

# **STATA BAYESIAN ANALYSIS REFERENCE MANUAL RELEASE 16**



A Stata Press Publication  
StataCorp LLC  
College Station, Texas



® Copyright © 1985–2019 StataCorp LLC  
All rights reserved  
Version 16

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845  
Typeset in  $\text{T}_{\text{E}}\text{X}$

ISBN-10: 1-59718-272-9

ISBN-13: 978-1-59718-272-0

This manual is protected by copyright. All rights are reserved. No part of this manual may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means—electronic, mechanical, photocopy, recording, or otherwise—without the prior written permission of StataCorp LLC unless permitted subject to the terms and conditions of a license granted to you by StataCorp LLC to use the software and documentation. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document.

StataCorp provides this manual “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. StataCorp may make improvements and/or changes in the product(s) and the program(s) described in this manual at any time and without notice.

The software described in this manual is furnished under a license agreement or nondisclosure agreement. The software may be copied only in accordance with the terms of the agreement. It is against the law to copy the software onto DVD, CD, disk, diskette, tape, or any other medium for any purpose other than backup or archival purposes.

The automobile dataset appearing on the accompanying media is Copyright © 1979 by Consumers Union of U.S., Inc., Yonkers, NY 10703-1057 and is reproduced by permission from CONSUMER REPORTS, April 1979.

Stata, **STATA** Stata Press, Mata, **MATA** and NetCourse are registered trademarks of StataCorp LLC.

Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations.

NetCourseNow is a trademark of StataCorp LLC.

Other brand and product names are registered trademarks or trademarks of their respective companies.

For copyright information about the software, type `help copyright` within Stata.

The suggested citation for this software is

StataCorp. 2019. *Stata: Release 16*. Statistical Software. College Station, TX: StataCorp LLC.



# Contents

Intro .....	Introduction to Bayesian analysis	1
Bayesian commands .....	Introduction to commands for Bayesian analysis	25
Bayesian estimation .....	Bayesian estimation commands	50
bayes .....	Bayesian regression models using the bayes prefix	52
bayesmh .....	Bayesian models using Metropolis–Hastings algorithm	112
bayesmh evaluators .....	User-defined evaluators with bayesmh	255
Bayesian postestimation .....	Postestimation tools for bayesmh and the bayes prefix	276
bayesgraph .....	Graphical summaries and convergence diagnostics	282
bayesstats .....	Bayesian statistics after Bayesian estimation	304
bayesstats ess .....	Effective sample sizes and related statistics	305
bayesstats grubin .....	Gelman–Rubin convergence diagnostics	314
bayesstats ic .....	Bayesian information criteria and Bayes factors	322
bayesstats ppvalues .....	Bayesian predictive p-values and other predictive summaries	332
bayesstats summary .....	Bayesian summary statistics	345
bayestest .....	Bayesian hypothesis testing	360
bayestest interval .....	Interval hypothesis testing	361
bayestest model .....	Hypothesis testing using model posterior probabilities	373
bayespredict .....	Bayesian predictions	385
set clevel .....	Set default credible level	419
bayes: betareg .....	Bayesian beta regression	422
bayes: binreg .....	Bayesian generalized linear models: Extensions to the binomial family	426
bayes: biprobit .....	Bayesian bivariate probit regression	430
bayes: clogit .....	Bayesian conditional logistic regression	434
bayes: cloglog .....	Bayesian complementary log-log regression	438
bayes: fracreg .....	Bayesian fractional response regression	442
bayes: glm .....	Bayesian generalized linear models	446
bayes: gnbreg .....	Bayesian generalized negative binomial regression	450
bayes: heckman .....	Bayesian Heckman selection model	454
bayes: heckprobit .....	Bayesian ordered probit model with sample selection	458
bayes: heckprobit .....	Bayesian probit model with sample selection	462
bayes: hetoprobit .....	Bayesian heteroskedastic ordered probit regression	466
bayes: hetoprobit .....	Bayesian heteroskedastic probit regression	470
bayes: hetregress .....	Bayesian heteroskedastic linear regression	474
bayes: intreg .....	Bayesian interval regression	478
bayes: logistic .....	Bayesian logistic regression, reporting odds ratios	482
bayes: logit .....	Bayesian logistic regression, reporting coefficients	486
bayes: mecloglog .....	Bayesian multilevel complementary log-log regression	490
bayes: meglm .....	Bayesian multilevel generalized linear model	495

bayes: meintreg	Bayesian multilevel interval regression	501
bayes: melogit	Bayesian multilevel logistic regression	506
bayes: menbreg	Bayesian multilevel negative binomial regression	511
bayes: meologit	Bayesian multilevel ordered logistic regression	516
bayes: meoprobit	Bayesian multilevel ordered probit regression	521
bayes: mepoisson	Bayesian multilevel Poisson regression	526
bayes: meprobit	Bayesian multilevel probit regression	531
bayes: mestreg	Bayesian multilevel parametric survival models	536
bayes: metobit	Bayesian multilevel tobit regression	541
bayes: mixed	Bayesian multilevel linear regression	546
bayes: mlogit	Bayesian multinomial logistic regression	551
bayes: mprobit	Bayesian multinomial probit regression	555
bayes: mvreg	Bayesian multivariate regression	559
bayes: nbreg	Bayesian negative binomial regression	563
bayes: ologit	Bayesian ordered logistic regression	567
bayes: oprobit	Bayesian ordered probit regression	571
bayes: poisson	Bayesian Poisson regression	575
bayes: probit	Bayesian probit regression	579
bayes: regress	Bayesian linear regression	583
bayes: streg	Bayesian parametric survival models	587
bayes: tnbreg	Bayesian truncated negative binomial regression	592
bayes: tobit	Bayesian tobit regression	596
bayes: tpoisson	Bayesian truncated Poisson regression	600
bayes: truncreg	Bayesian truncated regression	604
bayes: zinb	Bayesian zero-inflated negative binomial regression	608
bayes: zioprobit	Bayesian zero-inflated ordered probit regression	612
bayes: zip	Bayesian zero-inflated Poisson regression	616
Glossary		620
Subject and author index		629

# Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, [U] [27 Overview of Stata estimation commands](#); [R] [regress](#); and [D] [reshape](#). The first example is a reference to chapter 27, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	<i>Getting Started with Stata for Mac</i>
[GSU]	<i>Getting Started with Stata for Unix</i>
[GSW]	<i>Getting Started with Stata for Windows</i>
[U]	<i>Stata User's Guide</i>
[R]	<i>Stata Base Reference Manual</i>
[BAYES]	<i>Stata Bayesian Analysis Reference Manual</i>
[CM]	<i>Stata Choice Models Reference Manual</i>
[D]	<i>Stata Data Management Reference Manual</i>
[DSGE]	<i>Stata Dynamic Stochastic General Equilibrium Models Reference Manual</i>
[ERM]	<i>Stata Extended Regression Models Reference Manual</i>
[FMM]	<i>Stata Finite Mixture Models Reference Manual</i>
[FN]	<i>Stata Functions Reference Manual</i>
[G]	<i>Stata Graphics Reference Manual</i>
[IRT]	<i>Stata Item Response Theory Reference Manual</i>
[LASSO]	<i>Stata Lasso Reference Manual</i>
[XT]	<i>Stata Longitudinal-Data/Panel-Data Reference Manual</i>
[META]	<i>Stata Meta-Analysis Reference Manual</i>
[ME]	<i>Stata Multilevel Mixed-Effects Reference Manual</i>
[MI]	<i>Stata Multiple-Imputation Reference Manual</i>
[MV]	<i>Stata Multivariate Statistics Reference Manual</i>
[PSS]	<i>Stata Power, Precision, and Sample-Size Reference Manual</i>
[P]	<i>Stata Programming Reference Manual</i>
[RPT]	<i>Stata Reporting Reference Manual</i>
[SP]	<i>Stata Spatial Autoregressive Models Reference Manual</i>
[SEM]	<i>Stata Structural Equation Modeling Reference Manual</i>
[SVY]	<i>Stata Survey Data Reference Manual</i>
[ST]	<i>Stata Survival Analysis Reference Manual</i>
[TS]	<i>Stata Time-Series Reference Manual</i>
[TE]	<i>Stata Treatment-Effects Reference Manual: Potential Outcomes/Counterfactual Outcomes</i>
[I]	<i>Stata Glossary and Index</i>
[M]	<i>Mata Reference Manual</i>

# Title

Intro — Introduction to Bayesian analysis

Description      Remarks and examples      References      Also see

## Description

This entry provides a software-free introduction to Bayesian analysis. See [\[BAYES\]](#) **Bayesian commands** for an overview of the software for performing Bayesian analysis and for an [overview example](#).

## Remarks and examples

Remarks are presented under the following headings:

- What is Bayesian analysis?*
- Bayesian versus frequentist analysis, or why Bayesian analysis?*
- How to do Bayesian analysis*
- Advantages and disadvantages of Bayesian analysis*
- Brief background and literature review*
- Bayesian statistics*
  - Posterior distribution*
  - Selecting priors*
  - Point and interval estimation*
  - Comparing Bayesian models*
  - Posterior prediction*
- Bayesian computation*
- Markov chain Monte Carlo methods*
  - Metropolis–Hastings algorithm*
  - Adaptive random-walk Metropolis–Hastings*
  - Blocking of parameters*
  - Metropolis–Hastings with Gibbs updates*
  - Convergence diagnostics of MCMC*
- Summary*
- Video examples*

The first five sections provide a general introduction to Bayesian analysis. The remaining sections provide a more technical discussion of the concepts of Bayesian analysis.

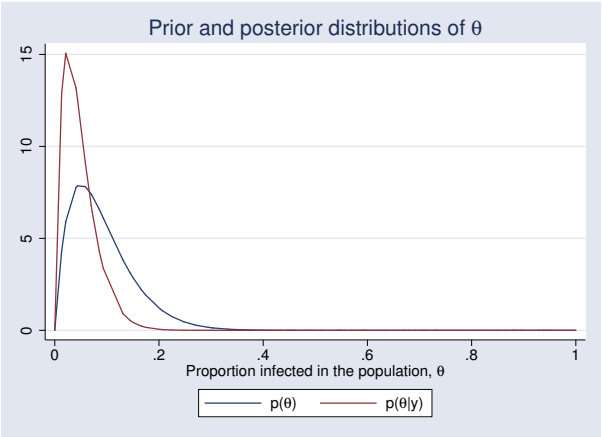
## What is Bayesian analysis?

Bayesian analysis is a statistical analysis that answers research questions about unknown parameters of statistical models by using probability statements. Bayesian analysis rests on the assumption that all model parameters are random quantities and thus can incorporate prior knowledge. This assumption is in sharp contrast with the more traditional, also called frequentist, statistical inference where all parameters are considered unknown but fixed quantities. Bayesian analysis follows a simple rule of probability, the Bayes rule, which provides a formalism for combining prior information with evidence from the data at hand. The Bayes rule is used to form the so called posterior distribution of model parameters. The posterior distribution results from updating the prior knowledge about model parameters with evidence from the observed data. Bayesian analysis uses the posterior distribution to form various summaries for the model parameters including point estimates such as posterior means, medians, percentiles, and interval estimates such as credible intervals. Moreover, all statistical tests about model parameters can be expressed as probability statements based on the estimated posterior distribution.

As a quick introduction to Bayesian analysis, we use an example, described in Hoff (2009, 3), of estimating the prevalence of a rare infectious disease in a small city. A small random sample of 20 subjects from the city will be checked for infection. The parameter of interest  $\theta \in [0, 1]$  is the fraction of infected individuals in the city. Outcome  $y$  records the number of infected individuals in the sample. A reasonable sampling model for  $y$  is a binomial model:  $y|\theta \sim \text{Binomial}(20, \theta)$ . Based on the studies from other comparable cities, the infection rate ranged between 0.05 and 0.20, with an average prevalence of 0.10. To use this information, we must conduct Bayesian analysis. This information can be incorporated into a Bayesian model with a prior distribution for  $\theta$ , which assigns a large probability between 0.05 and 0.20, with the expected value of  $\theta$  close to 0.10. One potential prior that satisfies this condition is a  $\text{Beta}(2, 20)$  prior with the expected value of  $2/(2 + 20) = 0.09$ . So, let's assume this prior for the infection rate  $\theta$ , that is,  $\theta \sim \text{Beta}(2, 20)$ . We sample individuals and observe none who have an infection, that is,  $y = 0$ . This value is not that uncommon for a small sample and a rare disease. For example, for a true rate  $\theta = 0.05$ , the probability of observing 0 infections in a sample of 20 individuals is about 36% according to the binomial distribution. So, our Bayesian model can be defined as follows:

$$\begin{aligned} y|\theta &\sim \text{Binomial}(20, \theta) \\ \theta &\sim \text{Beta}(2, 20) \end{aligned}$$

For this Bayesian model, we can actually compute the posterior distribution of  $\theta|y$ , which is  $\theta|y \sim \text{Beta}(2 + 0, 20 + 20 - 0) = \text{Beta}(2, 40)$ . The prior and posterior distributions of  $\theta$  are depicted below.



The posterior density (shown in red) is more peaked and shifted to the left compared with the prior distribution (shown in blue). The posterior distribution combined the prior information about  $\theta$  with the information from the data, from which  $y = 0$  provided evidence for a low value of  $\theta$  and shifted the prior density to the left to form the posterior density. Based on this posterior distribution, the posterior mean estimate of  $\theta$  is  $2/(2 + 40) = 0.048$  and the posterior probability that, for example,  $\theta < 0.10$  is about 93%.

If we compute a standard frequentist estimate of a population proportion  $\theta$  as a fraction of the infected subjects in the sample,  $\bar{y} = y/n$ , we will obtain 0 with the corresponding 95% confidence interval  $(\bar{y} - 1.96\sqrt{\bar{y}(1 - \bar{y})/n}, \bar{y} + 1.96\sqrt{\bar{y}(1 - \bar{y})/n})$  reducing to 0 as well. It may be difficult to convince a health policy maker that the prevalence of the disease in that city is indeed 0, given

the small sample size and the prior information available from comparable cities about a nonzero prevalence of this disease.

We used a beta prior distribution in this example, but we could have chosen another prior distribution that supports our prior knowledge. For the final analysis, it is important to consider a range of different prior distributions and investigate the sensitivity of the results to the chosen priors.

For more details about this example, see [Hoff \(2009\)](#). Also see [Beta-binomial model](#) in [\[BAYES\]](#) [bayesmh](#) for how to fit this model using [bayesmh](#).

## Bayesian versus frequentist analysis, or why Bayesian analysis?

Why use Bayesian analysis? Perhaps a better question is when to use Bayesian analysis and when to use frequentist analysis. The answer to this question mainly lies in your research problem. You should choose an analysis that answers your specific research questions. For example, if you are interested in estimating the probability that the parameter of interest belongs to some prespecified interval, you will need the Bayesian framework, because this probability cannot be estimated within the frequentist framework. If you are interested in a repeated-sampling inference about your parameter, the frequentist framework provides that.

Bayesian and frequentist approaches have very different philosophies about what is considered fixed and, therefore, have very different interpretations of the results. The Bayesian approach assumes that the observed data sample is fixed and that model parameters are random. The posterior distribution of parameters is estimated based on the observed data and the prior distribution of parameters and is used for inference. The frequentist approach assumes that the observed data are a repeatable random sample and that parameters are unknown but fixed and constant across the repeated samples. The inference is based on the sampling distribution of the data or of the data characteristics (statistics). In other words, Bayesian analysis answers questions based on the distribution of parameters conditional on the observed sample, whereas frequentist analysis answers questions based on the distribution of statistics obtained from repeated hypothetical samples, which would be generated by the same process that produced the observed sample given that parameters are unknown but fixed. Frequentist analysis consequently requires that the process that generated the observed data is repeatable. This assumption may not always be feasible. For example, in meta-analysis, where the observed sample represents the collected studies of interest, one may argue that the collection of studies is a one-time experiment.

Frequentist analysis is entirely data-driven and strongly depends on whether or not the data assumptions required by the model are met. On the other hand, Bayesian analysis provides a more robust estimation approach by using not only the data at hand but also some existing information or knowledge about model parameters.

In frequentist statistics, estimators are used to approximate the true values of the unknown parameters, whereas Bayesian statistics provides an entire distribution of the parameters. In our example of a prevalence of an infectious disease from [What is Bayesian analysis?](#), frequentist analysis produced one point estimate for the prevalence, whereas Bayesian analysis estimated the entire posterior distribution of the prevalence based on a given sample.

Frequentist inference is based on the sampling distributions of estimators of parameters and provides parameter point estimates and their standard errors as well as confidence intervals. The exact sampling distributions are rarely known and are often approximated by a large-sample normal distribution. Bayesian inference is based on the posterior distribution of the parameters and provides summaries of this distribution including posterior means and their MCMC standard errors (MCSE) as well as credible intervals. Although exact posterior distributions are known only in a number of cases, general posterior distributions can be estimated via, for example, Markov chain Monte Carlo (MCMC) sampling without any large-sample approximation.

Frequentist confidence intervals do not have straightforward probabilistic interpretations as do Bayesian credible intervals. For example, the interpretation of a 95% confidence interval is that if we repeat the same experiment many times and compute confidence intervals for each experiment, then 95% of those intervals will contain the true value of the parameter. For any given confidence interval, the probability that the true value is in that interval is either zero or one, and we do not know which. We may only infer that any given confidence interval provides a plausible range for the true value of the parameter. A 95% Bayesian credible interval, on the other hand, provides a range for a parameter such that the probability that the parameter lies in that range is 95%.

Frequentist hypothesis testing is based on a deterministic decision using a prespecified significance level of whether to accept or reject the null hypothesis based on the observed data, assuming that the null hypothesis is actually true. The decision is based on a  $p$ -value computed from the observed data. The interpretation of the  $p$ -value is that if we repeat the same experiment and use the same testing procedure many times, then given our null hypothesis is true, we will observe the result (test statistic) as extreme or more extreme than the one observed in the sample  $(100 \times p\text{-value})\%$  of the times. The  $p$ -value cannot be interpreted as a probability of the null hypothesis, which is a common misinterpretation. In fact, it answers the question of how likely are our data given that the null hypothesis is true, and not how likely is the null hypothesis given our data. The latter question can be answered by Bayesian hypothesis testing, where we can compute the probability of any hypothesis of interest.

## How to do Bayesian analysis

Bayesian analysis starts with the specification of a posterior model. The posterior model describes the probability distribution of all model parameters conditional on the observed data and some prior knowledge. The posterior distribution has two components: a likelihood, which includes information about model parameters based on the observed data, and a prior, which includes prior information (before observing the data) about model parameters. The likelihood and prior models are combined using the Bayes rule to produce the posterior distribution:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

If the posterior distribution can be derived in a closed form, we may proceed directly to the inference stage of Bayesian analysis. Unfortunately, except for some special models, the posterior distribution is rarely available explicitly and needs to be estimated via simulations. MCMC sampling can be used to simulate potentially very complex posterior models with an arbitrary level of precision. MCMC methods for simulating Bayesian models are often demanding in terms of specifying an efficient sampling algorithm and verifying the convergence of the algorithm to the desired posterior distribution. See [\[BAYES\] Bayesian estimation](#).

Inference is the next step of Bayesian analysis. If MCMC sampling is used for approximating the posterior distribution, the convergence of MCMC must be established before proceeding to inference (see, for example, [\[BAYES\] bayesgraph](#) and [\[BAYES\] bayesstats grubin](#)). Point and interval estimators are either derived from the theoretical posterior distribution or estimated from a sample simulated from the posterior distribution. Many Bayesian estimators, such as posterior mean and posterior standard deviation, involve integration. If the integration cannot be performed analytically to obtain a closed-form solution, sampling techniques such as Monte Carlo integration and MCMC and numerical integration are commonly used. See [\[BAYES\] Bayesian postestimation](#) and [\[BAYES\] bayesstats](#).

Another important step of Bayesian analysis is model checking, which is typically performed via posterior predictive checking. The idea behind posterior predictive checking is the comparison of various aspects of the distribution of the observed data with those of the replicated data. Replicated

data are simulated from the posterior predictive distribution of the fitted Bayesian model under the same conditions that generated the observed data, such as the same values of covariates, etc. The discrepancy between the distributions of the observed and replicated data is measured by test quantities (functions of the data and model parameters) and is quantified by so-called [posterior predictive  \$p\$ -values](#). See [\[BAYES\] bayesstats ppvalues](#) and [\[BAYES\] bayespredict](#).

Bayesian hypothesis testing can take two forms, which we refer to as interval-hypothesis testing and model-hypothesis testing. In an interval-hypothesis testing, the probability that a parameter or a set of parameters belongs to a particular interval or intervals is computed. In model hypothesis testing, the probability of a Bayesian model of interest given the observed data is computed. See [\[BAYES\] bayestest](#).

Model comparison is another common step of Bayesian analysis. The Bayesian framework provides a systematic and consistent approach to model comparison using the notion of posterior odds and related to them Bayes factors. See [\[BAYES\] bayesstats ic](#) for details.

Finally, prediction of some future unobserved data may also be of interest in Bayesian analysis. The prediction of a new data point is performed conditional on the observed data using the so-called posterior predictive distribution, which involves integrating out all parameters from the model with respect to their posterior distribution. Again, Monte Carlo integration is often the only feasible option for obtaining predictions. Prediction can also be helpful in estimating the goodness of fit of a model. See [\[BAYES\] bayespredict](#).

## Advantages and disadvantages of Bayesian analysis

Bayesian analysis is a powerful analytical tool for statistical modeling, interpretation of results, and prediction of data. It can be used when there are no standard frequentist methods available or the existing frequentist methods fail. However, one should be aware of both the advantages and disadvantages of Bayesian analysis before applying it to a specific problem.

The universality of the Bayesian approach is probably its main methodological advantage to the traditional frequentist approach. Bayesian inference is based on a single rule of probability, the Bayes rule, which is applied to all parametric models. This makes the Bayesian approach universal and greatly facilitates its application and interpretation. The frequentist approach, however, relies on a variety of estimation methods designed for specific statistical problems and models. Often, inferential methods designed for one class of problems cannot be applied to another class of models.

In Bayesian analysis, we can use previous information, either belief or experimental evidence, in a data model to acquire more balanced results for a particular problem. For example, incorporating prior information can mitigate the effect of a small sample size. Importantly, the use of the prior evidence is achieved in a theoretically sound and principled way.

By using the knowledge of the entire posterior distribution of model parameters, Bayesian inference is far more comprehensive and flexible than the traditional inference.

Bayesian inference is exact, in the sense that estimation and prediction are based on the posterior distribution. The latter is either known analytically or can be estimated numerically with an arbitrary precision. In contrast, many frequentist estimation procedures such as maximum likelihood rely on the assumption of asymptotic normality for inference.

Bayesian inference provides a straightforward and more intuitive interpretation of the results in terms of probabilities. For example, credible intervals are interpreted as intervals to which parameters belong with a certain probability, unlike the less straightforward repeated-sampling interpretation of the confidence intervals.



Bayesian models satisfy the likelihood principle (Berger and Wolpert 1988) that the information in a sample is fully represented by the likelihood function. This principle requires that if the likelihood function of one model is proportional to the likelihood function of another model, then inferences from the two models should give the same results. Some researchers argue that frequentist methods that depend on the experimental design may violate the likelihood principle.

Finally, as we briefly mentioned earlier, the estimation precision in Bayesian analysis is not limited by the sample size—Bayesian simulation methods may provide an arbitrary degree of precision.

Despite the conceptual and methodological advantages of the Bayesian approach, its application in practice is still considered controversial sometimes. There are two main reasons for this—the presumed subjectivity in specifying prior information and the computational challenges in implementing Bayesian methods. Along with the objectivity that comes from the data, the Bayesian approach uses potentially subjective prior distribution. That is, different individuals may specify different prior distributions. Proponents of frequentist statistics argue that for this reason, Bayesian methods lack objectivity and should be avoided. Indeed, there are settings such as clinical trial cases when the researchers want to minimize a potential bias coming from preexisting beliefs and achieve more objective conclusions. Even in such cases, however, a balanced and reliable Bayesian approach is possible. The trend in using noninformative priors in Bayesian models is an attempt to address the issue of subjectivity. On the other hand, some Bayesian proponents argue that the classical methods of statistical inference have built-in subjectivity such as a choice for a sampling procedure, whereas the subjectivity is made explicit in Bayesian analysis.

Building a reliable Bayesian model requires extensive experience from the researchers, which leads to the second difficulty in Bayesian analysis—setting up a Bayesian model and performing analysis is a demanding and involving task. This is true, however, to an extent for any statistical modeling procedure.

Lastly, one of the main disadvantages of Bayesian analysis is the computational cost. As a rule, Bayesian analysis involves intractable integrals that can only be computed using intensive numerical methods. Most of these methods such as MCMC are stochastic by nature and do not comply with the natural expectation from a user of obtaining deterministic results. Using simulation methods does not compromise the discussed advantages of Bayesian approach, but unquestionably adds to the complexity of its application in practice.

For more discussion about advantages and disadvantages of Bayesian analysis, see, for example, Thompson (2012), Bernardo and Smith (2000), and Berger and Wolpert (1988).

## Brief background and literature review

The principles of Bayesian analysis date back to the work of Thomas Bayes, who was a Presbyterian minister in Tunbridge Wells and Pierre Laplace, a French mathematician, astronomer, and physicist in the 18th century. Bayesian analysis started as a simple intuitive rule, named after Bayes, for updating beliefs on account of some evidence. For the next 200 years, however, Bayes's rule was just an obscure idea. Along with the rapid development of the standard or frequentist statistics in 20th century, Bayesian methodology was also developing, although with less attention and at a slower pace. One of the obstacles for the progress of Bayesian ideas has been the lasting opinion among mainstream statisticians of it being subjective. Another more-tangible problem for adopting Bayesian models in practice has been the lack of adequate computational resources. Nowadays, Bayesian statistics is widely accepted by researchers and practitioners as a valuable and feasible alternative.

Bayesian analysis proliferates in diverse areas including industry and government, but its application in sciences and engineering is particularly visible. Bayesian statistical inference is used in econometrics (Poirier [1995]; Chernozhukov and Hong [2003]; Kim, Shephard, and Chib [1998], Zellner [1997]);

education (Johnson 1997); epidemiology (Greenland 1998); engineering (Godsill and Rayner 1998); genetics (Iversen, Parmigiani, and Berry 1999); social sciences (Pollard 1986); hydrology (Parent et al. 1998); quality management (Rios Insua 1990); atmospheric sciences (Berliner et al. 1999); and law (DeGroot, Fienberg, and Kadane 1986), to name a few.

The subject of general statistics has been greatly influenced by the development of Bayesian ideas. Bayesian methodologies are now present in biostatistics (Carlin and Louis [2009]; Berry and Stangl [1996]); generalized linear models (Dey, Ghosh, and Mallick 2000); hierarchical modeling (Hobert 2000); statistical design (Chaloner and Verdinelli 1995); classification and discrimination (Neal [1996]; Neal [1999]); graphical models (Pearl 1998); nonparametric estimation (Müller and Vidakovic [1999]; Dey, Müller, and Sinha [1998]); survival analysis (Barlow, Clarotti, and Spizzichino 1993); sequential analysis (Carlin, Kadane, and Gelfand 1998); predictive inference (Aitchison and Dunsmore 1975); spatial statistics (Wolpert and Ickstadt [1998]; Besag and Higdon [1999]); testing and model selection (Kass and Raftery [1995]; Berger and Pericchi [1996]; Berger [2006]); and time series (Pole, West, and Harrison [1994]; West and Harrison [1997]).

Recent advances in computing allowed practitioners to perform Bayesian analysis using simulations. The simulation tools came from outside the statistics field—Metropolis et al. (1953) developed what is now known as a random-walk Metropolis algorithm to solve problems in statistical physics. Another landmark discovery was the Gibbs sampling algorithm (Geman and Geman 1984), initially used in image processing, which showed that exact sampling from a complex and otherwise intractable probability distribution is possible. These ideas were the seeds that led to the development of Markov chain Monte Carlo (MCMC)—a class of iterative simulation methods proved to be indispensable tools for Bayesian computations. Starting from the early 1990s, MCMC-based techniques slowly emerged in the mainstream statistical practice. More powerful and specialized methods appeared, such as perfect sampling (Propp and Wilson 1996), reversible-jump MCMC (Green 1995) for traversing variable dimension state spaces, and particle systems (Gordon, Salmond, and Smith 1993). Consequent widespread application of MCMC was imminent (Berger 2000) and influenced various specialized fields. For example, Gelman and Rubin (1992) investigated MCMC for the purpose of exploring posterior distributions; Geweke (1999) surveyed simulation methods for Bayesian inference in econometrics; Kim, Shephard, and Chib (1998) used MCMC simulations to fit stochastic volatility models; Carlin, Kadane, and Gelfand (1998) implemented Monte Carlo methods for identifying optimal strategies in clinical trials; Chib and Greenberg (1995) provided Bayesian formulation of a number of important econometrics models; and Chernozhukov and Hong (2003) reviewed some econometrics models involving Laplace-type estimators from an MCMC perspective. For more comprehensive exposition of MCMC, see, for example, Robert and Casella (2004); Tanner (1996); Gamerman and Lopes (2006); Chen, Shao, and Ibrahim (2000); and Brooks et al. (2011).

## Bayesian statistics

### Posterior distribution

To formulate the principles of Bayesian statistics, we start with a simple case when one is concerned with the interaction of two random variables, **A** and **B**. Let  $p(\cdot)$  denote either a probability mass function or a density, depending on whether the variables are discrete or continuous. The rule of conditional probability,

$$p(\mathbf{A}|\mathbf{B}) = \frac{p(\mathbf{A}, \mathbf{B})}{p(\mathbf{B})}$$

can be used to derive the so-called Bayes's theorem:

$$p(\mathbf{B}|\mathbf{A}) = \frac{p(\mathbf{A}|\mathbf{B})p(\mathbf{B})}{p(\mathbf{A})} \quad (1)$$

This rule also holds in the more general case when  $\mathbf{A}$  and  $\mathbf{B}$  are random vectors.

In a typical statistical problem, we have a data vector  $\mathbf{y}$ , which is assumed to be a sample from a probability model with an unknown parameter vector  $\boldsymbol{\theta}$ . We represent this model using the likelihood function  $L(\boldsymbol{\theta}; \mathbf{y}) = f(\mathbf{y}; \boldsymbol{\theta}) = \prod_{i=1}^n f(y_i|\boldsymbol{\theta})$ , where  $f(y_i|\boldsymbol{\theta})$  denotes the probability density function of  $y_i$  given  $\boldsymbol{\theta}$ . We want to infer some properties of  $\boldsymbol{\theta}$  based on the data  $\mathbf{y}$ . In Bayesian statistics, model parameters  $\boldsymbol{\theta}$  is a random vector. We assume that  $\boldsymbol{\theta}$  has a probability distribution  $p(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta})$ , which is referred to as a prior distribution. Because both  $\mathbf{y}$  and  $\boldsymbol{\theta}$  are random, we can apply Bayes's theorem (1) to derive the posterior distribution of  $\boldsymbol{\theta}$  given data  $\mathbf{y}$ ,

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})} = \frac{f(\mathbf{y}; \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{m(\mathbf{y})} \quad (2)$$

where  $m(\mathbf{y}) \equiv p(\mathbf{y})$ , known as the marginal distribution of  $\mathbf{y}$ , is defined by

$$m(\mathbf{y}) = \int f(\mathbf{y}; \boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \quad (3)$$

The marginal distribution  $m(\mathbf{y})$  in (3) does not depend on the parameter of interest  $\boldsymbol{\theta}$ , and we can, therefore, reduce (2) to

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto L(\boldsymbol{\theta}; \mathbf{y})\pi(\boldsymbol{\theta}) \quad (4)$$

Equation (4) is fundamental in Bayesian analysis and states that the posterior distribution of model parameters is proportional to their likelihood and prior probability distributions. We will often use (4) in the computationally more-convenient log-scale form

$$\ln\{p(\boldsymbol{\theta}|\mathbf{y})\} = l(\boldsymbol{\theta}; \mathbf{y}) + \ln\{\pi(\boldsymbol{\theta})\} - c \quad (5)$$

where  $l(\cdot; \cdot)$  denotes the log likelihood of the model. Depending on the analytical procedure involving the log-posterior  $\ln\{p(\boldsymbol{\theta}|\mathbf{y})\}$ , the actual value of the constant  $c = \ln\{m(\mathbf{y})\}$  may or may not be relevant. For valid statistical analysis, however, we will always assume that  $c$  is finite.

## Selecting priors

In Bayesian analysis, we seek a balance between prior information in a form of expert knowledge or belief and evidence from data at hand. Achieving the right balance is one of the difficulties in Bayesian modeling and inference. In general, we should not allow the prior information to overwhelm the evidence from the data, especially when we have a large data sample. A famous theoretical result, the Bernstein–von Mises theorem, states that in large data samples, the posterior distribution is independent of the prior distribution and, therefore, Bayesian and likelihood-based inferences should yield essentially the same results. On the other hand, we need a strong enough prior to support weak evidence that usually comes from insufficient data. It is always good practice to perform sensitivity analysis to check the dependence of the results on the choice of a prior.

The flexibility of choosing the prior freely is one of the main controversial issues associated with Bayesian analysis and the reason why some practitioners view the latter as subjective. It is also the reason why the Bayesian practice, especially in the early days, was dominated by noninformative priors. Noninformative priors, also called flat or vague priors, assign equal probabilities to all possible states of the parameter space with the aim of rectifying the subjectivity problem. One of the disadvantages of flat priors is that they are often improper; that is, they do not specify a legitimate probability distribution. For example, a uniform prior for a continuous parameter over an unbounded domain does not integrate to a finite number. However, this is not necessarily a problem because the corresponding posterior distribution may still be proper. Although Bayesian inference based on improper priors is possible, this is equivalent to discarding the terms  $\log\pi(\boldsymbol{\theta})$  and  $c$  in (5), which nullifies the benefit of Bayesian analysis because it reduces the latter to an inference based only on the likelihood. This is why there is a strong objection to the practice of noninformative priors. In recent years, an increasing number of researchers have advocated the use of sound informative priors, for example, [Thompson \(2014\)](#). For example, using informative priors is mandatory in areas such as genetics, where prior distributions have a physical basis and reflect scientific knowledge.

Another convenient preference for priors is to use [conjugate priors](#). Their choice is desirable from technical and computational standpoints but may not necessarily provide a realistic representation of the model parameters. Because of the limited arsenal of conjugate priors, an inclination to overuse them severely limits the flexibility of Bayesian modeling.

## Point and interval estimation

In Bayesian statistics, inference about parameters  $\boldsymbol{\theta}$  is based on the posterior distribution  $p(\boldsymbol{\theta}|\mathbf{y})$  and various ways of summarizing this distribution. Point and interval estimates can be used to summarize this distribution.

Commonly used point estimators are the posterior mean,

$$E(\boldsymbol{\theta}|\mathbf{y}) = \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta}$$

and the posterior median,  $q_{0.5}(\boldsymbol{\theta})$ , which is the 0.5 quantile of the posterior; that is,

$$P\{\boldsymbol{\theta} \leq q_{0.5}(\boldsymbol{\theta}|\mathbf{y})\} = 0.5$$

Another point estimator is the posterior mode, which is the value of  $\boldsymbol{\theta}$  that maximizes  $p(\boldsymbol{\theta}|\mathbf{y})$ .

Interval estimation is performed by constructing so-called credible intervals (CRIs). CRIs are special cases of credible regions. Let  $1 - \alpha \in (0, 1)$  be some predefined credible level. Then, an  $\{(1 - \alpha) \times 100\}\%$  credible set  $R$  of  $\boldsymbol{\theta}$  is such that

$$\Pr(\boldsymbol{\theta} \in R|\mathbf{y}) = \int_R p(\boldsymbol{\theta}|\mathbf{y}) d\boldsymbol{\theta} = 1 - \alpha$$

We consider two types of CRIs. The first one is based on quantiles. The second one is the highest posterior density (HPD) interval.

An  $\{(1 - \alpha) \times 100\}\%$  quantile-based, or also known as an equal-tailed CRI, is defined as  $(q_{\alpha/2}, q_{1-\alpha/2})$ , where  $q_a$  denotes the  $a$ th quantile of the posterior distribution. A commonly reported equal-tailed CRI is  $(q_{0.025}, q_{0.975})$ .

HPD interval is defined as an  $\{(1 - \alpha) \times 100\}\%$  CRI of the shortest width. As its name implies, this interval corresponds to the region of the posterior density with the highest concentration. For a unimodal posterior distribution, HPD is unique, but for a multimodal distribution it may not be unique. Computational approaches for calculating HPD are described in [Chen and Shao \(1999\)](#) and [Eberly and Casella \(2003\)](#).

Comparing Bayesian models

Model comparison is another important aspect of Bayesian statistics. We are often interested in comparing two or more plausible models for our data.

Let’s assume that we have models  $M_j$  parameterized by vectors  $\theta_j, j = 1, \dots, r$ . We may have varying degree of belief in each of these models given by prior probabilities  $p(M_j)$ , such that  $\sum_{j=1}^r p(M_j) = 1$ . By applying Bayes’s theorem, we find the posterior model probabilities

$$p(M_j|\mathbf{y}) = \frac{p(\mathbf{y}|M_j)p(M_j)}{p(\mathbf{y})}$$

where  $p(\mathbf{y}|M_j) = m_j(y)$  is the marginal likelihood of  $M_j$  with respect to  $\mathbf{y}$ . Because of the difficulty in calculating  $p(\mathbf{y})$ , it is a common practice to compare two models, say,  $M_j$  and  $M_k$ , using the posterior odds ratio

$$PO_{jk} = \frac{p(M_j|\mathbf{y})}{p(M_k|\mathbf{y})} = \frac{p(\mathbf{y}|M_j)p(M_j)}{p(\mathbf{y}|M_k)p(M_k)}$$

If all models are equally plausible, that is,  $p(M_j) = 1/r$ , the posterior odds ratio reduces to the so-called Bayes factors (BF) ([Jeffreys 1935](#)),

$$BF_{jk} = \frac{p(\mathbf{y}|M_j)}{p(\mathbf{y}|M_k)} = \frac{m_j(\mathbf{y})}{m_k(\mathbf{y})}$$

which are simply ratios of marginal likelihoods.

[Jeffreys \(1961\)](#) recommended an interpretation of  $BF_{jk}$  based on half-units of the log scale. The following table provides some rules of thumb:

$\log_{10}(BF_{jk})$	$BF_{jk}$	Evidence against $M_k$
0 to 1/2	1 to 3.2	Bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
>2	>100	Decisive

The Schwarz criterion BIC ([Schwarz 1978](#)) is an approximation of BF in case of arbitrary but proper priors. [Kass and Raftery \(1995\)](#) and [Berger \(2006\)](#) provide a detailed exposition of Bayes factors, their calculation, and their role in model building and testing.

## Posterior prediction

Prediction is another essential part of statistical analysis. In Bayesian statistics, prediction is performed using the posterior predictive distribution. The probability of observing some future data  $\mathbf{y}^*$  given the observed data  $\mathbf{y}$  can be obtained by the marginalization of

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\mathbf{y}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

which, assuming that  $\mathbf{y}^*$  is independent of  $\mathbf{y}$  given  $\boldsymbol{\theta}$ , can be simplified to

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta} \quad (6)$$

Equation (6) is called a posterior predictive distribution and is used for Bayesian prediction. See [\[BAYES\] bayespredict](#) and [\[BAYES\] bayesstats ppvalues](#).

## Bayesian computation

An unavoidable difficulty in performing Bayesian analysis is the need to compute integrals such as those expressing marginal distributions and posterior moments. The integrals involved in Bayesian inference are of the form  $E\{g(\boldsymbol{\theta})\} = \int g(\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$  for some function  $g(\cdot)$  of the random vector  $\boldsymbol{\theta}$ . With the exception of a few cases for which analytical integration is possible, the integration is performed via simulations.

Given a sample from the posterior distribution, we can use Monte Carlo integration to approximate the integrals. Let  $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_T$  be an independent sample from  $p(\boldsymbol{\theta}|\mathbf{y})$ .

The original integral of interest  $E\{g(\boldsymbol{\theta})\}$  can be approximated by

$$\hat{g} = \frac{1}{T} \sum_{t=1}^T g(\boldsymbol{\theta}_t)$$

Moreover, if  $g$  is a scalar function, under some mild conditions, the central limit theorem holds

$$\hat{g} \approx N[E\{g(\boldsymbol{\theta})\}, \sigma^2/T]$$

where  $\sigma^2 = \text{Cov}\{g(\boldsymbol{\theta}_i)\}$  can be approximated by the sample variance  $\sum_{t=1}^T \{g(\boldsymbol{\theta}_t) - \hat{g}\}^2/T$ . If the sample is not independent, then  $\hat{g}$  still approximates  $E\{g(\boldsymbol{\theta})\}$  but the variance  $\sigma^2$  is given by

$$\sigma^2 = \text{Var}\{g(\boldsymbol{\theta}_t)\} + 2 \sum_{k=1}^{\infty} \text{Cov}\{g(\boldsymbol{\theta}_t), g(\boldsymbol{\theta}_{t+k})\} \quad (7)$$

and needs to be approximated. Moreover, the conditions needed for the central limit theorem to hold involve the convergence rate of the chain and can be difficult to check in practice ([Tierney 1994](#)).

The Monte Carlo integration method solves the problem of Bayesian computation of computing a posterior distribution by sampling from that posterior distribution. The latter has been an important problem in computational statistics and a focus of intense research. Rejection sampling techniques serve as basic tools for generating samples from a general probability distribution ([von Neumann 1951](#)). They are based on the idea that samples from the target distribution can be obtained from another,

easy-to-sample distribution according to some acceptance–rejection rule for the samples from this distribution. It was soon recognized, however, that the acceptance–rejection methods did not scale well with the increase of dimensions, a problem known as the “curse of dimensionality”, essentially reducing the acceptance probability to zero. An alternative solution was to use the Markov chains to generate sequences of correlated sample points from the domain of the target distribution and keeping a reasonable rate of acceptance. It was not long before Markov chain Monte Carlo methods were accepted as effective tools for approximate sampling from general posterior distributions (Tanner and Wong 1987).

## Markov chain Monte Carlo methods

Every MCMC method is designed to generate values from a transition kernel such that the draws from that kernel converge to a prespecified target distribution. It simulates a Markov chain with the target distribution as the stationary or equilibrium distribution of the chain. By definition, a Markov chain is any sequence of values or states from the domain of the target distribution, such that each value depends on its immediate predecessor only. For a well-designed MCMC, the longer the chain, the closer the samples to the stationary distribution. MCMC methods differ substantially in their simulation efficiency and computational complexity.

The Metropolis algorithm proposed in Metropolis and Ulam (1949) and Metropolis et al. (1953) appears to be the earliest version of MCMC. The algorithm generates a sequence of states, each obtained from the previous one, according to a Gaussian proposal distribution centered at that state. Hastings (1970) described a more-general version of the algorithm, now known as a Metropolis–Hastings (MH) algorithm, which allows any distribution to be used as a proposal distribution. Below we review the general MH algorithm and some of its special cases.

### Metropolis–Hastings algorithm

Here we present the MH algorithm for sampling from a posterior distribution in a general formulation. It requires the specification of a proposal probability distribution  $q(\cdot)$  and a starting state  $\theta_0$  within the domain of the posterior, that is,  $p(\theta_0|\mathbf{y}) > 0$ . The algorithm generates a Markov chain  $\{\theta_t\}_{t=0}^{T-1}$  such that at each step  $t$  1) a proposal state  $\theta_*$  is generated conditional on the current state, and 2)  $\theta_*$  is accepted or rejected according to the suitably defined acceptance probability.

For  $t = 1, \dots, T - 1$ :

1. Generate a proposal state:  $\theta_* \sim q(\cdot|\theta_{t-1})$ .
2. Calculate the acceptance probability  $\alpha(\theta_*|\theta_{t-1}) = \min\{r(\theta_*|\theta_{t-1}), 1\}$ , where

$$r(\theta_*|\theta_{t-1}) = \frac{p(\theta_*|\mathbf{y})q(\theta_{t-1}|\theta_*)}{p(\theta_{t-1}|\mathbf{y})q(\theta_*|\theta_{t-1})}$$

3. Draw  $u \sim \text{Uniform}(0, 1)$ .
4. Set  $\theta_t = \theta_*$  if  $u < \alpha(\theta_*|\theta_{t-1})$ , and  $\theta_t = \theta_{t-1}$  otherwise.

We refer to the iteration steps 1 through 4 as an MH update. By design, any Markov chain simulated using this MH algorithm is guaranteed to have  $p(\theta|\mathbf{y})$  as its stationary distribution.

Two important criteria measuring the efficiency of MCMC are the acceptance rate of the chain and the degree of autocorrelation in the generated sample. When the acceptance rate is close to 0, then most of the proposals are rejected, which means that the chain failed to explore regions of appreciable posterior probability. The other extreme is when the acceptance probability is close to 1, in which

case the chain stays in a small region and fails to explore the whole posterior domain. An efficient MCMC has an acceptance rate that is neither too small nor too large and also has small autocorrelation. [Gelman, Gilks, and Roberts \(1997\)](#) showed that in the case of a multivariate posterior and proposal distributions, an acceptance rate of 0.234 is asymptotically optimal and, in the case of a univariate posterior, the optimal value is 0.45.

A special case of MH employs a Metropolis update with  $q(\cdot)$  being a symmetric distribution. Then, the acceptance ratio reduces to a ratio of posterior probabilities,

$$r(\theta_*|\theta_{t-1}) = \frac{p(\theta_*|\mathbf{y})}{p(\theta_{t-1}|\mathbf{y})}$$

The symmetric Gaussian distribution is a common choice for a proposal distribution  $q(\cdot)$ , and this is the one used in the original Metropolis algorithm.

Another important MCMC method that can be viewed as a special case of MH is Gibbs sampling ([Gelfand et al. 1990](#)), where the updates are the full conditional distributions of each parameter given the rest of the parameters. Gibbs updates are always accepted. If  $\theta = (\theta^1, \dots, \theta^d)$  and, for  $j = 1 \dots, d$ ,  $q_j$  is the conditional distribution of  $\theta^j$  given the rest  $\theta^{\{-j\}}$ , then the Gibbs algorithm is the following. For  $t = 1, \dots, T - 1$  and for  $j = 1, \dots, d$ :  $\theta_t^j \sim q_j(\cdot|\theta_{t-1}^{\{-j\}})$ . This step is referred to as a Gibbs update.

All MCMC methods share some limitations and potential problems. First, any simulated chain is influenced by its starting values, especially for short MCMC runs. It is required that the starting point has a positive posterior probability, but even when this condition is satisfied, if we start somewhere in a remote tail of the target distribution, it may take many iterations to reach a region of appreciable probability. Second, because there is no obvious stopping criterion, it is not easy to decide for how long to run the MCMC algorithm to achieve convergence to the target distribution. Third, the observations in MCMC samples are strongly dependent and this must be taken into account in any subsequent statistical inference. For example, the errors associated with the Monte Carlo integration should be calculated according to (7), which accounts for autocorrelation.

## Adaptive random-walk Metropolis–Hastings

The choice of a proposal distribution  $q(\cdot)$  in the MH algorithm is crucial for the mixing properties of the resulting Markov chain. The problem of determining an optimal proposal for a particular target posterior distribution is difficult and is still being researched actively. All proposed solutions are based on some form of an adaptation of the proposal distribution as the Markov chain progresses, which is carefully designed to preserve the ergodicity of the chain, that is, its tendency to converge to the target distribution. These methods are known as adaptive MCMC methods ([Haario, Saksman, and Tamminen \[2001\]](#); [Giordani and Kohn \[2010\]](#); and [Roberts and Rosenthal \[2009\]](#)).

The majority of adaptive MCMC methods are random-walk MH algorithms with updates of the form:  $\theta_* = \theta_{t-1} + Z_t$ , where  $Z_t$  follows some symmetric distribution. Specifically, we consider a Gaussian random-walk MH algorithm with  $Z_t \sim N(0, \rho^2 \Sigma)$ , where  $\rho$  is a scalar controlling the scale of random jumps for generating updates and  $\Sigma$  is a  $d$ -dimensional covariance matrix. One of the first important results regarding adaptation is from [Gelman, Gilks, and Roberts \(1997\)](#), where the authors derive the optimal scaling factor  $\rho = 2.38/\sqrt{d}$  and note that the optimal  $\Sigma$  is the true covariance matrix of the target distribution.



Haario, Saksman, and Tamminen (2001) proposes  $\Sigma$  to be estimated by the empirical covariance matrix plus a small diagonal matrix  $\epsilon \times I_d$  to prevent zero covariance matrices. Alternatively, Roberts and Rosenthal (2009) proposed a mixture of the two covariance matrices,

$$\Sigma_t = \beta \hat{\Sigma} + (1 - \beta) \Sigma_0$$

for some fixed covariance matrix  $\Sigma_0$  and  $\beta \in [0, 1]$ .

Because the proposal distribution of an adaptive MH algorithm changes at each step, the ergodicity of the chain is not necessarily preserved. However, under certain assumptions about the adaptation procedure, the ergodicity does hold; see Roberts and Rosenthal (2007), Andrieu and Moulines (2006), Atchadé and Rosenthal (2005), and Giordani and Kohn (2010) for details.

## Blocking of parameters

In the original MH algorithm, the update steps of generating proposals and applying the acceptance–rejection rule are performed for all model parameters simultaneously. For high-dimensional models, this may result in a poor mixing—the Markov chain may stay in the tails of the posterior distribution for long periods of time and traverse the posterior domain very slowly. Suboptimal mixing is manifested by either very high or very low acceptance rates. Adaptive MH algorithms are also prone to this problem, especially when model parameters have very different scales. An effective solution to this problem is called *blocking*—model parameters are separated into two or more subsets or blocks and MH updates are applied to each block separately in the order that the blocks are specified.

Let’s separate a vector of parameters into  $B$  blocks:  $\theta = \{\theta^1, \dots, \theta^B\}$ . The version of the Gaussian random-walk MH algorithm with blocking is as follows.

Let  $T_0$  be the number of burn-in iterations,  $T$  be the number of MCMC samples, and  $\rho_b^2 \Sigma^b$ ,  $b = 1, \dots, B$ , be block-specific proposal covariance matrices. Let  $\theta_0$  be the starting point within the domain of the posterior, that is,  $p(\theta_0 | \mathbf{y}) > 0$ .

1. At iteration  $t$ , let  $\theta_t = \theta_{t-1}$ .
2. For a block of parameters  $\theta_t^b$ :
  - 2.1. Let  $\theta_* = \theta_t$ . Generate a proposal for the  $b$ th block:  $\theta_*^b = \theta_{t-1}^b + \epsilon$ , where  $\epsilon \sim N(0, \rho_b^2 \Sigma^b)$ .
  - 2.2. Calculate the acceptance ratio,

$$r(\theta_* | \theta_t) = \frac{p(\theta_* | \mathbf{y})}{p(\theta_t | \mathbf{y})}$$

where  $\theta_* = (\theta_t^1, \theta_t^2, \dots, \theta_t^{b-1}, \theta_*^b, \theta_t^{b+1}, \dots, \theta_t^B)$ .

- 2.3. Draw  $u \sim \text{Uniform}(0, 1)$ .
- 2.4. Let  $\theta_t^b = \theta_*^b$  if  $u < \min\{r(\theta_* | \theta_t), 1\}$ .
3. Repeat step 2 for  $b = 1, \dots, B$ .
4. Repeat steps 1 through 3 for  $t = 1, \dots, T + T_0 - 1$ .
5. The final sequence is  $\{\theta_t\}_{t=T_0}^{T+T_0-1}$ .

Blocking may not always improve efficiency. For example, separating all parameters in individual blocks (the so-called one-at-a-time update regime) can lead to slow mixing when some parameters are highly correlated. A Markov chain may explore the posterior domain very slowly if highly correlated parameters are updated independently. There are no theoretical results about optimal blocking, so

you will need to use your judgment when determining the best set of blocks for your model. As a rule, parameters that are expected to be highly correlated are specified in one block. This will generally improve mixing of the chain unless the proposal correlation matrix does not capture the actual correlation structure of the block. For example, if there are two parameters in the block that have very different scales, adaptive MH algorithms that use the identity matrix for the initial proposal covariance may take a long time to approximate the optimal proposal correlation matrix. The user should, therefore, consider not only the probabilistic relationship between the parameters in the model, but also their scales to determine an optimal set of blocks.

## Metropolis–Hastings with Gibbs updates

The original Gibbs sampler updates each model parameter one at a time according to its full conditional distribution. We have already noted that Gibbs is a special case of the MH algorithm. Some of the advantages of Gibbs sampling include its high efficiency, because all proposals are automatically accepted, and that it does not require any additional tuning for proposal distributions in MH algorithms. Unfortunately, for most posterior distributions in practice, the full conditionals are either not available or are very difficult to sample from. It may be the case, however, that for some model parameters or groups of parameters, the full conditionals are available and are easy to generate samples from. This is done in a hybrid MH algorithm, which implements Gibbs updates for only some blocks of parameters. A hybrid MH algorithm combines Gaussian random-walk updates with Gibbs updates to improve the mixing of the chain.

The MH algorithm with blocking allows different samplers to be used for updating different blocks. If there is a group of model parameters with a conjugate prior (or semiconjugate prior), we can place this group of parameters in a separate block and use Gibbs sampling for it. This can greatly improve the overall sampling efficiency of the algorithm.

For example, suppose that the data are normally distributed with a known mean  $\mu$  and that we specify an inverse-gamma prior for  $\sigma^2$  with shape  $\alpha$  and scale  $\beta$ , which are some fixed constants.

$$y \sim N(\mu, \sigma^2), \quad \sigma^2 \sim \text{InvGamma}(\alpha, \beta)$$

The full conditional distribution for  $\sigma^2$  in this case is also an inverse-gamma distribution, but with different shape and scale parameters,

$$\sigma^2 \sim \text{InvGamma} \left\{ \tilde{\alpha} = \alpha + \frac{n}{2}, \quad \tilde{\beta} = \beta + \frac{1}{2} \sum_{i=1}^n (y_i - \mu)^2 \right\}$$

where  $n$  is the data sample size. So, an inverse-gamma prior for the variance is a conjugate prior in this model. We can thus place  $\sigma^2$  in a separate block and set up a Gibbs sampling for it using the above full conditional distribution.

See [Methods and formulas](#) in [\[BAYES\] bayesmh](#) for details.

## Convergence diagnostics of MCMC

Checking convergence of MCMC is an essential step in any MCMC simulation. Bayesian inference based on an MCMC sample is valid only if the Markov chain has converged and the sample is drawn from the desired posterior distribution. It is important that we verify the convergence for all model parameters and not only for a subset of parameters of interest. One difficulty with assessing

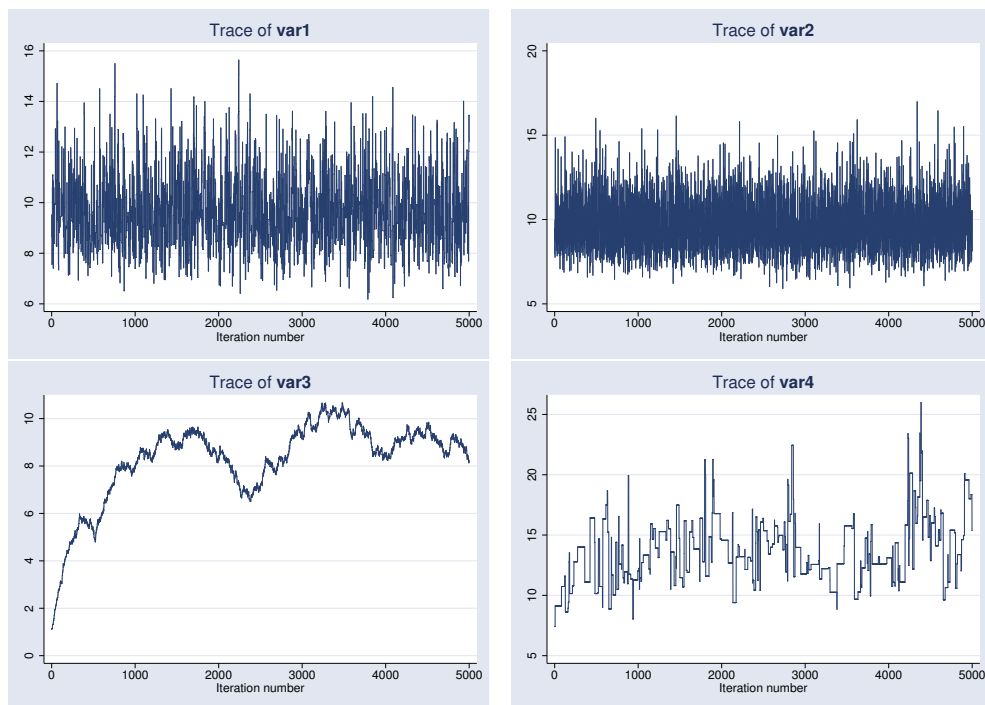
convergence of MCMC is that there is no single conclusive convergence criterion. The diagnostic usually involves checking for several necessary (but not necessarily sufficient) conditions for convergence. In general, the more aspects of the MCMC sample you inspect, the more reliable your results are.

The most extensive review of the methods for assessing convergence is [Cowles and Carlin \(1996\)](#). Other discussions about monitoring convergence can be found in [Gelman et al. \(2014\)](#) and [Brooks et al. \(2011\)](#).

There are at least two general approaches for detecting convergence issues. The first one is to inspect the mixing and time trends within the chains of individual parameters. The second one is to examine the mixing and time trends of multiple chains for each parameter. The lack of convergence in a Markov chain can be especially difficult to detect in a case of pseudoconvergence, which often occurs with multimodal posterior distributions. Pseudoconvergence occurs when the chain appears to have converged but it actually explored only a portion of the domain of a posterior distribution. To check for pseudoconvergence, [Gelman and Rubin \(1992\)](#) recommend running multiple chains from different starting states and comparing them; see [\[BAYES\] bayesstats grubin](#).

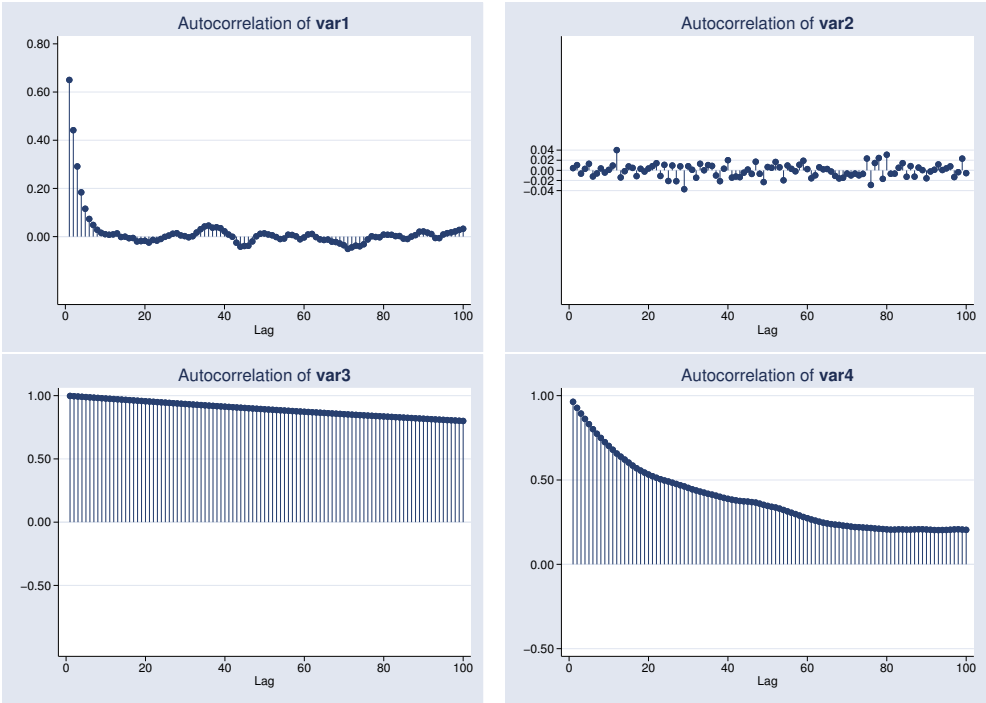
Trace plots are the most accessible convergence diagnostics and are easy to inspect visually. The trace plot of a parameter plots the simulated values for this parameter versus the iteration number. The trace plot of a well-mixing parameter should traverse the posterior domain rapidly and should have nearly constant mean and variance. See [\[BAYES\] bayesgraph](#) for details.

In the next figure, we show examples of trace plots for four parameters: `var1`, `var2`, `var3`, and `var4`. The first two parameters, `var1` and `var2`, have well-mixing chains, and the other two have poorly mixing chains. The chain for the parameter `var1` has a moderate acceptance rate, about 35%, and efficiency between 10% and 20%. This is a typical result for a Gaussian random-walk MH algorithm that has achieved convergence. The trace plot of `var2` in the top right panel shows almost perfect mixing—this is a typical example of Gibbs sampling with an acceptance rate close to 1 and efficiency above 95%. Although both chains traverse their marginal posterior domains, the right one does it more rapidly. On the downside, more efficient MCMC algorithms such as Gibbs sampling are usually associated with a higher computational cost.

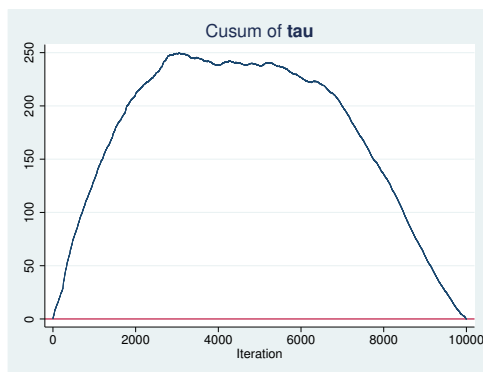
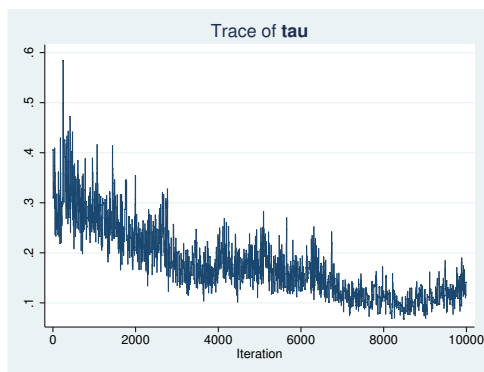


The bottom two trace plots illustrate cases of bad mixing and a lack of convergence. On the left, the chain for `var3` exhibits high acceptance rate but poor coverage of the posterior domain manifested by random drifting in isolated regions. This chain was produced by a Gaussian random-walk MH algorithm with a proposal distribution with a very small variance. On the right, the chain for the parameter `var4` has a very low acceptance rate, below 3%, because the used proposal distribution had a very large variance. In both cases, the chains do not converge; the simulation results do not represent the posterior distribution and should thus be discarded.

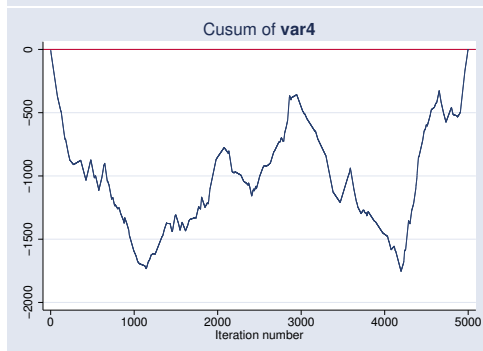
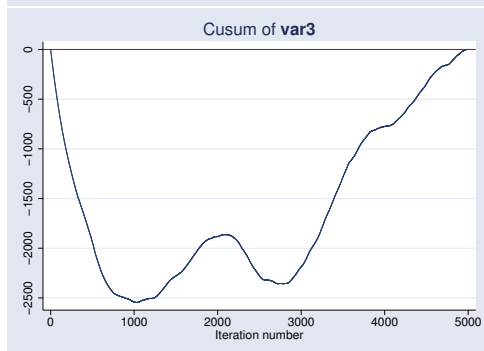
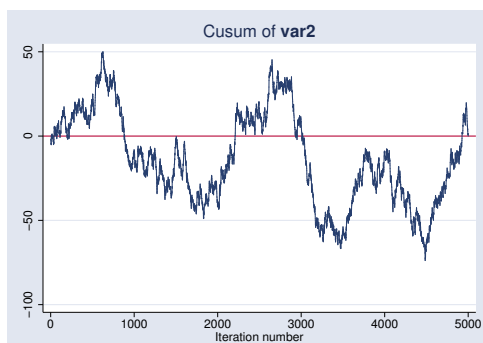
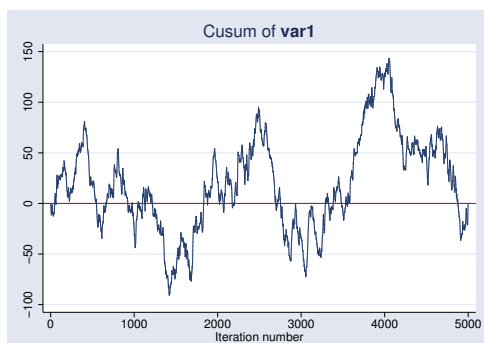
As we stated before, samples simulated using MCMC methods are correlated. The smaller the correlation, the more efficient the sampling process. Most of the MH algorithms typically generate highly correlated draws, whereas the Gibbs algorithm typically generates less-correlated draws. Below we show autocorrelation plots for the same four parameters using the same MCMC samples. The autocorrelation of `var1`, the one that comes from a well-mixing MH chain, becomes negligible fairly quickly, after about 10 lags. On the other hand, the autocorrelation of `var2` simulated using Gibbs sampling is essentially negligible for all positive lags. In the case of a poor mixing because of a small proposal variance (parameter `var3`), we observe very high positive correlation for at least 100 lags. The autocorrelation of `var4` is high but is lower than that of `var3`.



Yu and Mykland (1998) proposed a graphical procedure for assessing the convergence of individual parameters based on cumulative sums, also known as a cusum plot. By definition, any cusum plot starts at 0 and ends at 0. Cusum plots are useful for detecting drifts in the chain. For a chain without trend, the cusum plot should cross the  $x$  axis. For example, early drifts may indicate dependence on starting values. If we detect an early drift, we should discard an initial part of the chain and run it longer. Below, we show the trace plot of a poorly mixing parameter  $\tau$  and its corresponding cusum plot on the right. There is an apparent positive drift for approximately the first half of the chain followed by the drift in the negative direction. As a result, the cusum plot has a distinctive mountain-like shape and never crosses the  $x$  axis.



Cumsum plots can be also used for assessing how fast the chain is mixing. The slower the mixing of the chain, the smoother the cumsum plots. Conversely, the faster the mixing of the chain, the more jagged the cumsum plots. Below, we demonstrate the cumsum plots for the four variables considered previously. We can clearly see the contrast between the jagged lines of the fast mixing parameters `var1` and `var2` and the very smooth cumsum line of the poorly mixing parameter `var3`.



Besides graphical convergence diagnostics, there are some formal convergence tests (Geweke [1992]; Gelman and Rubin [1992]; Heidelberger and Welch [1983]; Raftery and Lewis [1992]; Zellner and Min [1995]). See *Convergence diagnostics using multiple chains* in [BAYES] `bayesmhm` and see [BAYES] `bayesstats grubin` for more details.

## Summary

Bayesian analysis is a statistical procedure that answers research questions by expressing uncertainty about unknown parameters using probabilities. Bayesian inference is based on the posterior distribution of model parameters conditional on the observed data. The posterior distribution is composed of a likelihood distribution of the data and the prior distribution of the model parameters. The likelihood model is specified in the same way it is specified with any standard likelihood-based analysis. The prior distribution is constructed based on the prior (before observing the data) scientific knowledge and results from previous studies. Sensitivity analysis is typically performed to evaluate the influence of different competing priors on the results.

Many posterior distributions do not have a closed form and must be simulated using MCMC methods such as MH methods or the Gibbs method or sometimes their combination. The convergence of MCMC must be verified before any inference can be made.

Marginal posterior distributions of the parameters are used for inference. These are summarized using point estimators such as posterior mean and median and interval estimators such as equal-tailed credible intervals and highest posterior density intervals. Credible intervals have an intuitive interpretation as fixed ranges to which a parameter is known to belong with a prespecified probability. Hypothesis testing provides a way to assign an actual probability to any hypothesis of interest. A number of criteria are available for comparing models of interest. Predictions and model checking are also available based on the posterior predictive distribution.

Bayesian analysis provides many advantages over the standard frequentist analysis, such as an ability to incorporate prior information in the analysis, higher robustness to sparse data, more-comprehensive inference based on the knowledge of the entire posterior distribution, and more intuitive and direct interpretations of results by using probability statements about parameters.

## Video examples

[Introduction to Bayesian statistics, part 1: The basic concepts](#)

[Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm](#)

**Thomas Bayes** (1701(?)–1761) was a Presbyterian minister with an interest in calculus, geometry, and probability theory. He was born in Hertfordshire, England. The son of a Nonconformist minister, Bayes was banned from English universities and so studied at Edinburgh University before becoming a clergyman himself. Only two works are attributed to Bayes during his lifetime, both published anonymously. He was admitted to the Royal Society in 1742 and never published thereafter.

The paper that gives us “Bayes’s Theorem” was published posthumously by Richard Price. The theorem has become an important concept for frequentist and Bayesian statisticians alike. However, the paper indicates that Bayes considered the theorem as relatively unimportant. His main interest appears to have been that probabilities were not fixed but instead followed some distribution. The notion, now foundational to Bayesian statistics, was largely ignored at the time.

Whether Bayes’s theorem is appropriately named is the subject of much debate. Price acknowledged that he had written the paper based on information he found in Bayes’s notebook, yet he never said how much he added beyond the introduction. Some scholars have also questioned whether Bayes’s notes represent original work or are the result of correspondence with other mathematicians of the time.

**Andrey Markov** (1856–1922) was a Russian mathematician who made many contributions to mathematics and statistics. He was born in Ryazan, Russia. In primary school, he was known as a poor student in all areas except mathematics. Markov attended St. Petersburg University, where he studied under Pafnuty Chebyshev and later joined the physicomathematical faculty. He was a member of the Russian Academy of the Sciences.

Markov's first interest was in calculus. He did not start his work in probability theory until 1883 when Chebyshev left the university and Markov took over his teaching duties. A large and influential body of work followed, including applications of the weak law of large numbers and what are now known as Markov processes and Markov chains. His work on processes and chains would later influence the development of a variety of disciplines such as biology, chemistry, economics, physics, and statistics.

Known in the Russian press as the “militant academician” for his frequent written protests about the czarist government's interference in academic affairs, Markov spent much of his adult life at odds with Russian authorities. In 1908, he resigned from his teaching position in response to a government requirement that professors report on students' efforts to organize protests in the wake of the student riots earlier that year. He did not resume his university teaching duties until 1917, after the Russian Revolution. His trouble with Russian authorities also extended to the Russian Orthodox Church. In 1912, he was excommunicated at his own request in protest over the Church's excommunication of Leo Tolstoy.

**Bruno de Finetti** (1906–1985) was born in Innsbruck, Austria. He received a degree in applied mathematics from the Polytechnic University of Milan. One of his first publications was in the field of genetics, in which he introduced what is now called the de Finetti diagram. Upon graduation, he began working for the Italian Central Statistical Institute and later moved to Trieste to work as an actuary. He became a professor at the University of Trieste in 1947 and later became a professor of the theory of probability at the University of Rome “La Sapienza”, a post he held for 15 years.

De Finetti made many contributions to the fields of probability and statistics. His text *Theory of Probability* helped lay the foundation for Bayesian theory. He also wrote papers on sequences of exchangeable random variables and processes with independent increments. In a paper published in 1955, de Finetti used an extension of the Lorenz–Gini concentration function to prove the Radon–Nikodym theorem. This extension has been employed in Bayesian statistics as a measure of robustness. His publications also include work on nonparametric estimation of a cumulative distribution function and group decision making, among other topics. For his many contributions, he was named a fellow of the Royal Statistical Society and the Institute of Mathematical Statistics.

## References

- Aitchison, J., and I. R. Dunsmore. 1975. *Statistical Prediction Analysis*. Cambridge: Cambridge University Press.
- Andrieu, C., and É. Moulines. 2006. On the ergodicity properties of some adaptive MCMC algorithms. *Annals of Applied Probability* 16: 1462–1505.
- Atchadé, Y. F., and J. S. Rosenthal. 2005. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli* 11: 815–828.
- Barlow, R. E., C. A. Clarotti, and F. Spizzichino, ed. 1993. *Reliability and Decision Making*. Cambridge: Chapman & Hall.
- Berger, J. O. 2000. Bayesian analysis: A look at today and thoughts of tomorrow. *Journal of the American Statistical Association* 95: 1269–1276.



- . 2006. “Bayes factors.” In *Encyclopedia of Statistical Sciences*, edited by Kotz, S., C. B. Read, N. Balakrishnan, and B. Vidakovic. Wiley. <http://onlinelibrary.wiley.com/doi/10.1002/0471667196.ess0985.pub2/abstract>.
- Berger, J. O., and L. R. Pericchi. 1996. The intrinsic Bayes factor for model selection and prediction. *Journal of the American Statistical Association* 91: 109–122.
- Berger, J. O., and R. L. Wolpert. 1988. *The Likelihood Principle: A Review, Generalizations, and Statistical Implications*. Hayward, CA: Institute of Mathematical Statistics.
- Berliner, L. M., J. A. Royle, C. K. Wikle, and R. F. Milliff. 1999. Bayesian methods in atmospheric sciences. In Vol. 6 of *Bayesian Statistics: Proceedings of the Sixth Valencia International Meeting*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 83–100. Oxford: Oxford University Press.
- Bernardo, J. M., and A. F. M. Smith. 2000. *Bayesian Theory*. Chichester, UK: Wiley.
- Berry, D. A., and D. K. Stangl, ed. 1996. *Bayesian Biostatistics*. New York: Dekker.
- Besag, J., and D. Higdon. 1999. Bayesian analysis for agricultural field experiments. *Journal of the Royal Statistical Society, Series B* 61: 691–746.
- Brooks, S. P., A. Gelman, G. L. Jones, and X.-L. Meng, ed. 2011. *Handbook of Markov Chain Monte Carlo*. Boca Raton, FL: Chapman & Hall/CRC.
- Carlin, B. P., J. B. Kadane, and A. E. Gelfand. 1998. Approaches for optimal sequential decision analysis in clinical trials. *Biometrics* 54: 964–975.
- Carlin, B. P., and T. A. Louis. 2009. *Bayesian Methods for Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Chaloner, K., and I. Verdinelli. 1995. Bayesian experimental design: A review. *Statistical Science* 10: 273–304.
- Chen, M.-H., and Q.-M. Shao. 1999. Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics* 8: 69–92.
- Chen, M.-H., Q.-M. Shao, and J. G. Ibrahim. 2000. *Monte Carlo Methods in Bayesian Computation*. New York: Springer.
- Chernozhukov, V., and H. Hong. 2003. An MCMC approach to classical estimation. *Journal of Econometrics* 115: 293–346.
- Chib, S., and E. Greenberg. 1995. Understanding the Metropolis–Hastings algorithm. *American Statistician* 49: 327–335.
- Cowles, M. K., and B. P. Carlin. 1996. Markov chain Monte Carlo convergence diagnostic: A comparative review. *Journal of the American Statistical Association* 91: 883–904.
- DeGroot, M. H., S. E. Fienberg, and J. B. Kadane. 1986. *Statistics and the Law*. New York: Wiley.
- Dey, D. D., P. Müller, and D. Sinha, ed. 1998. *Practical Nonparametric and Semiparametric Bayesian Statistics*. New York: Springer.
- Dey, D. K., S. K. Ghosh, and B. K. Mallick. 2000. *Generalized Linear Models: A Bayesian Perspective*. New York: Dekker.
- Eberly, L. E., and G. Casella. 2003. Estimating Bayesian credible intervals. *Journal of Statistical Planning and Inference* 112: 115–132.
- Gamerman, D., and H. F. Lopes. 2006. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Gelfand, A. E., S. E. Hills, A. Racine-Poon, and A. F. M. Smith. 1990. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association* 85: 972–985.
- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Gelman, A., W. R. Gilks, and G. O. Roberts. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability* 7: 110–120.
- Gelman, A., and D. B. Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical Science* 7: 457–472.
- Geman, S., and D. Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6: 721–741.

- Geweke, J. 1992. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In Vol. 4 of *Bayesian Statistics: Proceedings of the Fourth Valencia International Meeting, April 15–20, 1991*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 169–193. Oxford: Clarendon Press.
- . 1999. Using simulation methods for Bayesian econometric models: Inference, development, and communication. *Econometric Reviews* 18: 1–73.
- Giordani, P., and R. J. Kohn. 2010. Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics* 19: 243–259.
- Godsill, S. J., and P. J. W. Rayner. 1998. *Digital Audio Restoration*. Berlin: Springer.
- Gordon, N. J., D. J. Salmond, and A. F. M. Smith. 1993. novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing* 140: 107–113.
- Green, P. J. 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 82: 711–732.
- Greenland, S. 1998. Probability logic and probabilistic induction. *Epidemiology* 9: 322–332.
- Haario, H., E. Saksman, and J. Tamminen. 2001. An adaptive Metropolis algorithm. *Bernoulli* 7: 223–242.
- Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57: 97–109.
- Heidelberger, P., and P. D. Welch. 1983. Simulation run length control in the presence of an initial transient. *Operations Research* 31: 1109–1144.
- Hobert, J. P. 2000. Hierarchical models: A current computational perspective. *Journal of the American Statistical Association* 95: 1312–1316.
- Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.
- Iversen, E., Jr, G. Parmigiani, and D. A. Berry. 1999. Validating Bayesian Prediction Models: a Case Study in Genetic Susceptibility to Breast Cancer. In *Case Studies in Bayesian Statistics*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, vol. IV, 321–338. New York: Springer.
- Jeffreys, H. 1935. Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society* 31: 203–222.
- . 1961. *Theory of Probability*. 3rd ed. Oxford: Oxford University Press.
- Johnson, V. E. 1997. An alternative to traditional GPA for evaluating student performance. *Statistical Science* 12: 251–269.
- Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.
- Kim, S., N. Shephard, and S. Chib. 1998. Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Reviews of Economic Studies* 65: 361–393.
- Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21: 1087–1092.
- Metropolis, N., and S. Ulam. 1949. The Monte Carlo method. *Journal of the American Statistical Association* 44: 335–341.
- Müller, P., and B. Vidakovic, ed. 1999. *Bayesian Inference in Wavelet-Based Models*. New York: Springer.
- Neal, R. M. 1996. *Bayesian Learning for Neural Networks*. New York: Springer.
- . 1999. Regression and classification using gaussian process priors. In Vol. 6 of *Bayesian Statistics: Proceedings of the Sixth Valencia International Meeting*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 475–501. Oxford: Oxford University Press.
- Parent, E., P. Hubert, B. Bobee, and J. Miquel. 1998. *Statistical and Bayesian Methods in Hydrological Sciences*. Paris: UNESCO Press.
- Pearl, J. 1998. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann.
- Poirier, D. J. 1995. *Intermediate Statistics and Econometrics: A Comparative Approach*. Cambridge, MA: MIT Press.
- Pole, A., M. West, and J. Harrison. 1994. *Applied Bayesian Forecasting and Time Series Analysis*. Boca Raton, FL: Chapman and Hall.
- Pollard, W. E. 1986. *Bayesian Statistics for Evaluation Research: An Introduction*. Newbury Park, CA: SAGE.

- Propp, J. G., and D. B. Wilson. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms* 9: 223–252.
- Raftery, A. E., and S. M. Lewis. 1992. How many iterations in the Gibbs sampler? In Vol. 4 of *Bayesian Statistics: Proceedings of the Fourth Valencia International Meeting, April 15–20, 1991*, ed. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 763–773. Oxford: Clarendon Press.
- Rios Insua, D. 1990. *Sensitivity Analysis in Multi-Objective Decision Making*. New York: Springer.
- Robert, C. P., and G. Casella. 2004. *Monte Carlo Statistical Methods*. 2nd ed. New York: Springer.
- Roberts, G. O., and J. S. Rosenthal. 2007. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability* 44: 458–475.
- . 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* 18: 349–367.
- Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.
- Tanner, M. A. 1996. *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*. 3rd ed. New York: Springer.
- Tanner, M. A., and W. H. Wong. 1987. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association* 82: 528–550.
- Thompson, J. 2014. *Bayesian Analysis with Stata*. College Station, TX: Stata Press.
- Thompson, S. K. 2012. *Sampling*. 3rd ed. Hoboken, NJ: Wiley.
- Tierney, L. 1994. Markov chains for exploring posterior distributions. *Annals of Statistics* 22: 1701–1728.
- von Neumann, J. 1951. Various techniques used in connection with random digits. Monte Carlo methods. *Journal of Research of the National Bureau of Standards* 12: 36–38.
- West, M., and J. Harrison. 1997. *Bayesian Forecasting and Dynamic Models*. 2nd ed. New York: Springer.
- Wolpert, R. L., and K. Ickstadt. 1998. Poisson/gamma random field models for spatial statistics. *Biometrika* 85: 251–267.
- Yu, B., and P. Mykland. 1998. Looking at Markov samplers through cusum path plots: A simple diagnostic idea. *Statistics and Computing* 8: 275–286.
- Zellner, A. 1997. *Bayesian Analysis in Econometrics and Statistics: The Zellner View and Papers*. Northampton, MA: Edward Elgar.
- Zellner, A., and C.-K. Min. 1995. Gibbs sampler convergence criteria. *Journal of the American Statistical Association* 90: 921–927.

## Also see

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Glossary](#)

Description  
Also see

Remarks and examples

Acknowledgments

References

## Description

This entry describes commands to perform Bayesian analysis. Bayesian analysis is a statistical procedure that answers research questions by expressing uncertainty about unknown parameters using probabilities. It is based on the fundamental assumption that not only the outcome of interest but also all the unknown parameters in a statistical model are essentially random and are subject to prior beliefs.

### Estimation

<a href="#">Bayesian estimation</a>	Bayesian estimation commands
<a href="#">bayes</a>	Bayesian regression models using the <code>bayes</code> prefix
<a href="#">bayesmh</a>	Bayesian models using MH
<a href="#">bayesmh evaluators</a>	User-defined Bayesian models using MH

### Convergence tests and graphical summaries

<a href="#">bayesgraph</a>	Graphical summaries
<a href="#">bayesstats grubin</a>	Gelman–Rubin convergence diagnostics

### Postestimation statistics

<a href="#">bayesstats ess</a>	Effective sample sizes and related statistics
<a href="#">bayesstats summary</a>	Bayesian summary statistics
<a href="#">bayesstats ic</a>	Bayesian information criteria and Bayes factors

### Predictions

<a href="#">bayespredict</a>	Bayesian predictions
<a href="#">bayesstats ppvalues</a>	Bayesian predictive p-values

### Hypothesis testing

<a href="#">bayestest model</a>	Hypothesis testing using model posterior probabilities
<a href="#">bayestest interval</a>	Interval hypothesis testing

## Remarks and examples

This entry describes commands to perform Bayesian analysis. See [\[BAYES\] Intro](#) for an introduction to the topic of Bayesian analysis.

Bayesian estimation in Stata can be as easy as prefixing your estimation command with the `bayes` prefix ([\[BAYES\] bayes](#)). For example, if your estimation command is a linear regression of `y` on `x`

```
. regress y x
```

then Bayesian estimates for this model can be obtained by typing

```
. bayes: regress y x
```

See [\[BAYES\] Bayesian estimation](#) for a list of estimation commands that work with the `bayes` prefix.

In addition to the `bayes` prefix, there is a general-purpose Bayesian estimation command—the `bayesmh` command ([\[BAYES\] bayesmh](#)). `bayesmh` fits a variety of Bayesian models including multiple-equation linear and nonlinear models and, like the `bayes` prefix, estimates parameters using an adaptive MH Markov chain Monte Carlo (MCMC) method. You can choose from a variety of supported Bayesian models by specifying the `likelihood()` and `prior()` options. Or you can program your own Bayesian models by supplying a program evaluator for the posterior distributions of model parameters in the `evaluator()` option; see [\[BAYES\] bayesmh evaluators](#) for details.

After estimation, you can use `bayesgraph` to check convergence of MCMC visually. If you simulated multiple chains, you can use `bayesstats grubin` to compute Gelman–Rubin convergence diagnostics. You can also use `bayesstats ess` to compute effective sample sizes and related statistics for model parameters and functions of model parameters to assess the efficiency of the sampling algorithm and autocorrelation in the obtained MCMC sample. Once convergence is established, you can use `bayesstats summary` to obtain Bayesian summaries such as posterior means and standard deviations of model parameters and functions of model parameters and `bayesstats ic` to compute Bayesian information criteria and Bayes factors for models. You can use `bayestest model` to test hypotheses by comparing posterior probabilities of models. You can also use `bayestest interval` to test interval hypotheses about parameters and functions of parameters. You can use `bayespredict` and `bayesstats pvalues` for model diagnostics using posterior predictive checking. You can also use `bayespredict` to predict future observations.

Below we provide an overview example demonstrating the Bayesian suite of commands. In this entry, we mainly concentrate on the general command, `bayesmh`. For examples of using the simpler `bayes` prefix, see [example 11](#) and [Remarks and examples](#) in [\[BAYES\] bayes](#). Also, for more examples of `bayesmh`, see [Remarks and examples](#) in [\[BAYES\] bayesmh](#).

## Overview example

Consider an example from [Kuehl \(2000, 551\)](#) about the effects of exercise on oxygen uptake. The research objective is to compare the impact of the two exercise programs—12 weeks of step aerobic training and 12 weeks of outdoor running on flat terrain—on maximal oxygen uptake. Twelve healthy men were randomly assigned to one of the two groups, the “aerobic” group or the “running” group. Their changes in maximal ventilation (liters/minute) of oxygen for the 12-week period were recorded.

`oxygen.dta` contains 12 observations of changes in maximal ventilation of oxygen, recorded in variable `change`, from two groups, recorded in variable `group`. Additionally, ages of subjects are recorded in variable `age`, and an interaction between `age` and `group` is stored in variable `interaction`.

```
. use https://www.stata-press.com/data/r16/oxygen
(Oxygen Uptake Data)
. describe
Contains data from https://www.stata-press.com/data/r16/oxygen.dta
   obs:                12                Oxygen Uptake Data
   vars:                4                20 Jan 2018 15:56
                                   (_dta has notes)
```

variable name	storage type	display format	value label	variable label
change	float	%9.0g		Change in maximal oxygen uptake (liters/minute)
group	byte	%8.0g	grouplab	Exercise group (0: Running, 1: Aerobic)
age	byte	%8.0g		Age (years)
ageXgr	byte	%9.0g		Interaction between age and group

Sorted by:

Kuehl (2000) uses analysis of covariance to analyze these data. We use linear regression instead,

$$\text{change} = \beta_0 + \beta_{\text{group}}\text{group} + \beta_{\text{age}}\text{age} + \epsilon$$

where  $\epsilon$  is a random error with zero mean and variance  $\sigma^2$ . Also see Hoff (2009) for Bayesian analysis of these data.

Examples are presented under the following headings:

[Example 1: OLS](#)  
[Example 2: Bayesian normal linear regression with noninformative prior](#)  
[Example 3: Bayesian linear regression with informative prior](#)  
[Example 4: Bayesian normal linear regression with multivariate prior](#)  
[Example 5: Checking convergence](#)  
[Example 6: Postestimation summaries](#)  
[Example 7: Bayesian predictions](#)  
[Example 8: Model comparison](#)  
[Example 9: Hypothesis testing](#)  
[Example 10: Erasing simulation datasets](#)  
[Example 11: Bayesian linear regression using the bayes prefix](#)

## ► Example 1: OLS

Let's fit OLS regression to our data first.

```
. regress change group age
```

Source	SS	df	MS	Number of obs	=	12
Model	647.874893	2	323.937446	F(2, 9)	=	41.42
Residual	70.388768	9	7.82097423	Prob > F	=	0.0000
				R-squared	=	0.9020
				Adj R-squared	=	0.8802
Total	718.263661	11	65.2966964	Root MSE	=	2.7966

change	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
group	5.442621	1.796453	3.03	0.014	1.378763	9.506479
age	1.885892	.295335	6.39	0.000	1.217798	2.553986
_cons	-46.4565	6.936531	-6.70	0.000	-62.14803	-30.76498

From the table, both group and age are significant predictors of the outcome in this model.

For example, we reject the hypothesis of  $H_0: \beta_{\text{group}} = 0$  at a 5% level based on the  $p$ -value of 0.014. The actual interpretation of the reported  $p$ -value is that if we repeat the same experiment and use the same testing procedure many times, then given our null hypothesis of no effect of group, we will observe the result (test statistic) as extreme or more extreme than the one observed in this sample ( $t = 3.03$ ) only 1.4% of the times. The  $p$ -value cannot be interpreted as a probability of the null hypothesis, which is a common misinterpretation. In fact, it answers the question of how likely our data are, given that the null hypothesis is true, and not how likely the null hypothesis is, given our data. The latter question can be answered using Bayesian hypothesis testing, which we demonstrate in [example 9](#).

Confidence intervals are popular alternatives to  $p$ -values that eliminate some of the  $p$ -value shortcomings. For example, the 95% confidence interval for the coefficient for `group` is [1.38, 9.51] and does not contain the value of 0, so we consider `group` to be a significant predictor of `change`. The interpretation of a 95% confidence interval is that if we repeat the same experiment many times and compute confidence intervals for each experiment, then 95% of those intervals will contain the true value of the parameter. Thus we cannot conclude that the true coefficient for `group` lies between 1.38 and 9.51 with a probability of 0.95—a common misinterpretation of a confidence interval. This probability is either 0 or 1, and we do not know which for any particular confidence interval. All we know is that [1.38, 9.51] is a plausible range for the true value of the coefficient for `group`. Intervals that can actually be interpreted as probabilistic ranges for a parameter of interest may be constructed within the Bayesian paradigm; see [example 9](#).

◀

## ► Example 2: Bayesian normal linear regression with noninformative prior

In [example 1](#), we stated that frequentist methods cannot provide probabilistic summaries for the parameters of interest. This is because in frequentist statistics, parameters are viewed as unknown but fixed quantities. The only random quantity in a frequentist model is an outcome of interest. Bayesian statistics, on the other hand, in addition to the outcome of interest, also treats all model parameters as random quantities. This is what sets Bayesian statistics apart from frequentist statistics and enables one to make probability statements about the likely values of parameters and to assign probabilities to hypotheses of interest.

Bayesian statistics focuses on the estimation of various aspects of the posterior distribution of a parameter of interest, an initial or a prior distribution that has been updated with information about a parameter contained in the observed data. A posterior distribution is thus described by the prior distribution of a parameter and the likelihood function of the data given the parameter.

Let's now fit a Bayesian linear regression to `oxygen.dta`. To fit a Bayesian parametric model, we need to specify the likelihood function or the distribution of the data and prior distributions for all model parameters. Our Bayesian linear model has four parameters: three regression coefficients and the variance of the data. We assume a normal distribution for our outcome, `change`, and start with a noninformative Jeffreys prior for the parameters. Under the Jeffreys prior, the joint prior distribution of the coefficients and the variance is proportional to the inverse of the variance.

We can write our model as follows,

$$\begin{aligned}\text{change} &\sim N(X\beta, \sigma^2) \\ (\beta, \sigma^2) &\sim \frac{1}{\sigma^2}\end{aligned}$$

where  $X$  is our design matrix, and  $\beta = (\beta_0, \beta_{\text{group}}, \beta_{\text{age}})'$ , which is a vector of coefficients.

We use the `bayesmh` command to fit our Bayesian model. Let's consider the specification of the model first.

```
bayesmh change group age, likelihood(normal({var}))    ///
    prior({change:}, flat) prior({var}, jeffreys)
```

The specification of the regression function in `bayesmh` is the same as in any other Stata regression command—the name of the dependent variable follows the command, and the covariates of interest are specified next. Likelihood or outcome distribution is specified in the `likelihood()` option, and prior distributions are specified in the `prior()` options, which are repeated options.

All model parameters must be specified in curly braces, `{}`. `bayesmh` automatically creates parameters associated with the regression function—regression coefficients—but it is your responsibility to define the remaining model parameters. In our example, the only parameter we need to define is the variance parameter, which we define as `{var}`. The three regression coefficients `{change:group}`, `{change:age}`, and `{change:_cons}` are automatically created by `bayesmh`.

The last step is to specify the likelihood and the prior distributions. `bayesmh` provides several different built-in distributions for the likelihood and priors. If a certain distribution is not available or you have a particularly complicated Bayesian model, you may consider writing your own evaluator for the posterior distribution; see [\[BAYES\] bayesmh evaluators](#) for details. In our example, we specify distribution `normal({var})` in option `likelihood()` to request the likelihood function of the normal model with the variance parameter `{var}`. This specification together with the regression specification defines the likelihood model for our outcome `change`. We assign the `flat` prior, a prior with a density of 1, to all regression coefficients with `prior({change:}, flat)`, where `{change:}` is a shortcut for referring to all parameters with equation name `change`, our regression coefficients. Finally, we specify prior `jeffreys` for the variance parameter `{var}` to request the density  $1/\sigma^2$ .

Let's now run our command. `bayesmh` uses MCMC sampling, specifically, an adaptive random-walk MH MCMC method, to estimate marginal posterior distributions of parameters. Because `bayesmh` is using an MCMC method, which is stochastic, we must specify a random-number seed for reproducibility of our results. For consistency and simplicity, we use the same random seed of 14 in all of our examples throughout the manual.



```
. set seed 14

. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...

Model summary
-----
Likelihood:
  change ~ normal(xb_change,{var})

Priors:
  {change:group age _cons} ~ 1 (flat)
  {var} ~ jeffreys                                     (1)
-----

(1) Parameters are elements of the linear form xb_change.

Bayesian normal regression      MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =      2,500
                                         MCMC sample size =     10,000
                                         Number of obs   =       12
                                         Acceptance rate =     .1371
                                         Efficiency: min =     .02687
                                         avg             =     .03765
                                         max             =     .05724

Log marginal-likelihood = -24.703776
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	5.429677	2.007889	.083928	5.533821	1.157584	9.249262
age	1.8873	.3514983	.019534	1.887856	1.184714	2.567883
_cons	-46.49866	8.32077	.450432	-46.8483	-62.48236	-30.22105
var	10.27946	5.541467	.338079	9.023905	3.980325	25.43771

First, `bayesmh` provides a summary for the specified model. It is particularly useful for complicated models with many parameters and [hyperparameters](#). In fact, we recommend that you first specify the `dryrun` option, which provides only the summary of the model without estimation, to verify the specification of your model and then proceed with estimation. You can then use the `nomodelsummary` option during estimation to suppress the model summary, which may be rather long.

Next, `bayesmh` provides a header with various model summaries on the right-hand side. It reports the total number of MCMC iterations, 12,500, including the default 2,500 burn-in iterations, which are discarded from the analysis MCMC sample, and the number of iterations retained in the MCMC sample, or MCMC sample size, which is 10,000 by default. These default values should be viewed as initial estimates and further adjusted for the problem at hand to ensure convergence of the MCMC; see [example 5](#).

An acceptance rate and a summary of the parameter-specific [efficiencies](#) are also part of the output header. An acceptance rate specifies the proportion of proposed parameter values that was accepted by the algorithm. An acceptance rate of 0.14 in our example means that 14% out of 10,000 proposal parameter values were accepted by the algorithm. For the MH algorithm, this number rarely exceeds 50% and is typically below 30%. A low acceptance rate (for example, below 10%) may indicate convergence problems. In our example, the acceptance rate is a bit low, so we may need to investigate this further. In general, MH tends to have lower efficiencies compared with other MCMC methods. For example, efficiencies of 10% and higher are considered good. Efficiencies below 1% may be a source of concern. The efficiencies are somewhat low in our example, so we may consider tuning our MCMC sampler; see [Improving efficiency of the MH algorithm—blocking of parameters](#).

Finally, `bayesmh` reports a table with a summary of the results. The `Mean` column reports the estimates of posterior means, which are means of the marginal posterior distributions of the parameters. The posterior mean estimates are pretty close to the OLS estimates obtained in [example 1](#). This is expected, provided MCMC converged, because we used a noninformative prior. That is, we did not provide any additional information about parameters beyond that contained in the data.

The next column reports estimates of posterior standard deviations, which are standard deviations of the marginal posterior distribution. These values describe the variability in the posterior distribution of the parameter and are comparable to our OLS standard errors.

The precision of the posterior mean estimates is described by their Monte Carlo standard errors. These numbers should be small, relative to the scales of the parameters. Increasing the MCMC sample size should decrease these numbers.

The `Median` column provides estimates of the median of the posterior distribution and can be used to assess the symmetries of the posterior distribution. At a quick glance, the estimates of posterior means and medians are pretty close for the regression coefficients, so we suspect that their posterior distributions may be symmetric.

The last two columns provide credible intervals for the parameters. Unlike confidence intervals, as discussed in [example 1](#), these intervals have a straightforward probabilistic interpretation. For example, the probability that the coefficient for `group` is between 1.16 and 9.25 is about 0.95. The lower bound of the interval is greater than 0, so we conclude that there is an effect of the exercise program on the change in oxygen uptake. We can also use Bayesian hypothesis testing to test effects of parameters; see [example 9](#).

Before any interpretation of the results, however, it is important to verify the convergence of MCMC; see [example 5](#).

See [example 11](#) for how to fit Bayesian linear regression more easily using the `bayes` prefix.

◀

### ▷ Example 3: Bayesian linear regression with informative prior

In [example 2](#), we considered a noninformative prior for the model parameters. The strength (as well as the weakness) of Bayesian modeling is specifying an informative prior distribution, which may improve results. The strength is that if we have reliable prior knowledge about the distribution of a parameter, incorporating this in our model will improve results and potentially make certain analysis that would not be possible to perform in the frequentist domain feasible. The weakness is that a strong incorrect prior may lead to results that are not supported by the observed data. As with any modeling task, Bayesian or frequentist, a substantive research of the process generating the data and its parameters will be necessary for you to find appropriate models.

Let's consider an informative [conjugate prior](#) distribution for our normal regression model.

$$\begin{aligned}(\beta|\sigma^2) &\sim \text{i.i.d. } N(0, \sigma^2) \\ \sigma^2 &\sim \text{InvGamma}(2.5, 2.5)\end{aligned}$$

Here, for simplicity, all coefficients are assumed to be independently and identically distributed as normal with zero mean and variance  $\sigma^2$ , and the variance parameter is distributed according to the above inverse gamma distribution. In practice, a better prior would be to allow each parameter to have a different variance, at least for parameters with different scales.

Let's fit this model using `bayesmh`. Following the model above, we specify the `normal(0, {var})` prior for the coefficients and the `igamma(2.5, 2.5)` prior for the variance.

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, normal(0, {var}))
> prior({var}, igamma(2.5, 2.5))
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
change ~ normal(xb\_change,{var})

Priors:  
{change:group age \_cons} ~ normal(0,{var})  
{var} ~ igamma(2.5,2.5) (1)

(1) Parameters are elements of the linear form xb\_change.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.1984
	Efficiency: min =	.03732
	avg =	.04997
	max =	.06264

Log marginal-likelihood = -49.744054

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change	group	6.510807	2.812828	.129931	6.50829	.9605561 12.23164
	age	.2710499	.2167863	.009413	.2657002	-.1556194 .7173697
	_cons	-6.838302	4.780343	.191005	-6.683556	-16.53356 2.495631
	var	28.83438	10.53573	.545382	26.81462	14.75695 54.1965

The results from this model are substantially different from the results we obtained in [example 2](#). Considering that we used this simple prior for demonstration purposes only and did not use any external information about model parameters based on prior studies, we would be reluctant to trust the results from this model.



► Example 4: Bayesian normal linear regression with multivariate prior

Continuing with informative priors, we will consider Zellner’s *g*-prior ([Zellner 1986](#)), which is one of the more commonly used priors for the regression coefficients in a normal linear regression. [Hoff \(2009\)](#) provides more details about this example, and he includes the interaction between age and group as in [example 8](#). Here we concentrate on demonstrating how to fit our model using `bayesmh`.

The mathematical formulation of the priors is the following,

$$(\beta|\sigma^2) \sim \text{MVN}(0, g\sigma^2(X'X)^{-1})$$
$$\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)$$

where *g* reflects prior sample size,  $\nu_0$  is the prior degrees of freedom for the inverse gamma distribution, and  $\sigma_0^2$  is the prior variance for the inverse gamma distribution. This prior incorporates dependencies between coefficients. We use values of the parameters similar to those in [Hoff \(2009\)](#):  $g = 12$ ,  $\nu_0 = 1$ , and  $\sigma_0^2 = 8$ .

`bayesmh` provides the `zellnersg0()` prior to accommodate the above prior. The first argument is the dimension of the distribution, which is 3 in our example, the second argument is the prior degrees of freedom, which is 12 in our example, and the last argument is the variance parameter, which is `{var}` in our example. The mean is assumed to be a zero vector of the corresponding dimension. (You can use `zellnersg()` if you want to specify a nonzero mean vector; see [BAYES] `bayesmh`.)

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, zellnersg0(3,12,{var}))
> prior({var}, igamma(0.5, 4))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{var})
                                {var} ~ igamma(0.5,4)                                (1)
```

---

(1) Parameters are elements of the linear form `xb_change`.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.06169
	Efficiency: min =	.0165
	avg =	.02018
	max =	.02159

Log marginal-likelihood = -35.356501

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
change					
group	4.988881	2.260571	.153837	4.919351	.7793098 9.775568
age	1.713159	.3545698	.024216	1.695671	1.053206 2.458556
_cons	-42.31891	8.239571	.565879	-41.45385	-59.30145 -27.83421
var	12.29575	6.570879	.511475	10.3609	5.636195 30.93576

These results are more in agreement with results from [example 2](#) than with results of [example 3](#), but our acceptance rate and efficiencies are low and require further investigation.

◀

## □ Technical note

We can reproduce what `zellnersg0()` does above manually. First, we must compute  $(X'X)^{-1}$ . We can use Stata's matrix functions to do that.

```
. matrix accum xTx = group age
(obs=12)
. matrix S = syminv(xTx)
```

We now specify the desired multivariate normal prior for the coefficients, `mvnrmal0(3,12*{var}*S)`. The first argument of `mvnrmal0()` specifies the dimension of the distribution, and the second argument specifies the variance–covariance matrix. A mean of zero is assumed for all dimensions. One interesting feature of this specification is that the variance–covariance matrix is specified as a function of `{var}`.

```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, mvnnormal0(3,12*{var}*S))
> prior({var}, igamma(0.5, 4))
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
change ~ normal(xb\_change,{var})

Priors:  
{change:group age \_cons} ~ mvnnormal(3,0,0,0,12\*{var}\*S) (1)  
{var} ~ igamma(0.5,4)

(1) Parameters are elements of the linear form xb\_change.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.06169
	Efficiency: min =	.0165
	avg =	.02018
	max =	.02159

Log marginal-likelihood = -35.356501

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	4.988881	2.260571	.153837	4.919351	.7793098	9.775568
age	1.713159	.3545698	.024216	1.695671	1.053206	2.458556
_cons	-42.31891	8.239571	.565879	-41.45385	-59.30145	-27.83421
var	12.29575	6.570879	.511475	10.3609	5.636195	30.93576

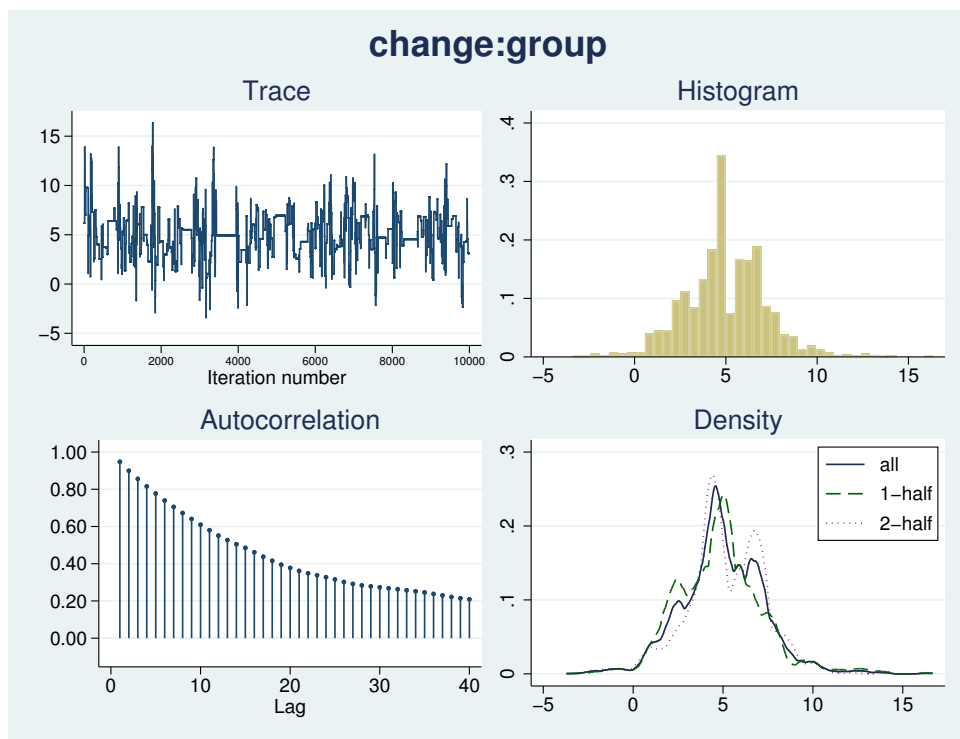


➤ Example 5: Checking convergence

We can use the `bayesgraph` command to visually check convergence of MCMC of parameter estimates. `bayesgraph` provides a variety of graphs. For several commonly used visual diagnostics displayed in a compact form, use `bayesgraph diagnostics`.

For example, we can look at graphical diagnostics for the coefficient for `group`.

```
. bayesgraph diagnostics {change:group}
```



The displayed diagnostics include a trace plot, an autocorrelation plot, a histogram, and a kernel density estimate overlaid with densities estimated using the first and the second halves of the MCMC sample. Both the trace plot and the autocorrelation plot demonstrate high autocorrelation. The shape of the histogram is not unimodal. We definitely have some convergence issues in this example.

Similarly, we can look at diagnostics for other model parameters. To see all graphs at once, type

```
bayesgraph diagnostics _all
```

Other useful summaries are effective sample sizes and statistics related to them. These can be obtained by using the `bayesstats ess` command.

```
. bayesstats ess
Efficiency summaries      MCMC sample size =      10,000
                          Efficiency:  min =       .0165
                              avg =       .02018
                              max =       .02159
```

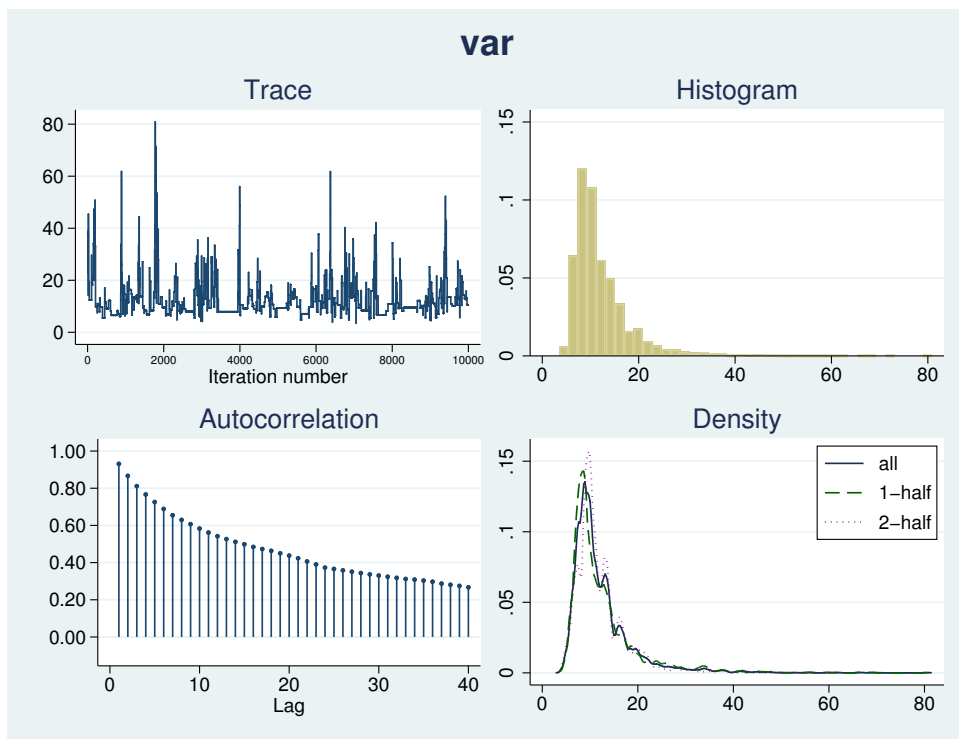
	ESS	Corr. time	Efficiency
change			
group	215.93	46.31	0.0216
age	214.39	46.64	0.0214
_cons	212.01	47.17	0.0212
var	165.04	60.59	0.0165

The closer ESS estimates are to the MCMC sample size, the less correlated the MCMC sample is, and the more precise our estimates of parameters are. Do not expect to see values close to the MCMC sample size with the MH algorithm, but values below 1% of the MCMC sample size are certainly red flags. In our example, ESS for {var} is somewhat low, so we may need to look into improving its sampling efficiency. For example, blocking on {var} should improve the efficiency for the variance; see [Improving efficiency of the MH algorithm—blocking of parameters](#). It is usually a good idea to sample regression coefficients and the variance in two separate blocks.

Correlation times may be viewed as estimates of autocorrelation lags in the MCMC samples. For example, correlation times of the coefficients range between 46 and 47, and the correlation time for the variance parameter is higher, 61. Consequently, the efficiency for the variance is lower than for the regression coefficients. More investigation of the MCMC for {var} is needed.

Indeed, the MCMC for the variance has very poor mixing and very high autocorrelation.

```
. bayesgraph diagnostics {var}
```



One remedy is to update the variance parameter separately from the regression coefficients by putting the variance parameter in a separate block; see [Improving efficiency of the MH algorithm—blocking of parameters](#) for details about this procedure. In `bayesmh`, this can be done by specifying the `block()` option.



```
. set seed 14
. bayesmh change group age, likelihood(normal({var}))
> prior({change:}, zellnersg(3,12,{var}))
> prior({var}, igamma(0.5, 4)) block({var})
> saving(agegroup_simdata)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{var})
                             {var} ~ igamma(0.5,4)                                     (1)
```

---

```
(1) Parameters are elements of the linear form xb_change.
Bayesian normal regression           MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =       2,500
                                           MCMC sample size =     10,000
                                           Number of obs   =        12
                                           Acceptance rate =      .3232
                                           Efficiency: min =    .06694
                                           avg =          .1056
                                           max =          .1443
Log marginal-likelihood = -35.460606
```

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	5.080653	2.110911	.080507	5.039834	.8564619	9.399672
age	1.748516	.3347172	.008875	1.753897	1.128348	2.400989
_cons	-43.12425	7.865979	.207051	-43.2883	-58.64107	-27.79122
var	12.09916	5.971454	.230798	10.67555	5.375774	27.32451

---

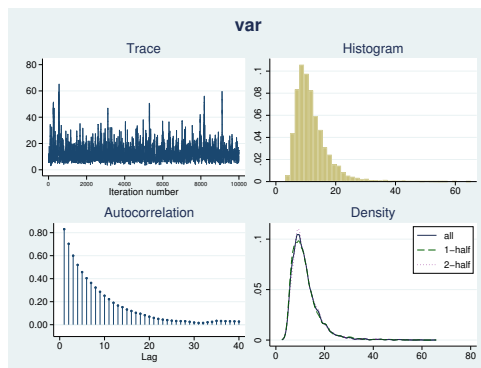
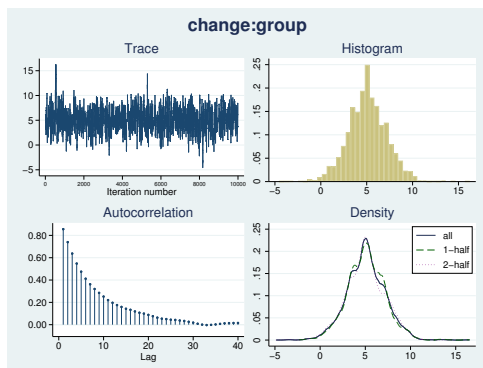
```
file agegroup_simdata.dta saved
. estimates store agegroup
```

Our acceptance rate and efficiencies are now higher.

In this example, we also used `estimates store agegroup` to store current estimation results as `agegroup` for future use. To use `estimates store` after `bayesmh`, we had to specify the `saving()` option with `bayesmh` to save the `bayesmh` simulation results to a permanent Stata dataset; see [Storing estimation results after Bayesian estimation](#).

The MCMC chains are now mixing much better. We may consider increasing the default MCMC sample size to achieve even lower autocorrelation.

```
. bayesgraph diagnostics {change:group} {var}
```



Multiple chains are often used to diagnose the convergence of MCMC; see [Convergence diagnostics using multiple chains](#) in [BAYES] [bayesmh](#) and [BAYES] [bayesstats grubin](#). Also see [Convergence of MCMC](#) in [BAYES] [bayesmh](#) for more information.

◀

## ► Example 6: Postestimation summaries

We can use the [bayesstats summary](#) command to compute postestimation summaries for model parameters and functions of model parameters. For example, we can compute an estimate of the standardized coefficient for `change`, which is  $\hat{\beta}_{\text{group}} \times \sigma_x / \sigma_y$ , where  $\sigma_x$  and  $\sigma_y$  are sample standard deviations of `group` and `change`, respectively.

We use [summarize](#) (see [R] [summarize](#)) to compute sample standard deviations and store them in respective scalars.

```
. summarize group
```

Variable	Obs	Mean	Std. Dev.	Min	Max
group	12	.5	.522233	0	1

```
. scalar sd_x = r(sd)
```

```
. summarize change
```

Variable	Obs	Mean	Std. Dev.	Min	Max
change	12	2.469167	8.080637	-10.74	17.05

```
. scalar sd_y = r(sd)
```

The standardized coefficient is an expression of the model parameter `{change:group}`, so we specify it in parentheses.

```
. bayesstats summary (group_std:{change:group}*sd_x/sd_y)
```

Posterior summary statistics MCMC sample size = 10,000

group\_std : {change:group}\*sd\_x/sd\_y

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
group_std	.3283509	.1364233	.005203	.3257128	.0553512 .6074792

The posterior mean estimate of the standardized group coefficient is 0.33 with a 95% credible interval of [0.055, 0.61].



► Example 7: Bayesian predictions

Bayesian predictions are useful for checking model fit and for predicting future observations. We can use the `bayespredict` command to generate replication samples for the outcome variable `change` and save them in a new dataset, `change_pred.dta`. Samples are drawn from the [posterior predictive distribution](#) of `change`. We specify `{_ysim}` with `bayespredict` to simulate the outcome values and use a random-number seed for reproducibility.

```
. bayespredict {_ysim}, saving(change_pred) rseed(16)
Computing predictions ...
file change_pred.dta saved
file change_pred.ster saved
```

`change_pred.dta` contains an MCMC sample of predicted values for each of the 12 observations. We can use `bayesstats` summary to calculate posterior summaries for the predicted observations by specifying using with the prediction dataset.

```
. bayesstats summary {_ysim} using change_pred
Posterior summary statistics                                MCMC sample size =      10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ysim1_1	-2.954378	3.763301	.060963	-2.930854	-10.39297	4.528522
_ysim1_2	-4.610688	3.771203	.059014	-4.660554	-11.9289	2.948378
_ysim1_3	-4.620784	3.758543	.057517	-4.645584	-12.03851	2.917013
_ysim1_4	.6417156	3.756645	.063162	.6019013	-6.83463	8.330498
_ysim1_5	4.069868	3.972042	.072874	4.065139	-3.780329	12.06363
_ysim1_6	-8.120147	3.832453	.061674	-8.096888	-15.54334	-.3579446
_ysim1_7	16.18539	4.076738	.072385	16.2033	8.105208	24.23569
_ysim1_8	2.156433	3.921	.072344	2.135557	-5.528265	10.00732
_ysim1_9	9.14268	3.780417	.071241	9.154486	1.571643	16.59816
_ysim1_10	10.91948	3.776916	.068083	10.92263	3.445305	18.59981
_ysim1_11	.3919052	3.969695	.079798	.344616	-7.389234	8.386358
_ysim1_12	3.902787	3.809399	.077872	3.884087	-3.530938	11.49579

The first column contains posterior means, MCMC estimates of the expected outcome observations with respect to the posterior predictive distribution. Both posterior means and medians can be used as Bayesian predictors.

One way to assess goodness of fit of the model is by comparing [replicated outcome](#) samples with the observed outcome sample. The discrepancy between these two can be measured using the so-called [posterior predictive \*p\*-values](#). We can use the `bayesstats ppvalues` command to compute these *p*-values. The posterior predictive *p*-values are typically computed for functions of the data or test statistics. Here, as a quick demonstration, we will compute them for each individual observation.

```
. bayesstats ppvalues {_ysim} using change_pred
Posterior predictive summary   MCMC sample size =    10,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
_ysim1_1	-2.954378	3.763301	-.87	.2786
_ysim1_2	-4.610688	3.771203	-10.74	.9512
_ysim1_3	-4.620784	3.758543	-3.27	.3479
_ysim1_4	.6417156	3.756645	-1.97	.773
_ysim1_5	4.069868	3.972042	7.5	.1819
_ysim1_6	-8.120147	3.832453	-7.25	.4034
_ysim1_7	16.18539	4.076738	17.05	.4124
_ysim1_8	2.156433	3.921	4.96	.2198
_ysim1_9	9.14268	3.780417	10.4	.3644
_ysim1_10	10.91948	3.776916	11.05	.4858
_ysim1_11	.3919052	3.969695	.26	.5106
_ysim1_12	3.902787	3.809399	2.51	.6498

Note: P(T>=T\_obs) close to 0 or 1 indicates lack of fit.

All estimated posterior predictive *p*-values are between 0.05 and 0.95 (except for `_ysim1_2`) and thus indicate adequate fit for the individual observations. However, more stringent [model checking](#) typically requires that various [test quantities](#) be computed using the entire replicated sample to inspect the distribution of replicated outcome values to assess the overall fit of the model. See [\[BAYES\] bayesstats ppvalues](#) for examples.

We can also use [bayespredict](#) to generate out-of-sample predictions. For illustration, let's add two new observations to the dataset: one for age 26 and group `Aerobic` (`group=1`) and another for age 26 and group `Running` (`group=0`).

```
. set obs 14
number of observations (_N) was 12, now 14

. replace group = 1 in 13
(1 real change made)

. replace group = 0 in 14
(1 real change made)

. replace age = 26 in 13/14
(2 real changes made)
```

We want to predict the outcome `change` for the new observations. Possible Bayesian predictors are the posterior means of the simulated outcome observations. These can be calculated using the `mean` option and saved in a new variable, say, `pname`.

```
. bayespredict pmean, mean rseed(16)
Computing predictions ...
. list change age group pmean
```

	change	age	group	pmean
1.	-.87	23	Running	-2.914124
2.	-10.74	22	Running	-4.613421
3.	-3.27	22	Running	-4.701283
4.	-1.97	25	Running	.545417
5.	7.5	27	Running	4.060798
6.	-7.25	20	Running	-8.111091
7.	17.05	31	Aerobic	16.15393
8.	4.96	23	Aerobic	2.183771
9.	10.4	27	Aerobic	9.155602
10.	11.05	28	Aerobic	10.87576
11.	.26	22	Aerobic	.4234267
12.	2.51	24	Aerobic	3.937901
13.	.	26	Aerobic	7.380203
14.	.	26	Running	2.405744

The predicted estimates for the out-of-sample observations 13 and 14 are 7.4 and 2.4 for the change in maximal oxygen uptake (liters/minute) for a 26-year old in the aerobic and running groups, respectively.

See [BAYES] [bayespredict](#) for more examples.

Finally, we drop the two new observations we added and erase the prediction dataset and the auxiliary estimation file created by `bayespredict`.

```
. drop in 13/14
(2 observations deleted)
. erase change_pred.dta
. erase change_pred.ster
```



➤ Example 8: Model comparison

As we can with frequentist analysis, we can use various information criteria to compare different models. There is great flexibility in which model can be compared: you can compare models with different distributions for the outcome, you can compare models with different priors, you can compare models with different forms for the regression function, and more. The only requirement is that the same data are used to fit the models. Comparisons using Bayes factors additionally require that parameters be sampled from the complete posterior distribution, which includes the normalizing constant.

Let's compare our reduced model with the full model including an interaction term. We again use a multivariate Zellner's *g*-prior for the coefficients and an inverse gamma prior for the variance. We use the same values as in [example 4](#) for prior parameters. (We use the interaction variable in this example for notational simplicity. We could have used the factor-variable notation `c.age#i.group` to include this interaction directly in our model; see [U] [11.4.3 Factor variables](#).)

```

. set seed 14
. bayesmh change group age ageXgr, likelihood(normal({var}))
> prior({change:}, zellnersg(4,12,{var}))
> prior({var}, igamma(0.5, 4)) block({var})
> saving(full_simdata)
Burn-in ...
Simulation ...
Model summary

```

---

```

Likelihood:
  change ~ normal(xb_change,{var})
Priors:
  {change:group age ageXgr _cons} ~ zellnersg(4,12,0,{var})
                                   {var} ~ igamma(0.5,4)

```

---

(1) Parameters are elements of the linear form xb\_change.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.3113
	Efficiency: min =	.0562
	avg =	.06425
	max =	.08478

Log marginal-likelihood = -36.738363

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	11.94079	16.74992	.706542	12.13983	-22.31056	45.11963
age	1.939266	.5802772	.023359	1.938756	.7998007	3.091072
ageXgr	-.2838718	.6985226	.028732	-.285647	-1.671354	1.159183
_cons	-47.57742	13.4779	.55275	-47.44761	-74.64672	-20.78989
var	11.72886	5.08428	.174612	10.68098	5.302265	24.89543

```

file full_simdata.dta saved
. estimates store full

```

We can use the `bayesstats ic` command to compare the models. We list the names of the corresponding estimation results following the command name.

```

. bayesstats ic full agegroup
Bayesian information criteria

```

	DIC	log(ML)	log(BF)
full	65.03326	-36.73836	.
agegroup	63.5884	-35.46061	1.277756

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

The smaller that DIC is and the larger that log(ML) is, the better. The model without interaction, `agegroup`, is preferred according to these statistics. The log Bayes-factor for the `agegroup` model relative to the full model is 1.28. Kass and Raftery (1995) provide a table of values for Bayes factors; see, for example, *Bayes factors* in [BAYES] `bayesstats ic`. According to their scale, because  $2 \times 1.28 = 2.56$  is greater than 2 (slightly), there is some mild evidence that model `agegroup` is better than model `full`.

➤ Example 9: Hypothesis testing

Continuing with [example 8](#), we can compute the actual probability associated with each of the models. We can use the `bayestest model` command to do this.

Similar to `bayesstats ic`, this command requires the names of estimation results corresponding to the models of interest.

```
. bayestest model full agegroup
Bayesian model tests
```

	log(ML)	P(M)	P(M y)
full	-36.7384	0.5000	0.2179
agegroup	-35.4606	0.5000	0.7821

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

Under the assumption that both models are equally probable a priori, the model without interaction, `agegroup`, has the probability of 0.78, whereas the `full` model has the probability of only 0.22. Despite the drastic disparity in the probabilities, according to the results from [example 8](#), model `agegroup` is only slightly preferable to model `full`. To have stronger evidence against `full`, we would expect to see higher probabilities (above 0.9) for `agegroup`.

We may be interested in testing an interval hypothesis about the parameter of interest. For example, for a model without interaction, let's compute the probability that the coefficient for `group` is between 4 and 8. We use `estimates restore` (see [\[R\] estimates store](#)) to load the results of the `agegroup` model back into memory.

```
. estimates restore agegroup
(results agegroup are active now)
. bayestest interval {change:group}, lower(4) upper(8)
Interval tests      MCMC sample size =      10,000
      prob1 : 4 < {change:group} < 8
```

	Mean	Std. Dev.	MCSE
prob1	.6159	0.48641	.0155788

The estimated probability or, technically, its posterior mean estimate is 0.62 with a standard deviation of 0.49 and Monte Carlo standard errors of 0.016.



➤ Example 10: Erasing simulation datasets

After you are done with your analysis, remember to erase any simulation datasets that you created using `bayesmh` and no longer need. If you want to save your estimation results to disk for future reference, use `estimates save`; see [\[R\] estimates save](#).

We are done with our analysis, and we do not need the datasets for future reference, so we remove both simulation files we created using `bayesmh`.

```
. erase agegroup_simdata.dta
. erase full_simdata.dta
```



### ► Example 11: Bayesian linear regression using the bayes prefix

Recall our OLS regression from [example 1](#). There is a more convenient way to obtain Bayesian estimates for this regression than using the `bayesmh` command as in previous examples. Because `regress` is one of the estimation commands that supports the `bayes` prefix ([\[BAYES\] Bayesian estimation](#)), we can simply type

```
. set seed 14
. bayes: regress change group age
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  change ~ regress(xb_change,{sigma2})

Priors:
  {change:group age _cons} ~ normal(0,10000)
  {sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form `xb_change`.

Bayesian linear regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.283
	Efficiency: min =	.02715
	avg =	.05779
	max =	.0692

Log marginal-likelihood = -45.562124

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	5.425311	2.111038	.080252	5.368975	1.104434	9.425197
age	1.885651	.3255098	.012472	1.887263	1.244666	2.517292
_cons	-46.47537	7.632058	.295505	-46.73244	-60.39245	-30.5054
sigma2	10.28431	7.614468	.462105	8.412747	3.595971	31.47161

Note: Default priors are used for model parameters.

With the `bayes` prefix command, the likelihood is determined automatically by the specified estimation command—`regress` in our example. The `bayes` prefix also provides the default prior specifications for model parameters, displaying this information as a note at the bottom of the output table; see [Default priors](#) in [\[BAYES\] bayes](#). Model summary provides details about the used default priors. For linear regression, the regression coefficients are assigned independent normal priors with zero mean and variance of 10,000, and the variance is assigned an inverse-gamma prior with the same shape and scale parameters of 0.01.

The default priors are provided for convenience and are chosen to be fairly uninformative for models with moderately scaled parameters. However, they are not guaranteed to be uninformative for all models and datasets; see [Linear regression: A case of informative default priors](#) in [\[BAYES\] bayes](#). You should choose priors carefully based on your research and model of interest.



As with bayesmh, the default MCMC method is an adaptive MH, but we can specify the gibbs option to request Gibbs sampling.

```
. set seed 14
. bayes, gibbs: regress change group age
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  change ~ normal(xb_change,{sigma2})
Priors:
  {change:group age _cons} ~ normal(0,10000)
                          {sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form xb\_change.

Bayesian linear regression	MCMC iterations =	12,500
Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	1
	Efficiency: min =	.556
	avg =	.889
	max =	1

Log marginal-likelihood = -45.83666

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	5.452439	2.062795	.020628	5.460372	1.360104	9.512987
age	1.875606	.330127	.003301	1.877129	1.228647	2.543129
_cons	-46.21334	7.746862	.077469	-46.18291	-61.82541	-31.09702
sigma2	9.929756	5.899176	.079113	8.426173	3.731261	24.76194

Note: Default priors are used for model parameters.

As expected, we obtain higher efficiency when using the Gibbs sampling. However, the gibbs option is available only with bayes: regress and bayes: mvreg and only for certain prior distributions.

We can easily change the default priors by specifying the `prior()` option, as with `bayesmh`. For example, we can reproduce `bayesmh`'s results from [example 4](#) but with the `bayes` prefix.

```
. set seed 14
. bayes, prior({change:}, zellnersg(3,12,{sigma2}))
> prior({sigma2}, igamma(0.5, 4)): regress change group age
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  change ~ regress(xb_change,{sigma2})
Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{sigma2})
                                {sigma2} ~ igamma(0.5,4)                                (1)
```

---

(1) Parameters are elements of the linear form `xb_change`.

Bayesian linear regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.2838
	Efficiency: min =	.06423
	avg =	.07951
	max =	.09277

Log marginal-likelihood = -35.448029

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	4.944955	2.184113	.086181	5.052278	.7065487	9.35098
age	1.747984	.3390581	.011132	1.747477	1.045677	2.416091
_cons	-43.09605	7.904334	.263186	-43.01961	-58.57942	-27.11278
sigma2	12.17932	5.87997	.220888	10.72651	5.511202	28.1211

The results are similar to those from [example 4](#) using `bayesmh` but not identical. By default, `bayes: regress` automatically splits the regression coefficients and the variance into two separate blocks, whereas `bayesmh` treats all parameters as one block; see [Improving efficiency of the MH algorithm—blocking of parameters](#) in [BAYES] `bayesmh` for details about blocking.

To match the results exactly, you can either specify the `block({var})` option with `bayesmh` in example 4 or specify the `noblocking` option to request no default blocking with the `bayes` prefix.

```
. set seed 14
. bayes, prior({change:}, zellnersg(3,12,{sigma2}))
> prior({sigma2}, igamma(0.5, 4)) noblocking: regress change group age
Burn-in ...
Simulation ...
Model summary

Likelihood:
  change ~ regress(xb_change,{sigma2})

Priors:
  {change:group age _cons} ~ zellnersg(3,12,0,{sigma2})
  {sigma2} ~ igamma(0.5,4)                                     (1)
```

(1) Parameters are elements of the linear form `xb_change`.

Bayesian linear regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	12
	Acceptance rate =	.06169
	Efficiency: min =	.0165
	avg =	.02018
	max =	.02159

Log marginal-likelihood = -35.356501

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
change						
group	4.988881	2.260571	.153837	4.919351	.7793098	9.775568
age	1.713159	.3545698	.024216	1.695671	1.053206	2.458556
_cons	-42.31891	8.239571	.565879	-41.45385	-59.30145	-27.83421
sigma2	12.29575	6.570879	.511475	10.3609	5.636195	30.93576

See [\[BAYES\]](#) `bayes` for more details.



## Acknowledgments

We thank John Thompson of the Department of Health Sciences at the University of Leicester, UK, and author of *Bayesian Analysis with Stata*, and Matthew J. Baker of Hunter College and the Graduate Center, CUNY for their software and contributions to Bayesian analysis in Stata.

## References

Baker, M. J. 2014. Adaptive Markov chain Monte Carlo sampling and estimation in Mata. *Stata Journal* 14: 623–661.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.

Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.

Kuehl, R. O. 2000. *Design of Experiments: Statistical Principles of Research Design and Analysis*. 2nd ed. Belmont, CA: Duxbury.

Zellner, A. 1986. On assessing prior distributions and Bayesian regression analysis with *g*-prior distributions. In Vol. 6 of *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti (Studies in Bayesian Econometrics and Statistics)*, ed. P. K. Goel and A. Zellner, 233–343. Amsterdam: North-Holland.

## Also see

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **Glossary**

Description

Bayesian estimation in Stata is similar to standard estimation—simply prefix the estimation commands with `bayes:` (see [\[BAYES\] bayes](#)). You can also refer to [\[BAYES\] bayesmh](#) and [\[BAYES\] bayesmlh evaluators](#) for fitting more general Bayesian models.

The following estimation commands support the `bayes` prefix.

Command	Entry	Description
Linear regression models		
<code>regress</code>	<a href="#">[BAYES] bayes: regress</a>	Linear regression
<code>hetregress</code>	<a href="#">[BAYES] bayes: hetregress</a>	Heteroskedastic linear regression
<code>tobit</code>	<a href="#">[BAYES] bayes: tobit</a>	Tobit regression
<code>intreg</code>	<a href="#">[BAYES] bayes: intreg</a>	Interval regression
<code>truncreg</code>	<a href="#">[BAYES] bayes: truncreg</a>	Truncated regression
<code>mvreg</code>	<a href="#">[BAYES] bayes: mvreg</a>	Multivariate regression
Binary-response regression models		
<code>logistic</code>	<a href="#">[BAYES] bayes: logistic</a>	Logistic regression, reporting odds ratios
<code>logit</code>	<a href="#">[BAYES] bayes: logit</a>	Logistic regression, reporting coefficients
<code>probit</code>	<a href="#">[BAYES] bayes: probit</a>	Probit regression
<code>cloglog</code>	<a href="#">[BAYES] bayes: cloglog</a>	Complementary log-log regression
<code>hetprobit</code>	<a href="#">[BAYES] bayes: hetprobit</a>	Heteroskedastic probit regression
<code>binreg</code>	<a href="#">[BAYES] bayes: binreg</a>	GLM for the binomial family
<code>biprobit</code>	<a href="#">[BAYES] bayes: biprobit</a>	Bivariate probit regression
Ordinal-response regression models		
<code>ologit</code>	<a href="#">[BAYES] bayes: ologit</a>	Ordered logistic regression
<code>oprobit</code>	<a href="#">[BAYES] bayes: oprobit</a>	Ordered probit regression
<code>hetoprobit</code>	<a href="#">[BAYES] bayes: hetoprobit</a>	Heteroskedastic ordered probit regression
<code>zioprobit</code>	<a href="#">[BAYES] bayes: zioprobit</a>	Zero-inflated ordered probit regression
Categorical-response regression models		
<code>mlogit</code>	<a href="#">[BAYES] bayes: mlogit</a>	Multinomial (polytomous) logistic regression
<code>mprobit</code>	<a href="#">[BAYES] bayes: mprobit</a>	Multinomial probit regression
<code>clogit</code>	<a href="#">[BAYES] bayes: clogit</a>	Conditional logistic regression
Count-response regression models		
<code>poisson</code>	<a href="#">[BAYES] bayes: poisson</a>	Poisson regression
<code>nbreg</code>	<a href="#">[BAYES] bayes: nbreg</a>	Negative binomial regression
<code>gnbreg</code>	<a href="#">[BAYES] bayes: gnbreg</a>	Generalized negative binomial regression
<code>tpoisson</code>	<a href="#">[BAYES] bayes: tpoisson</a>	Truncated Poisson regression
<code>tnbreg</code>	<a href="#">[BAYES] bayes: tnbreg</a>	Truncated negative binomial regression
<code>zip</code>	<a href="#">[BAYES] bayes: zip</a>	Zero-inflated Poisson regression
<code>zinb</code>	<a href="#">[BAYES] bayes: zinb</a>	Zero-inflated negative binomial regression

## Generalized linear models

`glm` [BAYES] **bayes: glm**

Generalized linear models

## Fractional-response regression models

`fracreg` [BAYES] **bayes: fracreg**

Fractional response regression

`betareg` [BAYES] **bayes: betareg**

Beta regression

## Survival regression models

`streg` [BAYES] **bayes: streg**

Parametric survival models

## Sample-selection regression models

`heckman` [BAYES] **bayes: heckman**

Heckman selection model

`heckprobit` [BAYES] **bayes: heckprobit**

Probit regression with sample selection

`heckoprobit` [BAYES] **bayes: heckoprobit**

Ordered probit model with sample selection

## Multilevel regression models

`mixed` [BAYES] **bayes: mixed**

Multilevel linear regression

`metobit` [BAYES] **bayes: metobit**

Multilevel tobit regression

`meintreg` [BAYES] **bayes: meintreg**

Multilevel interval regression

`melogit` [BAYES] **bayes: melogit**

Multilevel logistic regression

`meprobit` [BAYES] **bayes: meprobit**

Multilevel probit regression

`mecloglog` [BAYES] **bayes: mecloglog**

Multilevel complementary log-log regression

`meologit` [BAYES] **bayes: meologit**

Multilevel ordered logistic regression

`meoprobit` [BAYES] **bayes: meoprobit**

Multilevel ordered probit regression

`mepoisson` [BAYES] **bayes: mepoisson**

Multilevel Poisson regression

`menbreg` [BAYES] **bayes: menbreg**

Multilevel negative binomial regression

`meglm` [BAYES] **bayes: meglm**

Multilevel generalized linear model

`mestreg` [BAYES] **bayes: mestreg**

Multilevel parametric survival regression

## Video examples

[Introduction to Bayesian statistics, part 1: The basic concepts](#)[Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm](#)

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix[\[BAYES\] bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm[\[BAYES\] bayesmh evaluators](#) — User-defined evaluators with bayesmh[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix[\[BAYES\] Intro](#) — Introduction to Bayesian analysis[\[BAYES\] Glossary](#)

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">Reference</a>	<a href="#">Also see</a>		

## Description

The `bayes` prefix fits [Bayesian regression models](#). It provides Bayesian support for many likelihood-based estimation commands. The `bayes` prefix uses default or user-supplied priors for model parameters and estimates parameters using MCMC by drawing simulation samples from the corresponding posterior model. Also see [\[BAYES\] bayesmh](#) and [\[BAYES\] bayesmh evaluators](#) for fitting more general Bayesian models.

## Quick start

Bayesian linear regression of `y` on `x`, using default normal priors for the regression coefficients and an inverse-gamma prior for the variance

```
bayes: regress y x
```

As above, but use a standard deviation of 10 instead of 100 for the default normal priors and shape of 2 and scale of 1 instead of values of 0.01 for the default inverse-gamma prior

```
bayes, normalprior(10) igammaprior(2 1): regress y x
```

As above, but simulate four chains

```
bayes, normalprior(10) igammaprior(2 1) nchains(4): regress y x
```

Bayesian logistic regression of `y` on `x1` and `x2`, showing model summary without performing estimation

```
bayes, dryrun: logit y x1 x2
```

As above, but estimate model parameters and use uniform priors for all regression coefficients

```
bayes, prior({y: x1 x2 _cons}, uniform(-10,10)): logit y x1 x2
```

As above, but use a shortcut notation to refer to all regression coefficients

```
bayes, prior({y:}, uniform(-10,10)): logit y x1 x2
```

As above, but report odds ratios and use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///
      prior({y:_cons}, normal(0,10)) or: logit y x1 x2
```

Report odds ratios for the logit model on replay

```
bayes, or
```

Bayesian ordered logit regression of `y` on `x1` and `x2`, saving simulation results to `simdata.dta` and using a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ologit y x1 x2 x3
```

Bayesian multinomial regression of  $y$  on  $x_1$  and  $x_2$ , specifying 20,000 MCMC samples, setting length of the burn-in period to 5,000, and requesting that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): mlogit y x1 x2
```

Bayesian Poisson regression of  $y$  on  $x_1$  and  $x_2$ , putting regression slopes in separate blocks and showing block summary

```
bayes, block({y:x1}) block({y:x2}) blocksummary: poisson y x1 x2
```

Bayesian multivariate regression of  $y_1$  and  $y_2$  on  $x_1$ ,  $x_2$ , and  $x_3$ , using Gibbs sampling and requesting 90% HPD credible interval instead of the default 95% equal-tailed credible interval

```
bayes, gibbs clevel(90) hpd: mvreg y1 y2 = x1 x2 x3
```

As above, but use `mvreg`'s option `level()` instead of `bayes`'s option `clevel()`

```
bayes, gibbs hpd: mvreg y1 y2 = x1 x2 x3, level(90)
```

Suppress estimates of the covariance matrix from the output

```
bayes, noshow(Sigma, matrix)
```

Bayesian Weibull regression of `stset` survival-time outcome on  $x_1$  and  $x_2$ , specifying starting values of 1 for  $\{y:x_1\}$  and of 2 for  $\{y:x_2\}$

```
bayes, initial({y:x1} 1 {y:x2} 2): streg x1 x2, distribution(weibull)
```

Bayesian two-level linear regression of  $y$  on  $x_1$  and  $x_2$  with random intercepts by `id`

```
bayes: mixed y x1 x2 || id:
```

## Menu

Statistics > Bayesian analysis > Regression models > *estimation\_command*



## Syntax

```
bayes [ , bayesopts ] : estimation_command [ , estopts ]
```

*estimation\_command* is a likelihood-based estimation command, and *estopts* are command-specific estimation options; see [BAYES] **Bayesian estimation** for a list of supported commands, and see the command-specific entries for the supported estimation options, *estopts*.

<i>bayesopts</i>	Description
Priors	
* <b>gibbs</b>	specify Gibbs sampling; available only with <b>regress</b> or <b>mvreg</b> for certain prior combinations
* <b><u>normalprior</u></b> (#)	specify standard deviation of default normal priors for regression coefficients and other real scalar parameters; default is <b>normalprior</b> (100)
* <b><u>igammaprior</u></b> (# #)	specify shape and scale of default inverse-gamma prior for variances; default is <b>igammaprior</b> (0.01 0.01)
* <b><u>iwishartprior</u></b> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
Simulation	
<b>nchains</b> (#)	number of chains; default is to simulate one chain
<b>mcmcsize</b> (#)	MCMC sample size; default is <b>mcmcsize</b> (10000)
<b>burnin</b> (#)	burn-in period; default is <b>burnin</b> (2500)
<b>thinning</b> (#)	thinning interval; default is <b>thinning</b> (1)
<b>rseed</b> (#)	random-number seed
<b>exclude</b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>restubs</b> ( <i>restub1 restub2</i> ...)	specify stubs for random-effects parameters for all levels; allowed only with multilevel models
Blocking	
* <b>blocksize</b> (#)	maximum block size; default is <b>blocksize</b> (50)
<b>block</b> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b>blocksummary</b>	display block summary
* <b>noblocking</b>	do not block parameters by default
Initialization	
<b>initial</b> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<b>init#</b> ( <i>initspec</i> )	specify initial values for #th chain; requires <b>nchains</b> ()
<b>initall</b> ( <i>initspec</i> )	specify initial values for all chains; requires <b>nchains</b> ()
<b>nomleinitial</b>	suppress the use of maximum likelihood estimates as starting values
<b>initrandom</b>	specify random initial values
<b>initsummary</b>	display initial values used for simulation
* <b>noisily</b>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(adaptopts)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(cov)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform_option</code>	display coefficient table in exponentiated form
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <code>filename.dta</code>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary; allowed only with multilevel models
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is command-specific
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects([reref])</code>	specify that all or a subset of random-effects parameters be included in the output; allowed only with multilevel commands
<code>melabel</code>	display estimation table using the same row labels as <code>estimation_command</code> ; allowed only with multilevel commands
<code>nogroup</code>	suppress table summarizing groups; allowed only with multilevel models
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

The full specification of `iwishartprior()` is `iwishartprior(# [matname] [, relevel(levelvar)])`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

## Options

### Priors

**gibbs** specifies that Gibbs sampling be used to simulate model parameters instead of the default adaptive Metropolis–Hastings sampling. This option is allowed only with the **regress** and **mvreg** estimation commands. It is available only with certain prior combinations such as normal prior for regression coefficients and an inverse-gamma prior for the variance. Specifying the **gibbs** option is equivalent to specifying **block()**'s **gibbs** suboption for all default blocks of parameters. If you use the **block()** option to define your own blocks of parameters, the **gibbs** option will have no effect on those blocks, and an MH algorithm will be used to update parameters in those blocks unless you also specify **block()**'s **gibbs** suboption.

**normalprior(#)** specifies the standard deviation of the default normal priors. The default is **normalprior(100)**. The normal priors are used for scalar parameters defined on the whole real line; see [Default priors](#) for details.

**igammaprior(# #)** specifies the shape and scale parameters of the default inverse-gamma priors. The default is **igammaprior(0.01 0.01)**. The inverse-gamma priors are used for positive scalar parameters such as a variance; see [Default priors](#) for details. Instead of a number #, you can specify a missing value (.) to refer to the default value of 0.01.

**iwishartprior(# [matname] [, relevel(levelvar)])** specifies the degrees of freedom and, optionally, the scale matrix *matname* of the default inverse-Wishart priors used for unstructured covariances of random effects with multilevel models. The degrees of freedom # is a positive real scalar with the default value of  $d + 1$ , where  $d$  is the number of random-effects terms at the level of hierarchy *levelvar*. Instead of a number #, you can specify a missing value (.) to refer to the default value. Matrix name *matname* is the name of a positive-definite Stata matrix with the default of  $I(d)$ , the identity matrix of dimension  $d$ . If **relevel(levelvar)** is omitted, the specified parameters are used for inverse-Wishart priors for all levels with unstructured random-effects covariances. Otherwise, they are used only for the prior for the specified level *levelvar*. See [Default priors](#) for details.

**prior(priorspec)** specifies a prior distribution for model parameters. This option may be repeated. A prior may be specified for any of the model parameters, except the random-effects parameters in multilevel models. Model parameters with the same prior specifications are placed in a separate block. Model parameters that are not included in prior specifications are assigned default priors; see [Default priors](#) for details. Model parameters may be scalars or matrices, but both types may not be combined in one prior statement. If multiple scalar parameters are assigned a single univariate prior, they are considered independent, and the specified prior is used for each parameter. You may assign a multivariate prior of dimension  $d$  to  $d$  scalar parameters. Also see [Referring to model parameters](#) in [\[BAYES\] bayesmh](#).

All **prior()** distributions are allowed, but they are not guaranteed to correspond to proper posterior distributions for all likelihood models. You need to think carefully about the model you are building and evaluate its convergence thoroughly; see [Convergence of MCMC](#) in [\[BAYES\] bayesmh](#).

**dryrun** specifies to show the summary of the model that would be fit without actually fitting the model. This option is recommended for checking specifications of the model before fitting the model. The model summary reports the information about the likelihood model and about priors for all model parameters.

### Simulation

**nchains(#)** specifies the number of Markov chains to simulate. You must specify at least two chains. By default, only one chain is produced. Simulating multiple chains is useful for convergence

diagnostics and to improve precision of parameter estimates. Four chains are often recommended in the literature, but you can specify more or less depending on your objective. The reported estimation results are based on all chains. You can use `bayesstats summary` with option `sepchains` to see the results for each chain. The reported acceptance rate, efficiencies, and log marginal-likelihood are averaged over all chains. You can use option `chainsdetail` to see these simulation summaries for each chain. Also see *Convergence diagnostics using multiple chains* in [BAYES] `bayesmh` and *Gelman–Rubin convergence diagnostic* in [BAYES] `bayesstats grubin`.

`mcmcsize(#)` specifies the target MCMC sample size. The default MCMC sample size is `mcmcsize(10000)`. The total number of iterations for the MH algorithm equals the sum of the burn-in iterations and the MCMC sample size in the absence of thinning. If thinning is present, the total number of MCMC iterations is computed as `burnin() + (mcmcsize() - 1) × thinning() + 1`. Computation time of the MH algorithm is proportional to the total number of iterations. The MCMC sample size determines the precision of posterior summaries, which may be different for different model parameters and will depend on the efficiency of the Markov chain. With multiple chains, `mcmcsize()` applies to each chain. Also see *Burn-in period and MCMC sample size* in [BAYES] `bayesmh`.

`burnin(#)` specifies the number of iterations for the burn-in period of MCMC. The values of parameters simulated during burn-in are used for adaptation purposes only and are not used for estimation. The default is `burnin(2500)`. Typically, burn-in is chosen to be as long as or longer than the adaptation period. The burn-in period may need to be larger for multilevel models because these models introduce high-dimensional random-effects parameters and thus require longer adaptation periods. With multiple chains, `burnin()` applies to each chain. Also see *Burn-in period and MCMC sample size* in [BAYES] `bayesmh` and *Convergence of MCMC* in [BAYES] `bayesmh`.

`thinning(#)` specifies the thinning interval. Only simulated values from every  $(1 + k \times \#)$ th iteration for  $k = 0, 1, 2, \dots$  are saved in the final MCMC sample; all other simulated values are discarded. The default is `thinning(1)`; that is, all simulation values are saved. Thinning greater than one is typically used for decreasing the autocorrelation of the simulated MCMC sample. With multiple chains, `thinning()` applies to each chain.

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. With one chain, `rseed(#)` is equivalent to typing `set seed #` prior to calling the bayes prefix; see [R] `set seed`. With multiple chains, you should use `rseed()` for reproducibility; see *Reproducing results* in [BAYES] `bayesmh`.

`exclude(paramref)` specifies which model parameters should be excluded from the final MCMC sample. These model parameters will not appear in the estimation table, and postestimation features for these parameters and log marginal-likelihood will not be available. This option is useful for suppressing nuisance model parameters. For example, if you have a factor predictor variable with many levels but you are only interested in the variability of the coefficients associated with its levels, not their actual values, then you may wish to exclude this factor variable from the simulation results. If you simply want to omit some model parameters from the output, see the `noshow()` option. `paramref` can include individual random-effects parameters.

`restubs(restub1 restub2 ...)` specifies the stubs for the names of random-effects parameters. You must specify stubs for all levels—one stub per level. This option overrides the default random-effects stubs. See *Likelihood model* for details about the default names of random-effects parameters.

#### Blocking

`blocksize(#)` specifies the maximum block size for the model parameters; default is `blocksize(50)`. This option does not apply to random-effects parameters. Each group of random-effects parameters is placed in one block, regardless of the number of random-effects parameters in that group.

`block(paramref[ , blockopts ])` specifies a group of model parameters for the blocked MH algorithm.

By default, model parameters, except the random-effects parameters, are sampled as independent blocks of 50 parameters or of the size specified in option `blocksize()`. Regression coefficients from different equations are placed in separate blocks. Auxiliary parameters such as variances and correlations are sampled as individual separate blocks, whereas the cutpoint parameters of the ordinal-outcome regressions are sampled as one separate block. With multilevel models, each group of random-effects parameters is placed in a separate block, and the `block()` option is not allowed with random-effects parameters. The `block()` option may be repeated to define multiple blocks. Different types of model parameters, such as scalars and matrices, may not be specified in one `block()`. Parameters within one block are updated simultaneously, and each block of parameters is updated in the order it is specified; the first specified block is updated first, the second is updated second, and so on. See *Improving efficiency of the MH algorithm—blocking of parameters* in [BAYES] **bayesmh**.

*blockopts* include `gibbs`, `split`, `scale()`, `covariance()`, and `adaptation()`.

`gibbs` specifies to use Gibbs sampling to update parameters in the block. This option is allowed only for hyperparameters and only for specific combinations of prior and hyperprior distributions; see *Gibbs sampling for some likelihood-prior and prior-hyperprior configurations* in [BAYES] **bayesmh**. For more information, see *Gibbs and hybrid MH sampling* in [BAYES] **bayesmh**. `gibbs` may not be combined with `scale()`, `covariance()`, or `adaptation()`.

`split` specifies that all parameters in a block are treated as separate blocks. This may be useful for levels of factor variables.

`scale(#)` specifies an initial multiplier for the scale factor corresponding to the specified block. The initial scale factor is computed as  $\#/\sqrt{n_p}$  for continuous parameters and as  $\#/n_p$  for discrete parameters, where  $n_p$  is the number of parameters in the block. The default is `scale(2.38)`. If specified, this option overrides the respective setting from the `scale()` option specified with the command. `scale()` may not be combined with `gibbs`.

`covariance(matname)` specifies a scale matrix *matname* to be used to compute an initial proposal covariance matrix corresponding to the specified block. The initial proposal covariance is computed as  $\rho \times \text{Sigma}$ , where  $\rho$  is a scale factor and  $\text{Sigma} = \text{matname}$ . By default, *Sigma* is the identity matrix. If specified, this option overrides the respective setting from the `covariance()` option specified with the command. `covariance()` may not be combined with `gibbs`.

`adaptation(tarate())` and `adaptation(tolerance())` specify block-specific TAR and acceptance tolerance. If specified, they override the respective settings from the `adaptation()` option specified with the command. `adaptation()` may not be combined with `gibbs`.

`blocksummary` displays the summary of the specified blocks. This option is useful when `block()` is specified.

`noblocking` requests that no default blocking is applied to model parameters. By default, model parameters are sampled as independent blocks of 50 parameters or of the size specified in option `blocksize()`. For multilevel models, this option has no effect on random-effects parameters; blocking is always applied to them.

#### Initialization

`initial(initspec)` specifies initial values for the model parameters to be used in the simulation. With multiple chains, this option is equivalent to specifying option `init1()`. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example,

to initialize a scalar parameter `alpha` to 0.5 and a 2x2 matrix `Sigma` to the identity matrix `I(2)`, you can type

```
bayes, initial({alpha} 0.5 {Sigma,m} I(2)) : ...
```

You can also specify a list of parameters using any of the specifications described in [Referring to model parameters](#) in [BAYES] `bayesmh`. For example, to initialize all regression coefficients from equations `y1` and `y2` to zero, you can type

```
bayes, initial({y1:} {y2:} 0) : ...
```

The general specification of *initspec* is

```
paramref initval [paramref initval [...]]
```

where *initval* is either a number, a Stata expression that evaluates to a number, or a Stata matrix for initialization of matrix parameters.

Curly braces may be omitted for scalar parameters but must be specified for matrix parameters. Initial values declared using this option override the default initial values or any initial values declared during parameter specification in the `likelihood()` option. See [Initial values](#) for details.

`init#(initspec)` specifies initial values for the model parameters for the *#*th chain. This option requires option `nchains()`. `init1()` overrides the default initial values for the first chain, `init2()` for the second chain, and so on. You specify initial values in `init#()` just like you do in option `initial()`. See [Initial values](#) for details.

`initall(initspec)` specifies initial values for the model parameters for all chains. This option requires option `nchains()`. You specify initial values in `initall()` just like you do in option `initial()`. You should avoid specifying fixed initial values in `initall()` because then all chains will use the same initial values. `initall()` is useful to specify random initial values when you define your own priors within `prior()`'s `density()` and `logdensity()` suboptions. See [Initial values](#) for details.

`nomleinitial` suppresses using maximum likelihood estimates (MLEs) as starting values for model parameters. With multiple chains, this option and discussion below apply only to the first chain. By default, when no initial values are specified, MLE values from *estimation-command* are used as initial values. For multilevel commands, MLE estimates are used only for regression coefficients. Random effects are assigned zero values, and random-effects variances and covariances are initialized with ones and zeros, respectively. If `nomleinitial` is specified and no initial values are provided, the command uses ones for positive scalar parameters, zeros for other scalar parameters, and identity matrices for matrix parameters. `nomleinitial` may be useful for providing an alternative starting state when checking convergence of MCMC. This option cannot be combined with `initrandom`.

`initrandom` specifies that the model parameters be initialized randomly. Random initial values are generated from the prior distributions of the model parameters. If you want to use fixed initial values for some of the parameters, you can specify them in the `initial()` option or during parameter declarations in the `likelihood()` option. Random initial values are not available for parameters with `flat`, `jeffreys`, `density()`, `logdensity()`, and `jeffreys()` priors; you must provide your own initial values for such parameters. This option cannot be combined with `nomleinitial`. See [Specifying initial values](#) in [BAYES] `bayesmh` for details.

`initsummary` specifies that the initial values used for simulation be displayed.

`noisily` specifies that the output from the estimation command be shown during initialization. The estimation command is executed once to set up the model and calculate initial values for model parameters.

**adaptation**(*adaptopts*) controls adaptation of the MCMC procedure. Adaptation takes place every prespecified number of MCMC iterations and consists of tuning the proposal scale factor and proposal covariance for each block of model parameters. Adaptation is used to improve sampling efficiency. Provided defaults are based on theoretical results and may not be sufficient for all applications. See *Adaptation of the MH algorithm* in [BAYES] **bayesmh** for details about adaptation and its parameters.

*adaptopts* are any of the following options:

**every**(#) specifies that adaptation be attempted every #th iteration. The default is **every**(100).

To determine the adaptation interval, you need to consider the maximum block size specified in your model. The update of a block with  $k$  model parameters requires the estimation of a  $k \times k$  covariance matrix. If the adaptation interval is not sufficient for estimating the  $k(k+1)/2$  elements of this matrix, the adaptation may be insufficient.

**maxiter**(#) specifies the maximum number of adaptive iterations. Adaptation includes tuning of the proposal covariance and of the scale factor for each block of model parameters. Once the TAR is achieved within the specified tolerance, the adaptation stops. However, no more than # adaptation steps will be performed. The default is variable and is computed as  $\max\{25, \text{floor}(\text{burnin}()/\text{adaptation}(\text{every}()))\}$ .

**maxiter**() is usually chosen to be no greater than  $(\text{mcmcsz}() + \text{burnin}())/\text{adaptation}(\text{every}())$ .

**miniter**(#) specifies the minimum number of adaptive iterations to be performed regardless of whether the TAR has been achieved. The default is **miniter**(5). If the specified **miniter**() is greater than **maxiter**(), then **miniter**() is reset to **maxiter**(). Thus, if you specify **maxiter**(0), then no adaptation will be performed.

**alpha**(#) specifies a parameter controlling the adaptation of the AR. **alpha**() should be in  $[0, 1]$ . The default is **alpha**(0.75).

**beta**(#) specifies a parameter controlling the adaptation of the proposal covariance matrix. **beta**() must be in  $[0, 1]$ . The closer **beta**() is to zero, the less adaptive the proposal covariance. When **beta**() is zero, the same proposal covariance will be used in all MCMC iterations. The default is **beta**(0.8).

**gamma**(#) specifies a parameter controlling the adaptation rate of the proposal covariance matrix. **gamma**() must be in  $[0, 1]$ . The larger the value of **gamma**(), the less adaptive the proposal covariance. The default is **gamma**(0).

**tarate**(#) specifies the TAR for all blocks of model parameters; this is rarely used. **tarate**() must be in  $(0, 1)$ . The default AR is 0.234 for blocks containing continuous multiple parameters, 0.44 for blocks with one continuous parameter, and  $1/n\_maxlev$  for blocks with discrete parameters, where  $n\_maxlev$  is the maximum number of levels for a discrete parameter in the block.

**tolerance**(#) specifies the tolerance criterion for adaptation based on the TAR. **tolerance**() should be in  $(0, 1)$ . Adaptation stops whenever the absolute difference between the current AR and TAR is less than **tolerance**(). The default is **tolerance**(0.01).

**scale**(#) specifies an initial multiplier for the scale factor for all blocks. The initial scale factor is computed as  $\#/\sqrt{n_p}$  for continuous parameters and  $\#/n_p$  for discrete parameters, where  $n_p$  is the number of parameters in the block. The default is **scale**(2.38).

**covariance**(*cov*) specifies a scale matrix *cov* to be used to compute an initial proposal covariance matrix. The initial proposal covariance is computed as  $\rho \times \Sigma$ , where  $\rho$  is a scale factor and



$\Sigma = \text{matname}$ . By default,  $\Sigma$  is the identity matrix. Partial specification of  $\Sigma$  is also allowed. The rows and columns of *cov* should be named after some or all model parameters. According to some theoretical results, the optimal proposal covariance is the posterior covariance matrix of model parameters, which is usually unknown. This option does not apply to the blocks containing random-effects parameters.

#### Reporting

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is `clevel(95)` or as set by [\[BAYES\] set clevel](#).

`hpd` displays the HPD credible intervals instead of the default equal-tailed credible intervals.

`eform_option` causes the coefficient table to be displayed in exponentiated form; see [\[R\] eform\\_option](#).

The estimation command determines which `eform_option` is allowed (`eform(string)` and `eform` are always allowed).

`remargl` specifies to compute the log marginal-likelihood for multilevel models. It is not reported by default for multilevel models. Bayesian multilevel models contain many parameters because, in addition to regression coefficients and variance components, they also estimate individual random effects. The computation of the log marginal-likelihood involves the inverse of the determinant of the sample covariance matrix of all parameters and loses its accuracy as the number of parameters grows. For high-dimensional models such as multilevel models, the computation of the log marginal-likelihood can be time consuming, and its accuracy may become unacceptably low. Because it is difficult to access the levels of accuracy of the computation for all multilevel models, the log marginal-likelihood is not reported by default. For multilevel models containing a small number of random effects, you can use the `remargl` option to compute and display the log marginal-likelihood.

`batch(#)` specifies the length of the block for calculating batch means and MCSE using batch means. The default is `batch(0)`, which means no batch calculations. When `batch()` is not specified, MCSE is computed using effective sample sizes instead of batch means. Option `batch()` may not be combined with `corrlag()` or `corrtol()`.

`saving(filename[, replace])` saves simulation results in *filename.dta*. The `replace` option specifies to overwrite *filename.dta* if it exists. If the `saving()` option is not specified, the `bayes` prefix saves simulation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time the `bayes` prefix is run and will also be erased if the current estimation results are cleared. `saving()` may be specified during estimation or on replay.

The saved dataset has the following structure. Variable `_chain` records chain identifiers. Variable `_index` records iteration numbers. The `bayes` prefix saves only states (sets of parameter values) that are different from one iteration to another and the frequency of each state in variable `_frequency`. (Some states may be repeated for discrete parameters.) As such, `_index` may not necessarily contain consecutive integers. Remember to use `_frequency` as a frequency weight if you need to obtain any summaries of this dataset. Values for each parameter are saved in a separate variable in the dataset. Variables containing values of parameters without equation names are named as `eq0_p#`, following the order in which parameters are declared in the `bayes` prefix. Variables containing values of parameters with equation names are named as `eq#_p#`, again following the order in which parameters are defined. Parameters with the same equation names will have the same variable prefix `eq#`. For example,

```
. bayes, saving(mcmc): ...
```



will create a dataset, `mcmc.dta`, with variable names `eq1_p1` for  $\{y:x1\}$ , `eq1_p2` for  $\{y:._cons\}$ , and `eq0_p1` for  $\{var\}$ . Also see macros `e(parnames)` and `e(varnames)` for the correspondence between parameter names and variable names.

In addition, the `bayes` prefix saves variable `_loglikelihood` to contain values of the log likelihood from each iteration and variable `_logposterior` to contain values of the log posterior from each iteration.

`nomodelsummary` suppresses the detailed summary of the specified model. The model summary is reported by default.

`nomesummary` suppresses the summary about the multilevel structure of the model. This summary is reported by default for multilevel commands.

`chainsdetail` specifies that acceptance rates, efficiencies, and log marginal-likelihoods be reported separately for each chain. By default, the header reports these statistics averaged over all chains. This option requires option `nchains()`.

`nodots`, `dots`, and `dots(#)` specify to suppress or display dots during simulation. With multiple chains, these options affect all chains. `dots(#)` displays a dot every `#` iterations. During the adaptation period, a symbol `a` is displayed instead of a dot. If `dots(..., every(#))` is specified, then an iteration number is displayed every `#`th iteration instead of a dot or `a`. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100, every(1000))`. `dots` is the default with multilevel commands, and `nodots` is the default with other commands.

`show(paramref)` or `noshow(paramref)` specifies a list of model parameters to be included in the output or excluded from the output, respectively. By default, all model parameters (except random-effects parameters with multilevel models) are displayed. Do not confuse `noshow()` with `exclude()`, which excludes the specified parameters from the MCMC sample. When the `noshow()` option is specified, for computational efficiency, MCMC summaries of the specified parameters are not computed or stored in `e()`. *paramref* can include individual random-effects parameters.

`showreffects` and `showreffects(reref)` are used with multilevel commands and specify that all or a list *reref* of random-effects parameters be included in the output in addition to other model parameters. By default, all random-effects parameters are excluded from the output as if you have specified the `noshow()` option. This option computes, displays, and stores in `e()` MCMC summaries for the random-effects parameters.

`melabel` specifies that the `bayes` prefix use the same row labels as *estimation\_command* in the estimation table. This option is allowed only with multilevel commands. It is useful to match the estimation table output of `bayes: mecnd` with that of *mecnd*. This option implies `nomesummary` and `nomodelsummary`.

`nogroup` suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header. This option is for use with multilevel commands.

`notable` suppresses the estimation table from the output. By default, a summary table is displayed containing all model parameters except those listed in the `exclude()` and `noshow()` options. Regression model parameters are grouped by equation names. The table includes six columns and reports the following statistics using the MCMC simulation results: posterior mean, posterior standard deviation, MCMC standard error or MCSE, posterior median, and credible intervals.

`noheader` suppresses the output header either at estimation or upon replay.

`title(string)` specifies an optional title for the command that is displayed above the table of the parameter estimates. The default title is specific to the specified likelihood model.

*display\_options*: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fwrapon(style)`, and `nolstretch`; see [R] [Estimation options](#).

#### Advanced

`search(search_options)` searches for feasible initial values. *search\_options* are `on`, `repeat(#)`, and `off`.

`search(on)` is equivalent to `search(repeat(500))`. This is the default.

`search(repeat(k))`,  $k > 0$ , specifies the number of random attempts to be made to find a feasible initial-value vector, or initial state. The default is `repeat(500)`. An initial-value vector is feasible if it corresponds to a state with positive posterior probability. If feasible initial values are not found after  $k$  attempts, an error will be issued. `repeat(0)` (rarely used) specifies that no random attempts be made to find a feasible starting point. In this case, if the specified initial vector does not correspond to a feasible state, an error will be issued.

`search(off)` prevents the command from searching for feasible initial values. We do not recommend specifying this option.

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is  $\min\{500, \text{mcmcsize}()/2\}$ . The total autocorrelation is computed as the sum of all lag- $k$  autocorrelation values for  $k$  from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`. Options `corrlag()` and `batch()` may not be combined.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag- $k$  autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the  $k$ th lag are discarded. Options `corrtol()` and `batch()` may not be combined.

## Remarks and examples

Remarks and examples are presented under the following headings:

- Using the bayes prefix*
  - Likelihood model*
  - Default priors*
  - Initial values*
  - Command-specific options*
- Introductory example*
- Linear regression: A case of informative default priors*
- Logistic regression with perfect predictors*
- Multinomial logistic regression*
- Generalized linear model*
- Truncated Poisson regression*
- Zero-inflated negative binomial model*
- Parametric survival model*
- Heckman selection model*
- Multilevel models*
  - Two-level models*
  - Crossed-effects model*
- Video examples*

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using adaptive MH and Gibbs algorithms, see [BAYES] [bayesmh](#). See [BAYES] [Bayesian estimation](#) for a list of supported estimation commands. For a quick overview example of all Bayesian commands, see [Overview example](#) in [BAYES] [Bayesian commands](#).

## Using the bayes prefix

The `bayes` prefix provides Bayesian estimation for many likelihood-based regression models. Simply prefix your estimation command with `bayes` to get Bayesian estimates—`bayes: estimation_command`; see [BAYES] [Bayesian estimation](#) for a list of supported commands. Also see [BAYES] [bayesmh](#) for other Bayesian models.

Similarly to the `bayesmh` command, the `bayes` prefix sets up a Bayesian posterior model, uses MCMC to simulate parameters of this model, and summarizes and reports results. The process of specifying a Bayesian model is similar to that described in [Setting up a posterior model](#) in [BAYES] [bayesmh](#), except the likelihood model is now determined by the specified *estimation\_command* and default priors are used for model parameters. The `bayes` prefix and the `bayesmh` command share the same methodology of MCMC simulation and the same summarization and reporting of simulation results; see [BAYES] [bayesmh](#) for details. In the following sections, we provide information specific to the `bayes` prefix.

### Likelihood model

With the `bayes` prefix, the likelihood component of the Bayesian model is determined by the prefixed estimation command, and all posterior model parameters are defined by the likelihood model. For example, the parameters of the model

```
. bayes: streg age smoking, distribution(lognormal)
```

are the regression coefficients and auxiliary parameters you see when you fit

```
. streg age smoking, distribution(lognormal)
```

All estimation commands have regression coefficients as their model parameters. Some commands have additional parameters such as variances and correlation coefficients.

The `bayes` prefix typically uses the likelihood parameterization and the naming convention of the estimation command to define model parameters, but there are exceptions. For example, the `truncreg` command uses the standard deviation parameter `{sigma}` to parameterize the likelihood, whereas `bayes: truncreg` uses the variance parameter `{sigma2}`.

Most model parameters are scalar parameters supported on the whole real line such as regression coefficients, log-transformed positive parameters, and atanh-transformed correlation coefficients. For example, positive scalar parameters are the variance parameters in `bayes: regress`, `bayes: tobit`, and `bayes: truncreg`, and matrix parameters are the covariance matrix `{Sigma, matrix}` in `bayes: mvreg` and covariances of random effects in multilevel commands such as `bayes: meglm`.

The names of model parameters are provided in the model summary displayed by the `bayes` prefix. Knowing these names is useful when specifying the prior distributions, although the `bayes` prefix does provide default priors; see [Default priors](#). You can use the `dryrun` option with the `bayes` prefix to see the names of model parameters prior to the estimation. In general, the names of regression coefficients are formed as `{depvar: indepvar}`, where *depvar* is the name of the specified dependent variable and *indepvar* is the name of an independent variable. There are exceptions such as `bayes: streg`, for which *depvar* is replaced with `_t`. Variance parameters are named `{sigma2}`, log-standard-deviation parameters are named `{lnsigma}`, atanh-transformed correlation parameters are named `{athrho}`, and the covariance matrix of `bayes: mvreg` is named `{Sigma, matrix}` (or `{Sigma, m}` for short).

For multilevel models such as `bayes: meglm`, in addition to regression coefficients and variance components, the `bayes` prefix also estimates [random-effects parameters](#). This is different from the corresponding frequentist commands, such as `meglm`, in which random effects are integrated out and thus are not among the final model parameters. (They can be predicted after estimation.) As such, the

bayes prefix has its own naming convention for model parameters of multilevel commands. Before moving on, you should be familiar with the syntax of the multilevel commands; see, for example, [Syntax](#) in [ME] [meglm](#).

The regression coefficients are labeled as usual,  $\{depvar:indepvar\}$ . Random-effects parameters are labeled as outlined in tables 1 and 2. You can change the default names by specifying the `restubs()` option. The common syntax of  $\{rename\}$  is  $\{restub\# \}$ , where *restub* is a capital letter, U for the level specified first, or a sequence of capital letters that is unique to each random-effects level, and # refers to the group of random effects at that level: 0 for random intercepts, 1 for random coefficients associated with the variable specified first in the random-effects equation, 2 for random coefficients associated with the variable specified second, and so on. The full syntax of  $\{rename\}$ ,  $\{fullrename\}$ , is  $\{restub\#[levelvar] \}$ , where *levelvar* is the variable identifying the level of hierarchy and is often omitted from the specification for brevity. Random effects at the observation level or crossed effects, specified as `_all: R.varname` with multilevel commands, are labeled as {U0}, {V0}, {W0}, and so on. Random effects at nesting levels, or nested effects, are labeled using a sequence of capital letters starting with the letter corresponding to the top level. For example, the multilevel model

```
. bayes: melogit y x1 x2 || id1: x1 x2 || id2: x1 || id3:
```

will have random-effects parameters {U0}, {U1}, and {U2} to represent, respectively, random intercepts, random coefficients for *x1*, and random coefficients for *x2* at the *id1* level; parameters {UU0} and {UU1} for random intercepts and random coefficients for *x1* at the *id2* level; and random intercepts {UUU0} at the *id3* level. See [Multilevel models](#) for more examples. Also see [Different ways of specifying model parameters](#) for how to refer to individual random effects during postestimation.

Table 1. Random effects at nesting levels of hierarchy (nested effects)

Hierarchy	Random effects	$\{rename\}$
<i>lev1</i>	Random intercepts	{U0}
	Random coefficients	{U1}, {U2}, etc.
<i>lev1&gt;lev2</i>	Random intercepts	{UU0}
	Random coefficients	{UU1}, {UU2}, etc.
<i>lev1&gt;lev2&gt;lev3</i>	Random intercepts	{UUU0}
	Random coefficients	{UUU1}, {UUU2}, etc.
...		

Table 2. Random effects at the observation level, `_all` (crossed effects)

Hierarchy	Random effects	$\{rename\}$
<i>lev1</i>	Random intercepts	{U0}
<i>lev2</i>	Random intercepts	{V0}
<i>lev3</i>	Random intercepts	{W0}
...		

Variance components for independent random effects are labeled as  $\{rename:sigma2\}$ . In the above example, there are six variance components: {U0:sigma2}, {U1:sigma2}, {U2:sigma2}, {UU0:sigma2}, {UU1:sigma2}, and {UUU0:sigma2}.

Covariance matrices of correlated random effects are labeled as `{restub:Sigma,matrix}` (or `{restub:Sigma,m}` for short), where *restub* is the letter stub corresponding to the level at which random effects are defined. For example, if we specify an unstructured covariance for the random effects at the `id1` and `id2` levels (with `cov(un)` short for `covariance(unstructured)`)

```
. bayes: melogit y x1 x2 || id1: x1 x2, cov(un) || id2: x1, cov(un) || id3:
```

we will have two covariance matrix parameters, a  $3 \times 3$  covariance `{U:Sigma,m}` at the `id1` level and a  $2 \times 2$  covariance `{UU:Sigma,m}` at the `id2` level, and the variance component `{UUU0:sigma2}` at the `id3` level.

For Gaussian multilevel models such as `bayes: mixed`, the error variance component is labeled as `{e.devar:sigma2}`.

Also see command-specific entries for the naming convention of additional parameters such as cutpoints with ordinal models or overdispersion parameters with negative binomial models.

## Default priors

For convenience, the `bayes` prefix provides default priors for model parameters. The priors are chosen to be general across models and are fairly uninformative for a typical combination of a likelihood model and dataset. However, the default priors may not always be appropriate. You should always inspect their soundness and, if needed, override the prior specification for some or all model parameters using the `prior()` option.

All scalar parameters supported on the whole real line, such as regression coefficients and log-transformed positive parameters, are assigned a normal distribution with zero mean and variance  $\sigma_{\text{prior}}^2$ ,  $N(0, \sigma_{\text{prior}}^2)$ , where  $\sigma_{\text{prior}}$  is given by the `normalprior()` option. The default value for  $\sigma_{\text{prior}}$  is 100, and thus the default priors for these parameters are  $N(0, 10000)$ . These priors are fairly uninformative for parameters of moderate size but may become informative for large-scale parameters. See the [Linear regression: A case of informative default priors](#) example below.

All positive scalar parameters, such as the variance parameters in `bayes: regress` and `bayes: tobit`, are assigned an inverse-gamma prior with shape parameter  $\alpha$  and scale parameter  $\beta$ ,  $\text{InvGamma}(\alpha, \beta)$ . The default values for  $\alpha$  and  $\beta$  are 0.01, and thus the default prior for these parameters is  $\text{InvGamma}(0.01, 0.01)$ .

All cutpoint parameters of ordinal-outcome models, such as `bayes: ologit` and `bayes: oprobit` are assigned flat priors, improper uniform priors with a constant density of 1, equivalent to specifying the `flat` prior option. The reason for this choice is that the cutpoint parameters are sensitive to the range of the outcome variables, which is usually unknown a priori.

For multilevel models with `independent` and `identity` random-effects covariance structures, variances of random effects are assigned inverse-gamma priors,  $\text{InvGamma}(0.01, 0.01)$ . For `unstructured` random-effects covariances, covariance matrix parameters are assigned fairly uninformative inverse-Wishart priors,  $\text{InvWishart}(d + 1, I(d))$ , where  $d$  is the dimension of the random-effects covariance matrix and  $I(d)$  is the identity matrix of dimension  $d$ . Setting the degrees-of-freedom parameter of the inverse-Wishart prior to  $d + 1$  is equivalent to specifying uniform on  $(-1, 1)$  distributions for the individual correlation parameters.

The model summary displayed by the `bayes` prefix describes the chosen default priors, which you can see prior to estimation if you specify `bayes's dryrun` option. You can use the `prior()` option repeatedly to override the default prior specifications for some or all model parameters.

## Initial values

By default, the `bayes` prefix uses the ML estimates from the prefixed estimation command as initial values for all scalar model parameters.

For example, the specification

```
. bayes: logit y x
```

will use the ML estimates from

```
. logit y x
```

as default initial values for the regression coefficients.

You can override the default initial values by using the `initial()` option; see [Specifying initial values](#) in [BAYES] `bayesmh`.

If the `nomleinitial` option is specified, instead of using the estimates from the prefixed command, all scalar model parameters are initialized with zeros, except for the variance parameters, which are initialized with ones.

The covariance matrix parameter {`Sigma`, `matrix`} of `bayes: mvreg` is always initialized with the identity matrix.

For multilevel models, regression coefficients are initialized using the ML estimates from the corresponding model without random effects, variances of random effects are initialized with ones, covariances of random effects are initialized with zeros, and random effects themselves are initialized with zeros.

With multiple chains, the following default initialization takes place. The first chain is initialized as described above. The subsequent chains use random initial values. In general, random initial values are generated from the prior distributions. For some improper priors such as `flat` and `jeffreys`, to avoid extremely large values, random initial values are sampled from a normal distribution with the mean centered at the initial values of the first chain and with standard deviations proportional to the magnitudes of the respective initial estimates.

See [Specifying initial values](#) in [BAYES] `bayesmh` for more information about default initial values and for how to specify your own.

## Command-specific options

Not all command-specific options, that is, options specified with the estimation command, are applicable within the Bayesian framework. One example is the group of maximum-likelihood optimization options such as `technique()` and `gradient`. For a list of supported options, refer to the entry specific to each command; see [BAYES] [Bayesian estimation](#) for a list of commands.

Some of the command-specific reporting options, such as [eform\\_option](#) and display options, can be specified either with *estimation\_command* or with the `bayes` prefix. For example, to obtain estimates of odds ratios instead of coefficients after the logit model, you can specify the `or` option with the command

```
. bayes: logit y x, or
```

or with the `bayes` prefix

```
. bayes, or: logit y x
```

You can also specify this option on replay with the `bayes` prefix

```
. bayes: logit y x
. bayes, or
```

Introductory example

We start with a simple linear regression model applied to `womenwage.dta`, which contains income data for a sample of working women.

```
. use https://www.stata-press.com/data/r16/womenwage
(Wages of women)
```

Suppose we want to regress women’s yearly income, represented by the `wage` variable, on their age, represented by the `age` variable. We can fit this model using the `regress` command.

```
. regress wage age
```

Source	SS	df	MS	Number of obs	=	488
Model	3939.49247	1	3939.49247	F(1, 486)	=	43.53
Residual	43984.4891	486	90.503064	Prob > F	=	0.0000
				R-squared	=	0.0822
				Adj R-squared	=	0.0803
Total	47923.9816	487	98.406533	Root MSE	=	9.5133

wage	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
age	.399348	.0605289	6.60	0.000	.2804173	.5182787
_cons	6.033077	1.791497	3.37	0.001	2.513041	9.553112

➤ Example 1: Bayesian simple linear regression

We can fit a corresponding Bayesian regression model by simply adding `bayes:` in front of the `regress` command. Because the `bayes` prefix is simulation based, we set a random-number seed to get reproducible results.

```
. set seed 15
. bayes: regress wage age
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  wage ~ regress(xb_wage,{sigma2})
Priors:
  {wage:age _cons} ~ normal(0,10000)
  {sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form `xb_wage`.

Bayesian linear regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	488
	Acceptance rate =	.3739
	Efficiency: min =	.1411
	avg =	.1766
	max =	.2271

Log marginal-likelihood = -1810.1432

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
wage						
age	.4008591	.0595579	.001586	.4005088	.2798807	.5183574
_cons	5.969069	1.737247	.043218	5.997571	2.60753	9.396475
sigma2	90.76252	5.891887	.123626	90.43802	79.71145	102.8558

Note: Default priors are used for model parameters.

The Bayesian model has two regression coefficient parameters, `{wage:age}` and `{wage:_cons}`, and a positive scalar parameter, `{sigma2}`, representing the variance of the error term. The model summary shows the default priors used for the model parameters: `normal(0, 10000)` for the regression coefficients and `igamma(0.01, 0.01)` for the variance parameter. The default priors are provided for convenience and should be used with caution. These priors are fairly uninformative in this example, but this may not always be the case; see the example in [Linear regression: A case of informative default priors](#).

The first two columns of the `bayes` prefix’s estimation table report the posterior means and standard deviations of the model parameters. We observe that for the regression coefficients `{wage:age}` and `{wage:_cons}`, the posterior means and standard deviations are very similar to the least-square estimates and their standard errors as reported by the `regress` command. The posterior mean estimate for `{sigma2}`, 90.76, is close to the residual mean squared estimate, 90.50, listed in the ANOVA table of the `regress` command. The estimation table of the `bayes` prefix also reports Monte Carlo standard errors (MCSEs), medians, and equal-tailed credible intervals.

The Bayesian estimates are stochastic in nature and, by default, are based on an MCMC sample of size 10,000. It is important to verify that the MCMC simulation has converged; otherwise, the Bayesian estimates cannot be trusted. The simulation efficiencies reported in the header of the estimation table can serve as useful initial indicators of convergence problems. The minimum efficiency in our example is about 0.14, and the average efficiency is about 0.17. These numbers are typical for the MH sampling algorithm used by `bayes` and do not indicate convergence problems; see [example 1](#) in [\[BAYES\] bayesstats grubin](#) for convergence diagnostics using multiple chains for this example. Also see [Convergence of MCMC](#) in [\[BAYES\] bayesmh](#) for details about convergence diagnostics.



➤ Example 2: Predictions

There are several postestimation commands available after the bayes prefix; see [\[BAYES\] Bayesian postestimation](#). Among them is the bayesstats summary command, which we can use to compute simple predictions. Suppose that we want to predict the expected wage of a 40-year-old woman conditional on the above fitted posterior model. Based on our model, this expected wage corresponds to the linear combination {wage : \_cons} + {wage : age} × 40. We name this expression wage40 and supply it to the bayesstats summary command.

```
. bayesstats summary (wage40: {wage:_cons} + {wage:age}*40)
Posterior summary statistics                                MCMC sample size =    10,000
      wage40 : {wage:_cons} + {wage:age}*40
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
wage40	22.00343	.81679	.024045	21.99231	20.39435	23.6718

The posterior mean estimate for the expected wage is about 22 with a 95% credible interval between 20.39 and 23.67.



➤ Example 3: Gibbs sampling

The bayes prefix uses adaptive MH as its default sampling algorithm. However, in the special case of linear regression, a more efficient Gibbs sampling is available. We can request Gibbs sampling by specifying the gibbs option.

```
. set seed 15
. bayes, gibbs: regress wage age
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  wage ~ normal(xb_wage,{sigma2})
Priors:
  {wage:age _cons} ~ normal(0,10000)
  {sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form xb\_wage.

Bayesian linear regression	MCMC iterations =	12,500
Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	488
	Acceptance rate =	1
	Efficiency =	1

Log marginal-likelihood = -1810.087

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
wage						
age	.3999669	.0611328	.000611	.4005838	.2787908	.518693
_cons	6.012074	1.804246	.018042	6.000808	2.488816	9.549921
sigma2	90.84221	5.939535	.059395	90.54834	79.8132	103.0164

---

Note: Default priors are used for model parameters.

The posterior summary results obtained by Gibbs sampling and MH sampling are very close except for the MCSEs. The Gibbs sampler reports substantially lower MCSEs than the default sampler because of its higher efficiency. In fact, in this example, the Gibbs sampler achieves the highest possible efficiency of 1.



Linear regression: A case of informative default priors

Our example in *Introductory example* used the default priors, which were fairly uninformative for those data and that model. This may not always be true. Consider a linear regression model using the familiar `auto.dta`. Let us regress the response variable `price` on the covariate `length` and factor variable `foreign`.

```
. use https://www.stata-press.com/data/r16/auto, clear
(1978 Automobile Data)
. regress price length i.foreign
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	16.35
Model	200288930	2	100144465	Prob > F	=	0.0000
Residual	434776467	71	6123612.21	R-squared	=	0.3154
				Adj R-squared	=	0.2961
Total	635065396	73	8699525.97	Root MSE	=	2474.6

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
length	90.21239	15.83368	5.70	0.000	58.64092	121.7839
foreign						
Foreign	2801.143	766.117	3.66	0.000	1273.549	4328.737
_cons	-11621.35	3124.436	-3.72	0.000	-17851.3	-5391.401

➤ Example 4: Default priors

We first fit a Bayesian regression model using the `bayes` prefix with default priors. Because the range of the outcome variable `price` is at least an order of magnitude larger than the range of the predictor variables `length` and `foreign`, we anticipate that some of the model parameters may have large scale, and longer adaptation may be necessary for the MCMC algorithm to reach optimal sampling for these parameters. We allow for longer adaptation by increasing the burn-in period from the default value of 2,500 to 5,000.

```
. set seed 15
. bayes, burnin(5000): regress price length i.foreign
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  price ~ regress(xb_price,{sigma2})

Priors:
  {price:length 1.foreign _cons} ~ normal(0,10000)
                                {sigma2} ~ igamma(.01,.01)
```

(1) Parameters are elements of the linear form `xb_price`.

Bayesian linear regression  
Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = -699.23257

MCMC iterations = 15,000  
Burn-in = 5,000  
MCMC sample size = 10,000  
Number of obs = 74  
Acceptance rate = .3272  
Efficiency: min = .05887  
              avg = .1093  
              max = .1958

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
price						
length	33.03301	1.80186	.060848	33.07952	29.36325	36.41022
foreign						
Foreign	32.77011	98.97104	4.07922	34.3237	-164.1978	222.0855
_cons	-8.063175	102.9479	3.34161	-9.110308	-205.9497	196.9341
sigma2	7538628	1297955	29334.9	7414320	5379756	1.04e+07

Note: Default priors are used for model parameters.

The posterior mean estimates of the regression coefficients are smaller (in absolute value) than the corresponding estimates from the `regress` command, because the default prior for the coefficients, `normal(0, 10000)`, is informative and has a strong shrinkage effect. For example, the least-square estimate of the constant term from `regress` is about  $-11,621$ , and its scale is much larger than the default prior standard deviation of 100. As a result, the default prior shrinks the estimate of the constant toward 0 and, specifically, to  $-8.06$ .

You should be aware that the default priors are provided for convenience and are not guaranteed to be uninformative in all cases. They are designed to have little effect on model parameters, the maximum likelihood estimates of which are of moderate size, say, less than 100 in absolute value. For large-scale parameters, as in this example, the default priors can become informative.



► Example 5: Flat priors

Continuing with [example 4](#), we can override the default priors using the `prior()` option. We can, for example, apply the completely uninformative flat prior, a prior with the density of 1, for the coefficient parameters.

```
. set seed 15
. bayes, prior({price:}, flat) burnin(5000): regress price length i.foreign
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  price ~ regress(xb_price,{sigma2})
Priors:
  {price:length 1.foreign _cons} ~ 1 (flat)
                                {sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form `xb_price`.

Bayesian linear regression	MCMC iterations	=	15,000
Random-walk Metropolis-Hastings sampling	Burn-in	=	5,000
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.3404
	Efficiency: min	=	.07704
	avg	=	.1086
	max	=	.1898
Log marginal-likelihood = -669.62603			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
price						
length	89.51576	16.27187	.586237	89.60969	57.96996	122.7961
foreign						
Foreign	2795.683	770.6359	26.0589	2787.139	1305.773	4298.785
_cons	-11478.83	3202.027	113.271	-11504.65	-17845.87	-5244.189
sigma2	6270294	1089331	25002.1	6147758	4504695	8803268

Note: Default priors are used for some model parameters.

The posterior mean estimates for the coefficient parameters are now close to the least-square estimates from `regress`. For example, the posterior mean estimate for `{price:_cons}` is about  $-11,479$ , whereas the least-square estimate is  $-11,621$ .

However, the flat priors should be used with caution. Flat priors are improper and may result in improper posterior distributions for which Bayesian inference cannot be carried out. You should thus choose the priors carefully, accounting for the properties of the likelihood model.



► Example 6: Zellner’s *g*-prior

A type of prior specific to the normal linear regression model is Zellner’s *g*-prior. We can apply it to our example using the `zellnersg0()` prior. For this prior, we need to specify the dimension of the prior, which is the number of regression coefficients (3), a degree of freedom (50) and the variance parameter of the error term in the regression model, `{sigma2}`; the mean parameter is assumed to be 0 by `zellnersg0()`. See [example 9](#) in [\[BAYES\] bayesmh](#) for more details about Zellner’s *g*-prior.

```
. set seed 15
. bayes, prior({price:}, zellnersg0(3, 50, {sigma2})) burnin(5000):
> regress price length i.foreign
Burn-in ...
Simulation ...
Model summary

Likelihood:
  price ~ regress(xb_price,{sigma2})

Priors:
  {price:length 1.foreign _cons} ~ zellnersg(3,50,0,{sigma2})
                                {sigma2} ~ igamma(.01,.01)                                     (1)
```

(1) Parameters are elements of the linear form `xb_price`.

Bayesian linear regression  
Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = -697.84862

MCMC iterations = 15,000  
Burn-in = 5,000  
MCMC sample size = 10,000  
Number of obs = 74  
Acceptance rate = .3019  
Efficiency: min = .06402  
              avg = .105  
              max = .1944

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
price						
length	87.53039	16.24762	.569888	87.72965	55.5177	119.9915
foreign						
Foreign	2759.267	794.043	31.3829	2793.241	1096.567	4202.283
_cons	-11223.95	3211.553	113.34	-11308.39	-17534.25	-4898.139
sigma2	6845242	1159035	26286.9	6716739	4978729	9521252

Note: Default priors are used for some model parameters.

We see that using this Zellner’s *g*-prior has little effect on the coefficient parameters, and the simulated posterior mean estimates are close to the least-square estimates from `regress`.



Logistic regression with perfect predictors

Let’s revisit the example in [Logistic regression model: A case of nonidentifiable parameters](#) of [BAYES] `bayesmh`. The example uses `heartswitz.dta` to model the binary outcome `disease`, the presence of a heart disease, using the predictor variables `restecg`, `isfbs`, `age`, and `male`. The dataset is a sample from Switzerland.

```
. use https://www.stata-press.com/data/r16/heartswitz, clear
(Subest of Switzerland heart disease data from UCI Machine Learning Repository)
```

➤ Example 7: Perfect prediction

The logistic regression model for these data is

```
. logit disease restecg isfbs age male
(output omitted)
```

To fit a Bayesian logistic regression, we prefix the `logit` command with `bayes`. We also specify the `noisily` option to show the estimation output of the `logit` command, which is run by the `bayes` prefix to set up the model and compute starting values for the parameters.

```

. set seed 15
. bayes, noisily: logit disease restecg isfbs age male
note: restecg != 0 predicts success perfectly
      restecg dropped and 17 obs not used
note: isfbs != 0 predicts success perfectly
      isfbs dropped and 3 obs not used
note: male != 1 predicts success perfectly
      male dropped and 2 obs not used
Iteration 0:   log likelihood = -4.2386144
Iteration 1:   log likelihood = -4.2358116
Iteration 2:   log likelihood = -4.2358076
Iteration 3:   log likelihood = -4.2358076
Logistic regression               Number of obs   =           26
                                LR chi2(1)        =            0.01
                                Prob > chi2        =           0.9403
                                Pseudo R2         =           0.0007
Log likelihood = -4.2358076

```

disease	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
restecg	0 (omitted)					
isfbs	0 (omitted)					
age	-.0097846	.1313502	-0.07	0.941	-.2672263	.2476572
male	0 (omitted)					
_cons	3.763893	7.423076	0.51	0.612	-10.78507	18.31285

Burn-in ...  
 Simulation ...  
 Model summary

```

Likelihood:
  disease ~ logit(xb_disease)
Prior:
  {disease:age _cons} ~ normal(0,10000)

```

```

(1) Parameters are elements of the linear form xb_disease.
Bayesian logistic regression           MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in       =       2,500
                                           MCMC sample size =    10,000
                                           Number of obs  =       26
                                           Acceptance rate =     .2337
                                           Efficiency:  min =     .1076
                                           avg         =     .1113
Log marginal-likelihood = -14.795726    max         =     .115

```

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	(omitted)					
isfbs	(omitted)					
age	-.0405907	.1650514	.004868	-.0328198	-.4005246	.2592641
male	(omitted)					
_cons	6.616447	9.516872	.290075	5.491008	-8.852858	28.99392

Note: Default priors are used for model parameters.

As evident from the output of the logit command, the covariates restecg, isfbs, and male are dropped because of perfect prediction. Although these predictors cannot be identified using the likelihood alone, they can be identified, potentially, in a posterior model with an informative prior. The default prior normal(0, 10000), used by the bayes prefix for the regression coefficients, is not

informative enough to resolve the perfect prediction, and we must override it with a more informative prior.

◀

## ► Example 8: Informative prior

In the example in *Logistic regression model: A case of nonidentifiable parameters* of [BAYES] `bayesmh`, we use information from another similar dataset, `hearthungary.dta`, to come up with informative priors for the regression coefficients. We use the same priors with the `bayes` prefix. We specify the `asis` option with the `logit` command to prevent dropping the perfect predictors from the model. We also specify the `nomleinitial` option to prevent the `bayes` prefix from trying to obtain ML estimates to use as starting values; reliable ML estimates cannot be provided by the `logit` command when the perfect predictors are retained.

```
. set seed 15
. bayes, prior({disease:restecg age}, normal(0,10))
> prior({disease:isfbs male}, normal(1,10))
> prior({disease:_cons}, normal(-4,10)) nomleinitial:
> logit disease restecg isfbs age male, asis
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  disease ~ logit(xb_disease)
Priors:
  {disease:restecg age} ~ normal(0,10)
  {disease:isfbs male} ~ normal(1,10)
  {disease:_cons} ~ normal(-4,10)
```

---

(1) Parameters are elements of the linear form `xb_disease`.

```
Bayesian logistic regression          MCMC iterations =    12,500
Random-walk Metropolis-Hastings sampling  Burn-in       =     2,500
                                          MCMC sample size =   10,000
                                          Number of obs   =     48
                                          Acceptance rate =    .2121
                                          Efficiency: min =    .01885
                                          avg           =    .04328
                                          max           =    .06184
```

Log marginal-likelihood = -11.006071

---

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	1.965122	2.315475	.115615	1.655961	-2.029873	6.789415
isfbs	1.708631	2.726071	.113734	1.607439	-3.306837	7.334592
age	.1258811	.0707431	.003621	.1245266	-.0016807	.2719748
male	.2671381	2.237349	.162967	.3318061	-4.106425	4.609955
_cons	-2.441911	2.750613	.110611	-2.538183	-7.596747	3.185172

For this posterior model with informative priors, we successfully estimate all regression parameters in the logistic regression model.

The informative prior in this example is based on information from an independent dataset, `hearthungary.dta`, which is a sample of observations on the same heart condition and predictor attributes as `heartswitz.dta` but sampled from Hungary's population. Borrowing information from independent datasets to construct informative priors is justified only when the datasets are compatible with the currently analyzed data.

◀



Multinomial logistic regression

Consider the health insurance dataset, `sysdsn1.dta`, to model the insurance outcome, `insure`, which takes the values `Indemnity`, `Prepaid`, and `Uninsure`, using the predictor variables `age`, `male`, `nonwhite`, and `site`. This model is considered in more detail in [example 4](#) in [\[R\] mlogit](#).

```
. use https://www.stata-press.com/data/r16/sysdsn1, clear
(Health insurance data)
```

First, we use the `mlogit` command to fit the model

```
. mlogit insure age male nonwhite i.site, nolog
Multinomial logistic regression      Number of obs      =          615
                                      LR chi2(10)           =          42.99
                                      Prob > chi2           =          0.0000
Log likelihood = -534.36165          Pseudo R2           =          0.0387
```

insure	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
Indemnity	(base outcome)					
Prepaid						
age	-.011745	.0061946	-1.90	0.058	-.0238862	.0003962
male	.5616934	.2027465	2.77	0.006	.1643175	.9590693
nonwhite	.9747768	.2363213	4.12	0.000	.5115955	1.437958
site						
2	.1130359	.2101903	0.54	0.591	-.2989296	.5250013
3	-.5879879	.2279351	-2.58	0.010	-1.034733	-.1412433
_cons	.2697127	.3284422	0.82	0.412	-.3740222	.9134476
Uninsure						
age	-.0077961	.0114418	-0.68	0.496	-.0302217	.0146294
male	.4518496	.3674867	1.23	0.219	-.268411	1.17211
nonwhite	.2170589	.4256361	0.51	0.610	-.6171725	1.05129
site						
2	-1.211563	.4705127	-2.57	0.010	-2.133751	-.2893747
3	-.2078123	.3662926	-0.57	0.570	-.9257327	.510108
_cons	-1.286943	.5923219	-2.17	0.030	-2.447872	-.1260134

Next, we use the `bayes` prefix to perform Bayesian estimation of the same multinomial logistic regression model.

```
. set seed 15
. bayes: mlogit insure age male nonwhite i.site
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  Prepaid Uninsure ~ mlogit(xb_Prepaid,xb_Uninsure)
Priors:
  {Prepaid:age male nonwhite i.site _cons} ~ normal(0,10000)      (1)
  {Uninsure:age male nonwhite i.site _cons} ~ normal(0,10000)    (2)
```

---

(1) Parameters are elements of the linear form `xb_Prepaid`.  
(2) Parameters are elements of the linear form `xb_Uninsure`.

Bayesian multinomial logistic regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
Base outcome: Indemnity	Number of obs =	615
	Acceptance rate =	.2442
	Efficiency: min =	.01992
	avg =	.03086
Log marginal-likelihood = -614.49286	max =	.05659

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
Prepaid						
age	-.0125521	.006247	.000396	-.0125871	-.024602	-.0005809
male	.5462718	.2086422	.012818	.5573004	.1263754	.9271802
nonwhite	.9796293	.2275709	.015746	.9737777	.53642	1.401076
site						
2	.098451	.214039	.012887	.0994476	-.3172914	.5260208
3	-.6043961	.2348319	.011596	-.6072807	-1.045069	-.1323191
_cons	.3183984	.3309283	.021325	.3219128	-.3423583	.956505
Uninsure						
age	-.008377	.0118479	.000581	-.0082922	-.0323571	.0140366
male	.4687524	.3537416	.02376	.4748359	-.2495656	1.147333
nonwhite	.1755361	.42708	.022566	.198253	-.7214481	.938098
site						
2	-1.298562	.4746333	.033628	-1.27997	-2.258622	-.4149035
3	-.2057122	.3533365	.020695	-.2009649	-.904768	.4924401
_cons	-1.305083	.5830491	.02451	-1.296332	-2.463954	-.1758435

Note: Default priors are used for model parameters.

For this model and these data, the default prior specification of the `bayes` prefix is fairly uninformative and, as a result, the posterior mean estimates for the parameters are close to the ML estimates obtained with `mlogit`.

We can report posterior summaries for the relative-risk ratios instead of the regression coefficients. This is equivalent to applying an exponential transformation,  $\exp(b)$ , to the simulated values of each of the regression coefficients,  $b$ , and then summarizing them. We can obtain relative-risk ratio summaries by replaying the `bayes` command with the `rrr` option specified. We use the already available simulation results from the last estimation and do not refit the model. We could have also specified the `rrr` option during the estimation.

```
. bayes, rrr
Model summary

Likelihood:
  Prepaid Uninsure ~ mlogit(xb_Prepaid,xb_Uninsure)

Priors:
  {Prepaid:age male nonwhite i.site _cons} ~ normal(0,10000)      (1)
  {Uninsure:age male nonwhite i.site _cons} ~ normal(0,10000)    (2)
```

(1) Parameters are elements of the linear form `xb_Prepaid`.  
(2) Parameters are elements of the linear form `xb_Uninsure`.

Bayesian multinomial logistic regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
Base outcome: Indemnity	Number of obs	=	615
	Acceptance rate	=	.2442
	Efficiency: min	=	.02149
	avg	=	.03181
	max	=	.06007
Log marginal-likelihood = -614.49286			

	RRR	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
Prepaid						
age	.9875456	.0061686	.000391	.9874918	.9756982	.9994192
male	1.764212	.3634348	.022268	1.745953	1.134708	2.527372
nonwhite	2.732931	.6240495	.042568	2.647929	1.709875	4.059566
site						
2	1.129077	.2450092	.015242	1.104561	.7281185	1.692189
3	.5617084	.1338774	.00665	.5448304	.3516675	.8760614
_cons	1.451983	.4904589	.029972	1.379764	.7100938	2.60259
Uninsure						
age	.9917276	.0117452	.000575	.991742	.9681608	1.014136
male	1.699605	.6045513	.040763	1.60775	.7791391	3.149782
nonwhite	1.301138	.5448086	.027742	1.219271	.4860479	2.555117
site						
2	.3045686	.1461615	.009698	.2780457	.1044944	.6604046
3	.8663719	.3155926	.01806	.8179411	.4046357	1.636304
_cons	.3203309	.1976203	.008063	.2735332	.0850978	.8387492

Note: \_cons estimates baseline relative risk for each outcome.  
Note: Default priors are used for model parameters.

Generalized linear model

Consider the insecticide experiment dataset, `beetle.dta`, to model the number of beetles killed, `r`, on the number of subjected beetles, `n`; the type of beetles, `beetle`; and the log-dose of insecticide, `ldose`. More details can be found in [example 2](#) of [\[R\] glm](#).

```
. use https://www.stata-press.com/data/r16/beetle, clear
```

Consider a generalized linear model with a binomial family and a complementary log-log link function for these data.

```
. glm r i.beetle ldose, family(binomial n) link(cloglog) nolog
Generalized linear models      Number of obs   =      24
Optimization      : ML        Residual df    =      20
                               Scale parameter =      1
Deviance           = 73.76505595      (1/df) Deviance = 3.688253
Pearson            = 71.8901173      (1/df) Pearson  = 3.594506
Variance function: V(u) = u*(1-u/n)      [Binomial]
Link function      : g(u) = ln(-ln(1-u/n)) [Complementary log-log]
                               AIC           = 6.74547
Log likelihood     = -76.94564525      BIC           = 10.20398
```

r	OIM		z	P> z	[95% Conf. Interval]	
	Coef.	Std. Err.				
beetle						
Red flour	-.0910396	.1076132	-0.85	0.398	-.3019576	.1198783
Mealworm	-1.836058	.1307125	-14.05	0.000	-2.09225	-1.579867
ldose	19.41558	.9954265	19.50	0.000	17.46458	21.36658
_cons	-34.84602	1.79333	-19.43	0.000	-38.36089	-31.33116

To fit a Bayesian generalized linear model with default priors, we type

```
. set seed 15
. bayes: glm r i.beetle ldose, family(binomial n) link(cloglog)
Burn-in ...
Simulation ...
Model summary

Likelihood:
  r ~ glm(xb_r)
Prior:
  {r:i.beetle ldose _cons} ~ normal(0,10000) (1)
```

(1) Parameters are elements of the linear form xb\_r.

```
Bayesian generalized linear models      MCMC iterations = 12,500
Random-walk Metropolis-Hastings sampling Burn-in          = 2,500
                                         MCMC sample size = 10,000
Family : binomial n                    Number of obs   = 24
Link    : complementary log-log        Scale parameter = 1
                                         Acceptance rate = .2003
                                         Efficiency: min = .03414
                                         avg          = .05094
                                         max          = .08012
Log marginal-likelihood = -102.9776
```

r	Mean	Std. Dev.	MCSE	Median	Equal-tailed	
					[95% Cred. Interval]	
beetle						
Red flour	-.0903569	.106067	.004527	-.093614	-.2964984	.112506
Mealworm	-1.843952	.130297	.004603	-1.848374	-2.091816	-1.594582
ldose	19.52814	.9997765	.054106	19.52709	17.6146	21.6217
_cons	-35.04832	1.800461	.096777	-35.0574	-38.81427	-31.61378

Note: Default priors are used for model parameters.

The posterior mean estimates of the regression parameters are not that different from the ML estimates obtained with `glm`.

If desired, we can request highest posterior density intervals be reported instead of default equal-tailed credible intervals by specifying the `hpd` option. We can also change the credible-interval level; for example, to request 90% credible intervals, we specify the `clevel(90)` option. We also could specify these options during estimation.

```
. bayes, clevel(90) hpd
Model summary

Likelihood:
  r ~ glm(xb_r)

Prior:
  {r:i.beetle ldose _cons} ~ normal(0,10000)                                     (1)
```

(1) Parameters are elements of the linear form `xb_r`.

Bayesian generalized linear models	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
Family : binomial n	Number of obs =	24
Link : complementary log-log	Scale parameter =	1
	Acceptance rate =	.2003
	Efficiency: min =	.03414
	avg =	.05094
	max =	.08012

Log marginal-likelihood = -102.9776

r	HPD					
	Mean	Std. Dev.	MCSE	Median	[90% Cred. Interval]	
beetle						
Red flour	-.0903569	.106067	.004527	-.093614	-.2444412	.1020305
Mealworm	-1.843952	.130297	.004603	-1.848374	-2.03979	-1.620806
ldose	19.52814	.9997765	.054106	19.52709	17.86148	21.16389
_cons	-35.04832	1.800461	.096777	-35.0574	-37.96057	-32.00411

Note: Default priors are used for model parameters.

Truncated Poisson regression

The semiconductor manufacturing dataset, `probe.dta`, contains observational data of failure rates, `failure`, of silicon wafers with width, `width`, and depth, `depth`, tested at four different probes, `probe`. A wafer is rejected if more than 10 failures are detected. See [example 2](#) in [\[R\] tpoisson](#).

```
. use https://www.stata-press.com/data/r16/probe, clear
```

We fit a truncated Poisson regression model with a truncation point of 10. We suppress the constant regression term from the likelihood equation using the `noconstant` option to retain all four probe levels by including `ibn.probe` in the list of covariates, which declares `probe` to be a factor variable with no base level.

```
. tpoisson failures ibn.probe depth width, noconstant ll(10) nolog
Truncated Poisson regression
Limits: lower =      10          Number of obs   =      88
        upper =    +inf          Wald chi2(6)    =    11340.50
Log likelihood = -239.35746      Prob > chi2    =      0.0000
```

failures	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
probe						
1	2.714025	.0752617	36.06	0.000	2.566515	2.861536
2	2.602722	.0692732	37.57	0.000	2.466949	2.738495
3	2.725459	.0721299	37.79	0.000	2.584087	2.866831
4	3.139437	.0377137	83.24	0.000	3.065519	3.213354
depth	-.0005034	.0033375	-0.15	0.880	-.0070447	.006038
width	.0330225	.015573	2.12	0.034	.0025001	.063545

## ► Example 9: Default priors

We first apply the bayes prefix with default priors to perform Bayesian estimation of the model. The estimation takes a little longer, so we specify the dots option to see the progress.

```
. set seed 15
. bayes, dots: tpoisson failures ibn.probe depth width, noconstant ll(10)
Burn-in 2500 aaaaaaaaaa1000.....2000..... done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

```
Likelihood:
failures ~ tpoisson(xb_failures)

Prior:
{failures:i.probe depth width} ~ normal(0,10000) (1)
```

(1) Parameters are elements of the linear form xb\_failures.

```
Bayesian truncated Poisson regression          MCMC iterations =    12,500
Random-walk Metropolis-Hastings sampling        Burn-in       =     2,500
                                                MCMC sample size =    10,000
Limits: lower =      10          Number of obs   =      88
        upper =    +inf          Acceptance rate =     .1383
                                                Efficiency: min =    .004447
                                                avg =    .01322
Log marginal-likelihood = -288.22663            max =    .04082
```

failures	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
probe						
1	2.689072	.0696122	.008596	2.688881	2.557394	2.833737
2	2.581567	.0644141	.00966	2.588534	2.436973	2.701187
3	2.712054	.0695932	.006415	2.717959	2.55837	2.844429
4	3.13308	.0397521	.004592	3.133433	3.055979	3.208954
depth	-.000404	.0033313	.000165	-.000504	-.0067928	.0061168
width	.036127	.0165308	.001821	.0360637	.001239	.067552

Note: Default priors are used for model parameters.

Note: There is a high autocorrelation after 500 lags.

With the default prior specification, the posterior mean estimates for the regression parameters are similar to the ML estimates obtained with the `tpoisson` command. However, the `bayes` prefix issues a high autocorrelation warning note and reports a minimum efficiency of only 0.004. The posterior model with default priors seems to be somewhat challenging for the MH sampler. We could allow for longer burn-in and increase the MCMC sample size to improve the MCMC convergence and increase the estimation precision. Instead, we will provide an alternative prior specification that will increase the model flexibility and improve its fit to the data.



➤ **Example 10: Hyperpriors**

We now assume that the four probe coefficients, `{failures:ibn.probe}`, have a normal prior distribution with mean parameter `{probe_mean}` and a variance of 10,000. It is reasonable to assume that all four probes have positive failure rates and that `{probe_mean}` is a positive hyperparameter. We decide to assign `{probe_mean}` a `gamma(2, 1)` hyperprior, which is a distribution with a positive domain and a mean of 2. We use this prior for the purpose of illustration; this prior is not informative for this model and these data. We initialize `{probe_mean}` with 1 to give it a starting value compatible with its hyperprior.

```
. set seed 15
. bayes, prior({failures:ibn.probe}, normal({probe_mean}, 10000))
> prior({probe_mean}, gamma(2, 1)) initial({probe_mean} 1) dots:
> tpoisson failures ibn.probe depth width, noconstant ll(10)
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done

Model summary
-----
Likelihood:
    failures ~ tpoisson(xb_failures)

Priors:
    {failures:i.probe} ~ normal({probe_mean},10000)           (1)
    {failures:depth width} ~ normal(0,10000)                 (1)

Hyperprior:
    {probe_mean} ~ gamma(2,1)
-----
(1) Parameters are elements of the linear form xb_failures.
```

Bayesian truncated Poisson regression

Random-walk Metropolis-Hastings sampling

Limits: lower = 10

upper = +inf

Log marginal-likelihood = -287.91504

MCMC iterations = 12,500

Burn-in = 2,500

MCMC sample size = 10,000

Number of obs = 88

Acceptance rate = .304

Efficiency: min = .04208

avg = .0775

max = .127

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
failures						
probe						
1	2.703599	.0770656	.003757	2.704613	2.551404	2.848774
2	2.592738	.0711972	.002796	2.594628	2.446274	2.728821
3	2.716223	.0755001	.003549	2.719622	2.568376	2.863064
4	3.137069	.0388127	.001317	3.136773	3.062074	3.211616
depth	-.000461	.0033562	.000109	-.0004457	-.0067607	.0062698
width	.0337508	.0152654	.000532	.0337798	.003008	.0622191
probe_mean	2.051072	1.462867	.041051	1.71286	.2211973	5.809428

Note: Default priors are used for some model parameters.

The MCMC simulation achieves an average efficiency of about 8% with no indication of convergence problems. Not only are the posterior mean estimates for the regression parameters similar to the ML estimates, but the MCMC standard errors are much lower than those achieved by the previous model with default priors. By introducing the hyperparameter {probe\_mean}, we have improved the goodness of fit of the model.



Zero-inflated negative binomial model

In this example, we consider a Bayesian model using zero-inflated negative binomial likelihood. We revisit [example 1](#) in [R] [zinb](#), which models the number of fish caught by visitors to a national park. The probability that a particular visitor fished is assumed to depend on the variables `child` and `camper`, which are supplied as covariates to the `inflate()` option of `zinb`.



```
. use https://www.stata-press.com/data/r16/fish, clear
. zinb count persons livebait, inflate(child camper) nolog
Zero-inflated negative binomial regression      Number of obs      =      250
                                                Nonzero obs        =      108
                                                Zero obs           =      142

Inflation model = logit                      LR chi2(2)          =      82.23
Log likelihood = -401.5478                  Prob > chi2         =      0.0000
```

count	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
count						
persons	.9742984	.1034938	9.41	0.000	.7714543	1.177142
livebait	1.557523	.4124424	3.78	0.000	.7491503	2.365895
_cons	-2.730064	.476953	-5.72	0.000	-3.664874	-1.795253
inflate						
child	3.185999	.7468551	4.27	0.000	1.72219	4.649808
camper	-2.020951	.872054	-2.32	0.020	-3.730146	-.3117567
_cons	-2.695385	.8929071	-3.02	0.003	-4.44545	-.9453189
/lnalpha	.5110429	.1816816	2.81	0.005	.1549535	.8671323
alpha	1.667029	.3028685			1.167604	2.380076

Let’s fit a Bayesian model with default normal prior distributions.

```
. set seed 15
. bayes, dots: zinb count persons livebait, inflate(child camper)
Burn-in 2500 aaaaaaaaa1000aaaaaaaa2000aaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

---

```
Likelihood:
  count ~ zinb(xb_count,xb_inflate,{lnalpha})

Priors:
  {count:persons livebait _cons} ~ normal(0,10000) (1)
  {inflate:child camper _cons} ~ normal(0,10000) (2)
  {lnalpha} ~ normal(0,10000)
```

---

(1) Parameters are elements of the linear form xb\_count.  
(2) Parameters are elements of the linear form xb\_inflate.

Bayesian zero-inflated negative binomial model

Random-walk Metropolis-Hastings sampling

Inflation model: logit

Log marginal-likelihood = -438.47876

MCMC iterations = 12,500

Burn-in = 2,500

MCMC sample size = 10,000

Number of obs = 250

Acceptance rate = .3084

Efficiency: min = .03716

avg = .0791

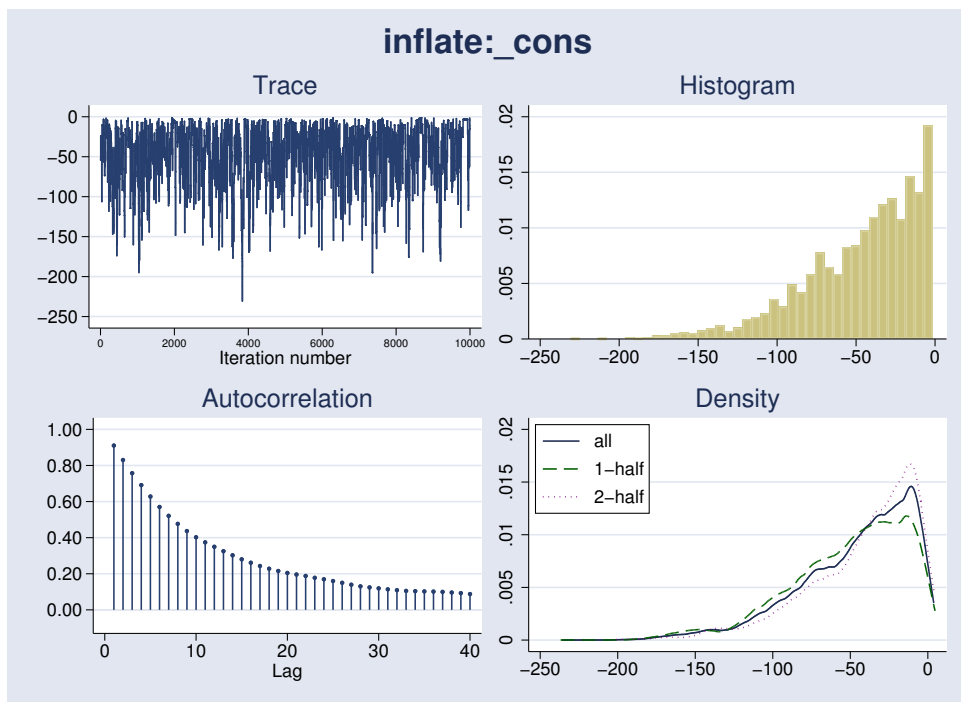
max = .1613

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
count						
persons	.9851217	.1084239	.003601	.985452	.7641609	1.203561
livebait	1.536074	.4083865	.013509	1.515838	.753823	2.3539
_cons	-2.805915	.4700702	.014974	-2.795244	-3.73847	-1.89491
inflate						
child	46.95902	36.33974	1.87977	38.77997	3.612863	138.3652
camper	-46.123	36.34857	1.88567	-37.66796	-137.4568	-2.544566
_cons	-46.62439	36.36232	1.88355	-38.5171	-137.5522	-3.272469
lnalpha	.7055935	.1591234	.003962	.7048862	.3959316	1.025356

Note: Default priors are used for model parameters.

The posterior mean estimates for the main regression coefficients {count:persons}, {count:livebait}, and {count:\_cons} are relatively close to the ML estimates from the zinb command, but the inflation coefficients, {inflate:child}, {inflate:camper}, and {inflate:\_cons}, are quite different. For example, zinb estimates {inflate:\_cons} are about -2.7, whereas the corresponding posterior mean estimate is about -46.6. To explain this large discrepancy, we draw the diagnostic plot of {inflate:\_cons}.

```
. bayesgraph diagnostic {inflate:_cons}
```



The marginal posterior distribution of `{inflate:_cons}` is highly skewed to the left, and it is apparent that its posterior mean is much smaller than its posterior mode. In large samples, under proper noninformative priors, the posterior mode estimator and the ML estimator are equivalent. Therefore, it is not surprising that the posterior mean of `{inflate:_cons}` is much smaller than its ML estimate. We can obtain a rough estimate of the posterior mode in this example.

First, we need to save the simulation results in a dataset, say, `sim_zinb.dta`. You can do this during estimation or on replay by specifying the `saving()` option with the `bayes` prefix.

```
. bayes, saving(sim_zinb)
note: file sim_zinb.dta saved
```

Next, we load the dataset and identify the variable that represents the parameter {inflate:\_cons}.

```
. use sim_zinb, clear
. describe
Contains data from sim_zinb.dta
  obs:      6,874
  vars:      12                               8 Mar 2019 10:44
```

variable name	storage type	display format	value label	variable label
_chain	int	%8.0g		Chain identifier
_index	int	%8.0g		Iteration number
_loglikelihood	double	%10.0g		Log likelihood
_logposterior	double	%10.0g		Log posterior
eq1_p1	double	%10.0g		{count:persons}
eq1_p2	double	%10.0g		{count:livebait}
eq1_p3	double	%10.0g		{count:_cons}
eq2_p1	double	%10.0g		{inflate:child}
eq2_p2	double	%10.0g		{inflate:camper}
eq2_p3	double	%10.0g		{inflate:_cons}
eq0_p1	double	%10.0g		{lnalpha}
_frequency	int	%8.0g		Frequency weight

Sorted by:

Variable eq2\_p3 with the variable label {inflate:\_cons} contains MCMC estimates for the {inflate:\_cons} parameter.

We use the egen's mode() function to generate a constant variable, mode, which contains the mode estimate for {inflate:\_cons}.

```
. egen mode = mode(eq2_p3)
. display mode[1]
-3.417458
```

The mode estimate for {inflate:\_cons} is about  $-3.42$ , and it is indeed much closer to the ML estimate of  $-2.70$  than its posterior mean estimate.

The inflation parameter  $\alpha$  in the likelihood of the zero-inflated negative binomial model is log-transformed, and it is represented by {lnalpha} in our posterior model. To summarize the simulation result for  $\alpha$  directly, we can use the bayesstats summary command to exponentiate {lnalpha}.

```
. bayesstats summary (alpha: exp({lnalpha}))
Posterior summary statistics                                MCMC sample size =    10,000
      alpha : exp({lnalpha})
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
alpha	2.050889	.3292052	.008191	2.023616	1.485768	2.788087

## Parametric survival model

Consider [example 7](#) in [ST] [streg](#), which analyzes the effect of a hip-protection device, age, and sex on the risk of hip fractures in patients. The survival dataset is hip3.dta with time to event variable time1 and failure variable fracture. The data are already stset.

```
. use https://www.stata-press.com/data/r16/hip3, clear
(hip fracture study)

. stset
-> stset time1, id(id) failure(fracture) time0(time0)

           id: id
    failure event: fracture != 0 & fracture < .
obs. time interval: (time0, time1]
    exit on or before: failure
```

---

206	total observations	
0	exclusions	

---

206	observations remaining, representing	
148	subjects	
37	failures in single-failure-per-subject data	
1,703	total analysis time at risk and under observation	
	at risk from t =	0
	earliest observed entry t =	0
	last observed exit t =	39

It is assumed that the hazard curves for men and women have different shapes. We use the `streg` command to fit a model with Weibull survival distribution and the ancillary variable `male` to account for the difference between men and women.

```
. streg protect age, distribution(weibull) ancillary(male) nolog
    failure _d: fracture
analysis time _t: time1
           id: id

Weibull PH regression
No. of subjects =          148           Number of obs   =          206
No. of failures =           37
Time at risk   =          1703
Log likelihood = -69.323532           LR chi2(2)         =          39.80
                                           Prob > chi2        =          0.0000
```

_t		Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
_t	protect	-2.130058	.3567005	-5.97	0.000	-2.829178	-1.430938
	age	.0939131	.0341107	2.75	0.006	.0270573	.1607689
	_cons	-10.17575	2.551821	-3.99	0.000	-15.17722	-5.174269
ln_p	male	-.4887189	.185608	-2.63	0.008	-.8525039	-.1249339
	_cons	.4540139	.1157915	3.92	0.000	.2270667	.6809611

We then perform Bayesian analysis of the same model using the `bayes` prefix. We apply more conservative normal priors, `normal(0, 100)`, by specifying the `normalprior(10)` option. To allow for longer adaptation of the MCMC sampler, we increase the burn-in period to 5,000, `burnin(5000)`.

```
. set seed 15
. bayes, normalprior(10) burnin(5000) dots:
> streg protect age, distribution(weibull) ancillary(male)
      failure _d: fracture
      analysis time _t: time1
      id: id
Burn-in 5000 aaaaaaaaaa1000aaaaaaaaa2000aaaaaaaaa3000aaaaaaaaa4000aaaaaaaaa5000
> done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

```
Likelihood:
  _t ~ streg_weibull(xb__t,xb_ln_p)
Priors:
  {_t:protect age _cons} ~ normal(0,100) (1)
  {ln_p:male _cons} ~ normal(0,100) (2)
```

(1) Parameters are elements of the linear form xb\_\_t.  
(2) Parameters are elements of the linear form xb\_ln\_p.

Bayesian Weibull PH regression	MCMC iterations	=	15,000
Random-walk Metropolis-Hastings sampling	Burn-in	=	5,000
	MCMC sample size	=	10,000
No. of subjects =	148	Number of obs	= 206
No. of failures =	37		
No. at risk =	1703		
	Acceptance rate	=	.3418
	Efficiency: min	=	.01
	avg	=	.03421
	max	=	.05481

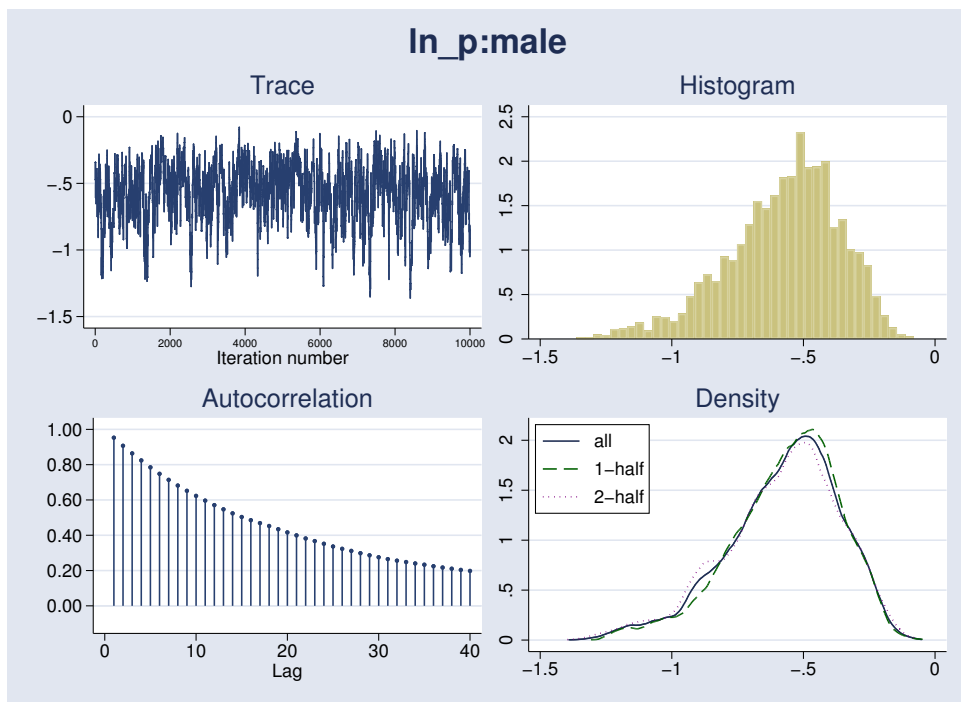
Log marginal-likelihood = -91.348814

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_t	protect	-2.114715	.3486032	.017409	-2.105721	-2.818483	-1.46224
	age	.0859305	.0328396	.001403	.0862394	.0210016	.1518009
	_cons	-9.57056	2.457818	.117851	-9.551418	-14.49808	-4.78585
ln_p	male	-.5753907	.2139477	.014224	-.5468488	-1.07102	-.2317242
	_cons	.4290642	.11786	.011786	.4242712	.203933	.6548229

Note: Default priors are used for model parameters.

The posterior mean estimates for the regression parameters {\_t:protect}, {\_t:age}, and {\_t:\_cons} are close to the estimates reported by the streg command. However, the estimate for {ln\_p:male} is somewhat different. If we inspect the diagnostic plot for {ln\_p:male}, we will see that the reason for this is the asymmetrical shape of its marginal posterior distribution.

```
. bayesgraph diagnostic {ln_p:male}
```



As evident from the density plot, the posterior distribution of `{ln_p:male}` is skewed to the left, so the posterior mean estimate,  $-0.58$ , is expected to be smaller than the ML estimate,  $-0.49$ , given that we used fairly uninformative priors; see [Zero-inflated negative binomial model](#) for the comparison of posterior mean, posterior mode, and ML estimates for highly skewed posterior distributions.

## Heckman selection model

### ► Example 11

A representative example of a Heckman selection model is provided by `wagenwk.dta`, which contains observations on the income of women who choose to work. See [example 1](#) in [\[R\] heckman](#).

```
. use https://www.stata-press.com/data/r16/womenwk, clear
```

The women's income (`wage`) is assumed to depend on their education (`educ`) and their age (`age`). In addition, the selection decision, or the choice of a woman to work, is assumed to depend on their marital status (`married`), number of children (`children`), education, and age. We fit this selection model using the `heckman` command.

```
. heckman wage educ age, select(married children educ age) nolog
Heckman selection model          Number of obs   =      2,000
(regression model with sample selection)      Selected   =      1,343
                                           Nonselected =      657
                                           Wald chi2(2)   =      508.44
Log likelihood = -5178.304          Prob > chi2    =      0.0000
```

wage	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
wage						
education	.9899537	.0532565	18.59	0.000	.8855729	1.094334
age	.2131294	.0206031	10.34	0.000	.1727481	.2535108
_cons	.4857752	1.077037	0.45	0.652	-1.625179	2.59673
select						
married	.4451721	.0673954	6.61	0.000	.3130794	.5772647
children	.4387068	.0277828	15.79	0.000	.3842534	.4931601
education	.0557318	.0107349	5.19	0.000	.0346917	.0767718
age	.0365098	.0041533	8.79	0.000	.0283694	.0446502
_cons	-2.491015	.1893402	-13.16	0.000	-2.862115	-2.119915
/athrho	.8742086	.1014225	8.62	0.000	.6754241	1.072993
/lnsigma	1.792559	.027598	64.95	0.000	1.738468	1.84665
rho	.7035061	.0512264			.5885365	.7905862
sigma	6.004797	.1657202			5.68862	6.338548
lambda	4.224412	.3992265			3.441942	5.006881

LR test of indep. eqns. (rho = 0): chi2(1) = 61.20 Prob > chi2 = 0.0000

We then apply the bayes prefix to perform Bayesian estimation of the Heckman selection model.

```
. set seed 15
. bayes, dots: heckman wage educ age, select(married children educ age)
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

---

```
Likelihood:
  wage ~ heckman(xb_wage,xb_select,{athrho} {lnsigma})
Priors:
      {wage:education age _cons} ~ normal(0,10000)      (1)
      {select:married children education age _cons} ~ normal(0,10000)  (2)
      {athrho lnsigma} ~ normal(0,10000)
```

---

- (1) Parameters are elements of the linear form xb\_wage.  
(2) Parameters are elements of the linear form xb\_select.



Bayesian Heckman selection model	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	2,000
	Selected	=	1,343
	Nonselected	=	657
	Acceptance rate	=	.3484
	Efficiency: min	=	.02314
	avg	=	.03657
	max	=	.05013
Log marginal-likelihood = -5260.2024			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
wage						
education	.9919131	.051865	.002609	.9931531	.8884407	1.090137
age	.2131372	.0209631	.001071	.2132548	.1720535	.2550835
_cons	.4696264	1.089225	.0716	.4406188	-1.612032	2.65116
select						
married	.4461775	.0681721	.003045	.4456493	.3178532	.5785857
children	.4401305	.0255465	.001156	.4402145	.3911135	.4903804
education	.0559983	.0104231	.000484	.0556755	.0360289	.076662
age	.0364752	.0042497	.000248	.0362858	.0280584	.0449843
_cons	-2.494424	.18976	.011327	-2.498414	-2.861266	-2.114334
athrho	.868392	.099374	.005961	.8699977	.6785641	1.062718
lnsigma	1.793428	.0269513	.001457	1.793226	1.740569	1.846779

Note: Default priors are used for model parameters.

The posterior mean estimates for the Bayesian model with default normal priors are similar to the ML estimates obtained with the `heckman` command.

We can calculate posterior summaries for the correlation parameter,  $\rho$ , and the standard error,  $\sigma$ , in their natural scale by inverse-transforming the model parameters `{athrho}` and `{lnsigma}` using the `bayesstats summary` command. We also include posterior summaries for the selectivity effect  $\lambda = \rho\sigma$ .

```
. bayesstats summary (rho:1-2/(exp(2*{athrho})+1))
> (sigma:exp({lnsigma}))
> (lambda:exp({lnsigma})*(1-2/(exp(2*{athrho})+1)))
Posterior summary statistics                                MCMC sample size =    10,000
    rho : 1-2/(exp(2*{athrho})+1)
    sigma : exp({lnsigma})
    lambda : exp({lnsigma})*(1-2/(exp(2*{athrho})+1))
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
rho	.6970522	.0510145	.003071	.701373	.5905851	.7867018
sigma	6.012205	.1621422	.008761	6.008807	5.700587	6.339366
lambda	4.196646	.3937209	.024351	4.212609	3.411479	4.946325

Again, the posterior mean estimates of  $\rho$ ,  $\sigma$ , and  $\lambda$  agree with the ML estimates reported by `heckman`.

## Multilevel models

The `bayes` prefix supports several [multilevel commands](#) such as `mixed` and `meglm`; see [\[BAYES\] Bayesian estimation](#). Multilevel models introduce effects at different levels of hierarchy such as hospital effects and doctor-nested-within-hospital effects, which are often high-dimensional. These effects are commonly referred to as [random effects](#) in frequentist models. Bayesian multilevel models estimate random effects together with other model parameters. In contrast, frequentist multilevel models integrate random effects out, but provide ways to predict them after estimation, conditional on other estimated model parameters. Thus, in addition to regression coefficients and variance components (variances and covariances of random effects), Bayesian multilevel models include random effects themselves as model parameters. With a slight abuse of the terminology, we will sometimes refer to regression coefficients as [fixed effects](#), keeping in mind that they are still random quantities from a Bayesian perspective.

Multilevel models are more difficult to simulate from because of the existence of high-dimensional random-effects parameters. They typically require longer burn-in periods to achieve convergence and larger MCMC sample sizes to obtain precise estimates of random effects and variance components.

Prior specification is particularly important for multilevel models. Using noninformative priors for all model parameters will likely result in nonconvergence or high autocorrelation of the MCMC sample, especially with small datasets. The default priors provided by the `bayes` prefix are chosen to be fairly uninformative, which may often lead to low simulation efficiencies for model parameters and, especially, for variance components; see [Default priors](#). So, do not be surprised to see high autocorrelation with default priors, and be prepared to investigate various prior specifications during your analysis. For example, you may need to use the `iwishartprior()` option to increase the degrees of freedom and to specify a different scale matrix of the inverse-Wishart prior distribution used for the covariance matrices of random effects.

To change the default priors, you will need to know the names of the model parameters. See [Likelihood model](#) to learn how the `bayes` prefix labels the parameters. You can specify your own name stubs for the groups of random-effects parameters using the `restubs()` option. After simulation, see [Different ways of specifying model parameters](#) for how to refer to individual random effects to evaluate MCMC convergence or to obtain their MCMC summaries.

By default, the `bayes` prefix does not compute or display MCMC summaries of individual random effects to conserve computation time and space. You can specify the `showeffects()` or `show()` option to compute and display them for chosen groups of random effects.

Also, the `bayes` prefix does not compute the log marginal-likelihood by default for multilevel models. The computation involves the inverse of the determinant of the sample covariance matrix of all parameters and loses accuracy as the number of parameters grows. For high-dimensional models such as multilevel models, the computation can be time consuming, and its accuracy may become unacceptably low. Because it is difficult to access the levels of accuracy of the computation for all multilevel models, the log marginal-likelihood is not computed by default. For multilevel models containing a small number of random effects, you can use the `remarg1` option to compute and display it.

Assessing convergence of MCMC for multilevel models is challenging because of the high dimensionality. Technically, the convergence of all parameters, including the random-effects parameters, must be explored. In practice, this may not always be feasible. Many applications focus on the regression coefficients and variance components and treat random-effects parameters as nuisance. In this case, it may be sufficient to check convergence only for the parameters of interest, especially because their convergence is adversely affected whenever there are convergence problems for many of the random-effects parameters. If the random-effects parameters are of primary interest in your study, you should evaluate their convergence. For models with a small to moderate number of random-effects

parameters, it may be beneficial to always check the convergence of the random-effects parameters. Also see *Convergence of MCMC* in [BAYES] **bayesmh**.

Two-level models

Consider [example 1](#) from [ME] **mixed** that analyzed the weight gain of 48 pigs over 9 successive weeks. Detailed Bayesian analysis of these data using **bayesmh** are presented in *Panel-data and multilevel models* in [BAYES] **bayesmh**. Here, we use **bayes: mixed** to fit Bayesian two-level random-intercept and random-coefficient models to these data.

```
. use https://www.stata-press.com/data/r16/pig
(Longitudinal analysis of pig weights)
```

► Example 12: Random-intercept model, using option melabel

We first consider a simple random-intercept model of dependent variable **weight** on covariate **week** with variable **id** identifying pigs. The random-intercept model assumes that all pigs share a common growth rate but have different initial weight.

For comparison purposes, we first use the **mixed** command to fit this model by maximum likelihood.

```
. mixed weight week || id:
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -1014.9268
Iteration 1:   log likelihood = -1014.9268
Computing standard errors:
Mixed-effects ML regression              Number of obs      =      432
Group variable: id                      Number of groups   =       48
Obs per group:
      min =          9
      avg =         9.0
      max =          9
Wald chi2(1)                            =    25337.49
Prob > chi2                             =      0.0000

Log likelihood = -1014.9268
```

weight	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974059	32.40	0.000	18.18472	20.52651

Random-effects Parameters		Estimate	Std. Err.	[95% Conf. Interval]	
id: Identity	var(_cons)	14.81751	3.124226	9.801716	22.40002
	var(Residual)	4.383264	.3163348	3.805112	5.04926

```
LR test vs. linear model: chibar2(01) = 472.65          Prob >= chibar2 = 0.0000
```

To fit a Bayesian analog of this model, we simply prefix the mixed command with bayes. We also specify the melabel option with bayes to label model parameters in the output table as mixed does.

```
. set seed 15
. bayes, melabel: mixed weight week || id:
note: Gibbs sampling is used for regression coefficients and variance
      components
Burn-in 2500 aaaaaaaaaa1000aaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Bayesian multilevel regression          MCMC iterations =    12,500
Metropolis-Hastings and Gibbs sampling   Burn-in       =     2,500
                                          MCMC sample size =   10,000
Group variable: id                     Number of groups =     48
                                          Obs per group:
                                          min =          9
                                          avg =         9.0
                                          max =          9
                                          Number of obs   =    432
                                          Acceptance rate =    .8112
                                          Efficiency: min =   .007005
                                          avg =         .5064
                                          max =          1
Log marginal-likelihood
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
	week	6.209734	.0390718	.000391	6.209354	6.133233 6.285611
	_cons	19.46511	.6239712	.07455	19.48275	18.2534 20.67396
id						
	var(_cons)	15.7247	3.436893	.049048	15.26104	10.31182 23.60471
	var(Residual)	4.411155	.3193582	.004397	4.396044	3.834341 5.080979

Note: Default priors are used for model parameters.

The estimates of posterior means and posterior standard deviations are similar to the ML estimates and standard errors from mixed. The results are also close to those from bayesmh in [example 23](#) in [\[BAYES\] bayesmh](#).

The average efficiency of the simulation is about 51% and there is no indication of any immediate convergence problems, but we should investigate convergence more thoroughly; see, for example, [example 5](#) in [\[BAYES\] Bayesian commands](#) and, more generally, [Convergence of MCMC](#) in [\[BAYES\] bayesmh](#).

Because Bayesian multilevel models are generally slower than other commands, the bayes prefix displays dots by default with multilevel commands. You can specify the nodots option to suppress them.

Also, as we described in [Multilevel models](#), the log marginal-likelihood is not computed for multilevel models by default because of the high dimensionality of the models. This is also described in the help file that appears when you click on Log marginal-likelihood in the output header in the Results window. For models with a small number of random effects, you can specify the remargl option to compute the log marginal-likelihood.

An important note about `bayes: mixed` is the default simulation method. Most `bayes` prefix commands use an adaptive MH algorithm to sample model parameters. The high-dimensional nature of multilevel models greatly decreases the simulation efficiency of this algorithm. For Gaussian multilevel models, such as `bayes: mixed`, model parameters can be sampled using a more efficient, albeit slower, Gibbs algorithm under certain prior distributions. The default priors used for regression coefficients and variance components allow the `bayes` prefix to use Gibbs sampling for these parameters with the `mixed` command. If you change the prior distributions or the default blocking structure for some parameters, Gibbs sampling may not be available for those parameters and an adaptive MH sampling will be used instead.



► **Example 13: Random-intercept model, default output**

When we specified the `melabel` option with `bayes` in [example 12](#), we intentionally suppressed some of the essential output from `bayes: mixed`. Here is what we would have seen had we not specified `melabel`.

```
. bayes
Multilevel structure
-----
id
  {U0}: random intercepts
-----
Model summary
-----
Likelihood:
  weight ~ normal(xb_weight,{e.weight:sigma2})
Priors:
  {weight:week _cons} ~ normal(0,10000) (1)
  {U0} ~ normal(0,{U0:sigma2}) (1)
  {e.weight:sigma2} ~ igamma(.01,.01)
Hyperprior:
  {U0:sigma2} ~ igamma(.01,.01)
-----
(1) Parameters are elements of the linear form xb_weight.
```

Bayesian multilevel regression  
Metropolis-Hastings and Gibbs sampling  
  
Group variable: id

MCMC iterations = 12,500  
Burn-in = 2,500  
MCMC sample size = 10,000  
Number of groups = 48  
Obs per group:  
    min = 9  
    avg = 9.0  
    max = 9  
Number of obs = 432  
Acceptance rate = .8112  
Efficiency: min = .007005  
            avg = .5064  
            max = 1

Log marginal-likelihood

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.209734	.0390718	.000391	6.209354	6.133233	6.285611
_cons	19.46511	.6239712	.07455	19.48275	18.2534	20.67396
id						
U0:sigma2	15.7247	3.436893	.049048	15.26104	10.31182	23.60471
e.weight						
sigma2	4.411155	.3193582	.004397	4.396044	3.834341	5.080979

Note: Default priors are used for model parameters.

Let’s go over the default output in detail, starting with the model summary. For multilevel models, in addition to the model summary, which describes the likelihood model and prior distributions, the bayes prefix displays information about the multilevel structure of the model.

Multilevel structure

id	{U0}: random intercepts
----	-------------------------

Our multilevel model has one set of random effects, labeled as U0, which represent random intercepts at the id level. Recall that in Bayesian models, random effects are not integrated out but estimated together with other model parameters. So, {U0}, or using its full name {U0[id]}, represent [random-effects parameters](#) in our model. See [Likelihood model](#) to learn about the default naming convention for random-effects parameters.

According to the model summary below, the likelihood of the model is a normal linear regression with the linear predictor containing regression parameters {weight:week} and {weight:\_cons} and random-effects parameters {U0}, and with the error variance labeled as {e.weight:sigma2}. Regression coefficients {weight:week} and {weight:\_cons} have default normal priors with zero means and variances of 10,000. The random intercepts {U0} are normally distributed with mean zero and variance {U0:sigma2}. The variance components, error variance {e.weight:sigma2}, and random-intercept variance {U0:sigma2} have default inverse-gamma priors, InvGamma(0.01,0.01). The random-intercept variance is a hyperparameter in our model.

```
Model summary
-----
Likelihood:
  weight ~ normal(xb_weight,{e.weight:sigma2})
Priors:
  {weight:week _cons} ~ normal(0,10000)                (1)
                        {U0} ~ normal(0,{U0:sigma2})    (1)
                        {e.weight:sigma2} ~ igamma(.01,.01)
Hyperprior:
  {U0:sigma2} ~ igamma(.01,.01)
-----
(1) Parameters are elements of the linear form xb_weight.
```

The default output table of `bayes: mixed` uses the names of model parameters as they are defined by the `bayes` prefix.

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.209734	.0390718	.000391	6.209354	6.133233	6.285611
_cons	19.46511	.6239712	.07455	19.48275	18.2534	20.67396
id						
U0:sigma2	15.7247	3.436893	.049048	15.26104	10.31182	23.60471
e.weight						
sigma2	4.411155	.3193582	.004397	4.396044	3.834341	5.080979

Note: Default priors are used for model parameters.

Becoming familiar with the native parameter names of the `bayes` prefix is important for prior specification and for later postestimation. The `melabel` option is provided for easier comparison of the results between the `bayes` prefix and the corresponding frequentist multilevel command.

► **Example 14: Displaying random effects**

By default, the `bayes` prefix does not compute or display MCMC summaries for the random-effects parameters to conserve space and computational time. You can specify the `showeffects` option to display all random effects or the `showeffects()` or `show()` option to display specific random effects. For example, continuing [example 13](#), we can display the random-effects estimates for the first five pigs as follows.

```
. bayes, show({U0[1/5]}) noheader
```

U0[id]	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
1	-1.778442	.8873077	.074832	-1.761984	-3.542545	.0062218
2	.7831408	.8775376	.071421	.7961802	-.9547035	2.491798
3	-2.052634	.9038672	.072325	-2.061559	-3.822966	-.3246834
4	-1.891103	.878177	.075611	-1.858056	-3.642227	-.1028766
5	-3.316584	.8894319	.074946	-3.320502	-5.0469	-1.568927

These posterior mean estimates of random-effects parameters should be comparable with those predicted by `predict`, `reffects` after `mixed`. Posterior standard deviations, however, will generally be larger than the corresponding standard errors of random effects predicted after `mixed`, because the latter do not incorporate the uncertainty about the estimated model parameters.

You can also use [\[BAYES\] bayesstats summary](#) to obtain MCMC summaries of random-effects parameters after estimation:

```
. bayesstats summary {U0[1/5]}
(output omitted)
```

If you decide to use the `showreffects` option to display all random-effects parameters, beware of the increased computation time for models with many random effects. Then, the `bayes` prefix will compute and display the MCMC summaries for only the first  $M$  random-effects parameters, where  $M$  is the maximum matrix dimension (`c(max_matdim)`). The number of parameters displayed and stored in `e(b)` cannot exceed `c(max_matdim)`. You can specify the `show()` option with `bayes` or use `bayesstats summary` to obtain results for other random-effects parameters.

◀

## ► Example 15: Random-coefficient model

Continuing [example 13](#), let's consider a random-coefficient model that allows the growth rate to vary among pigs.

Following `mixed`'s specification, we include the random slope for `week` at the `id` level by specifying the `week` variable in the random-effects equation.



```
. set seed 15

. bayes: mixed weight week || id: week
note: Gibbs sampling is used for regression coefficients and variance
      components
Burn-in 2500 aaaaaaaaa1000aaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Multilevel structure
```

```
id
  {U0}: random intercepts
  {U1}: random coefficients for week
```

Model summary

```
Likelihood:
  weight ~ normal(xb_weight,{e.weight:sigma2})

Priors:
  {weight:week _cons} ~ normal(0,10000) (1)
  {U0} ~ normal(0,{U0:sigma2}) (1)
  {U1} ~ normal(0,{U1:sigma2}) (1)
  {e.weight:sigma2} ~ igamma(.01,.01)

Hyperpriors:
  {U0:sigma2} ~ igamma(.01,.01)
  {U1:sigma2} ~ igamma(.01,.01)
```

(1) Parameters are elements of the linear form xb\_weight.

Bayesian multilevel regression	MCMC iterations =	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	10,000
Group variable: id	Number of groups =	48
	Obs per group:	
	min =	9
	avg =	9.0
	max =	9
	Number of obs =	432
	Acceptance rate =	.7473
	Efficiency: min =	.003057
	avg =	.07487
	max =	.1503
Log marginal-likelihood		

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.233977	.0801192	.01449	6.237648	6.05268	6.387741
_cons	19.44135	.3426786	.044377	19.44532	18.76211	20.11843
id						
U0:sigma2	7.055525	1.649394	.050935	6.844225	4.466329	10.91587
U1:sigma2	.3941786	.0901945	.002717	.3825387	.2526756	.6044887
e.weight						
sigma2	1.613775	.1261213	.003254	1.609296	1.386427	1.880891

Note: Default priors are used for model parameters.  
Note: There is a high autocorrelation after 500 lags.

In addition to random intercepts  $\{U_0\}$ , we now have random coefficients for `week`, labeled as  $\{U_1\}$ , with the corresponding variance parameter  $\{U_1:\text{sigma2}\}$ . Compared with the random-intercept model, by capturing the variability of slopes on `week`, we reduced the estimates of the error variance and the random-intercept variance.

The average simulation efficiency decreased to only 7%, and we now see a note about a high autocorrelation after 500 lags. We can use, for example, `bayesgraph` diagnostics to verify that the high autocorrelation in this example is not an indication of nonconvergence but rather of a slow mixing of our MCMC sample. If we use `bayesstats` `ess`, we will see that the coefficient on `weight` and the constant term have the lowest efficiency, which suggests that these parameters are likely to be correlated with some of the random-effects estimates. If we want to reduce the autocorrelation and improve precision of the estimates for these parameters, we can increase the MCMC sample size by specifying the `mcmcsize()` option or thin the MCMC chain by specifying the `thinning()` option.

◀

## ► Example 16: Random-coefficient model, unstructured covariance

In [example 15](#), we assumed independence between random intercepts  $\{U_0\}$  and random slopes on `week`,  $\{U_1\}$ . We relax this assumption here by specifying an unstructured covariance matrix.

Before we proceed with estimation, let's review our model summary first by specifying the `dryrun` option.

```
. bayes, dryrun: mixed weight week || id: week, covariance(unstructured)
Multilevel structure
-----
id
    {U0}: random intercepts
    {U1}: random coefficients for week
-----
Model summary
-----
Likelihood:
    weight ~ normal(xb_weight,{e.weight:sigma2})
Priors:
    {weight:week _cons} ~ normal(0,10000) (1)
    {U0}{U1} ~ mvnormal(2,{U:Sigma,m}) (1)
    {e.weight:sigma2} ~ igamma(.01,.01)
Hyperprior:
    {U:Sigma,m} ~ iwishart(2,3,I(2))
-----
(1) Parameters are elements of the linear form xb_weight.
```

The prior distributions for random effects  $\{U_0\}$  and  $\{U_1\}$  are no longer independent. Instead, they have a joint prior—a bivariate normal distribution with covariance matrix parameter  $\{U:\text{Sigma},m\}$ , which is short for  $\{U:\text{Sigma},\text{matrix}\}$ . The random-effects stub `U` is used to label the covariance matrix. The covariance matrix  $\{U:\text{Sigma},m\}$  is assigned a fairly uninformative inverse-Wishart prior with three degrees of freedom and an identity scale matrix; see [Default priors](#) for details.

Let’s now fit the model but suppress the model summary for brevity.

```
. set seed 15
. bayes, nomodelsummary: mixed weight week || id: week, covariance(unstructured)
note: Gibbs sampling is used for regression coefficients and variance
      components
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Multilevel structure
```

```
id
  {U0}: random intercepts
  {U1}: random coefficients for week
```

Bayesian multilevel regression	MCMC iterations	=	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
Group variable: id	Number of groups	=	48
	Obs per group:		
	min	=	9
	avg	=	9.0
	max	=	9
	Number of obs	=	432
	Acceptance rate	=	.7009
	Efficiency: min	=	.003683
	avg	=	.07461
Log marginal-likelihood	max	=	.1602

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.207086	.0878022	.014469	6.204974	6.041093	6.384891
_cons	19.39551	.4077822	.050353	19.40187	18.53869	20.1993
id						
U:Sigma_1_1	6.872161	1.627769	.061568	6.673481	4.282284	10.62194
U:Sigma_2_1	-.0866373	.2702822	.009861	-.0796118	-.645439	.4341423
U:Sigma_2_2	.399525	.0904532	.002488	.3885861	.2575883	.6104775
e.weight						
sigma2	1.611889	.1263131	.003155	1.605368	1.381651	1.872563

Note: Default priors are used for model parameters.  
Note: There is a high autocorrelation after 500 lags.

The 95% credible interval for the covariance between {U0} and {U1}, labeled as {U:Sigma\_2\_1} in the output, is [-.65, 0.43], which suggests independence between {U0} and {U1}.

The high autocorrelation note is due to the lower sampling efficiency of some of the regression coefficients as can be seen from the output of `bayesstats ess`:

```
. bayesstats ess
Efficiency summaries      MCMC sample size =    10,000
                          Efficiency: min =    .003683
                              avg =    .07461
                              max =    .1602
```

	ESS	Corr. time	Efficiency
weight			
week	36.83	271.55	0.0037
_cons	65.58	152.48	0.0066
id			
U:Sigma_1_1	698.99	14.31	0.0699
U:Sigma_2_1	751.20	13.31	0.0751
U:Sigma_2_2	1321.67	7.57	0.1322
e.weight			
sigma2	1602.39	6.24	0.1602

We explore the impact of this high autocorrelation on MCMC convergence in [example 17](#).



## ► Example 17: Random-coefficient model, multiple chains

We continue with the random-coefficient model with unstructured covariance from [example 16](#). Some of the parameters such as the coefficients `{weight:week}` and `{weight:_cons}` have low sampling efficiency, which raises convergence and precision concerns. Simulating multiple Markov chains of the model may help address these concerns.

We will simulate three chains by specifying the `nchains(3)` option. We will use the `rseed(15)` option to ensure reproducibility with multiple chains; see [Reproducing results](#) in [\[BAYES\] bayesmh](#). We will also suppress various model summaries by specifying the `nomodelsummary` and `nomesummary` options.

When using multiple chains to assess convergence, it is important to apply overdispersed initial values for different chains. It is difficult to quantify overdispersion because it is specific to the data and model. The default initialization provided by the `bayes: mixed` command may or may not be sufficient. To be certain, we recommend that you provide initial values explicitly, at least for the main parameters of interest. In the following specification, we provide initial values for the two regression coefficients referred to as `{weight:}`, the variance parameter `{e.weight:sigma2}`, and the covariance matrix `{U:Sigma, matrix}`. We try to generate initial values that are sufficiently separated. For example, we use `rnormal(-10, 100)` for the regression coefficients in the second chain and `rnormal(10, 100)` in the third chain. Specifying initial values for the random effects `{U0}` and `{U1}` would be more tedious, so we let them be sampled from their corresponding prior distributions. Because the hyperparameters of these priors have overdispersed initial values, we indirectly provide some overdispersion for the initial random effects as well.

```
. bayes, nchains(3) rseed(15) nomodelsummary nomesummary
> init2({weight:} rnormal(-10,100) {e.weight:sigma2} 0.1 {U:Sigma,m} 100*I(2))
> init3({weight:} rnormal(10,100) {e.weight:sigma2} 100 {U:Sigma,m} (10,-5\5,10)):
> mixed weight week || id: week, covariance(unstructured)
note: Gibbs sampling is used for regression coefficients and variance components
Chain 1
  Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaa done
  Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Chain 2
  Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaa done
  Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Chain 3
  Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaa done
  Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done

Bayesian multilevel regression                Number of chains      =           3
Metropolis-Hastings and Gibbs sampling        Per MCMC chain:
                                                Iterations              =        12,500
                                                Burn-in                 =         2,500
                                                Sample size             =        10,000

Group variable: id                            Number of groups        =           48
                                                Obs per group:
                                                    min =                   9
                                                    avg  =                  9.0
                                                    max  =                   9

                                                Number of obs           =         432
                                                Avg acceptance rate     =         .6981
                                                Avg efficiency: min     =        .003059
                                                    avg  =                  .07659
                                                    max  =                  .1663

Log marginal-likelihood                       Max Gelman-Rubin Rc    =         1.055
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.201475	.0874855	.009133	6.200176	6.032975	6.374917
_cons	19.3941	.4344171	.035266	19.38919	18.52954	20.2323
id						
U:Sigma_1_1	6.863804	1.6219	.035988	6.653249	4.329726	10.62575
U:Sigma_2_1	-.0799526	.2684949	.005546	-.0723027	-.6351419	.4354943
U:Sigma_2_2	.3983365	.0890525	.001378	.3869276	.258562	.6048894
e.weight						
sigma2	1.612452	.1254983	.001777	1.605632	1.383175	1.874105

Note: Default priors are used for model parameters.  
Note: Default initial values are used for multiple chains.  
Note: There is a high autocorrelation after 500 lags in at least one of the chains.

While the sampling efficiency of the chains is about the same as in [example 16](#), having three MCMC samples instead of one improves the precision of the estimation results, as evident from the lower MCMC errors for all model parameters.

Let’s compute Gelman–Rubin diagnostics as a convergence check. We can already see in the header of `bayes: mixed` that the maximum Gelman–Rubin statistic `Rc` of 1.055 is close to 1.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains      =          3
MCMC size, per chain =    10,000
Max Gelman-Rubin Rc   =    1.055383
```

	Rc
weight	
week	1.006404
_cons	1.055383
id	
U:Sigma_1_1	1.000567
U:Sigma_2_1	1.001168
U:Sigma_2_2	1.002119
e.weight	
sigma2	.9999899

Convergence rule: `Rc < 1.1`

The convergence diagnostic estimates `Rc` for all reported parameters are lower than 1.1, suggesting the convergence of the chains. We can also explore MCMC convergence visually; see [\[BAYES\] bayesgraph](#).



Crossed-effects model

Let’s revisit [example 4](#) from [\[ME\] meglm](#), which analyzes salamander cross-breeding data. Two populations of salamanders are considered: whiteside males and females (variables `wsm` and `wsf`) and roughbutt males and females (variables `rbm` and `rbf`). Male and female identifiers are recorded in the `male` and `female` variables. The outcome binary variable `y` indicates breeding success or failure.

In [example 4](#) of [\[ME\] meglm](#), we fit a crossed-effects logistic regression for successful mating, in which the effects of male and female were crossed. For the purpose of illustration, we will fit a crossed-effects probit regression here using [meglm](#) with the probit link.

```
. use https://www.stata-press.com/data/r16/salamander
. meglm y wsm##wsf || _all: R.male || female:, family(bernoulli) link(probit)
note: crossed random-effects model specified; option intmethod(laplace)
implied

Fitting fixed-effects model:
Iteration 0:   log likelihood = -223.01026
Iteration 1:   log likelihood = -222.78736
Iteration 2:   log likelihood = -222.78735

Refining starting values:
Grid node 0:   log likelihood = -216.49485

Fitting full model:
Iteration 0:   log likelihood = -216.49485   (not concave)
Iteration 1:   log likelihood = -214.34477   (not concave)
Iteration 2:   log likelihood = -212.34737
Iteration 3:   log likelihood = -209.42226   (not concave)
Iteration 4:   log likelihood = -209.38329   (not concave)
Iteration 5:   log likelihood = -209.37943
Iteration 6:   log likelihood = -208.11953
Iteration 7:   log likelihood = -208.1119
Iteration 8:   log likelihood = -208.11182
Iteration 9:   log likelihood = -208.11182

Mixed-effects GLM                Number of obs      =          360
Family:                          Bernoulli
Link:                             probit
```

Group Variable	No. of Groups	Observations per Group		
		Minimum	Average	Maximum
_all	1	360	360.0	360
female	60	6	6.0	6

Integration method: laplace

Log likelihood = -208.11182      Wald chi2(3) = 40.61  
 Prob > chi2 = 0.0000

y	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
1.wsm	-.4122062	.2606844	-1.58	0.114	-.9231383	.0987259
1.wsf	-1.720334	.3185787	-5.40	0.000	-2.344736	-1.095931
wsm#wsf						
1 1	2.121125	.3590666	5.91	0.000	1.417368	2.824883
_cons	.5950983	.2215291	2.69	0.007	.1609093	1.029287
_all>male var(_cons)	.3867447	.1793565			.1558374	.9597919
female var(_cons)	.44641	.1980304			.1871265	1.064958

LR test vs. probit model: chi2(2) = 29.35      Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

To fit the corresponding Bayesian model, we prefix the above command with bayes:.

```
. set seed 15
. bayes: meglm y wsm#wsf || _all: R.male || female:, family(bernoulli)
> link(probit)
Burn-in 2500 aaaaaaaaa1000aaaaaaaa2000aaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Multilevel structure
```

---

```
male
    {U0}: random intercepts
female
    {V0}: random intercepts
```

---

Model summary

---

```
Likelihood:
  y ~ meglm(xb_y)

Priors:
  {y:1.wsm 1.wsf 1.wsm#1.wsf _cons} ~ normal(0,10000)          (1)
                                     {U0} ~ normal(0,{U0:sigma2}) (1)
                                     {V0} ~ normal(0,{V0:sigma2}) (1)

Hyperpriors:
  {U0:sigma2} ~ igamma(.01,.01)
  {V0:sigma2} ~ igamma(.01,.01)
```

---

(1) Parameters are elements of the linear form xb\_y.



Bayesian multilevel GLM

Random-walk Metropolis-Hastings sampling

MCMC iterations = 12,500

Burn-in = 2,500

MCMC sample size = 10,000

Group Variable	No. of Groups	Observations per Group		
		Minimum	Average	Maximum
_all	1	360	360.0	360
female	60	6	6.0	6

Family : Bernoulli

Link : probit

Number of obs = 360

Acceptance rate = .3223

Efficiency: min = .008356

avg = .02043

max = .02773

Log marginal-likelihood

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
y	1.wsm	-.411886	.28122	.016889	-.4158334	-.9645049	.156521
	1.wsf	-1.722195	.3329918	.023312	-1.713574	-2.381169	-1.094443
	wsm#wsf						
	1 1	2.110366	.3671998	.022643	2.09234	1.443113	2.831923
	_cons	.5858733	.2512646	.015407	.5906893	.0812177	1.077352
male							
	U0:sigma2	.4291858	.2195246	.024015	.3876708	.1347684	.9648611
female							
	V0:sigma2	.4928416	.2189307	.019043	.4576824	.1648551	1.003193

Note: Default priors are used for model parameters.

The variance components for male and female, {U0:sigma2} and {V0:sigma2}, are slightly higher than the corresponding ML estimates, but the regression coefficients are similar.

## Video examples

Introduction to Bayesian statistics, part 1: The basic concepts

Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm

A prefix for Bayesian regression in Stata

Bayesian linear regression using the bayes prefix

Bayesian linear regression using the bayes prefix: How to specify custom priors

Bayesian linear regression using the bayes prefix: Checking convergence of the MCMC chain

Bayesian linear regression using the bayes prefix: How to customize the MCMC chain

## Stored results

In addition to the results stored by `bayesmh`, the `bayes` prefix stores the following in `e()`:

### Scalars

<code>e(priorsigma)</code>	standard deviation of default normal priors
<code>e(priorshape)</code>	shape of default inverse-gamma priors
<code>e(priorscale)</code>	scale of default inverse-gamma priors
<code>e(blocksize)</code>	maximum size for blocks of model parameters

### Macros

<code>e(prefix)</code>	<code>bayes</code>
<code>e(cmdname)</code>	command name from <i>estimation_command</i>
<code>e(cmd)</code>	same as <code>e(cmdname)</code>
<code>e(command)</code>	estimation command line

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayesmh`.

## Reference

Balov, N. 2017. Bayesian logistic regression with Cauchy priors using the bayes prefix. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2017/09/08/bayesian-logistic-regression-with-cauchy-priors-using-the-bayes-prefix/>.

## Also see

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

## Description

`bayesmh` fits a variety of Bayesian models using an adaptive Metropolis–Hastings (MH) algorithm. It provides various likelihood models and prior distributions for you to choose from. Likelihood models include univariate normal linear and nonlinear regressions, multivariate normal linear and nonlinear regressions, generalized linear models such as logit and Poisson regressions, and multiple-equations linear models. Prior distributions include continuous distributions such as uniform, Jeffreys, normal, gamma, multivariate normal, and Wishart and discrete distributions such as Bernoulli and Poisson. You can also program your own Bayesian models; see [\[BAYES\] bayesmh evaluators](#).

Also see [\[BAYES\] Bayesian estimation](#) for a list of Bayesian regression models that can be fit more conveniently with the `bayes` prefix ([\[BAYES\] bayes](#)).

## Quick start

Bayesian normal linear regression of `y1` on `x1` with flat priors for coefficient on `x1` and the intercept and with a Jeffreys prior on the variance parameter `{var}`

```
bayesmh y1 x1, likelihood(normal({var})) ///
      prior({y1: x1 _cons}, flat) prior({var}, jeffreys)
```

Add binary variable `a` using [factor-variable notation](#)

```
bayesmh y1 x1 i.a, likelihood(normal({var})) ///
      prior({y1: x1 i.a _cons}, flat) prior({var}, jeffreys)
```

Same as above

```
bayesmh y1 x1 i.a, likelihood(normal({var})) ///
      prior({y1:}, flat) prior({var}, jeffreys)
```

Specify a different prior for `a = 1`

```
bayesmh y1 x1 i.a, likelihood(normal({var})) ///
      prior({y1:x1 _cons}, flat) prior({y1: 1.a}, normal(0,100)) ///
      prior({var}, jeffreys)
```

Specify a starting value of 1 for parameter `{var}`

```
bayesmh y1 x1 i.a, likelihood(normal({var})) ///
      prior({y1:}, flat) prior({var}, jeffreys) initial({var} 1)
```

Same as above

```
bayesmh y1 x1 i.a, likelihood(normal({var=1})) ///
      prior({y1:}, flat) prior({var}, jeffreys)
```

A normal prior with  $\mu = 2$  and  $\sigma^2 = 0.5$  for the coefficient on `x1`, a normal prior with  $\mu = -40$  and  $\sigma^2 = 100$  for the intercept, and an inverse-gamma prior with shape parameter of 0.1 and scale parameter of 1 for `{var}`

```
bayesmh y1 x1, likelihood(normal({var})) ///
      prior({y1:x1}, normal(2,.5))      ///
      prior({y1:_cons}, normal(-40,100)) ///
      prior({var}, igamma(0.1,1))
```

Place `{var}` into a separate block

```
bayesmh y1 x1, likelihood(normal({var})) ///
      prior({y1:x1}, normal(2,.5))      ///
      prior({y1:_cons}, normal(-40,100)) ///
      prior({var}, igamma(0.1,1)) block({var})
```

As above, but simulate four chains

```
bayesmh y1 x1, likelihood(normal({var})) ///
      prior({y1:x1}, normal(2,.5))      ///
      prior({y1:_cons}, normal(-40,100)) ///
      prior({var}, igamma(0.1,1)) block({var}) ///
      nchains(4)
```

Zellner's  $g$  prior to allow `{y1:x1}` and `{y1:_cons}` to be correlated, specifying 2 dimensions,  $df = 30$ ,  $\mu = 2$  for `{y1:x1}`,  $\mu = -40$  for `{y1:_cons}`, and variance parameter `{var}`

```
bayesmh y1 x1, likelihood(normal({var})) ///
      prior({var}, igamma(0.1,1))      ///
      prior({y1:}, zellnersg(2,30,2,-40,{var}))
```

Model for dichotomous dependent variable `y2` regressed on `x1` with a logit likelihood

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
```

As above, and save model results to `simdata.dta`, and store estimates in memory as `m1`

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, ///
      normal(0,100)) saving(simdata.dta)
estimates store m1
```

As above, but save the results on replay

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
bayesmh, saving(simdata.dta)
estimates store m1
```

Show model summary without performing estimation

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) dryrun
```

Fit model without showing model summary

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
      nomodelsummary
```

As above, and specify the random-number seed for reproducibility

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
      rseed(1234)
```

Same as above (set seed method useful only for a single chain)

```
set seed 1234
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100))
```

Specify 20,000 MCMC samples, and set length of the burn-in period to 5,000

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
mcmcsize(20000) burnin(5000)
```

Specify that only observations  $1 + 5k$ , for  $k = 0, 1, \dots$ , be saved to the MCMC sample

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
thinning(5)
```

Set the maximum number of adaptive iterations of the MCMC procedure to 30, and specify that adaptation of the MCMC procedure be attempted every 25 iterations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
adaptation(maxiter(30) every(25))
```

Request that a dot be displayed every 100 simulations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
dots(100)
```

Also request that an iteration number be displayed every 1,000 iterations

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
dots(100, every(1000))
```

Same as above

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
dots
```

Request that the 90% equal-tailed credible interval be displayed

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
clevel(90)
```

Request that the default 95% highest posterior density credible interval be displayed

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) hpd
```

Use the batch-means estimator of MCSE with the length of the block of 5

```
bayesmh y2 x1, likelihood(logit) prior({y2:}, normal(0,100)) ///
batch(5)
```

Multivariate normal regression of  $y_1$  and  $y_3$  on  $x_1$  and  $x_2$ , using normal priors with  $\mu = 0$  and  $\sigma^2 = 100$  for the regression coefficients and intercepts, an inverse-Wishart prior for the covariance matrix parameter  $\{S, \text{matrix}\}$  of dimension 2,  $df = 100$ , and an identity scale matrix

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S, matrix})) ///
prior({y1:} {y3:}, normal(0,100)) ///
prior({S, matrix}, iwishart(2,100,I(2)))
```

As above, but use abbreviated declaration for the covariance matrix

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S,m})) ///
prior({y1:} {y3:}, normal(0,100)) ///
prior({S,m}, iwishart(2,100,I(2)))
```

As above, and specify starting values for matrix  $\{S, m\}$  using previously defined matrix  $W$

```
bayesmh y1 y3 = x1 x2, likelihood(mvnormal({S,m})) ///
prior({y1:} {y3:}, normal(0,100)) ///
prior({S,m}, iwishart(2,100,I(2))) initial({S,m} W)
```

Multivariate normal regression with outcome-specific regressors

```
baysmh (y1 x1 x2) (y3 x1 x3), likelihood(mvnormal({S,m})) ///
prior({y1:} {y3:}, normal(0,100)) ///
prior({S,m}, iwishart(2,100,I(2)))
```

Linear multiple-equation model of y1 on x1 and of y3 on y1, x1, and x2 with separate variance parameters for each equation

```
baysmh (y1 x1, likelihood(normal({var1}))) ///
(y3 y1 x1 x2, likelihood(normal({var2}))), ///
prior({y1:} {y3:}, flat) ///
prior({var1}, jeffreys) prior({var2}, jeffreys)
```

Nonlinear model with parameters {a}, {b}, {c}, and {var} specified using a substitutable expression

```
baysmh y1 = ({a}+{b}*x1^{c}), likelihood(normal({var})) ///
prior({a b}, normal(0,100)) prior({c}, normal(0,2)) ///
prior({var}, igamma(0.1,1))
```

Multivariate nonlinear model with distinct parameters in each equation

```
baysmh (y1 = ({a1} + {b1}*x1^{c1})) ///
(y3 = ({a2} + {b2}*x1^{c2})), likelihood(mvnormal({S,m})) ///
prior({a1 a2 b1 b2}, normal(0,100)) ///
prior({c1 c2}, normal(0,2)) prior({S,m}, iwishart(2,100,I(2)))
```

Random-intercept logistic regression of y1 on x1 with group variable gr and zero-mean normal prior with variance parameter {var} for the random-intercept parameters

```
baysmh y1 x1, likelihood(logit) reffects(gr) ///
prior({y1:i.gr}, normal(0, {var})) ///
prior({y1: x1 _cons}, flat) prior({var}, jeffreys)
```

## Menu

Statistics > Bayesian analysis > General estimation and regression

# Syntax

## Univariate linear models

```
bayesmh depvar [indepvars] [if] [in] [weight],
      likelihood(modelspec) prior(priorspec) [reffects(varname) options]
```

## Multivariate linear models

*Multivariate normal linear regression with common regressors*

```
bayesmh depvars = [indepvars] [if] [in] [weight],
      likelihood(mvnormal(...)) prior(priorspec) [options]
```

*Multivariate normal regression with outcome-specific regressors*

```
bayesmh ([eqname1:]depvar1 [indepvars1])
      ([eqname2:]depvar2 [indepvars2]) [...] [if] [in] [weight],
      likelihood(mvnormal(...)) prior(priorspec) [options]
```

## Multiple-equation linear models

```
bayesmh (eqspec) [(eqspec)] [...] [if] [in] [weight], prior(priorspec) [options]
```

## Nonlinear models

*Univariate nonlinear regression*

```
bayesmh depvar = (subexpr) [if] [in] [weight],
      likelihood(modelspec) prior(priorspec) [options]
```

*Multivariate normal nonlinear regression*

```
bayesmh (depvars1 = (subexpr1))
      (depvars2 = (subexpr2)) [...] [if] [in] [weight],
      likelihood(mvnormal(...)) prior(priorspec) [options]
```

## Probability distributions

*Univariate distributions*

```
bayesmh depvar [if] [in] [weight],
      likelihood(distribution) prior(priorspec) [options]
```

*Multiple-equation distribution specifications*

```
bayesmh (deqspec) [(deqspec)] [...] [if] [in] [weight],
      prior(priorspec) [options]
```

The syntax of *eqspec* is

```
varspec [if] [in] [weight], likelihood(modelspec) [noconstant]
```

The syntax of *varspec* is one of the following:

for single outcome

```
[eqname:]depvar [indepvars]
```

for multiple outcomes with common regressors

```
depvars = [indepvars]
```

for multiple outcomes with outcome-specific regressors

```
([eqname1:]depvar1 [indepvars1]) ([eqname2:]depvar2 [indepvars2]) [...]
```

The syntax of *deqspect* is

```
[eqname:] depvar [if] [in] [weight], likelihood(distribution)
```

*subexpr*, *subexpr1*, *subexpr2*, and so on are substitutable expressions; see [Substitutable expressions](#) for details.

The syntax of *modelspec* is

```
model [, modelopts]
```

<i>model</i>	Description
<hr/>	
Model	
<u>normal</u> (var)	normal regression with variance <i>var</i>
<u>t</u> (sigma2, df)	<i>t</i> regression with squared scale <i>sigma2</i> and degrees of freedom <i>df</i>
<u>lognormal</u> (var)	lognormal regression with variance <i>var</i>
<u>lnormal</u> (var)	synonym for <code>lognormal()</code>
<u>exponential</u>	exponential regression
<u>mvnormal</u> (Sigma)	multivariate normal regression with covariance matrix <i>Sigma</i>
<u>probit</u>	probit regression
<u>logit</u>	logistic regression
<u>logistic</u>	logistic regression; synonym for <code>logit</code>
<u>binomial</u> (n)	binomial regression with logit link and number of trials <i>n</i>
<u>binlogit</u> (n)	synonym for <code>binomial()</code>
<u>oprobit</u>	ordered probit regression
<u>ologit</u>	ordered logistic regression
<u>poisson</u>	Poisson regression
<u>llf</u> (subexpr)	substitutable expression for observation-level log-likelihood function

A distribution argument is a number for scalar arguments such as *var*; a variable name, *varname* (except for matrix arguments); a matrix for matrix arguments such as *Sigma*; a model parameter, *paramspec*; an expression, *expr*; or a substitutable expression, *subexpr*. See [Specifying arguments of likelihood models and prior distributions](#).



<i>modelopts</i>	Description
Model	
<code>offset(<i>varname</i><sub>o</sub>)</code>	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1; not allowed with <code>normal()</code> and <code>mvnormal()</code>
<code>exposure(<i>varname</i><sub>e</sub>)</code>	include <code>ln(<i>varname</i><sub>e</sub>)</code> in model with coefficient constrained to 1; allowed only with <code>poisson</code>

<i>distribution</i>	Description
Model	
<code>dexponential(<i>beta</i>)</code>	exponential distribution with scale parameter <i>beta</i>
<code>dbernoulli(<i>p</i>)</code>	Bernoulli distribution with success probability <i>p</i>
<code>dbinomial(<i>p</i>,<i>n</i>)</code>	binomial distribution with success probability <i>p</i> and number of trials <i>n</i>
<code>dpoisson(<i>mu</i>)</code>	Poisson distribution with mean <i>mu</i>

A distribution argument is a model parameter, *paramspec*, or a substitutable expression, *subexpr*, containing model parameters. An *n* argument may be a number; an expression, *expr*; or a variable name, *varname*. See [Specifying arguments of likelihood models and prior distributions](#).

The syntax of *priorspec* is

*paramref*, *priordist*

where the simplest specification of *paramref* is

*paramspec* [*paramspec* [...]]

Also see [Referring to model parameters](#) for other specifications.

The syntax of *paramspec* is

{ [*eqname*:]*param*[, *matrix*] }

where the parameter label *eqname* and parameter name *param* are valid Stata names. Model parameters are either scalars such as {*var*}, {*mean*}, and {*shape:alpha*}, or matrices such as {*Sigma*, *matrix*} and {*Scale:V*, *matrix*}. For scalar parameters, you can use {*param*=#} to specify an initial value. For example, you can specify, {*var*=1}, {*mean*=1.267}, or {*shape:alpha*=3}.

<i>priordist</i>	Description
Model	
<u>normal</u> ( <i>mu</i> , <i>var</i> )	normal with mean <i>mu</i> and variance <i>var</i>
<u>t</u> ( <i>mu</i> , <i>sigma2</i> , <i>df</i> )	location–scale <i>t</i> with mean <i>mu</i> , squared scale <i>sigma2</i> , and degrees of freedom <i>df</i>
<u>lognormal</u> ( <i>mu</i> , <i>var</i> )	lognormal with mean <i>mu</i> and variance <i>var</i>
<u>lnormal</u> ( <i>mu</i> , <i>var</i> )	synonym for <u>lognormal</u> ()
<u>uniform</u> ( <i>a</i> , <i>b</i> )	uniform on ( <i>a</i> , <i>b</i> )
<u>gamma</u> ( <i>alpha</i> , <i>beta</i> )	gamma with shape <i>alpha</i> and scale <i>beta</i>
<u>igamma</u> ( <i>alpha</i> , <i>beta</i> )	inverse gamma with shape <i>alpha</i> and scale <i>beta</i>
<u>exponential</u> ( <i>beta</i> )	exponential with scale <i>beta</i>
<u>beta</u> ( <i>a</i> , <i>b</i> )	beta with shape parameters <i>a</i> and <i>b</i>
<u>laplace</u> ( <i>mu</i> , <i>beta</i> )	Laplace with mean <i>mu</i> and scale <i>beta</i>
<u>cauchy</u> ( <i>loc</i> , <i>beta</i> )	Cauchy with location <i>loc</i> and scale <i>beta</i>
<u>chi2</u> ( <i>df</i> )	central $\chi^2$ with degrees of freedom <i>df</i>
<u>pareto</u> ( <i>alpha</i> , <i>beta</i> )	Pareto with shape <i>alpha</i> and scale <i>beta</i>
<u>jeffreys</u>	Jeffreys prior for variance of a normal distribution
<u>mvnormal</u> ( <i>d</i> , <i>mean</i> , <i>Sigma</i> )	multivariate normal of dimension <i>d</i> with mean vector <i>mean</i> and covariance matrix <i>Sigma</i> ; <i>mean</i> can be a matrix name or a list of <i>d</i> means separated by comma: <i>mu</i> <sub>1</sub> , <i>mu</i> <sub>2</sub> , ..., <i>mu</i> <sub><i>d</i></sub>
<u>mvnormal0</u> ( <i>d</i> , <i>Sigma</i> )	multivariate normal of dimension <i>d</i> with zero mean vector and covariance matrix <i>Sigma</i>
<u>mvn0</u> ( <i>d</i> , <i>Sigma</i> )	synonym for <u>mvnormal0</u> ()
<u>zellnersg</u> ( <i>d</i> , <i>g</i> , <i>mean</i> ,{ <i>var</i> })	Zellner's <i>g</i> -prior of dimension <i>d</i> with <i>g</i> degrees of freedom, mean vector <i>mean</i> , and variance parameter { <i>var</i> }; <i>mean</i> can be a matrix name or a list of <i>d</i> means separated by comma: <i>mu</i> <sub>1</sub> , <i>mu</i> <sub>2</sub> , ..., <i>mu</i> <sub><i>d</i></sub>
<u>zellnersg0</u> ( <i>d</i> , <i>g</i> ,{ <i>var</i> })	Zellner's <i>g</i> -prior of dimension <i>d</i> with <i>g</i> degrees of freedom, zero mean vector, and variance parameter { <i>var</i> }
<u>dirichlet</u> ( <i>a</i> <sub>1</sub> , <i>a</i> <sub>2</sub> ,..., <i>a</i> <sub><i>d</i></sub> )	Dirichlet (multivariate beta) of dimension <i>d</i> with shape parameters <i>a</i> <sub>1</sub> , <i>a</i> <sub>2</sub> , ..., <i>a</i> <sub><i>d</i></sub>
<u>wishart</u> ( <i>d</i> , <i>df</i> , <i>V</i> )	Wishart of dimension <i>d</i> with degrees of freedom <i>df</i> and scale matrix <i>V</i>
<u>iwishart</u> ( <i>d</i> , <i>df</i> , <i>V</i> )	inverse Wishart of dimension <i>d</i> with degrees of freedom <i>df</i> and scale matrix <i>V</i>
<u>jeffreys</u> ( <i>d</i> )	Jeffreys prior for covariance of a multivariate normal distribution of dimension <i>d</i>

<code>bernoulli(<i>p</i>)</code>	Bernoulli with success probability <i>p</i>
<code>geometric(<i>p</i>)</code>	geometric for the number of failures before the first success with success probability on one trial <i>p</i>
<code>index(<i>p</i><sub>1</sub>, ..., <i>p</i><sub><i>k</i></sub>)</code>	discrete indices 1, 2, ..., <i>k</i> with probabilities <i>p</i> <sub>1</sub> , <i>p</i> <sub>2</sub> , ..., <i>p</i> <sub><i>k</i></sub>
<code>poisson(<i>mu</i>)</code>	Poisson with mean <i>mu</i>
<code>flat</code>	flat prior; equivalent to <code>density(1)</code> or <code>logdensity(0)</code>
<code>density(<i>f</i>)</code>	generic density <i>f</i>
<code>logdensity(<i>logf</i>)</code>	generic log density <i>logf</i>

Dimension *d* is a positive number #.

A distribution argument is a number for scalar arguments such as *var*, *alpha*, *beta*; a Stata matrix for matrix arguments such as *Sigma* and *V*; a model parameter, *paramspec*; an expression, *expr*; or a substitutable expression, *subexpr*. See [Specifying arguments of likelihood models and prior distributions](#).

*f* is a nonnegative number, #; an expression, *expr*; or a substitutable expression, *subexpr*.

*logf* is a number, #; an expression, *expr*; or a substitutable expression, *subexpr*.

When `mvnormal()` or `mvnormal0()` of dimension *d* is applied to *paramref* with *n* parameters (*n* ≠ *d*), *paramref* is reshaped into a matrix with *d* columns, and its rows are treated as independent samples from the specified `mvnormal()` distribution. If such reshaping is not possible, an error is issued. See [example 25](#) for application of this feature.

<i>options</i>	Description
----------------	-------------

#### Model

<code>noconstant</code>	suppress constant term; not allowed with ordered models, nonlinear models, and probability distributions
* <code>likelihood(<i>lspec</i>)</code>	distribution for the likelihood model
* <code>prior(<i>priorspec</i>)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation

#### Model 2

<code>redefine(<i>label</i>:<i>i.varname</i>)</code>	specify a random-effects linear form; this option may be repeated
<code>xbdefine(<i>label</i>:<i>varlist</i>)</code>	specify a linear form

#### Simulation

<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(<i>paramref</i>)</code>	specify model parameters to be excluded from the simulation results

#### Blocking

<code>block(<i>paramref</i> [ , <i>blockopts</i> ])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary

#### Initialization

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation

## Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>noexpression</code>	suppress output of expressions from model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[ (<i>refref</i>) ]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Options `likelihood()` and `prior()` are required. `prior()` must be specified for all model parameters.

Options `prior()`, `redefine()`, and `block()` may be repeated.

*indepsvars* and *paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

With multiple-equations specifications, a local *if* specified within an equation is applied together with the global *if* specified with the command.

Only *fweights* are allowed; see [U] 11.1.6 weight.

With multiple-equations specifications, local weights or (weights specified within an equation) override global weights (weights specified with the command).

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

<i>blockopts</i>	Description
<code>gibbs</code>	requests Gibbs sampling; available for selected models only and not allowed with <code>scale()</code> , <code>covariance()</code> , or <code>adaptation()</code>
<code>split</code>	requests that all parameters in a block be treated as separate blocks
<code>reffects</code>	requests that all parameters in a block be treated as random-effects parameters
<code>scale(#)</code>	initial multiplier for scale factor for current block; default is <code>scale(2.38)</code> ; not allowed with <code>gibbs</code>
<code>covariance(cov)</code>	initial proposal covariance for the current block; default is the identity matrix; not allowed with <code>gibbs</code>
<code>adaptation(adaptopts)</code>	control the adaptive MCMC procedure of the current block; not allowed with <code>gibbs</code>

Only `tarate()` and `tolerance()` may be specified in the `adaptation()` option.

<i>adaptopts</i>	Description
<code>every(#)</code>	adaptation interval; default is <code>every(100)</code>
<code>maxiter(#)</code>	maximum number of adaptation loops; default is <code>maxiter(25)</code> or <code>max{25, floor(burnin()/every())}</code> whenever default values of these options are modified
<code>miniter(#)</code>	minimum number of adaptation loops; default is <code>miniter(5)</code>
<code>alpha(#)</code>	parameter controlling acceptance rate (AR); default is <code>alpha(0.75)</code>
<code>beta(#)</code>	parameter controlling proposal covariance; default is <code>beta(0.8)</code>
<code>gamma(#)</code>	parameter controlling adaptation rate; default is <code>gamma(0)</code>
<code>*tarate(#)</code>	target acceptance rate (TAR); default is parameter specific
<code>*tolerance(#)</code>	tolerance for AR; default is <code>tolerance(0.01)</code>

\*Only starred options may be specified in the `adaptation()` option specified within `block()`.

## Options

Model

- `noconstant` suppresses the constant term (intercept) from the regression model. By default, `bayesmh` automatically includes a model parameter `{depname: _cons}` in all regression models except ordered and nonlinear models. Excluding the constant term may be desirable when there is a factor variable, the base level of which absorbs the constant term in the linear combination.
- `likelihood(lspec)` specifies the distribution of the data. This option specifies the likelihood portion of the Bayesian model. This option is required. *lspec* is one of *modelspec* or *distribution*.
- modelspec* specifies one of the supported likelihood distributions for regression models. A location parameter of these distributions is automatically parameterized as a linear combination of the specified independent variables and needs not be specified. Other parameters may be specified as arguments to the distribution separated by commas. Each argument may be a real number (`#`), a variable name (except for matrix parameters), a predefined matrix, a model parameter specified in `{}`, a Stata expression, or a substitutable expression containing model parameters; see *Declaring model parameters* and *Specifying arguments of likelihood models and prior distributions*.

*distribution* specifies one of the supported distributions for modeling the dependent variable. A distribution argument must be a model parameter specified in `{}` or a substitutable expression containing model parameters; see *Declaring model parameters* and *Specifying arguments of likelihood models and prior distributions*. A number of trials,  $n$ , of the binomial distribution may be a real number (`#`), a Stata expression, or a variable name. For an example of modeling outcome distributions directly, see *Beta-binomial model*.

For some regression models, option `likelihood()` provides suboptions *subopts* in `likelihood(..., subopts)`. *subopts* is `offset()` and `exposure()`.

`offset(varnameo)` specifies that *varname<sub>o</sub>* be included in the regression model with the coefficient constrained to be 1. This option is available with `probit`, `logit`, `binomial()`, `binlogit()`, `oprobit`, `ologit`, and `poisson`.

`exposure(varnamee)` specifies a variable that reflects the amount of exposure over which the *devar* events were observed for each observation; `ln(varnamee)` with coefficient constrained to be 1 is entered into the log-link function. This option is available with `poisson`.

`prior(priorspec)` specifies a prior distribution for model parameters. This option is required and may be repeated. A prior must be specified for each model parameter. Model parameters may be scalars or matrices, but both types may not be combined in one prior statement. If multiple scalar parameters are assigned a single univariate prior, they are considered independent, and the specified prior is used for each parameter. You may assign a multivariate prior of dimension  $d$  to  $d$  scalar parameters. Also see *Referring to model parameters* and *Specifying arguments of likelihood models and prior distributions*.

All `likelihood()` and `prior()` combinations are allowed, but they are not guaranteed to correspond to proper posterior distributions. You need to think carefully about the model you are building and evaluate its convergence thoroughly; see *Convergence of MCMC*.

`dryrun` specifies to show the summary of the model that would be fit without actually fitting the model. This option is recommended for checking specifications of the model before fitting the model. The model summary reports the information about the likelihood model and about priors for all model parameters.

## Model 2

`reffects(varname)` specifies a random-effects variable, a variable identifying the group structure for the random effects, with univariate linear models. This option is useful for fitting two-level random-intercept models. A random-effects variable is treated as a factor variable with no base level. As such, you can refer to random-effects parameters or, simply, random effects associated with *varname* using a conventional factor-variable notation. For example, you can use `{devar:i.varname}` to refer to all random-effects parameters of *varname*. These parameters must be included in a single prior statement, usually a normal distribution with variance specified by an additional parameter. The random-effects parameters are assumed to be conditionally independent across levels of *varname* given all other model parameters. The random-effects parameters are automatically grouped in one block and are thus not allowed in the `block()` option. See *example 23*.

`redefine(label:i.varname)` specifies a random-effects linear form that can be used in substitutable expressions. You can use `{label:}` to refer to the linear form in substitutable expressions. You can specify `{label:i.varname}` to refer to all random-effects parameters associated with *varname*. The random-effects parameters are automatically grouped in one block and are thus not allowed in the `block()` option. This option is useful for fitting multilevel models and may be repeated. See *example 29*.

`xbdefine(label:varlist)` specifies a linear form of the variables in *varlist* that can be used in substitutable expressions. You can use the specification `{label:}` to refer to the linear form in substitutable expressions. For any *varname* in *varlist*, you can use `{label:varname}` to refer to the corresponding parameter. This option is useful with nonlinear specifications when the linear form contains many variables and provides more efficient computation in such cases.

#### Simulation

`nchains(#)` specifies the number of Markov chains to simulate. You must specify at least two chains. By default, only one chain is produced. Simulating multiple chains is useful for convergence diagnostics and to improve precision of parameter estimates. Four chains are often recommended in the literature, but you can specify more or less depending on your objective. The reported estimation results are based on all chains. You can use [bayesstats summary](#) with option `sepchains` to see the results for each chain. The reported acceptance rate, efficiencies, and log marginal-likelihood are averaged over all chains. You can use option `chainsdetail` to see these simulation summaries for each chain. Also see [Convergence diagnostics using multiple chains](#) and [Gelman–Rubin convergence diagnostic](#) in [BAYES] [bayesstats grubin](#).

`mcmcsize(#)` specifies the target MCMC sample size. The default MCMC sample size is `mcmcsize(10000)`. The total number of iterations for the MH algorithm equals the sum of the burn-in iterations and the MCMC sample size in the absence of thinning. If thinning is present, the total number of MCMC iterations is computed as `burnin() + (mcmcsize() - 1) × thinning() + 1`. Computation time of the MH algorithm is proportional to the total number of iterations. The MCMC sample size determines the precision of posterior summaries, which may be different for different model parameters and will depend on the efficiency of the Markov chain. With multiple chains, `mcmcsize()` applies to each chain. Also see [Burn-in period and MCMC sample size](#).

`burnin(#)` specifies the number of iterations for the burn-in period of MCMC. The values of parameters simulated during burn-in are used for adaptation purposes only and are not used for estimation. The default is `burnin(2500)`. Typically, burn-in is chosen to be as long as or longer than the adaptation period. With multiple chains, `burnin()` applies to each chain. Also see [Burn-in period and MCMC sample size](#) and [Convergence of MCMC](#).

`thinning(#)` specifies the thinning interval. Only simulated values from every  $(1 + k \times \#)$ th iteration for  $k = 0, 1, 2, \dots$  are saved in the final MCMC sample; all other simulated values are discarded. The default is `thinning(1)`; that is, all simulation values are saved. Thinning greater than one is typically used for decreasing the autocorrelation of the simulated MCMC sample. With multiple chains, `thinning()` applies to each chain.

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. With one chain, `rseed(#)` is equivalent to typing `set seed #` prior to calling `bayesmh`; see [R] [set seed](#). With multiple chains, you should use `rseed()` for reproducibility; see [Reproducing results](#).

`exclude(paramref)` specifies which model parameters should be excluded from the final MCMC sample. These model parameters will not appear in the estimation table, and postestimation features for these parameters and log marginal-likelihood will not be available. This option is useful for suppressing nuisance model parameters. For example, if you have a factor predictor variable with many levels but you are only interested in the variability of the coefficients associated with its levels, not their actual values, then you may wish to exclude this factor variable from the simulation results. If you simply want to omit some model parameters from the output, see the `noshow()` option. *paramref* can include individual random-effects parameters.

### Blocking

`block(paramref[ , blockopts ])` specifies a group of model parameters for the blocked MH algorithm.

By default, all parameters except matrices are treated as one block, and each matrix parameter is viewed as a separate block. You can use the `block()` option to separate scalar parameters in multiple blocks. Technically, you can also use `block()` to combine matrix parameters in one block, but this is not recommended. The `block()` option may be repeated to define multiple blocks. Different types of model parameters, such as scalars and matrices, may not be specified in one `block()`. Parameters within one block are updated simultaneously, and each block of parameters is updated in the order it is specified; the first specified block is updated first, the second is updated second, and so on. See [Improving efficiency of the MH algorithm—blocking of parameters](#).

*blockopts* include `gibbs`, `split`, `reflects`, `scale()`, `covariance()`, and `adaptation()`.

`gibbs` specifies to use Gibbs sampling to update parameters in the block. This option is allowed only for specific combinations of likelihood models and prior distributions; see [Gibbs sampling for some likelihood-prior and prior-hyperprior configurations](#). For more information, see [Gibbs and hybrid MH sampling](#). `gibbs` may not be combined with `reflects`, `scale()`, `covariance()`, or `adaptation()`.

`split` specifies that all parameters in a block are treated as separate blocks. This may be useful for levels of factor variables.

`reflects` specifies that the parameters associated with the levels of a factor variable included in the likelihood specification be treated as random-effects parameters. Random-effects parameters must be included in one prior statement and are assumed to be conditionally independent across levels of a grouping variable given all other model parameters. `reflects` requires that parameters be specified as `{devar:i.varname}`, where *i.varname* is the corresponding factor variable in the likelihood specification, and may not be combined with `block()`'s suboptions `gibbs` and `split`. This option is useful for fitting hierarchical or multilevel models. See [example 25](#) for details.

`scale(#)` specifies an initial multiplier for the scale factor corresponding to the specified block.

The initial scale factor is computed as  $\#/\sqrt{n_p}$  for continuous parameters and as  $\#/n_p$  for discrete parameters, where  $n_p$  is the number of parameters in the block. The default is `scale(2.38)`. If specified, this option overrides the respective setting from the `scale()` option specified with the command. `scale()` may not be combined with `gibbs`.

`covariance(matname)` specifies a scale matrix *matname* to be used to compute an initial proposal covariance matrix corresponding to the specified block. The initial proposal covariance is computed as  $\rho \times \text{Sigma}$ , where  $\rho$  is a scale factor and  $\text{Sigma} = \text{matname}$ . By default, *Sigma* is the identity matrix. If specified, this option overrides the respective setting from the `covariance()` option specified with the command. `covariance()` may not be combined with `gibbs`.

`adaptation(tarate())` and `adaptation(tolerance())` specify block-specific TAR and acceptance tolerance. If specified, they override the respective settings from the `adaptation()` option specified with the command. `adaptation()` may not be combined with `gibbs`.

`blocksummary` displays the summary of the specified blocks. This option is useful when `block()` is specified.

### Initialization

`initial(initspec)` specifies initial values for the model parameters to be used in the simulation.

With multiple chains, this option is equivalent to specifying option `init1()`. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example,



to initialize a scalar parameter `alpha` to 0.5 and a 2x2 matrix `Sigma` to the identity matrix `I(2)`, you can type

```
bayesmh ... , initial({alpha} 0.5 {Sigma,m} I(2)) ...
```

You can also specify a list of parameters using any of the specifications described in [Referring to model parameters](#). For example, to initialize all regression coefficients from equations `y1` and `y2` to zero, you can type

```
bayesmh ... , initial({y1:} {y2:} 0) ...
```

The general specification of *initspec* is

```
paramref initval [paramref initval [...]]
```

where *initval* is either a number, a Stata expression that evaluates to a number, or a Stata matrix for initialization of matrix parameters.

Curly braces may be omitted for scalar parameters but must be specified for matrix parameters. Initial values declared using this option override the default initial values or any initial values declared during parameter specification in the `likelihood()` option. See [Specifying initial values](#) for details.

`init#(initspec)` specifies initial values for the model parameters for the *#*th chain. This option requires option `nchains()`. `init1()` overrides the default initial values for the first chain, `init2()` for the second chain, and so on. You specify initial values in `init#()` just like you do in option `initial()`. See [Specifying initial values](#) for details.

`initall(initspec)` specifies initial values for the model parameters for all chains. This option requires option `nchains()`. You specify initial values in `initall()` just like you do in option `initial()`. You should avoid specifying fixed initial values in `initall()` because then all chains will use the same initial values. `initall()` is useful to specify random initial values when you define your own priors within `prior()`'s `density()` and `logdensity()` suboptions. See [Specifying initial values](#) for details.

`nomleinitial` suppresses using maximum likelihood estimates (MLEs) as starting values for model parameters. With multiple chains, this option and discussion below apply only to the first chain. By default, when no initial values are specified, MLE values (when available) are used as initial values. If `nomleinitial` is specified and no initial values are provided, the command uses ones for positive scalar parameters, zeros for other scalar parameters, and identity matrices for matrix parameters. `nomleinitial` may be useful for providing an alternative starting state when checking convergence of MCMC. This option cannot be combined with `initrandom`.

`initrandom` specifies that the model parameters be initialized randomly. Random initial values are generated from the prior distributions of the model parameters. If you want to use fixed initial values for some of the parameters, you can specify them in the `initial()` option or during parameter declarations in the `likelihood()` option. Random initial values are not available for parameters with `flat`, `jeffreys`, `density()`, `logdensity()`, and `jeffreys()` priors; you must provide your own initial values for such parameters. This option cannot be combined with `nomleinitial`. See [Specifying initial values](#) for details.

`initsummary` specifies that the initial values used for simulation be displayed.

---

#### Adaptation

`adaptation(adaptopts)` controls adaptation of the MCMC procedure. Adaptation takes place every prespecified number of MCMC iterations and consists of tuning the proposal scale factor and proposal covariance for each block of model parameters. Adaptation is used to improve sampling

efficiency. Provided defaults are based on theoretical results and may not be sufficient for all applications. See [Adaptation of the MH algorithm](#) for details about adaptation and its parameters.

*adaptopts* are any of the following options:

**every(#)** specifies that adaptation be attempted every *#*th iteration. The default is **every(100)**.

To determine the adaptation interval, you need to consider the maximum block size specified in your model. The update of a block with  $k$  model parameters requires the estimation of a  $k \times k$  covariance matrix. If the adaptation interval is not sufficient for estimating the  $k(k+1)/2$  elements of this matrix, the adaptation may be insufficient.

**maxiter(#)** specifies the maximum number of adaptive iterations. Adaptation includes tuning of the proposal covariance and of the scale factor for each block of model parameters. Once the TAR is achieved within the specified tolerance, the adaptation stops. However, no more than *#* adaptation steps will be performed. The default is variable and is computed as  $\max\{25, \text{floor}(\text{burnin}()/\text{adaptation}(\text{every}()))\}$ .

**maxiter()** is usually chosen to be no greater than  $(\text{mcmcsize}() + \text{burnin}())/\text{adaptation}(\text{every}())$ .

**miniter(#)** specifies the minimum number of adaptive iterations to be performed regardless of whether the TAR has been achieved. The default is **miniter(5)**. If the specified **miniter()** is greater than **maxiter()**, then **miniter()** is reset to **maxiter()**. Thus, if you specify **maxiter(0)**, then no adaptation will be performed.

**alpha(#)** specifies a parameter controlling the adaptation of the AR. **alpha()** should be in  $[0, 1]$ . The default is **alpha(0.75)**.

**beta(#)** specifies a parameter controlling the adaptation of the proposal covariance matrix. **beta()** must be in  $[0, 1]$ . The closer **beta()** is to zero, the less adaptive the proposal covariance. When **beta()** is zero, the same proposal covariance will be used in all MCMC iterations. The default is **beta(0.8)**.

**gamma(#)** specifies a parameter controlling the adaptation rate of the proposal covariance matrix. **gamma()** must be in  $[0, 1]$ . The larger the value of **gamma()**, the less adaptive the proposal covariance. The default is **gamma(0)**.

**tarate(#)** specifies the TAR for all blocks of model parameters; this is rarely used. **tarate()** must be in  $(0, 1)$ . The default AR is 0.234 for blocks containing continuous multiple parameters, 0.44 for blocks with one continuous parameter, and  $1/n\_maxlev$  for blocks with discrete parameters, where *n\_maxlev* is the maximum number of levels for a discrete parameter in the block.

**tolerance(#)** specifies the tolerance criterion for adaptation based on the TAR. **tolerance()** should be in  $(0, 1)$ . Adaptation stops whenever the absolute difference between the current AR and TAR is less than **tolerance()**. The default is **tolerance(0.01)**.

**scale(#)** specifies an initial multiplier for the scale factor for all blocks. The initial scale factor is computed as  $\#/\sqrt{n_p}$  for continuous parameters and  $\#/n_p$  for discrete parameters, where  $n_p$  is the number of parameters in the block. The default is **scale(2.38)**.

**covariance(cov)** specifies a scale matrix *cov* to be used to compute an initial proposal covariance matrix. The initial proposal covariance is computed as  $\rho \times \Sigma$ , where  $\rho$  is a scale factor and  $\Sigma = \text{matname}$ . By default,  $\Sigma$  is the identity matrix. Partial specification of  $\Sigma$  is also allowed. The rows and columns of *cov* should be named after some or all model parameters. According to some theoretical results, the optimal proposal covariance is the posterior covariance matrix of model parameters, which is usually unknown. This option does not apply to the blocks containing random-effects parameters.

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals.

The default is `clevel(95)` or as set by **[BAYES] set clevel**.

`hpd` displays the HPD credible intervals instead of the default equal-tailed credible intervals.

`eform` and `eform(string)` specify that the coefficient table be displayed in exponentiated form and that `exp(b)` and `string`, respectively, be used to label the exponentiated coefficients in the table.

`batch(#)` specifies the length of the block for calculating batch means and MCSE using batch means.

The default is `batch(0)`, which means no batch calculations. When `batch()` is not specified, MCSE is computed using effective sample sizes instead of batch means. Option `batch()` may not be combined with `corrlag()` or `corrtol()`.

`saving(filename[, replace])` saves simulation results in `filename.dta`. The `replace` option specifies to overwrite `filename.dta` if it exists. If the `saving()` option is not specified, **bayesmh** saves simulation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time **bayesmh** is run and will also be erased if the current estimation results are cleared. `saving()` may be specified during estimation or on replay.

The saved dataset has the following structure. Variable `_chain` records chain identifiers. Variable `_index` records iteration numbers. **bayesmh** saves only states (sets of parameter values) that are different from one iteration to another and the frequency of each state in variable `_frequency`. (Some states may be repeated for discrete parameters.) As such, `_index` may not necessarily contain consecutive integers. Remember to use `_frequency` as a frequency weight if you need to obtain any summaries of this dataset. Values for each parameter are saved in a separate variable in the dataset. Variables containing values of parameters without equation names are named as `eq0_p#`, following the order in which parameters are declared in **bayesmh**. Variables containing values of parameters with equation names are named as `eq#_p#`, again following the order in which parameters are defined. Parameters with the same equation names will have the same variable prefix `eq#`. For example,

```
. bayesmh y x1, likelihood(normal({var})) saving(mcmc) ...
```

will create a dataset, `mcmc.dta`, with variable names `eq1_p1` for `{y:x1}`, `eq1_p2` for `{y:_cons}`, and `eq0_p1` for `{var}`. Also see macros `e(parnames)` and `e(varnames)` for the correspondence between parameter names and variable names.

In addition, **bayesmh** saves variable `_loglikelihood` to contain values of the log likelihood from each iteration and variable `_logposterior` to contain values of the log posterior from each iteration.

`nomodelsummary` suppresses the detailed summary of the specified model. The model summary is reported by default.

`noexpression` suppresses the output of expressions from the model summary. Expressions (when specified) are reported by default.

`chainsdetail` specifies that acceptance rates, efficiencies, and log marginal-likelihoods be reported separately for each chain. By default, the header reports these statistics averaged over all chains. This option requires option `nchains()`.

`nodots`, `dots`, and `dots(#)` specify to suppress or display dots during simulation. With multiple chains, these options affect all chains. `dots(#)` displays a dot every # iterations. During the adaptation period, a symbol `a` is displayed instead of a dot. If `dots(..., every(#))` is specified, then an iteration number is displayed every #th iteration instead of a dot or `a`. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers

every 1,000 iterations; it is a synonym for `dots(100, every(1000))`. By default, no dots are displayed (`nodots` or `dots(0)`).

`show(paramref)` or `noshow(paramref)` specifies a list of model parameters to be included in the output or excluded from the output, respectively. By default, all model parameters (except random-effects parameters when `reffects()` is specified) are displayed. Do not confuse `noshow()` with `exclude()`, which excludes the specified parameters from the MCMC sample. When the `noshow()` option is specified, for computational efficiency, MCMC summaries of the specified parameters are not computed or stored in `e()`. *paramref* can include individual random-effects parameters.

`showreffects` and `showreffects(reref)` are used with option `reffects()` and specify that all or a list *reref* of random-effects parameters be included in the output in addition to other model parameters. By default, all random-effects parameters introduced by `reffects()` are excluded from the output as if you have specified the `noshow()` option. This option computes, displays, and stores in `e()` MCMC summaries for the random-effects parameters.

`notable` suppresses the estimation table from the output. By default, a summary table is displayed containing all model parameters except those listed in the `exclude()` and `noshow()` options. Regression model parameters are grouped by equation names. The table includes six columns and reports the following statistics using the MCMC simulation results: posterior mean, posterior standard deviation, MCMC standard error or MCSE, posterior median, and credible intervals.

`noheader` suppresses the output header either at estimation or upon replay.

`title(string)` specifies an optional title for the command that is displayed above the table of the parameter estimates. The default title is specific to the specified likelihood model.

*display\_options*: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fwrapon(style)`, and `nolstretch`; see [R] [Estimation options](#).

#### Advanced

`search(search_options)` searches for feasible initial values. *search\_options* are `on`, `repeat(#)`, and `off`.

`search(on)` is equivalent to `search(repeat(500))`. This is the default.

`search(repeat(k))`,  $k > 0$ , specifies the number of random attempts to be made to find a feasible initial-value vector, or initial state. The default is `repeat(500)`. An initial-value vector is feasible if it corresponds to a state with positive posterior probability. If feasible initial values are not found after  $k$  attempts, an error will be issued. `repeat(0)` (rarely used) specifies that no random attempts be made to find a feasible starting point. In this case, if the specified initial vector does not correspond to a feasible state, an error will be issued.

`search(off)` prevents the command from searching for feasible initial values. We do not recommend specifying this option.

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is `min{500, mcmcsizes()/2}`. The total autocorrelation is computed as the sum of all lag- $k$  autocorrelation values for  $k$  from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrctl()` if the latter is less than `corrlag()`. Options `corrlag()` and `batch()` may not be combined.

`corrctl(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrctl(0.01)`. For a given model parameter, if the absolute value of the lag- $k$  autocorrelation is less than `corrctl()`, then all autocorrelation lags beyond the  $k$ th lag are discarded. Options `corrctl()` and `batch()` may not be combined.

## Remarks and examples

Remarks are presented under the following headings:

- Using bayesmh*
- Setting up a posterior model*
  - Likelihood model*
  - Prior distributions*
  - Declaring model parameters*
  - Referring to model parameters*
  - Specifying arguments of likelihood models and prior distributions*
  - Substitutable expressions*
  - Checking model specification*
- Specifying MCMC sampling procedure*
  - Reproducing results*
  - Burn-in period and MCMC sample size*
  - Improving efficiency of the MH algorithm—blocking of parameters*
  - Gibbs and hybrid MH sampling*
  - Adaptation of the MH algorithm*
  - Specifying initial values*
- Summarizing and reporting results*
  - Posterior summaries and credible intervals*
  - Saving MCMC results*
- Convergence of MCMC*
- Video examples*

Examples are presented under the following headings:

- Getting started examples*
  - Mean of a normal distribution with a known variance*
  - Mean of a normal distribution with an unknown variance*
  - Simple linear regression*
  - Multiple linear regression*
  - Improving efficiency of the MH sampling*
- Convergence diagnostics using multiple chains*
  - Multiple chains using default initial values*
  - Multiple chains using overdispersed initial values*
- Bayesian predictions*
  - Simulating replicated outcomes*
  - Posterior predictive checks*
- Logistic regression model: A case of nonidentifiable parameters*
- Ordered probit regression*
- Beta-binomial model*
- Multivariate regression*
- Panel-data and multilevel models*
  - Two-level random-intercept model or panel-data model*
  - Linear growth curve model—a random-coefficient model*
  - Mixed-effects logistic regression*
- Bayesian analysis of change-point problem*
- Bioequivalence in a crossover trial*
- Random-effects meta-analysis of clinical trials*
- Item response theory*

For a quick overview example of all Bayesian commands, see *Overview example* in [BAYES] **Bayesian commands**.

## Using bayesmh

The `bayesmh` command for Bayesian analysis includes three functional components: setting up a posterior model, performing MCMC simulation, and summarizing and reporting results. The first component, the model-building step, requires some experience in the practice of Bayesian statistics

and, as any modeling task, is probably the most demanding. You should specify a posterior model that is statistically correct and that represents the observed data. Another important aspect is the computational feasibility of the model in the context of the MH MCMC procedure implemented in `bayesmh`. The provided MH algorithm is adaptive and, to a degree, can accommodate various statistical models and data structures. However, careful model parameterization and well-specified initial values and MCMC sampling scheme are crucial for achieving a fast-converging Markov chain and consequently good results. Simulation of MCMC must be followed by a thorough investigation of the convergence of the MCMC algorithm. Once you are satisfied with the convergence of the simulated chains, you may proceed with posterior summaries of the results and their interpretation. Below we discuss the three major steps of using `bayesmh` and provide recommendations.

## Setting up a posterior model

Any posterior model includes a likelihood model that specifies the conditional distribution of the data given model parameters and prior distributions for all model parameters. The prior distribution of a parameter can itself be specified conditional on other parameters, also referred to as *hyperparameters*. We will refer to their prior distributions as *hyperpriors*.

### Likelihood model

The likelihood model describes the data. You build your likelihood model the same way you do this in frequentist likelihood-based analysis.

The `bayesmh` command provides various likelihood models, which are specified in the `likelihood()` option. For a univariate response, there are normal models, generalized linear models for binary and count response, and more. For a multivariate model, you may choose between a multivariate normal model with covariates common to all variables and with covariates specific to each variable. You can also build likelihood models for multiple variables by specifying a distribution and a regression function for each variable by using `bayesmh`'s multiple-equation specification.

`bayesmh` is primarily designed for fitting regression models. As we said above, you specify the likelihood or outcome distribution in the `likelihood()` option. The regression specification of the model is the same as for other regression commands. For a univariate response, you specify the dependent and all independent variables following the command name. (Here we also include the `prior()` option that specifies prior distributions to emphasize that it is required in addition to `likelihood()`. See the next subsection for details about this option.)

```
. bayesmh y x1 x2, likelihood() prior() ...
```

For a multivariate response, you separate the dependent variables from the independent variables with the equal sign.

```
. bayesmh y1 y2 = x1 x2, likelihood(mvnormal(...)) prior() ...
```

With the multiple-equation specification, you follow the syntax for the univariate response, but you specify each equation in parentheses and you specify the `likelihood()` option within each equation.

```
. bayesmh (y1 x1, likelihood()) (y2 x2, likelihood()), prior() ...
```

In the above models, the regression function is modeled using a linear combination of the specified independent variables and regression coefficients. The constant is included by default, but you can specify the `noconstant` option to omit it from the linear predictor.

**bayesmh** also allows you to model the regression function as a nonlinear function of independent variables and regression parameters. In this case, you must use the equal sign to separate the dependent variable from the expression and specify the expression in parentheses:

```
. bayesmh y = ({a}+{b}*x^{c}), likelihood(normal()) prior() ...

. bayesmh (y1 = ({a1}+{b1}*x^{c1})    ///
            (y2 = ({a2}+{b2}*x^{c2})), likelihood(mvnormal()) prior() ...
```

You can also model an outcome distribution directly by specifying one of the supported [probability distributions](#).

For a not-supported or nonstandard likelihood, you can use the `llf()` option within `likelihood()` to specify a generic expression for the observation-level likelihood function; see [Substitutable expressions](#). When you use the `llf()` option, it is your responsibility to ensure that the provided expression corresponds to a valid density. For more complicated Bayesian models, you may consider writing your own likelihood or posterior function evaluators; see [\[BAYES\] bayesmh evaluators](#).

## Prior distributions

In addition to the likelihood, you must also specify prior distributions for all model parameters in a Bayesian model. Prior distributions or priors are key components in a Bayesian model specification and should be chosen carefully. They are used to quantify some expert knowledge or existing information about model parameters. For example, priors can be used for constraining the domain of some parameters to localize values that we think are more probable for reasons that are not considered in the likelihood specification. Improper priors (priors with densities that do not integrate to finite numbers) are also allowed, as long as they yield valid posterior distributions. Priors are often categorized as informative (subjective) or noninformative (objective). Noninformative priors are also known as vague priors. Uniform distributions are often used as noninformative priors and can even be applied to parameters with unbounded domains, in which case they become improper priors. Normal and gamma distributions with very large variances relative to the expected values of the parameters are also used as noninformative priors. Another family of noninformative priors, often chosen for their invariance under reparameterization, are so-called Jeffreys priors, named after Harold Jeffreys ([Jeffreys 1946](#)). For example, the **bayesmh** command provides built-in Jeffreys priors for the normal family of distributions. Jeffreys priors are usually improper. As discussed by many researchers, however, the overuse of noninformative priors contradicts the principles of Bayesian approach—analysis of a posterior model with noninformative priors would be close to one based on the likelihood only. Noninformative priors may also negatively influence the MCMC convergence. It is thus important to find good priors based on earlier studies and use them in the model as well as perform sensitivity analysis for competing priors. A good choice of prior should minimize the MCMC standard errors of the parameter estimates.

As for likelihoods, the **bayesmh** command provides several priors you can choose from by specifying the `prior()` options. For example, continuous univariate priors include normal, lognormal, uniform, inverse gamma, and exponential; discrete priors include Bernoulli and Poisson; multivariate priors include multivariate normal and inverse Wishart. There are also special priors: `jeffreys` and `jeffreys(#)`, which specify Jeffreys priors for the variance of the normal and multivariate normal distributions, and `zellnersg()` and `zellnersg0()`, which specify multivariate priors for regression coefficients ([Zellner and Revankar 1969](#)).

The `prior()` option is required and may be repeated. You can use the `prior()` option for each parameter or you can combine multiple parameters in one `prior()` specification.

For example, we can specify different priors for parameters `{y:x}` and `{y:_cons}` by

```
. bayesmh y x, ... prior({y:x}, normal(10,100)) prior({y:_cons}, normal(20,200)) ...
```



or the same univariate prior using one `prior()` statement, using

```
. bayesmh y x, ... prior({y:x _cons}, normal(10,100)) ...
```

or a multivariate prior with zero mean and fixed variance–covariance `S`, as follows:

```
. bayesmh y x, ... prior({y:x _cons}, mvnormal0(2,S)) ...
```

In the `prior()` option, we list model parameters following any of the specifications described in [Referring to model parameters](#) and then, following the comma, we specify one of the prior distributions *priordist*.

If you want to specify a nonstandard prior or if the prior you need is not supported, you can use the `density()` or `logdensity()` option within the `prior()` option to specify an expression for a generic density or log density of the prior distribution; see [Substitutable expressions](#). When you use the `density()` or `logdensity()` option, it is your responsibility to ensure that the provided expression corresponds to a valid density. For a complicated Bayesian model, you may consider writing your own posterior function evaluator; see [\[BAYES\] bayesmh evaluators](#).

Sometimes, you may need to specify a flat prior (a prior with the density equal to one) for some of the parameters. This is often needed when specifying a noninformative prior. You can specify the `flat` option instead of the prior distribution in the `prior()` option to request the flat prior. This option is equivalent to specifying `density(1)` or `logdensity(0)` in `prior()`.

The specified likelihood model for the data and prior distributions for the parameters are not guaranteed to result in proper posterior distributions of the parameters. Therefore, unless you are using one of the standard Bayesian models, you should always check the validity of the posterior model you specified.

## Declaring model parameters

Model parameters are typically declared, meaning first introduced, in the arguments of distributions specified in options `likelihood()` and `prior()`. We will refer to model parameters that are declared in the prior distributions (and not the likelihood distributions) as hyperparameters. Model parameters may also be declared within the parameter specification of the `prior()` option, but this is more rare.

`bayesmh` distinguishes between two types of model parameters: scalar and matrix. All parameters must be specified in curly braces, `{}`. There are two ways for declaring a scalar parameter: `{param}` and `{eqname:param}`, where *param* and *eqname* are valid Stata names.

The specification of a matrix parameter is similar, but you must use the `matrix` suboptions: `{param, matrix}` and `{eqname:param, matrix}`. The most common application of matrix model parameters is for specifying the variance–covariance matrix of a multivariate normal distribution.

All matrices are assumed to be symmetric and only the elements in the lower diagonal are reported in the output. Only a few multivariate prior distributions are available for matrix parameters: `wishart()`, `iwishart()`, and `jeffreys()`. In addition to being symmetric, these distributions require that the matrices be positive definite.

It is your responsibility to declare all parameters of your model, except regression coefficients in linear models. For a linear model, `bayesmh` automatically creates a regression coefficient with the name `{depvar:indepvar}` for each independent variable *indepvar* in the model and, if `noconstant` is not specified, an intercept parameter `{depvar:_cons}`. In the presence of factor variables, `bayesmh` will create a parameter `{depvar:level}` for each level indicator *level* and a parameter `{depvar:inter}` for each interaction indicator *inter*; see [\[U\] 11.4.3 Factor variables](#). (It is still your responsibility, however, to specify prior distributions for the regression parameters.)



For example,

```
. bayesmh y x, ...
```

will automatically have two regression parameters: `{y:x}` and `{y:_cons}`, whereas

```
. bayesmh y x, noconstant ...
```

will have only one: `{y:x}`.

For a univariate normal linear regression, we may want to additionally declare the scalar variance parameter by

```
. bayesmh y x, likelihood(normal({sig2})) ...
```

We can label the variance parameter, as follows:

```
. bayesmh y x, likelihood(normal({var:sig2})) ...
```

We can declare a hyperparameter for `{sig2}` using

```
. bayesmh y x, likelihood(normal({sig2})) prior({sig2}, igamma({df},2)) ...
```

where the hyperparameter `{df}` is declared in the inverse-gamma prior distribution for `{sig2}`.

For a multivariate normal linear regression, in addition to four regression parameters declared automatically by `bayesmh`: `{y1:x}`, `{y1:_cons}`, `{y2:x}`, and `{y2:_cons}`, we may also declare a parameter for the variance–covariance matrix:

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma, matrix})) ...
```

or abbreviate `matrix` to `m` for short:

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma, m})) ...
```

## Referring to model parameters

After a model parameter is declared, we may need to refer to it in our further model specification. We will definitely need to refer to it when we specify its prior distribution. We may also need to use it as an argument in the prior distributions of other parameters or need to specify it in the `block()` option for blocking of model parameters; see [Improving efficiency of the MH algorithm—blocking of parameters](#).

To refer to one parameter, we simply use its definition: `{param}`, `{eqname:param}`, `{param, matrix}`, or `{eqname:param, matrix}`. There are several ways in which you can refer to multiple parameters. You can refer to multiple model parameters in the parameter specification `paramref` of the `prior(paramref, ...)` option, of the `block(paramref, ...)` option, or of the `initial(paramref #)` option.

The most straightforward way to refer to multiple scalar model parameters is to simply list them individually, as follows:

```
{param1} {param2} ...
```

but there are shortcuts.

For example, the alternative to the above is

```
{param1 param2} ...
```

where we simply list the names of all parameters inside one set of curly braces.

If parameters have the same equation name, you can refer to all the parameters with that equation name as follows. Suppose that we have three parameters with the same equation name `eqname`, then the specification

```
{eqname:param1} {eqname:param2} {eqname:param3}
```

is the same as the specification

```
{eqname:}
```

or the specification

```
{eqname:param1 param2 param3}
```

The above specification is useful if we want to refer to a subset of parameters with the same equation name. For example, in the above, if we wanted to refer to only `param1` and `param2`, we could type

```
{eqname:param1 param2}
```

If a factor variable is used in the specification of the regression function, you can use the same factor-variable specification within *paramref* to refer to the coefficients associated with the levels of that factor variable; see [U] 11.4.3 **Factor variables**.

For example, factor variables are useful for constructing multilevel Bayesian models. Suppose that variable `id` defines the second level of hierarchy in a two-level random-effects model. We can fit a Bayesian random-intercept model as follows:

```
. bayesmh y x i.id, likelihood(normal({var})) prior({y:i.id}, normal(0,{tau})) ...
```

Here we used `{y:i.id}` in the prior specification to refer to all levels of `id`.

Similarly, we can add a random coefficient for a continuous covariate `x` by typing

```
. bayesmh y c.x##i.id, likelihood(normal({var}))
>                                prior({y:i.id}, normal(0,{tau1}))
>                                prior({y:c.x#i.id}, normal(0,{tau2})) ...
```

You can mix and match all the specifications above in one parameter specification, *paramref*.

To refer to multiple matrix model parameters, you can use `{paramlist, matrix}` to refer to matrix parameters with names *paramlist* and `{eqname:paramlist, matrix}` to refer to matrix parameters with names in *paramlist* and with equation name *eqname*.

For example, the specification

```
{eqname:Sigma1,m} {eqname:Sigma2,m} {Sigma3,m} {Sigma4,m}
```

is the same as the specification

```
{eqname:Sigma1 Sigma2,m} {Sigma3 Sigma4,m}
```

You cannot refer to both scalar and matrix parameters in one *paramref* specification.

For referring to model parameters in postestimation commands, see *Different ways of specifying model parameters* in [BAYES] **Bayesian postestimation**.

## Specifying arguments of likelihood models and prior distributions

As previously mentioned, likelihood distributions (or more precisely, likelihood models), *modelspec*, are specified in the `likelihood(modelspec)` option and prior distributions *priordist* are specified following the comma in the `prior(paramref, priordist)` option. For a list of supported models and distributions, see the corresponding tables in the syntax diagram.

In a likelihood model, mean and location parameters are determined by the specified regression function and thus need not be specified in the likelihood distributions. For example, for a normal linear regression, we use `likelihood(normal(var))`, where we specify only the variance parameter—the mean is already parameterized as a linear combination of the specified independent variables. In the prior distributions, we must specify all parameters of the distribution. For example, for a normal prior specification, we use `prior(paramref, normal(mu, var))`, where we must specify both mean *mu* and variance *var*. In addition, all multivariate prior distributions require that you specify the dimension *d* as the first argument.

Scalar arguments of the distributions may be specified as a number or as a scalar expression *exp*. Matrix arguments of the distributions may be specified as a matrix or as a matrix expression *exp*. Both types of arguments may be specified as a parameter (see [Declaring model parameters](#)) or as a substitutable expression, *subexp* (see [Substitutable expressions](#)). All distribution arguments, except the dimension *d* of multivariate prior distributions, support the above specifications. For likelihood models, arguments of the distributions may also contain variable names.

For example, in a normal linear regression, we can specify the variance as a known value of 25,

```
. bayesmh y x, likelihood(normal(25)) ...
```

or as a squared standard deviation of 5 (scalar expression),

```
. bayesmh y x, likelihood(normal(5^2)) ...
```

or as an unknown variance parameter `{var}`,

```
. bayesmh y x, likelihood(normal({var})) ...
```

or as a function of an unknown standard-deviation parameter `{sd}` (substitutable expression),

```
. bayesmh y x, likelihood(normal({sd}^2)) ...
```

In a multivariate normal linear regression, we can specify the variance–covariance matrix as a known matrix *S*,

```
. bayesmh y1 y2 = x, likelihood(mvnormal(S)) ...
```

or as a matrix function  $S = R \cdot R'$  using its Cholesky decomposition,

```
. bayesmh y1 y2 = x, likelihood(mvnormal(R*R')) ...
```

or as an unknown matrix parameter `{Sigma,m}`,

```
. bayesmh y1 y2 = x, likelihood(mvnormal({Sigma,m})) ...
```

or as a function of an unknown variance parameter `{var}` (substitutable expression),

```
. bayesmh y1 y2 = x, likelihood(mvnormal({var}*S)) ...
```

## Substitutable expressions

You may use substitutable expressions in `bayesmh` to define nonlinear expressions *subexpr*, arguments of outcome distributions in option `likelihood()`, observation-level log likelihood in option `llf()`, arguments of prior distributions in option `prior()`, and generic prior distributions in `prior()`'s suboptions `density()` and `logdensity()`. Substitutable expressions are just like any other mathematical expression in Stata, except that they may include model parameters.

To specify a substitutable expression in your `bayesmh` model, you must comply with the following rules:

1. Model parameters are bound in braces: `{mu}`, `{var:sigma2}`, `{Sigma, matrix}`, and `{Cov:Sigma, matrix}`.
2. Linear combinations can be specified using the notation

```
{eqname: varlist[, xb noconstant] }
```

For example, `{lc:mpg price weight}` is equivalent to

```
{lc:mpg}*mpg + {lc:price}*price + {lc:weight}*weight + {mpg:_cons}
```

The `xb` option is used to distinguish between the linear combination that contains one variable and a free parameter that has the same name as the variable and the same group name as the linear combination. For example, `{lc:weight, xb}` is equivalent to `{lc:_cons} + {lc:weight}*weight`, whereas `{lc:weight}` refers to either a free parameter `weight` with a group name `lc` or the coefficient of the `weight` variable, if `{lc:}` has been previously defined in the expression as a linear combination that involves variable `weight`. Thus the `xb` option indicates that the specification is a linear combination rather than a single parameter to be estimated.

When you define a linear combination, a constant term is included by default. The `noconstant` option suppresses the constant.

See [Linear combinations](#) in [ME] `menl` for details about specifying linear combinations.

3. Initial values are given by including an equal sign and the initial value inside the braces, for example, `{b1=1.267}`, `{gamma=3}`, etc. If you do not specify an initial value, that parameter is initialized to one for positive scalar parameters and to zero for other scalar parameters, or it is initialized to its MLE, if available. The `initial()` option overrides initial values provided in substitutable expressions. Initial values for matrices must be specified in the `initial()` option. By default, matrix parameters are initialized with identity matrices.

**Specifying linear combinations.** We can use substitutable expressions to specify linear combinations.

For example, a normal linear regression,

```
. bayesmh y x1 x2, likelihood(normal(1)) prior({y:}, normal(0,100))
```

may be equivalently (but less efficiently) fit using a nonlinear regression,

```
. bayesmh y = ({y:x1 x2}), likelihood(normal(1)) prior({y:}, normal(0,100))
```

The above nonlinear specification is essentially,

```
. bayesmh y = ({y:x1}*x1+{y:x2}*x2+{y:_cons}), likelihood(normal(1))
> prior({y:}, normal(0,100))
```

**Specifying nonstandard densities.** We can use substitutable expressions to define nonstandard or not-supported probability distributions.

For example, suppose we want to specify a Cauchy distribution with location  $a$  and scale  $b$ . We can specify the expression for the observation-level likelihood function in the `llf()` option within `likelihood()`.

```
. bayesmh y, likelihood(llf(ln({b})-ln({b}^2+(y-{a})^2)-ln(_pi))) noconstant ...
```

You can also use substitutable expressions to define nonstandard or not-supported prior distributions. For example, as suggested by [Gelman et al. \(2014\)](#), we can specify a Cauchy prior with location  $a = 0$  and scale  $b = 2.5$  for logistic regression coefficients, where continuous covariate  $x$  is standardized to have mean 0 and standard deviation 0.5. If `bayesmh` did not support the Cauchy prior (option `prior(, cauchy())`), we could have specified this prior using the substitutable expressions as follows:

```
. bayesmh y x, likelihood(logit)
> prior({y:x}, logdensity(ln(2.5)-ln(2.5^2+{y:x}^2)-ln(_pi)))
> prior({y:_cons}, logdensity(ln(10)-ln(10^2+{y:_cons}^2)-ln(_pi)))
```

## Checking model specification

Specifying a Bayesian model may be a tedious task when there are many model parameters and possibly hyperparameters. It is thus essential to verify model specification before starting a potentially time-consuming estimation.

`bayesmh` displays the summary of the specified model as a part of its standard output. You can use the `dryrun` option to obtain the model summary without estimation or simulation. Once you are satisfied with the specified model, you can use the `nomodelsummary` option to suppress a potentially long model summary during estimation. Even if you specify `nomodelsummary` during estimation, you will still be able to see the model summary, if desired, by simply replaying the results:

```
. bayesmh
```

## Specifying MCMC sampling procedure

Once you specify a correct posterior model, `bayesmh` uses an adaptive random-walk MH algorithm to obtain MCMC samples of model parameters from their posterior distribution.

## Reproducing results

Because `bayesmh` uses MCMC simulation—a stochastic procedure for sampling from a complicated and possibly nontractable distribution—it will produce different results each time you run the command. If the MCMC algorithm converged, the results should not change drastically. To obtain reproducible results, you must specify the random-number seed.

To specify a random-number seed, you can use `bayesmh`'s `rseed()` option. With a single chain, you can instead use `set seed #` prior to calling `bayesmh`; see [\[R\] set seed](#). With multiple chains, you should use `rseed()` for reproducibility because, as we explain later, using `set seed` is no longer sufficient.

With a single chain, if you forgot to specify the random-number seed before calling `bayesmh`, you can retrieve the random-number state used by the command from `e(rngstate)` and use it later with `set rngstate`. With multiple chains, reproducing results after the simulation without specifying the seed is more difficult. We strongly recommend that you specify the `rseed()` option with `bayesmh` when simulating multiple chains.

When you specify the `nchains()` option to simulate multiple chains, each chain uses its own stream of random numbers; see [R] [set rngstream](#). This is important to ensure that the chains are independent. To reproduce the simulation results, a random-number seed must be used for each stream. This is why using `set seed` prior to calling `bayesmh` will not be sufficient to reproduce results from multiple chains—`set seed` will affect only the first random-number stream. `bayesmh`'s `rseed()` option, however, will use the specified random-number seed with each stream. If you forgot to specify the seed with multiple chains, you can retrieve chain-specific random-number states from stored scalars `e(rngstate1)`, `e(rngstate2)`, etc. and use them with chain-specific random-number streams; see [R] [set rngstream](#) and `set rngstate` in [R] [set seed](#). For example, suppose you simulated two chains and forgot to specify the random-number seed:

```
. bayesmh ..., nchains(2) ...
```

You can type the following directly after the simulation to reproduce the results:

```
. set rng mt64s
. set rngstate 'e(rngstate2)'
. set rngstate 'e(rngstate1)'
. bayesmh ..., nchains(2) ...
```

Stata's default random-number generator is `mt64`; see [R] [set rng](#). To simulate multiple chains, the `nchains()` option temporarily switches to the stream random-number generator `mt64s`. To manually reproduce the results from multiple chains, you need to use `mt64s`, but we recommend that you switch back to `mt64` for the rest of your analysis. The `set rngstate` command sets the corresponding stream automatically; you do not need to use `set rngstream` to do this yourself. It is important, however, that you set the state of the first chain last, just before the next call to `bayesmh`, so that the stream used by the first chain is the current stream. Although you can reproduce results after estimation, we strongly recommend that you use the `rseed()` option during estimation if you want reproducibility.

## Burn-in period and MCMC sample size

`bayesmh` has the default burn-in period of 2,500 iterations and the default MCMC sample size of 10,000 iterations. That is, the first 2,500 iterations of the MCMC sampler are discarded and the next 10,000 iterations are used to form the MCMC samples of values of model parameters. You can change these numbers by specifying options `burnin()` and `mcmcsize()`.

The burn-in period must be long enough for the algorithm to reach convergence or, in other words, for the Markov chain to reach its stationary distribution or the desired posterior distribution of model parameters. The sample size for the MCMC sample is typically determined based on the autocorrelation present in the MCMC sample. The higher the autocorrelation, the larger the MCMC sample should be to achieve the same precision of the parameter estimates as obtained from the chain with low or negligible autocorrelation. Because of the nature of the sampling algorithm, all MCMC exhibit some autocorrelation and thus MCMC samples tend to have large sizes.

The defaults provided by `bayesmh` may not be sufficient for all Bayesian models and data types. You will need to explore the convergence of the MCMC algorithm for your particular data problem and modify the settings, if needed.

After the burn-in period, `bayesmh` includes every iteration in the MCMC sample. You can specify the `thinning(#)` option to store results from a subset of iterations. This option is useful if you want to subsample the chain to decrease autocorrelation in the final MCMC sample. If you use this option, `bayesmh` will perform a total of `thinning() × (mcmcsize() - 1) + 1` iterations, excluding burn-in iterations, to obtain MCMC sample of size `mcmcsize()`.

When you specify the `nchains()` option to produce multiple chains, the `mcmcsize()`, `burnin()`, and `thinning()` options apply to each chain.

## Improving efficiency of the MH algorithm—blocking of parameters

Although the MH algorithm is very general and can be applied to any Bayesian model, it is not the most optimal sampler and may require tuning to achieve higher efficiency.

Efficiency describes mixing properties of the Markov chain. High efficiency means good mixing (low autocorrelation) in the MCMC sample, and low efficiency means bad mixing (high autocorrelation) in the MCMC sample.

An AR is the number of accepted proposals of model parameters relative to the total number of proposals. It should not be confused with sampling efficiency. High AR does not mean high efficiency.

An efficient MH sampler has an AR between 15% and 50% (Roberts and Rosenthal 2001) and low autocorrelation and thus relatively large effective sample size (ESS) for all model parameters.

One way to improve efficiency of the MH algorithm is by blocking of model parameters. Blocking of model parameters is an important functional aspect of the MH sampler. By default, all parameters are used as one block and their covariance matrix is used to adapt the proposal distribution. With many parameters, estimation of this covariance matrix becomes difficult and imprecise and may lead to the loss of efficiency of the MH algorithm. In many cases, this matrix has a block diagonal structure because of independence of some blocks or sets of model parameters and its estimation may be replaced with estimation of the corresponding blocks, which are typically of smaller dimension. This may improve the efficiency of the sampler. To achieve optimal blocking, you need to identify the sets of approximately independent (a posteriori) model parameters and specify them in separate blocks.

To achieve an optimal blocking, you need to know or have some idea about the dependence between the parameters as determined by the posterior distribution. To improve efficiency, follow this principle: correlated parameters should be specified together, while independent groups of parameters should be specified in separate blocks. Because the posterior is usually defined indirectly, the relationship between the parameters is generally unknown. Often, however, we have some knowledge, either deduced from the model specification or based on prior experience with the model, about which parameters are highly correlated. In the worst case, you may need to run some preliminary simulations and determine an optimal blocking by using trial and error.

An ideal case for the MH algorithm is when all model parameters are independent with respect to the posterior distribution and are thus placed in separate blocks and sampled independently. In practice, this is not a realistic or interesting case, but it gives us an idea that we should always try to parameterize the model in such a way that the correlation between model parameters is minimized.

With `bayesmh`, you can use options `block()` to perform blocking. You specify one `block()` option for each set of independent model parameters. Model parameters that are dependent with each other are specified in the same `block()` option.

To illustrate a typical case, consider the following simple linear regression model:

$$y = \{a\} + \{b\} \times x + \epsilon, \epsilon \sim N(0, \{var\})$$

Even when `{a}` and `{b}` have independent prior specifications, the location parameters `{a}` and `{b}` are expected to be correlated a posteriori because of their common dependence on `y`. Alternatively, if the variance parameter `{var}` is independent of `{a}` and `{b}` a priori, it is generally less correlated with the location parameters a posteriori. A good blocking scheme is to use options `block({a} {b})` and `block({var})` with `bayesmh`. We can also reparameterize our model to reduce the correlation

between `{a}` and `{b}` by recentring. To center the slope parameter, we replace `{b}` with `{b} - #`, where `#` is a constant close to the mean of `{b}`. Now `{a}` and `{b} - #` can also be placed in separate blocks. See, for example, [Thompson \(2014\)](#) for more discussion related to model parameterization.

Other options that control MCMC sampling efficiency are `scale()`, `covariance()`, and `adaptation()`; see [Adaptation of the MH algorithm](#) for details.

With multiple chains, the `block()` option and other options that control MCMC sampling efficiency apply to all chains.

## Gibbs and hybrid MH sampling

In [Improving efficiency of the MH algorithm—blocking of parameters](#), we discussed blocking of model parameters as a way of improving efficiency of the MH algorithm. For certain Bayesian models, further improvement is possible by using Gibbs sampling for certain blocks of parameters. This leads to what we call a hybrid MH sampling with Gibbs updates.

Gibbs sampling is the most effective sampling procedure with the maximum possible AR of one and with often very high efficiency. Using Gibbs sampling for some blocks of parameters will typically lead to higher efficiency of the hybrid MH sampling compared with the simple MH sampling.

To apply Gibbs sampling to a set of parameters, we need to know the full conditional distribution for each parameter and be able to generate random samples from it. Usually, the full conditionals are known in various special cases but are not available for general posterior distributions. Thus, Gibbs sampling is not available for all likelihood and prior combinations. `baysmh` provides Gibbs sampling for Bayesian models with conjugate, or more specifically, semiconjugate prior distributions. See [Gibbs sampling for some likelihood-prior and prior-hyperprior configurations](#) for a list of supported models.

For a supported semiconjugate model, you can request Gibbs sampling for a block of parameters by specifying the `gibbs` suboption within option `block()`. In some cases, the `gibbs` suboption may be used in all parameter blocks, in which case we will have full Gibbs sampling.

To use Gibbs sampling for a set of parameters, you must first place them in separate `prior()` statements and specify semiconjugate prior distributions and then place them in a separate block and include the `gibbs` suboption, `block(..., gibbs)`.

Here is a standard application of a full Gibbs sampling to a normal mean-only model. Under the normal–inverse-gamma prior, the conditional posterior distributions of the mean parameter is normal and of the variance parameter is inverse gamma.

```
. baysmh y, likelihood(normal({var}))
>      prior({y:_cons}, normal(1,10))
>      prior({var}, igamma(10,1))
>      block({y:_cons}, gibbs)
>      block({var}, gibbs)
```

Because `{y:_cons}` and `{var}` are approximately independent a posteriori, we specified them in separate blocks.

Gibbs sampling can be applied to hyperparameters, which are not directly involved in the likelihood specification of the model. For example, we can use Gibbs sampling for the covariance matrix of regression coefficients.

```
. baysmh y x, likelihood(normal(var))
>      prior(var, igamma(10,1))
>      prior({y:_cons x}, mvnormal(2,1,0,{Sigma,m}))
>      prior({Sigma,m}, iwishart(2,10,V))
>      block({Sigma,m}, gibbs)
```



In the next example, the matrix parameter `{Sigma,m}` specifies the covariance matrix in the multivariate normal prior for a pair of model parameters, `{y:1.cat}` and `{y:2.cat}`. `{Sigma,m}` is a hyperparameter—it is not a model parameter of the likelihood but a parameter of a prior distribution, and it has an inverse-Wishart hyperprior distribution, which is a semiconjugate prior with respect to the multivariate normal prior distribution. Therefore, we can request a Gibbs sampler for `{Sigma,m}`.

```
bayesmh y x i.cat, likelihood(probit)
>      prior(y:x _cons,      normal(0, 1000))
>      prior(y:1.cat 2.cat, mvnormal0(2,{Sigma,m}))
>      prior({Sigma,m}, iwishart(2,10,V))
>      block({Sigma,m}, gibbs)
```

In general, Gibbs sampling, when available, is useful for covariance matrices because MH sampling has low efficiency for sampling positive-definite symmetric matrices. In a multivariate normal regression, the inverse Wishart distribution is a conjugate prior for the covariance matrix and thus inverse Wishart is the most common prior specification for a covariance matrix parameter. If an inverse-Wishart prior (`iwishart()`) is used for a covariance matrix, you can specify Gibbs sampling for the covariance matrix. You can do so by placing the matrix in a separate block and specifying the `gibbs` suboption in that block, as we showed above. Using Gibbs sampling for the covariance matrix usually greatly improves the sampling efficiency.

## Adaptation of the MH algorithm

The MH algorithm simulates Markov chains by generating small moves or jumps from the current parameter values (or current state) according to the proposal distribution. At each iteration of the algorithm, the proposed new state is accepted with a probability that is calculated based on the newly proposed state and the current state. The choice of a proposal distribution is crucial for the mixing properties of the Markov chain, that is, the rate at which the chain explores its stationary distribution. (In a Bayesian context, a Markov chain state is a vector of model parameters, and a stationary distribution is the target posterior distribution.) If the jumps are too small, almost all moves will be accepted. If the jumps are too large, almost all moves will be rejected. Either case will cause the chain to explore the entire posterior domain slowly and will thus lead to poor mixing. Adaptive MH algorithms try to tune the proposal distribution so that some optimal AR is achieved ([Haario, Saksman, and Tamminen \[2001\]](#); [Roberts and Rosenthal \[2009\]](#); [Andrieu and Thoms \[2008\]](#)).

In the random-walk MH algorithm, the proposal distribution is a Gaussian distribution with a zero mean and is completely determined by its covariance matrix. It is useful to represent the proposal covariance matrix as a product of a (scalar) scale factor and a positive-definite scale matrix. [Gelman, Gilks, and Roberts \(1997\)](#) show that the optimal scale matrix is the true covariance matrix of the target distribution, and the optimal scale factor is inversely proportional to the number of parameters. Therefore, in the ideal case when the true covariance matrix is available, it can be used as a proposal covariance and an MCMC adaptation can be avoided altogether. In practice, the true covariance is rarely known and the adaptation is thus unavoidable.

In the `bayesmh` command, the scale factor and the scale matrix that form the proposal covariance are constantly tuned during the adaptation phase of an MCMC so that the current AR approaches some predefined value.

You can use `scale()`, `covariance()`, and `adaptation()` options to control adaptation of the MH algorithm. The TAR is controlled by option `adaptation(tarate())`. The initial scale factor and scale matrix can be modified using the `scale()` and `covariance()` options. In the presence of blocks of parameters, these options can be specified separately for each block within the `block()` option. At each adaptation step, a new scale matrix is formed as a mixture (a linear combination) of the previous scale matrix and the current empirical covariance matrix of model parameters. The mixture of the two matrices

is controlled by option `adaptation(beta())`. A positive `adaptation(beta())` is recommended to have a more stable scale matrix between adaptation periods. The adaptation lasts until the maximum number `adaptation(every()) × adaptation(maxiter())` of adaptive iterations is reached or until `adaptation(tarate())` is reached within the `adaptation(tolerance())` limit. The default for `maxiter()` depends on the specified burn-in and `adaptation(every())` and is computed as `max{25, floor(burnin()/adaptation(every()))}`. The default for `adaptation(every())` is 100. If you change the default values of these parameters, you may want to increase the `burnin()` to be as long as the specified adaptation period so that adaptation is finished before the final simulated sample is obtained. (There are adaptation regimes in which adaptation is performed during the simulation phase as well, such as continuous adaptation.) Two additional adaptation options, `adaptation(alpha())` and `adaptation(gamma())` control the AR and the adaptation rate. For a detailed description of the adaptation process, see [Adaptive random-walk Metropolis–Hastings](#) in [BAYES] [Intro](#) and [Adaptive MH algorithm](#) in *Methods and formulas*.

With multiple chains, adaptation options apply to all chains.

## Specifying initial values

When exploring convergence of MCMC, it may be useful to try different initial values to verify that the convergence is unaffected by starting values. Using different initial values is also essential for multiple chains. We first describe how to specify initial values for a single chain and later for multiple chains.

**Single chain.** There are two different ways to specify initial values of model parameters in `bayesmh` for a single chain. First is by specifying an initial value when declaring a model parameter. Second is by specifying an initial value in the `initial()` option. Initial values for matrix model parameters may be specified only in the `initial()` option.

For example, below we initialize variance parameter `{var}` with value of 1 using two equivalent ways, as follows:

```
. bayesmh y x, likelihood(normal({var}=1)) ...
```

or

```
. bayesmh y x, likelihood(normal({var})) initial({var} 1) ...
```

If both initial-value specifications are used, initial values specified in the `initial()` option override any initial values specified during parameter declaration for the corresponding parameters.

You can initialize multiple parameters with the same value by supplying a list of parameters by using any of the specifications described in [Referring to model parameters](#) to `initial()`. For example, to initialize all regression coefficients from equations `y1` and `y2` to zero, you can type

```
. bayesmh ..., initial({y1:} {y2:} 0) ...
```

Stata expressions that evaluate to a number can also be used to specify initial values for scalar parameters. One particularly useful application of this is specifying random initial values using Stata's random-number functions; see [FN] [Random-number functions](#). For example, we can generate random initial values for parameters `{y1:}` from a normal distribution with mean 0 and standard deviation 10 and for parameters `{y2:}` from a uniform on (0, 1) distribution as follows:

```
. bayesmh ..., initial({y1:} rnormal(0,10) {y2:} runiform(0,1)) ...
```

You may also specify the `initrando` option to request random initial values for all model parameters. In that case, initial values are generated from the prior distributions of the parameters, except for parameters that are assigned `flat`, `jeffreys`, `density()`, `logdensity()`, or `jeffreys()`

prior distributions. For such parameters, you must specify your own initial values, or **bayesmh** will issue an error message.

**Multiple chains.** In the presence of multiple chains, you can use the `init#()` options to specify initial values for each chain: the `init1()` option specifies initial values for the first chain, `init2()` for the second chain, and so on. You specify initial values within the `init#()` options just like you do this within `initial()` for a single chain. (With multiple chains, `initial()` is synonymous to `init1()`.)

For example,

```
. bayesmh y x, likelihood(normal({var})) nchains(2) init1({var} 1) init2({var} 10) ...
```

You can use the `initall()` option to specify initial values for all chains. This is useful, for instance, when you want to generate random initial values from the same distribution for all chains. You should avoid specifying fixed initial values within `initall()` because then all chains will use the same starting values.

**Default initial values.** By default, if no initial value is specified and option `nomleinitial` is not used, **bayesmh** uses MLEs, whenever available, as starting values for model parameters for a single chain.

For example, for the previous regression model, **bayesmh** uses regression coefficients and mean squared error from linear regression `regress y x` as the respective starting values for the regression model parameters and variance parameter `{var}`.

If MLE is not available and an initial value is not provided, then a scalar model parameter is initialized with 1 for positive parameters and 0 for other parameters, and a matrix model parameter is initialized with an identity matrix. Note, however, that this default initialization is not guaranteed to correspond to the feasible state for the specified posterior model; that is, posterior probability of the initial state can be 0. When initial values are not feasible, **bayesmh** makes 500 random attempts to find a feasible initial-value vector. An initial-value vector is feasible if it corresponds to a state with positive posterior probability. If feasible initial values are not found after 500 attempts, **bayesmh** will issue the following error:

```
could not find feasible initial state
r(498);
```

You may use the `search()` option to modify the default settings for finding feasible initial values.

In the presence of multiple chains, each chain uses a different set of initial values for model parameters. The above description of default initial values applies to the first chain only. The subsequent chains use random initial values, which generally are generated from the prior distributions.

For improper priors `flat`, `jeffreys`, and `jeffreys(#)`, **bayesmh** cannot draw random initial values directly from these priors. Doing so would typically produce extreme values for model parameters for which log likelihood would be missing. Instead, the command generates initial values from a normal distribution centered at the initial values of the first chain with standard deviations proportional to the magnitudes of the respective initial estimates. This approach is also used to generate default initial values with user-defined priors `density()` and `logdensity()`.

Random initial values may not always be feasible. Extreme values may be produced for model parameters for some prior distributions, which may lead to missing log-likelihood values. **bayesmh** will attempt to generate several different sets of initial values before terminating the simulation of a particular chain and issuing a warning message. In this case, you must specify your own initial values for that chain.

Default initial values are provided for convenience! To detect nonconvergence, **overdispersed initial values** should be used with multiple chains. Randomly generated default initial values are not

guaranteed to produce overdispersed initial values for all chains. To fully explore convergence, we recommend that you specify your own initial values with multiple chains, especially with improper or noninformative priors.

See *Convergence diagnostics using multiple chains* for an example of specifying initial values with multiple chains.

You can use the `initsummary` option to see the initial values used for simulation. The initial values are also stored in the `e(init)` matrix after estimation.

## Summarizing and reporting results

As we discussed in *Checking model specification*, it is useful to verify the details about your model specification before estimation. The `dryrun` model will display the model summary without estimation. Once you are satisfied with the model specification, you can use the `nomodelsummary` option during estimation to suppress a potentially long model summary from the final output.

In the presence of blocking, you may also display the information about specified blocks by using the `blocksummary` option.

Simulation may be time consuming for large datasets and for models with many parameters. You can specify one of `dots` or `dots(#)` option to display a dot every `#` iterations to see the simulation progress.

You can also use the `initsummary` option to see the initial values used in the simulation, which may be useful with multiple chains.

## Posterior summaries and credible intervals

After simulation, `bayesmh` reports various summaries about the model parameters in the output table. The summaries include posterior mean and median estimates, estimates of posterior standard deviation and MCSE, and credible intervals. By default, 95% equal-tailed credible intervals are reported. You can use the `hpd` option to request HPD intervals instead. You can also use the `clevel()` option to change the default credible level.

`bayesmh` provides two estimators for MCSE: one using ESS and one using batch means. The ESS-based estimator is the default. You can request the batch-means estimator by specifying the `batch()` option. Options `corrlag()` and `corrto1()` affect how ESS is estimated when computing MCSE; see *Methods and formulas* in [BAYES] `bayesstats summary` for details.

In the presence of multiple chains, all chains are used to produce posterior summaries. You can use `bayesstats summary`'s `sepchains` option to see the results for each chain separately. Also, the reported acceptance rate, efficiencies, and log marginal-likelihood are averaged over all chains. You can use the `chainsdetail` option to see these simulation summaries for each chain.

## Saving MCMC results

In addition to postestimation summaries, `bayesmh` saves simulation results containing MCMC samples for all model parameters to a temporary Stata dataset. You can use the `saving()` option to save simulation results to a permanent dataset. In fact, if you want to store your estimation results in memory or save them to a disk, you must specify the `saving()` option with `bayesmh`; see *Storing estimation results after Bayesian estimation* in [BAYES] `Bayesian postestimation`. You can also specify the `saving()` option on replay.

```
. bayesmh, saving(...)
```

By default, all model parameters are saved in the dataset. If desired, you can exclude some of the parameters from the dataset by specifying the `exclude()` option. Beware that you will not be able to obtain posterior summaries for these parameters or use them in any way in your analysis, because no simulation results will be available for them. Also, the Laplace–Metropolis approximation for the log marginal-likelihood will not be available because its computation requires simulation results for all model parameters.

## Convergence of MCMC

As we discuss in *Convergence diagnostics of MCMC* in [BAYES] [Intro](#), checking convergence is an essential step of any MCMC simulation. Bayesian inference based on an MCMC sample is only valid if the Markov chain has converged and the sample is drawn from the desired posterior distribution. It is important to emphasize that we need to verify the convergence for all model parameters and not only for a subset of parameters of interest. Another difficulty in assessing convergence of MCMC is the lack of a single conclusive convergence criterion. The diagnostic usually involves checking for several necessary (but not necessarily sufficient) conditions for convergence. In general, the more aspects of the MCMC sample you inspect, the more reliable your results are.

An MCMC is said to have converged if it reached its stationary distribution. In the Bayesian context, the stationary distribution is the true posterior distribution of model parameters. Provided that the considered Bayesian model is well specified (that is, it defines a proper posterior distribution of model parameters), the convergence of MCMC is determined by the properties of its sampling algorithm.

The main component of the MH algorithm, or any MCMC algorithm, is the number of iterations it takes for the chain to approach its stationary distribution or for the MCMC sample to become representative of a sample from the true posterior distribution of model parameters. The period during which the chain is converging to its stationary distribution from its initial state is called the burn-in period. The iterations of the burn-in period are discarded from the MCMC sample used for analysis. Another complication is that adjacent observations from the MCMC sample tend to be positively correlated; that is, autocorrelation is typically present in MCMC samples. In theory, this should not be a problem provided that the MCMC sample size is sufficiently large. In practice, the autocorrelation in the MCMC sample may be so high that obtaining a sample of the necessary size becomes infeasible and finding ways to reduce autocorrelation becomes important.

Two aspects of the MH algorithm that affect the length of the burn-in (and convergence) are the starting values of model parameters or, in other words, a starting state and a proposal distribution. `bayesmh` has the default burn-in of 2,500 iterations, but you can change it by specifying the `burnin()` option. `bayesmh` uses a Gaussian normal distribution with a zero mean and a covariance matrix that is updated with current sample values during the adaptation period. You can control the proposal distribution by changing the initial scale factor in option `scale()` and an initial scale matrix in option `covariance()`; see *Adaptation of the MH algorithm*.

For the starting values of a single chain, `bayesmh` uses MLEs whenever available, but you can specify your own initial values in option `initial()`; see *Specifying initial values*. Good initial values help to achieve fast convergence of MCMC and bad initial values may slow convergence down. A common approach for eliminating the dependence of the chain on the initial values is to discard an initial part of the simulated sample: a burn-in period. The burn-in period must be sufficiently large for a chain to “forget” its initial state and approach its stationary distribution or the desired posterior distribution.

There are some researchers (for example, [Geyer \[2011\]](#)) who advocate that any starting point in the posterior domain is equally good and there should be no burn-in. While this is a sensible approach for a fixed, nonadaptive MH algorithm, it may not be as sensible for an adaptive MH algorithm because the proposal distribution is changing (possibly drastically) during the adaptation period. Therefore,

adaptive iterations are better discarded from the analysis MCMC sample and thus it is recommended that the burn-in period is at least as long as the adaptation period. (There are adaptive regimes such as continuous adaptation in which adaptation continues after the burn-in period as well.)

In addition to fast convergence, an “ideal” MCMC chain will also have good mixing (or low autocorrelation). A good mixing can be viewed as a rapid movement of the chain around the parameter space. High autocorrelation in MCMC and consequently low efficiencies are usually indications of bad mixing. To improve the mixing of the chain, you may need to improve the efficiency of the algorithm (see [Improving efficiency of the MH algorithm—blocking of parameters](#)) or sometimes reparameterize your model. In the presence of high autocorrelation, you may also consider subsampling or thinning the chain, option `thinning()`, to reduce autocorrelation, but this may not always be the best approach.

Even when the chain appears to have converged and has good mixing, you may still have a case of pseudoconvergence, which is common for multimodal posterior distributions. Specifying different sets of initial values may help detect pseudoconvergence.

Multiple chains are often used to assess the convergence of MCMC; see [Convergence diagnostics using multiple chains](#) and Balov (2016c). For more information about convergence of MCMC and its diagnostics, see [Convergence diagnostics of MCMC](#) in [BAYES] [Intro](#), [BAYES] [bayesgraph](#), [BAYES] [bayesstats ess](#), and [BAYES] [bayesstats grubin](#).

In what follows, we concentrate on demonstrating various specifications of `bayesmh`, which may not always correspond to the optimal Bayesian analysis for the considered problem. In addition, although we skip checking convergence for some of our models to keep the exposition short, it is important that you always check the convergence of all parameters in your model in your analysis before you make any inferential conclusions. If you are also interested in any functions of model parameters, you must check convergence of those functions as well.

## Video examples

[Introduction to Bayesian statistics, part 1: The basic concepts](#)

[Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm](#)

## Getting started examples

We will use the familiar `auto.dta` for our introductory examples. This dataset contains information about 74 automobiles, including their make and model, price, and mileage (variable `mpg`). In our examples, we are interested in estimating the average fuel efficiency as measured by the `mpg` variable and its relationship with other automobile characteristics such as `weight`.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
```

```
. describe mpg weight length
```

variable name	storage type	display format	value label	variable label
<code>mpg</code>	int	<code>%8.0g</code>		Mileage (mpg)
<code>weight</code>	int	<code>%8.0gc</code>		Weight (lbs.)
<code>length</code>	int	<code>%8.0g</code>		Length (in.)

Mean of a normal distribution with a known variance

We start with an example of estimating the mean of a normal distribution with known variance. This corresponds to a constant-only normal linear regression with an unknown constant (or intercept) and a known error variance.

Suppose we are interested in estimating the average fuel efficiency as measured by the mpg variable. For illustration purposes, let’s assume that mpg is normally distributed. We are interested in estimating its mean. Let’s also assume that we know the variance of mpg and it is 36.

► Example 1: Noninformative prior for the mean when variance is known

To fit a Bayesian model, we must specify the likelihood model and priors for all model parameters. We have only one parameter in this model—the constant (or the mean) of mpg. We first consider a noninformative prior for the constant: the prior distribution with a density equal to one.

To specify this model in bayesmh, we use the likelihood specification mpg, likelihood(normal(36)) and the prior specification prior({mpg:\_cons}, flat), where suboption flat requests a flat prior distribution with the density equal to one. This prior is an improper prior for the constant—the prior distribution does not integrate to one. {mpg:\_cons}, the constant or the mean of mpg, is the only model parameter and is declared automatically by bayesmh as a part of the regression function. (For this reason, we also did not need to specify the mean of the normal() distribution in the likelihood specification.) All other simulation and reporting options are left at default.

Because bayesmh uses MCMC sampling, a stochastic procedure, to obtain results, we specify a random-number seed (for example, 14) for reproducibility of results.

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, flat)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ 1 (flat)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4161
	Efficiency =	.2292

---

Log marginal-likelihood = -233.96144

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29812	.703431	.014693	21.28049	19.93155	22.69867

---

bayesmh first reports the summary of the model. The likelihood model specified for mpg is normal with mean {mpg:\_cons} and fixed variance of 36. The prior for {mpg:\_cons} is flat or completely noninformative.

Our model is very simple, so its summary is very short. For other models, the model summary may get very long. You can use the nomodelsummary option to suppress it from the output.



It is useful, however, to review the model summary before estimation for models with many parameters and complicated specifications. You can use the `dryrun` option to see the model summary without estimation. Once you verified the correctness of your model specification, you can specify `nomodelsummary` during estimation.

Next, `bayesmh` reports the header including the title for the fitted model, the used MCMC algorithm, and various numerical summaries of the sampling procedure. `bayesmh` performed 12,500 MCMC iterations, of which 2,500 were discarded as burn-in iterations and the next 10,000 iterations were kept in the final MCMC sample. An overall AR is 0.42, meaning that 42% out of 10,000 proposal parameter values were excepted by the algorithm. This is a good AR for the MH algorithm. Values below 10% may be a cause for concern and may indicate problems with convergence of MCMC. Very low ARs may also mean high autocorrelation. The efficiency is 0.23 and is also considered good for the MH algorithm. Efficiencies below 1% should be investigated further and would require further tuning of the algorithm and possibly revisiting the considered model.

Finally, `bayesmh` reports an estimation table that includes the posterior mean, posterior standard deviation, MCMC standard error (MCSE), posterior median, and the 95% credible interval.

The estimated posterior mean for `{mpg:_cons}` is 21.298 with a posterior standard deviation of 0.70. The efficiency of the estimator of the posterior mean is about 23%, which is relatively high for the random-walk MH sampling. In general, you should expect to see lower efficiencies from this algorithm for models with more parameters. The MCSE, which is an approximation of the error in estimating the true posterior mean, is about 0.015. Therefore, provided that the MCMC simulation has converged, the posterior mean of the constant is accurate to 1 decimal position, 21.3. If you want an estimation precision of, say, 2 decimal positions, you may need to increase the MCMC sample size  $10^1$  times; that is, use `mcmsize(100000)`.

The estimated posterior mean and medians are very close, suggesting that the posterior distribution of `{mpg:_cons}` may be symmetric. In fact, the posterior distribution of a mean in this model is known to be a normal distribution.

According to the reported 95% credible interval, the probability that the mean of `mpg` is between 19.9 and 22.7 is about 0.95. You can use the `clevel()` option to change the default credible level; also see [\[BAYES\] set clevel](#).

Because we used a completely noninformative prior, our results should be the same as frequentist results. In this Bayesian model, the posterior distribution of the constant parameter is known to be normal with a mean equal to the sample average. In the frequentist domain, the MLE of the constant is also the sample average, so the posterior mean estimate and the MLE should be the same in this model.

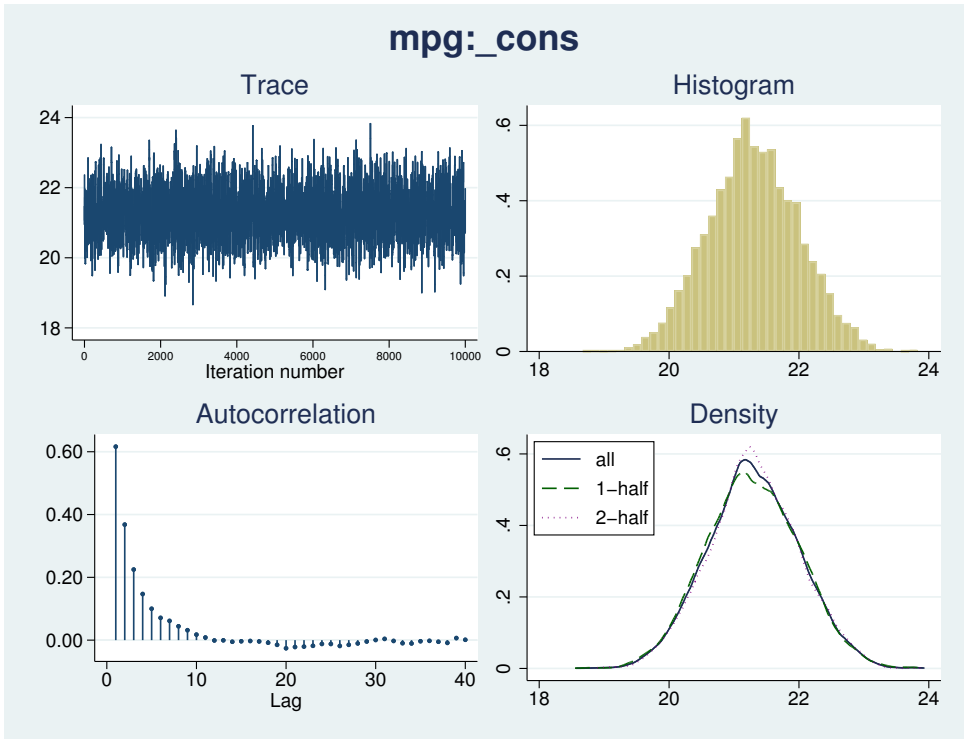
The sample average of `mpg` is 21.2973. Our posterior mean estimate is 21.298, which is very close. The reason it is not exactly the same is because we estimated the posterior mean of the constant based on an MCMC sample simulated from its posterior distribution instead of using the known formula. Closed-form expressions for posterior mean estimators are available only for some Bayesian models. In general, posterior distributions of parameters are unknown and posterior summaries may only be estimated from the MCMC samples of parameters.

In practice, we must verify the convergence of MCMC before making any inferential conclusions about the obtained results.



We start by looking at various graphical diagnostics as produced by `bayesgraph` diagnostics.

```
. bayesgraph diagnostics {mpg:_cons}
```



The trace plot represents a “perfect” trace plot. It does not exhibit any trends, and it traverses the distribution quickly. The chain is centered around 21.3, but also explores the portions of the distribution where the density is low, which is indicative of good mixing of the chain. The autocorrelation dies off very quickly. The posterior distribution looks normal. The kernel density estimates based on the first and second halves of the sample are very similar to each other and are close to the overall density estimate. We can see that MCMC converged and mixes well. See [BAYES] [bayesgraph](#) for details about this command.

See [Convergence diagnostics using multiple chains](#) for an example of using multiple chains to assess convergence. Also see [Convergence diagnostics of MCMC](#) for more discussion about convergence of MCMC.

◀

## ► Example 2: Informative prior for the mean when variance is known

In [example 1](#), we used a noninformative prior for `{mpg:_cons}`. Here, we consider a conjugate normal prior for `{mpg:_cons}`. A parameter is said to have a conjugate prior when the corresponding posterior belongs to the same family as the prior. In our example, if we assume a normal prior for the constant, its posterior is known to be normal too.

Suppose that based on previous studies, the distribution of the mean mileage was found to be normal with mean of 25 and variance of 10. We change the flat prior in `bayesmh`'s `prior()` option from [example 1](#) with `normal(25,10)`.

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(25,10))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(25,10)
```

---

```
Bayesian normal regression          MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =       2,500
                                          MCMC sample size =    10,000
                                          Number of obs  =       74
                                          Acceptance rate =     .4169
Log marginal-likelihood = -236.71627      Efficiency    =     .2293
```

---

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_cons	21.47952	.6820238	.014243	21.47745	20.13141	22.82153

Compared with [example 1](#), our results change only slightly: the estimated posterior mean is 21.48 with a posterior standard deviation of 0.68. The 95% credible interval is [20.1, 22.82].

The reason we obtained such similar results is that our specified prior is in close agreement with what we observed in this sample. The prior mean of 25 with a standard deviation of  $\sqrt{10} = 3.16$  overlaps greatly with what we observe for `{mpg:_cons}` in the data.

If we place a very strong prior on the value for the mean by, for example, substantially decreasing the variance of the normal prior distribution,

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(25,0.1))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(25,0.1)
```

---

Bayesian normal regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.4194
	Efficiency	=	.2352

---

```
Log marginal-likelihood = -246.2939
```

---

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_cons	24.37211	.292777	.006037	24.36588	23.79701	24.94403

we obtain very different results. Now the posterior mean and standard deviation estimates are very close to their prior values, as one would expect with such strong prior information.

Which results are correct? The answer depends on how confident we are in our prior knowledge. If we previously observed many samples in which the average mileage for the considered population of cars was essentially 25, our last results are consistent with this and the information about the mean of {mpg:\_cons} contained in the observed sample was not enough to counteract our belief. If, on the other hand, we had no prior information about the mean mileage, then we would use a noninformative or mildly informative prior in our Bayesian analysis. Also, if we believe that our observed data should have more weight in our analysis, we would not specify a very strong prior.



➤ Example 3: Noninformative normal prior for the mean when variance is known

In [example 1](#), we used a completely noninformative, flat prior for {mpg:\_cons}. In [example 2](#), we considered a conjugate normal prior for {mpg:\_cons}. We also saw that by varying the variance of the normal prior distribution, we could control the “informativeness” of our prior. The larger the variance, the less informative the prior. In fact, if we let the variance approach infinity, we will arrive at the same posterior distribution of the constant as with the flat prior.

For example, if we specify a very large variance in the normal prior,

```
. set seed 14
. bayesmh mpg, likelihood(normal(36)) prior({mpg:_cons}, normal(0,1000000))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},36)
Prior:
  {mpg:_cons} ~ normal(0,1000000)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4161
	Efficiency =	.2292

---

Log marginal-likelihood = -241.78836						
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29812	.7034313	.014693	21.28049	19.93155	22.69868

we will obtain results that are very similar to the results from [example 1](#) with the flat prior.

We do not need to use such an extreme value of the variance for the results to become less sensitive to the prior specification. As we saw in [example 2](#), using the variance of 10 in that example resulted in very little impact of the prior on the results.



Mean of a normal distribution with an unknown variance

Let’s now consider the case where both mean and variance of the normal distribution are unknown.

➤ Example 4: Noninformative Jeffreys prior when mean and variance are unknown

A noninformative prior commonly used for the normal model with unknown mean and variance is the Jeffreys prior, under which the prior for the mean is flat and the prior for the variance is the reciprocal of the variance. We use the same flat prior for {mpg:\_cons} as in [example 1](#) and specify the jeffreys prior for {var} using a separate prior() statement.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2668
	Efficiency: min =	.09718
	avg =	.1021
	max =	.1071

Log marginal-likelihood =	-234.645
---------------------------	----------

---

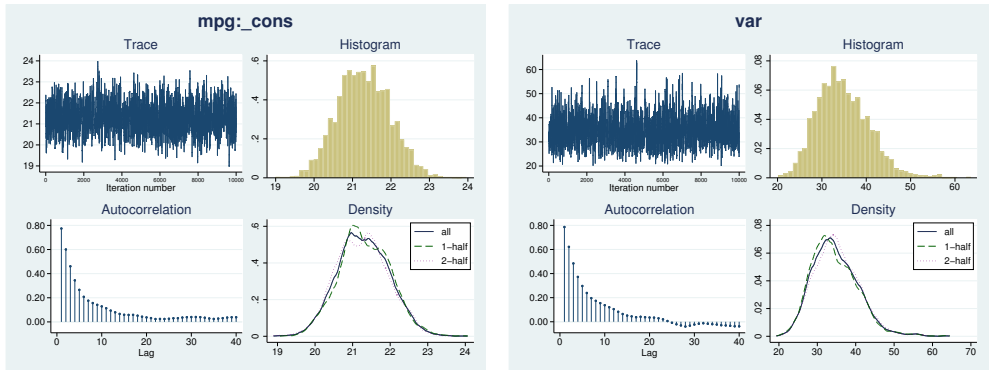
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

Because we used a noninformative prior, our results should be similar to the frequentist results apart from simulation uncertainty. Compared with [example 1](#), the average efficiency of the MH algorithm decreased to 10%, as is expected with more parameters, but is still considered a good efficiency for the MH algorithm.

The posterior mean estimate of {mpg:\_cons} is close to the OLS estimate of 21.297, and the posterior standard deviation is close to the standard error of the OLS estimate 0.673. MCSE is slightly larger than in [example 1](#) because we have lower efficiency. If we wanted to make MCSE smaller, we could increase our MCMC sample size. The posterior mean estimate of {var} agrees with the MLE of the variance 33.02, but we would not expect the two to be necessarily the same. We estimated the posterior mean of {var}, not the posterior mode, and because posterior distribution of {var} is not symmetric, the two estimates may not be the same.

Again, as with any MCMC analysis, we must verify the convergence of our MCMC sample before we can trust our results.

```
. bayesgraph diagnostics _all
```



Graphical diagnostic plots do not show any signs of nonconvergence for either of the parameters. We can also check convergence more formally using multiple chains; see [\[BAYES\] bayesstats grubin](#) and [Convergence diagnostics using multiple chains](#).

Recall that to access convergence of MCMC, we must explore convergence for all model parameters. ◀

### ► Example 5: Informative conjugate prior when mean and variance are unknown

For a normal distribution with unknown mean and variance, the informative conjugate prior is a normal prior for the mean and an inverse-gamma prior for the variance. Specifically, if  $y \sim N(\mu, \sigma^2)$ , then the informative conjugate prior for the parameters is

$$\begin{aligned}\mu | \sigma^2 &\sim N(\mu_0, \sigma^2) \\ \sigma^2 &\sim \text{InvGamma}(\nu_0/2, \nu_0 \sigma_0^2/2)\end{aligned}$$

where  $\mu_0$  is the prior mean of the normal distribution and  $\nu_0$  and  $\sigma_0^2$  are the prior degrees of freedom and prior variance for the inverse-gamma distribution. Let's assume  $\mu_0 = 25$ ,  $\nu_0 = 10$ , and  $\sigma_0^2 = 30$ .

Notice that in the specification of the prior for {mpg:\_cons}, we specify the parameter {var} as the variance of the normal distribution. We use `igamma(5,150)` as the prior for the variance parameter {var}.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(25,{var}))
> prior({var}, igamma(5,150))
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ normal(25,{var})
  {var} ~ igamma(5,150)
```

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1971
	Efficiency: min =	.09822
	avg =	.09923
	max =	.1002
Log marginal-likelihood = -237.77006		

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.314	.6639278	.02097	21.29516	20.08292	22.63049
var	33.54699	5.382861	.171756	32.77635	24.88107	46.0248

Compared with [example 4](#), the variance is slightly smaller, but the results are still very similar. ◀

► **Example 6: Noninformative inverse-gamma prior when mean and variance are unknown**

The Jeffreys prior for the variance from [example 4](#) can be viewed as a limiting case of an inverse-gamma distribution with the degrees of freedom approaching zero.

Indeed, if we replace the `jeffreys` prior in [example 4](#) with an inverse-gamma distribution with very small degrees of freedom,

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat)
> prior({var}, igamma(0.0001,0.0001))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ igamma(0.0001,0.0001)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2668
	Efficiency: min =	.09718
	avg =	.1021
	max =	.1071

Log marginal-likelihood = -243.85656

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
mpg					
_cons	21.29223	.6828811	.021905	21.27899	19.99154 22.61903
var	34.76569	5.915305	.180753	34.18389	24.91294 47.61275

we obtain results that are very close to the results from [example 4](#).

◀

## Simple linear regression

In this example, we consider a simple linear regression with one independent variable. We continue with `auto.dta`, but this time we regress `mpg` on a rescaled covariate `weight`.

```
. use https://www.stata-press.com/data/r16/auto
. replace weight = weight/100
variable weight was int now float
(74 real changes made)
```

We will have three model parameters: the slope and the intercept for the linear predictor and the variance parameter for the error term. Regression parameters, `{mpg:weight}` and `{mpg:_cons}`, will be declared implicitly by `bayesmh`, but we will need to explicitly specify the variance parameter `{var}`. We will also need to assign appropriate priors for all parameters.



➤ **Example 7: Noninformative prior for regression coefficients and variance**

As in our earlier examples, we start with a noninformative prior. For this model, a common noninformative prior for the parameters includes flat priors for {mpg:weight} and {mpg:\_cons} and a Jeffreys prior for {var}.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)
  {var} ~ jeffreys
```

---

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1768
	Efficiency: min =	.04557
	avg =	.06624
	max =	.07961

Log marginal-likelihood = -198.14389

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.6019838	.0512557	.001817	-.6018433	-.7015638	-.5021532
	_cons	39.47227	1.589082	.058601	39.49735	36.26465	42.43594
	var	12.22248	2.214665	.10374	11.92058	8.899955	17.47372

Our model summary shows the likelihood model for mpg, flat priors for the two regression coefficients, and a Jeffreys prior for the variance parameter. Now that we have a covariate in the model, the mean of the normal distribution is labeled as xb\_mpg to emphasize that it is now a linear combination of independent variables. Regression coefficients involved in the linear predictor are marked with (1) on the right.

The results are again very similar to the frequentist results. Posterior mean estimates of the coefficients are very similar to the OLS estimates obtained by using regress below. Posterior standard deviations are similar to the standard errors from regress.

```
. regress mpg weight
```

Source	SS	df	MS	Number of obs	=	74
Model	1591.99021	1	1591.99021	F(1, 72)	=	134.62
Residual	851.469254	72	11.8259619	Prob > F	=	0.0000
				R-squared	=	0.6515
				Adj R-squared	=	0.6467
Total	2443.45946	73	33.4720474	Root MSE	=	3.4389

mpg	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
weight	-.6008687	.0517878	-11.60	0.000	-.7041058	-.4976315
_cons	39.44028	1.614003	24.44	0.000	36.22283	42.65774

◀

### ► Example 8: Conjugate prior for regression coefficients and variance

In this example, we use a conjugate prior for the parameters, which corresponds to normal priors for {mpg:weight} and {mpg:\_cons} and an inverse-gamma prior for {var},

$$\begin{aligned}\beta_{\text{weight}}|\sigma^2 &\sim N(\mu_{\text{weight}}, \sigma^2) \\ \beta_{\text{cons}}|\sigma^2 &\sim N(\mu_{\text{cons}}, \sigma^2) \\ \sigma^2 &\sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)\end{aligned}$$

where regression coefficients have different means but equal variances.  $\mu_{\text{weight}}$  and  $\mu_{\text{cons}}$  are the prior means of the normal distributions, and  $\nu_0$  and  $\sigma_0^2$  are the prior degrees of freedom and prior variance for the inverse-gamma distribution. Let's assume  $\mu_{\text{weight}} = -0.5$ ,  $\mu_{\text{cons}} = 40$ ,  $\nu_0 = 10$ , and  $\sigma_0^2 = 10$ .

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:weight}, normal(-0.5,{var}))
> prior({mpg:_cons}, normal(40,{var}))
> prior({var}, igamma(5,50))
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:weight} ~ normal(-0.5,{var}) (1)  
{mpg:\_cons} ~ normal(40,{var}) (1)  
{var} ~ igamma(5,50)

---

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1953
	Efficiency: min =	.05953
	avg =	.06394
	max =	.06932

Log marginal-likelihood = -202.74075

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.6074375	.0480685	.001916	-.6078379	-.6991818	-.5119767
	_cons	39.65274	1.499741	.05696	39.63501	36.59486	42.47547
	var	11.696	1.929562	.079083	11.52554	8.570938	16.26954

For this mildly informative prior, our regression coefficients are still very similar to the results obtained using the noninformative prior in [example 7](#), but the variance estimate is slightly smaller.

► Example 9: Zellner’s *g* prior for regression coefficients

In [example 8](#), we assumed that {mpg:weight} and {mpg:\_cons} are independent a priori. We can specify Zellner’s *g* prior ([Zellner 1986](#)), often used for regression coefficients in a multiple regression, which allows correlation between the regression coefficients.

The prior for the coefficients can be written as

$$\beta|\sigma^2 \sim \text{MVN}(\mu_0, g\sigma^2(X'X)^{-1})$$

where  $\beta$  is a vector of coefficients,  $\mu_0$  is the vector of prior means,  $g$  is the prior degrees of freedom, and  $X$  is the design matrix. Let’s, for example, use  $g = 30$  and  $\mu_0 = (\mu_{\text{weight}}, \mu_{\text{cons}}) = (-0.5, 40)$ . Zellner’s *g* prior is not strictly a conventional Bayesian prior because it depends on the data.

In `bayesmh`, we can use `prior zellnersg()` to specify this prior. The first argument for this prior is the dimension (2), the second argument is the degrees of freedom (30), the next parameters are prior means (-0.5 and 40), and the last parameter is the name of the parameter corresponding to the variance term ({var}).

```

. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, zellnersg(2,30,-0.5,40,{var}))
> prior({var}, igamma(5,50))
Burn-in ...
Simulation ...
Model summary

```

---

```

Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ zellnersg(2,30,-0.5,40,{var})
  {var} ~ igamma(5,50)

```

---

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2576
	Efficiency: min =	.05636
	avg =	.08661
	max =	.1025

Log marginal-likelihood = -201.1662

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
weight	-.6004123	.0510882	.001595	-.5998094	-.7040552	-.5058665
_cons	39.55017	1.590016	.050051	39.49377	36.56418	42.79701
var	12.18757	2.038488	.085865	11.90835	8.913695	16.88978

The results are now closer to the results using noninformative prior obtained in [example 7](#), because we are introducing some information from the observed data by using  $(X'X)^{-1}$ .

◀

## ▶ Example 10: Specifying expressions as distributional arguments

We can actually reproduce what prior `zellnersg()` does in [example 9](#) manually.

First, we need to create a matrix that contains  $(X'X)^{-1}$ , `S`.

```

. matrix accum xTx = weight
(obs=74)
. matrix S = invsym(xTx)

```

Then, we can use the multivariate normal prior `mvnormal()` with the variance specified as an expression `30*var*S`.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, mvnormal(2,-0.5,40,30*{var}*S))
> prior({var}, igamma(5,50))
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:weight \_cons} ~ mvnormal(2,-0.5,40,30\*{var}\*S) (1)  
{var} ~ igamma(5,50)

---

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2576
	Efficiency: min =	.05636
	avg =	.08661
	max =	.1025

Log marginal-likelihood = -201.1662

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
weight	-.6004123	.0510882	.001595	-.5998094	-.7040552	-.5058665
_cons	39.55017	1.590016	.050051	39.49377	36.56418	42.79701
var	12.18757	2.038488	.085865	11.90835	8.913695	16.88978

We obtain results identical to those from [example 9](#).



Multiple linear regression

For a detailed example of a multiple linear regression, see [Overview example](#) in [\[BAYES\] Bayesian commands](#).

Improving efficiency of the MH sampling

In this section, we demonstrate how one can improve efficiency of the MH algorithm by using blocking of parameters and Gibbs sampling, whenever available. We continue with our simple linear regression of mpg on rescaled weight from [Simple linear regression](#), but we use different values for the parameters of prior distributions. We also assume that regression coefficients and the variance parameter are independent a priori. We use the `blocksummary` option to include a summary about each block.

### Example 11: First simulation run

Our first simulation is performed using the default settings for the algorithm. Specifically, all three model parameters are placed in one simulation block and are updated simultaneously, as our block summary indicates.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10)) blocksummary
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)
                      {var} ~ igamma(10,10)                                (1)
```

(1) Parameters are elements of the linear form xb\_mpg.

Block summary

1: {mpg:weight \_cons} {var}

Bayesian normal regression

Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = -226.63723

MCMC iterations = 12,500

Burn-in = 2,500

MCMC sample size = 10,000

Number of obs = 74

Acceptance rate = .2432

Efficiency: min = .06871

avg = .08318

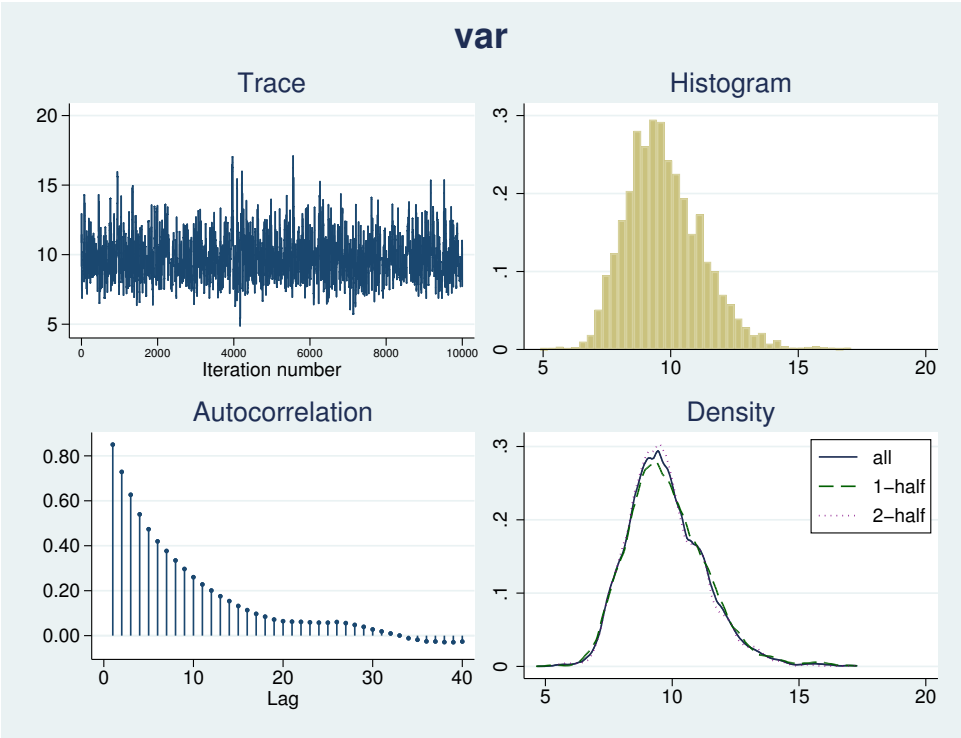
max = .09063

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]		
mpg	weight	-.5759855	.0471288	.001569	-.5750919	-.6676517	-.4868595
	_cons	38.65481	1.468605	.048784	38.70029	35.88062	41.49839
	var	9.758003	1.514112	.057762	9.601339	7.302504	13.13189

The mean estimates based on the simulated sample are {mpg:weight}= −0.58, {mpg:\_cons}= 38.65, and {var}= 9.8. The MH algorithm achieves an overall AR of 24% and an average efficiency of about 8%.

Our next step is to perform a visual inspection of the convergence of the chain.

```
. bayesgraph diagnostics {var}
```



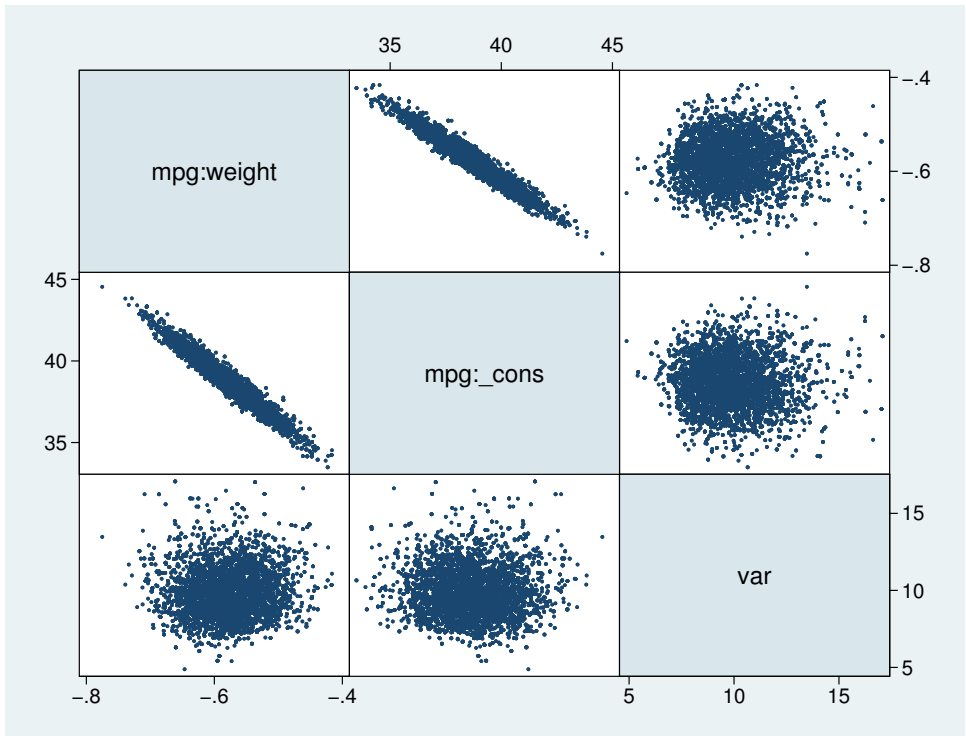
A graphical summary for the {var} parameter does not show any obvious problems. The trace plot reveals a good coverage of the domain of the marginal distribution, while the histogram and kernel density plots resemble the shape of an expected inverse-gamma distribution. The autocorrelation dies off after about lag 20.



### ► Example 12: Second simulation run—blocking of variance

Next, we show how to improve the mixing of the MCMC chain by using more careful blocking of model parameters. We can use the `bayesgraph matrix` command to view the scatterplots of the simulated values for `{mpg:weight}`, `{mpg:_cons}`, and `{var}`.

```
. bayesgraph matrix _all
```



The scatterplots reveal high correlation between `{mpg:weight}` and `{mpg:_cons}`. On the other hand, there is no significant correlation between `{var}` and the other two parameters.

In cases like this, we can expect higher sampling efficiency if we place `{var}` in a separate block. We can do this by including the option `block({var})`. The other two parameters, `{mpg:weight}` and `{mpg:_cons}`, will be automatically considered as a second block.



```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}) blocksummary
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:weight \_cons} ~ normal(0,100)  
{var} ~ igamma(10,10)

(1) Parameters are elements of the linear form xb\_mpg.

Block summary

---

1:	{var}
2:	{mpg:weight _cons}

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.3309
	Efficiency: min =	.09023
	avg =	.1202
	max =	.1784

Log marginal-likelihood = -226.73992

---

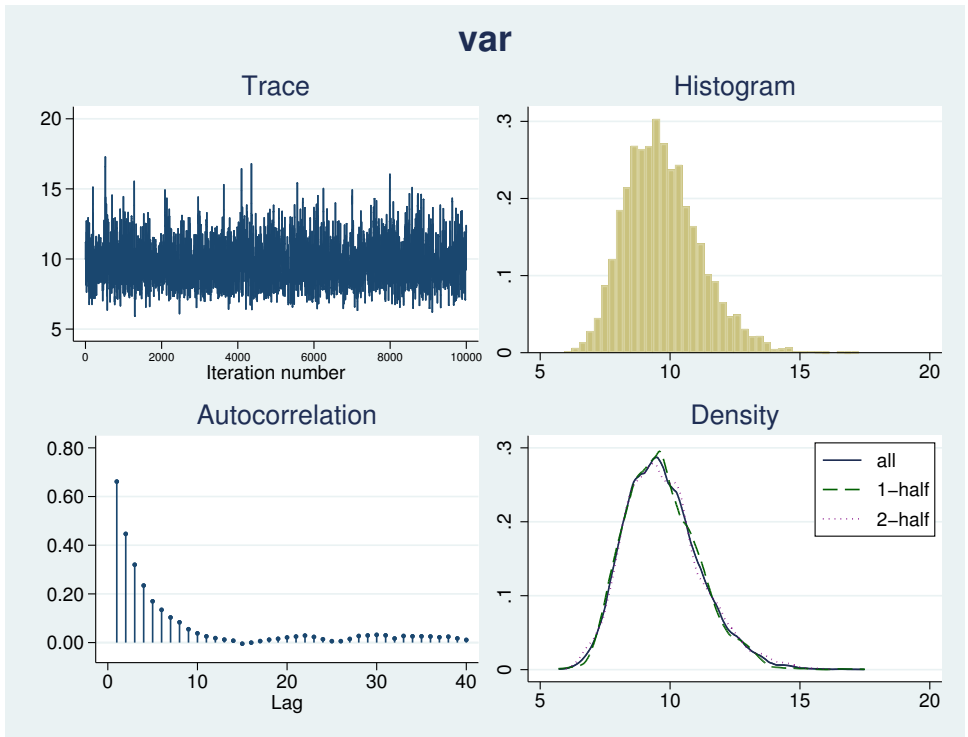
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5744536	.0450094	.001484	-.576579	-.663291
	_cons	38.59206	1.397983	.04654	38.63252	35.80229
	var	9.721684	1.454193	.034432	9.570546	7.303129

---

In this second run, we achieve higher simulation efficiency, about 12% on average. The MCSE for {var} is 0.034 and is about half the value of 0.058 from [example 11](#), which leads to twice as much accuracy in the estimation of the posterior mean of {var}.

Again, we can verify the convergence of the MCMC run for `{var}` by inspecting the `bayesgraph` diagnostics plot.

```
. bayesgraph diagnostics {var}
```



The improved sampling efficiency for `{var}` is evident by observing that the autocorrelation becomes negligible after about lag 10. The trace plot reveals more rapid traversing of the marginal posterior domain as well.

◀

### ► Example 13: Third simulation run—Gibbs update of variance

Further improvement of the mixing can be achieved by requesting a Gibbs sampling for the variance parameter. This is possible because `{var}` has an inverse-gamma prior, which is independent of the mean and is a semiconjugate prior in this model.

To request Gibbs sampling, we specify suboption gibbs within option block().

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}, gibbs) blocksummary
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:weight \_cons} ~ normal(0,100)  
{var} ~ igamma(10,10) (1)

(1) Parameters are elements of the linear form xb\_mpg.

Block summary

---

1: {var} (Gibbs)  
2: {mpg:weight \_cons}

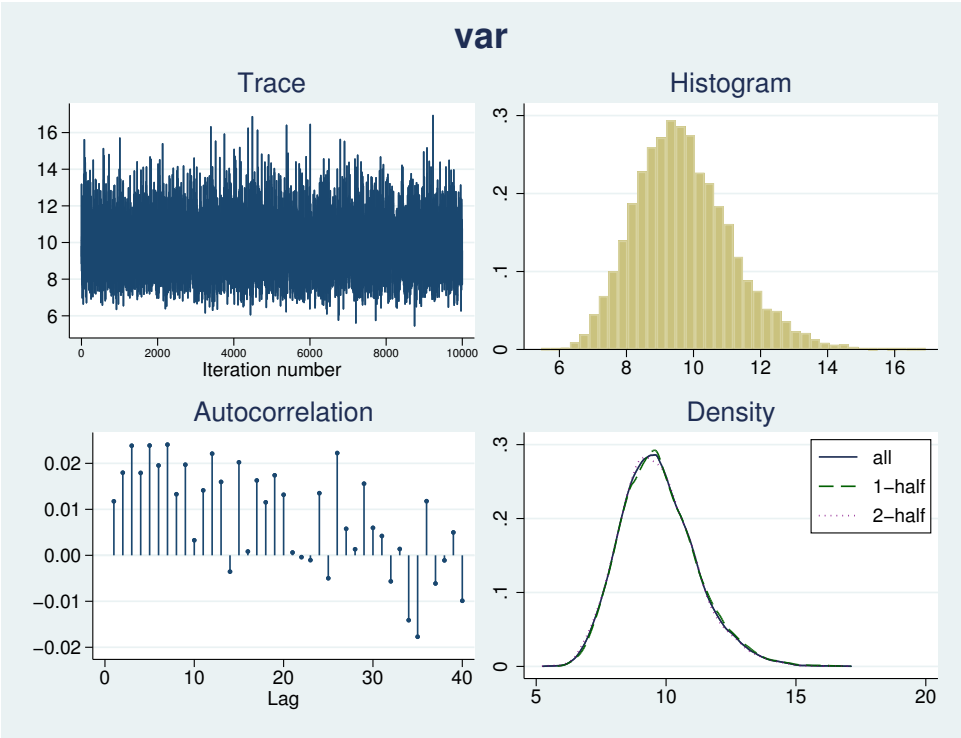
Bayesian normal regression	MCMC iterations	=	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.6285
	Efficiency: min	=	.1141
	avg	=	.3259
	max	=	.7441
Log marginal-likelihood = -226.72192			

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5764752	.0457856	.001324	-.5764938	-.6654439	-.486788
	_cons	38.64148	1.438705	.04259	38.6177	35.82136	41.38734
	var	9.711499	1.454721	.016865	9.585728	7.236344	12.95503

The average efficiency is now 0.33 with the maximum of 0.74 corresponding to the variance parameter.

The diagnostics plot for {var} is an example of almost perfect mixing.

```
. bayesgraph diagnostics {var}
```



➤ Example 14: Fourth simulation run—full Gibbs sampling

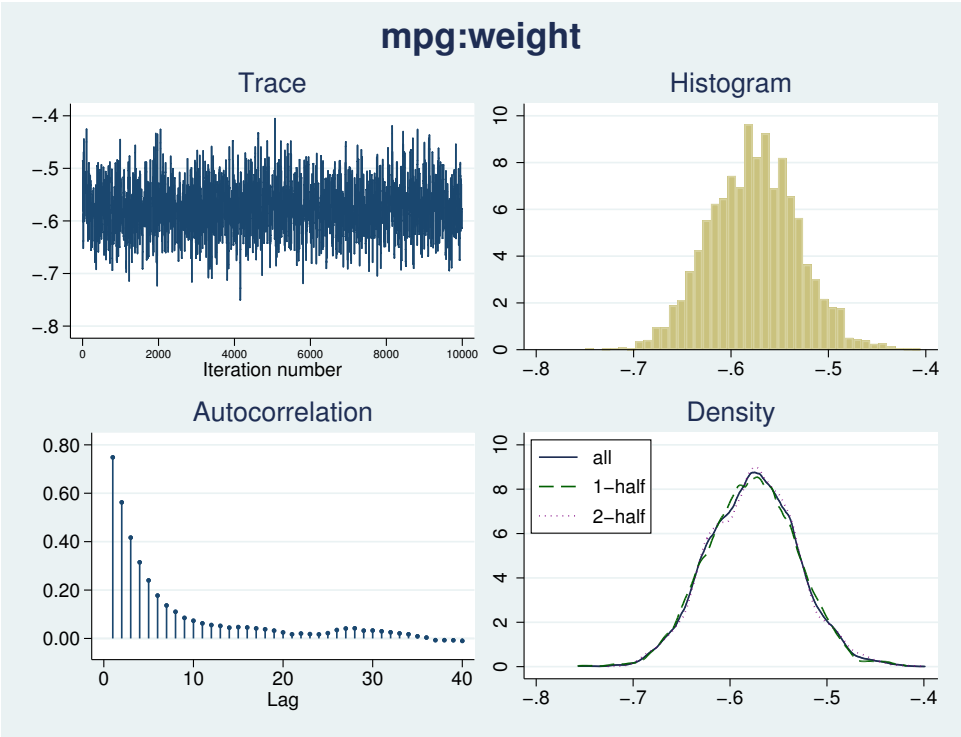
Continuing [example 13](#), there is still room for improvement in our model in terms of sampling efficiency. The efficiency of the regression coefficients is now low relative to the variance efficiency.

```
. bayesstats ess
Efficiency summaries      MCMC sample size =    10,000
                        Efficiency:  min =      .1141
                                avg =      .3259
                                max =      .7441
```

		ESS	Corr. time	Efficiency
mpg	weight	1195.57	8.36	0.1196
	_cons	1141.12	8.76	0.1141
var		7440.67	1.34	0.7441

For example, diagnostic plots for {weight:\_cons} do not look as good as diagnostic plots for the variance parameter in [example 13](#).

```
. bayesgraph diagnostics {mpg:weight}
```



Further improvement of the mixing can be achieved by requesting Gibbs sampling for the two blocks of parameters: regression coefficients and variance. Again, this is possible only because {mpg:weight}, {mpg:\_cons}, and {var} have normal and an inverse-gamma priors, which are independent and are semiconjugate in this model.

To request Gibbs sampling for the regression coefficients, we must place them in a separate block.

```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100))
> prior({var}, igamma(10,10))
> block({var}, gibbs)
> block({mpg:}, gibbs) blocksummary
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ normal(0,100)
                      {var} ~ igamma(10,10)                                (1)
```

---

(1) Parameters are elements of the linear form xb\_mpg.

Block summary

1: {var}	(Gibbs)
2: {mpg:weight _cons}	(Gibbs)

---

Bayesian normal regression	MCMC iterations =	12,500
Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	1
	Efficiency: min =	.9423
	avg =	.9808
	max =	1

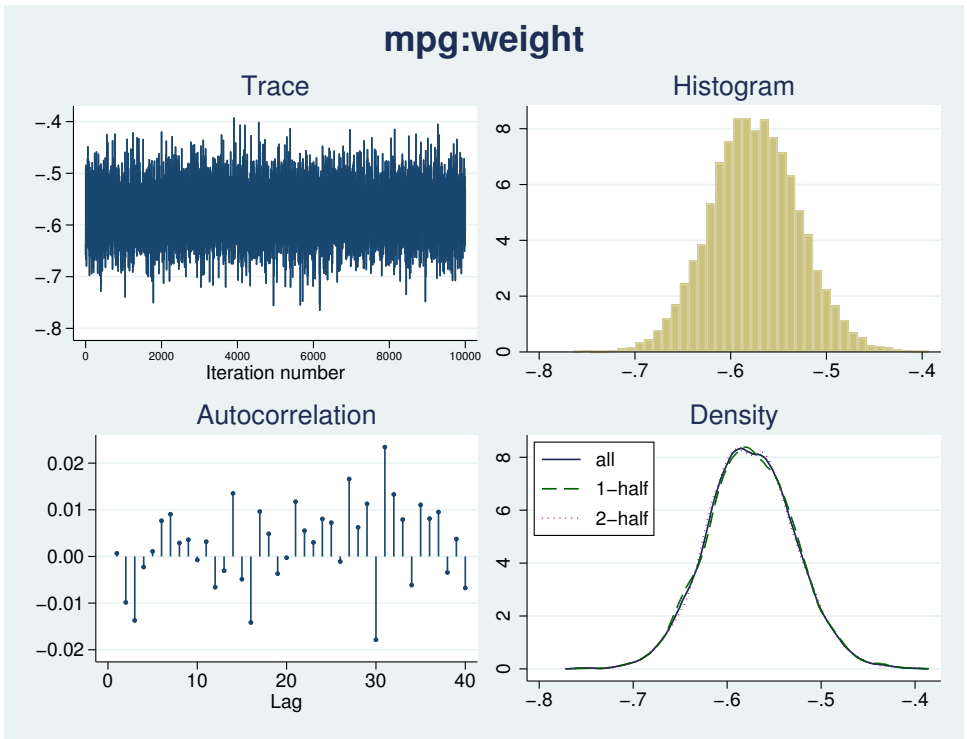
Log marginal-likelihood = -226.67227

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
	weight	-.5751071	.0467837	-.5757037	-.6659412	-.4823263
	_cons	38.61033	1.459511	38.61058	35.79156	41.45336
	var	9.703432	1.460435	9.564502	7.216982	12.96369

Now we have perfect sampling efficiency (with an average of 0.98) with essentially no autocorrelation. The estimators of posterior means have the lowest MCSEs among the four simulations.

For example, diagnostic plots for `{mpg:weight}` now look noticeably better.

```
. bayesgraph diagnostics {mpg:weight}
```



You can verify that the diagnostic plots of all parameters demonstrate almost perfect mixing as well.

```
. bayesgraph diagnostics _all  
(output omitted)
```

◀

## Convergence diagnostics using multiple chains

To assess the convergence of MCMC simulations of a Bayesian model, the literature often recommends comparing the results of multiple simulation sequences or multiple chains; see, for example, [Gelman et al. \(2014, chap. 11.4\)](#). In this section, we show how one can simulate multiple chains using `bayesmh`, visually compare the results using trace and density plots, and perform formal tests for convergence.

To simulate multiple Markov chains, you can use the `nchains()` option with `bayesmh`. When running multiple chains, it is essential for the chains to have different initial values **dispersed** over the range of values of model parameters. `bayesmh`, `nchains()` provides default initial values that are different for each chain, but these values are not guaranteed to be overdispersed and are provided strictly for your convenience. Often, you may want to specify your own initial values, which you can do using the `init#()` options; see [Specifying initial values](#) and [Multiple chains using overdispersed initial values](#).

## Multiple chains using default initial values

Let's continue with the Bayesian multiple linear regression model from [example 11](#). We specify the `nchains(4)` option to simulate four Markov chains of default size 10,000. We use the `rseed()` option to ensure reproducibility when running multiple chains. Specifying `set seed` is not sufficient in this case; see [Reproducing results](#). We also use `nomodelsummary` to suppress the output of the model summary.

```
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> nomodelsummary nchains(4) rseed(16)
```

Chain 1  
Burn-in ...  
Simulation ...

Chain 2  
Burn-in ...  
Simulation ...

Chain 3  
Burn-in ...  
Simulation ...

Chain 4  
Burn-in ...  
Simulation ...

Bayesian normal regression	Number of chains	=	4
Random-walk Metropolis-Hastings sampling	Per MCMC chain:		
	Iterations	=	12,500
	Burn-in	=	2,500
	Sample size	=	10,000
	Number of obs	=	74
	Avg acceptance rate	=	.2275
	Avg efficiency: min	=	.07897
	avg	=	.08265
	max	=	.08827
Avg log marginal-likelihood = -226.73271	Max Gelman-Rubin Rc	=	1.002

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5749136	.0463642	.000816	-.5760212	-.6649088	-.4847602
	_cons	38.59661	1.447703	.025758	38.62636	35.7311	41.40999
	var	9.713168	1.431891	.024098	9.605324	7.332055	12.84306

Note: Default initial values are used for multiple chains.

The important change in the output header of `bayesmh` with multiple chains is the presence of the maximum Gelman–Rubin convergence statistic, Max Gelman–Rubin Rc. This is the maximum value of the statistics across all model parameters. A convergence rule often used in practice is to declare convergence when convergence statistics of all model parameters are less than 1.1. In our example, the maximum statistic of 1.002 is less than 1.1, so the convergence rule is satisfied. See [\[BAYES\] bayesstats grubin](#) for details. Of course, it is important to also inspect convergence visually, as we demonstrate later in this example.

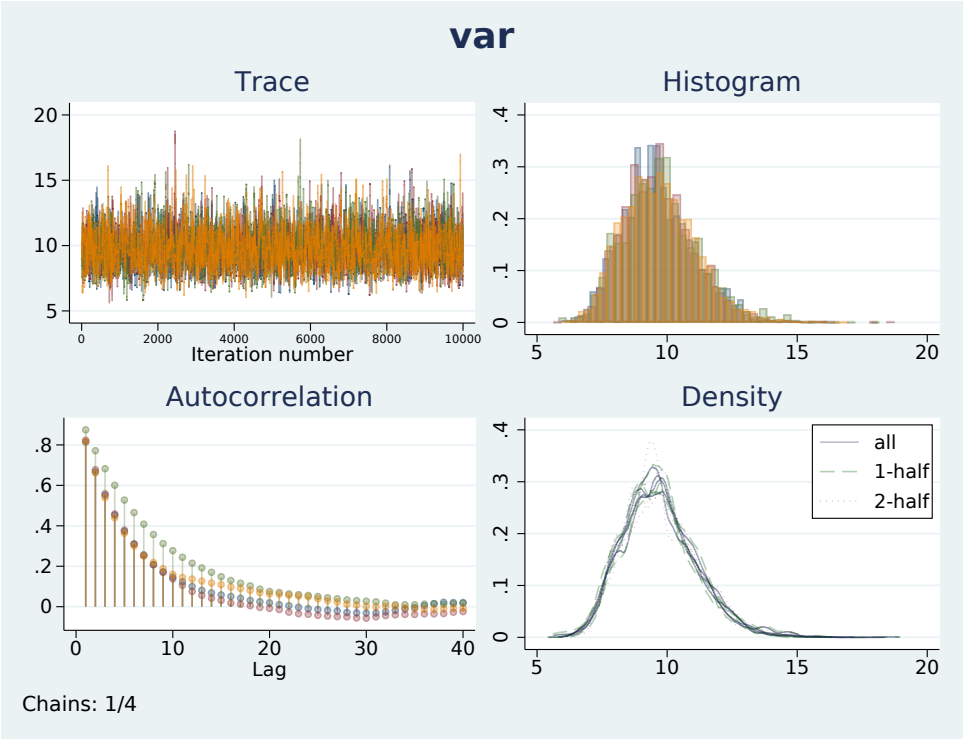
Because there are multiple simulation chains, `bayesmh` reports the simulation summaries averaged over the chains such as the average acceptance rate, average efficiencies, and the average log marginal-likelihood. You can use the `chainsdetail` option to see those summaries separately for each chain.



The average simulation efficiency for all chains is above 8% and seems adequate. The Gelman–Rubin convergence rule is met. There is no indication of convergence problems. Nevertheless, inspecting the simulation chains visually can provide additional reassurance. For instance, by comparing the trace plots of different simulation sequences for a model parameter, we can detect convergence irregularities and assess the overlap of the simulated marginal distributions for this parameter. If Markov chains have converged, we should not observe substantial differences between the trace plots or between the sampled marginal distributions.

For a single chain, we used `bayesgraph` diagnostics to explore the convergence of MCMC visually. We can use this command with multiple chains as well. Let’s plot graphical summaries for the variance parameter `{var}`.

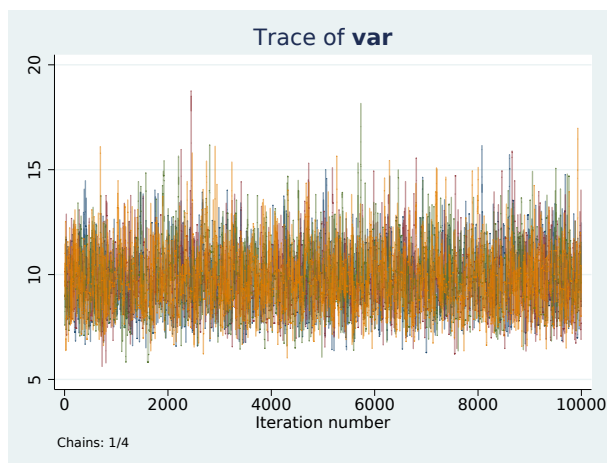
```
. bayesgraph diagnostics {var}
```



Graphical diagnostics look somewhat messy for multiple chains, but the main takeaway from this graph is that the results of the chains do not look drastically different. The trace plots overlap, the autocorrelations die off, and the histograms and density plots are similar for all chains. If desired, you can produce separate plots or graphs for each chain using `bayesgraph`’s `bychain()` or `sepchains` option; see [\[BAYES\] bayesgraph](#).

You can also focus separately on each type of plot. For instance, let's look more closely at the trace and density plots.

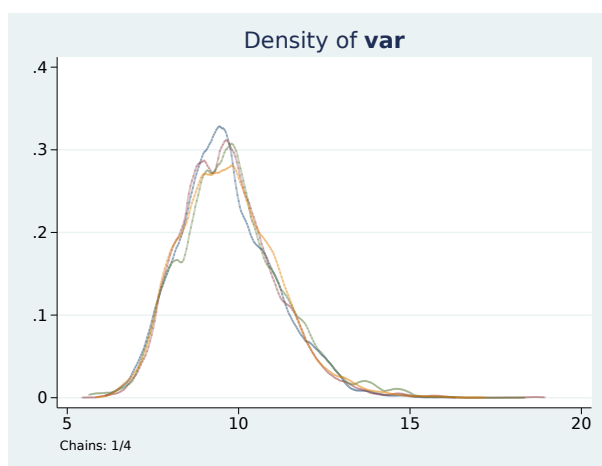
```
. bayesgraph trace {var}
```



The `bayesgraph trace` command overlays the traces of the simulated chains for convenient visual comparison of the chains. The trace plots are similar in terms of coverage and variation.

The overlaid density plots shown by `bayesgraph kdensity` provide another aspect of comparing multiple simulation sequences.

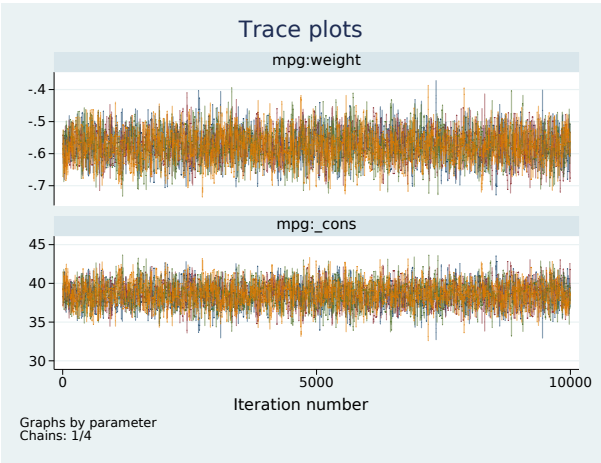
```
. bayesgraph kdensity {var}
```



The density plots of `{var}` from all chains mostly overlap with some variations about the marginal mode.

Similarly, we can explore the MCMC convergence visually for other parameters. For example, we can draw the trace plots for the coefficient parameters {mpg:\_cons} and {mpg:weight} and use bayesgraph’s byparm option to place plots of both parameters on one graph.

```
. bayesgraph trace {mpg:}, byparm
```



Again, the overlaid trace plots of {mpg:\_cons} and {mpg:weight} do not show any substantial differences and indicate good mixing of the chains.

We can use the bayesstats grubin command to compute Gelman–Rubin convergence diagnostics using multiple chains.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains      =          4
MCMC size, per chain =    10,000
Max Gelman-Rubin Rc   =    1.002068
```

		Rc
mpg	weight	1.000783
	_cons	1.000557
var		1.002068

```
Convergence rule: Rc < 1.1
```

Estimates of convergence statistics, Rc, larger than 1.2 indicate possible nonconvergence. In our case, the Rc estimates for all parameters are very close to 1 and do not raise any convergence concerns. Note that the largest estimate, 1.002, as reported by bayesmh, corresponds to parameter {var}.

Once MCMC convergence is established, we can proceed with our estimation results. We replay them here for your convenience (without the table header information).

```
. bayesmh, noheader
```

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5749136	.0463642	.000816	-.5760212	-.6649088	-.4847602
	_cons	38.59661	1.447703	.025758	38.62636	35.7311	41.40999
	var	9.713168	1.431891	.024098	9.605324	7.332055	12.84306

The summary results in the estimation table are based on all chains. Because we used more chains, our results are now more precise (have smaller MCSEs) compared with [example 11](#).

To inspect posterior summaries of each chain, we can use the `bayesstats summary` command with the `sepchains` option.

```
. bayesstats summary, sepchains
```

Posterior summary statistics

Chain 1 MCMC sample size = 10,000

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5736929	.0458934	.001611	-.5745238	-.6629738	-.4877666
	_cons	38.5649	1.425768	.052564	38.60731	35.75694	41.37725
	var	9.64884	1.386373	.044099	9.513188	7.251423	12.70699

Chain 2 MCMC sample size = 10,000

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5747026	.0456178	.001699	-.5759074	-.6618918	-.4851731
	_cons	38.59502	1.441276	.053339	38.57138	35.72466	41.40902
	var	9.683921	1.39533	.043302	9.60479	7.420058	12.73925

Chain 3 MCMC sample size = 10,000

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5740745	.0468218	.00169	-.576532	-.6631272	-.4817094
	_cons	38.57018	1.469792	.053026	38.62822	35.68724	41.37469
	var	9.802202	1.508294	.059519	9.68037	7.339275	13.32406

Chain 4		MCMC sample size = 10,000					
		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5771844	.0470114	.001543	-.5773599	-.6678485	-.4862513
	_cons	38.65634	1.451485	.047729	38.69004	35.82901	41.49365
	var	9.717709	1.428596	.048662	9.614184	7.33145	12.89246

The results from all chains are similar. The differences between posterior means, for instance, are within the ranges of the MCMC standard errors of the estimates.

In the presence of multiple chains, `bayesmh` displays a note beneath the estimation table about default initial values being used for the chains. The default initial values are provided for convenience, and often you may want to specify your own; see [Specifying initial values](#) for details. Also see [Multiple chains using overdispersed initial values](#) next.

**Multiple chains using overdispersed initial values**

We continue with our multiple-chains example from [Multiple chains using default initial values](#), but here we simulate Markov chains using overdispersed initial values. We specify random initial values manually using the `init#()` options.

For simplicity, we use only two chains. We generate initial values that are highly overdispersed and are far away from the maximum-likelihood estimates of model parameters. For the first chain, we generate initial values for the regression coefficients from the normal distribution with mean 10 and standard deviation 10 and for the variance from the gamma distribution with shape 1 and scale 50. For the second chain, we use the same distributions but different parameters, except for the standard deviation: we use the mean of  $-10$ , the standard deviation of 10, the shape of 50, and the scale of 1. We use the `init1()` and `init2()` options, respectively, to specify these initial values. To see the initial values used, we also specify the `initsummary` option.

```

. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> init1({mpg:} rnormal( 10, 10) {var} rgamma(50, 1))
> init2({mpg:} rnormal(-10, 10) {var} rgamma(1, 50))
> nomodelsummary nchains(2) rseed(16) initsummary
Chain 1
  Burn-in ...
  Simulation ...
Chain 2
  Burn-in ...
  Simulation ...
Initial values:
Chain 1: {mpg:weight} .168372 {mpg:_cons} 10.2646 {var} 46.3212
Chain 2: {mpg:weight} -9.07515 {mpg:_cons} -22.1665 {var} 39.3092
Bayesian normal regression          Number of chains      =          2
Random-walk Metropolis-Hastings sampling  Per MCMC chain:
                                           Iterations         =        12,500
                                           Burn-in             =         2,500
                                           Sample size         =        10,000
                                           Number of obs       =          74
                                           Avg acceptance rate =         .2256
                                           Avg efficiency: min =         .04544
                                           avg                 =         .07662
                                           max                 =         .09876
Avg log marginal-likelihood = -245.37212  Max Gelman-Rubin Rc =        42.57

```

		Mean	Std. Dev.	MCSE	Median	Equal-tailed	
						[95% Cred. Interval]	
mpg	weight	-.5334204	.0939955	.002271	-.5468147	-.6670521	-.3335525
	_cons	37.27179	2.977634	.067	37.70683	30.95118	41.41418
	var	27.45511	25.17659	.835183	30.3807	7.549151	45.8256

Note: There is a high autocorrelation after 500 lags in at least one of the chains.

The reported maximum Gelman–Rubin convergence statistic, 42.57, is very high and is much larger than 1. A note beneath the table reports high autocorrelation in one of the chains. Clearly, we have a problem.

We check the sampling efficiency of the parameters for each chain separately:

```
. bayesstats ess, sepchains
Efficiency summaries
Chain 1                MCMC sample size =    10,000
                        Efficiency:  min =    .07407
                                avg =    .07956
                                max =    .08962
```

		ESS	Corr. time	Efficiency
mpg	weight	749.91	13.33	0.0750
	_cons	740.66	13.50	0.0741
	var	896.19	11.16	0.0896

```
Chain 2                MCMC sample size =    10,000
                        Efficiency:  min =    .001253
                                avg =    .07369
                                max =    .1234
```

		ESS	Corr. time	Efficiency
mpg	weight	963.73	10.38	0.0964
	_cons	1234.44	8.10	0.1234
	var	12.53	798.09	0.0013

The {var} parameter in the second chain has the lowest ESS of 12.53.

Let’s check the Gelman–Rubin convergence statistics for all parameters.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains      =          2
MCMC size, per chain =    10,000
Max Gelman-Rubin Rc   =    42.57122
```

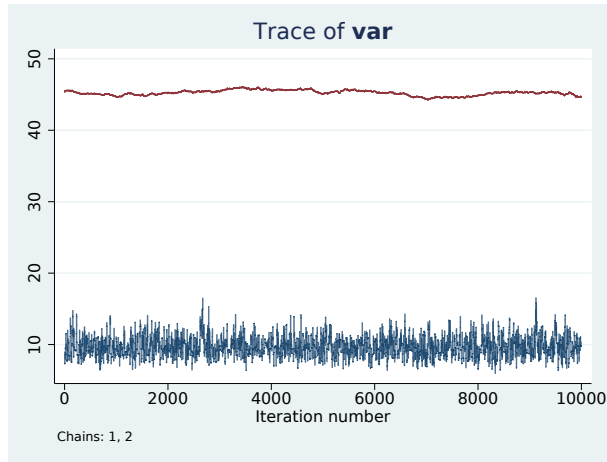
		Rc
mpg	weight	1.622996
	_cons	1.665635
	var	42.57122

Convergence rule: Rc < 1.1

The Rc estimates for all three parameters exceed 1, confirming nonconvergence, but {var} has a particularly large value of the convergence statistic of 42.57.

To investigate the convergence problem further visually, we inspect the trace plots of the `{var}` parameter from each chain.

```
. bayesgraph trace {var}
```

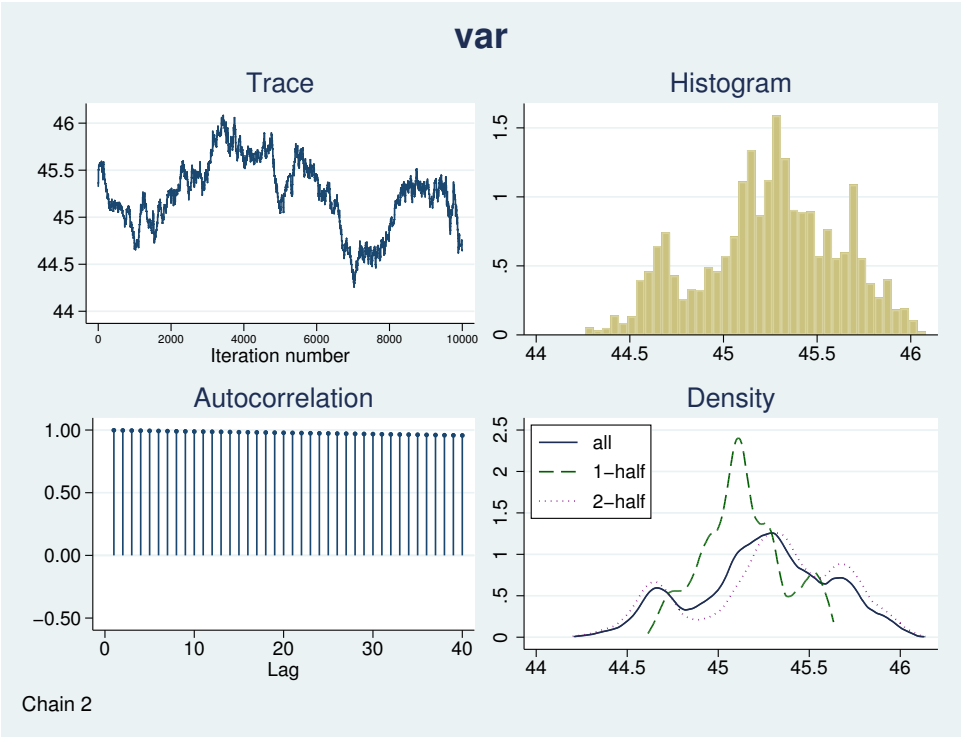


The two trace plots are completely separated and show that the chains explore different domains of the posterior distribution. The trace plot of the second chain, shown in red, has a mean value of about 45. Given a large initial value for `{var}` and the stochastic nature of the algorithm, the second chain did not converge by the default number of 2,500 burn-in iterations.



If we look at graphical diagnostics of {var} for the second chain,

```
. bayesgraph diagnostics {var}, chains(2)
```



we notice that the autocorrelation stays close to 1 and the trace plot exhibits a slow random walk behavior, failing to stabilize in a particular region.

When you specify overdispersed initial values, you should give the chains enough time to converge. This second chain simply has not run long enough to converge to the domain with a high posterior density. To fix this, we can use a longer burn-in of 10,000, `burnin(10000)`, and longer adaptation by lowering the adaptation tolerance to 0.002, `adaptation(tolerance(0.002))`.

```
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> nomodelsummary nchains(2) rseed(16)
> init1({mpg:} rnormal( 10, 10) {var} rgamma(50, 1))
> init2({mpg:} rnormal(-10, 10) {var} rgamma(1, 50))
> burnin(10000) adapt(tolerance(0.002))
```

Chain 1  
Burn-in ...  
Simulation ...

Chain 2  
Burn-in ...  
Simulation ...

Bayesian normal regression	Number of chains	=	2
Random-walk Metropolis-Hastings sampling	Per MCMC chain:		
	Iterations	=	20,000
	Burn-in	=	10,000
	Sample size	=	10,000
	Number of obs	=	74
	Avg acceptance rate	=	.296
	Avg efficiency: min	=	.08096
	avg	=	.09193
	max	=	.1002
Avg log marginal-likelihood = -226.70215	Max Gelman-Rubin Rc	=	1.001

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.5759702	.0461691	.001061	-.5772111	-.665917	-.4826217
	_cons	38.64229	1.440565	.032185	38.66686	35.73169	41.42428
	var	9.691232	1.472907	.036603	9.530698	7.264868	13.0381

The maximum Gelman–Rubin statistic is now only 1.001. We use `bayesstats grubin` for details.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains = 2
MCMC size, per chain = 10,000
Max Gelman-Rubin Rc = 1.001315
```

		Rc
mpg	weight	1.001315
	_cons	1.00095
	var	1.000061

Convergence rule:  $Rc < 1.1$

All Rc estimates satisfy the convergence rule,  $Rc < 1.1$ .

## Bayesian predictions

Bayesian predictions provide a powerful set of tools for model evaluation and assessing goodness of fit, in addition to predicting future observations; see *Overview of Bayesian predictions* in [BAYES] **bayespredict** for details. You can use **bayespredict**, **bayesreps**, and **bayesstats pp-values** to obtain Bayesian predictions and perform model checks. Here we illustrate some of the features of Bayesian predictions, which are available after fitting a model using **bayesmh**. We continue with the Bayesian multiple linear regression model from [example 11](#).

## Simulating replicated outcomes

As a quick model check, we can explore the distribution of the replicated outcomes and compare them with the observed outcome distribution. Replicated outcomes are new outcome values simulated from the **posterior predictive distribution** conditional on the observed set of covariates. Generally, replicated outcomes compose a sample of  $T$  observations, MCMC replicates, and  $n$  variables, one for each observation in the original data. The entire prediction sample is rarely needed in most applications. Often, it is sufficient to explore a small random subset from all  $T$  MCMC replicates. We can use **bayesreps** to generate such a subset and save the generated replicates as new variables in our dataset.

To use **bayesreps** and **bayespredict**, we must first save the simulation results from **bayesmh**. Let's refit the linear regression model and save the simulation results in **linregsim.dta**. We suppress the output with **quietly**.

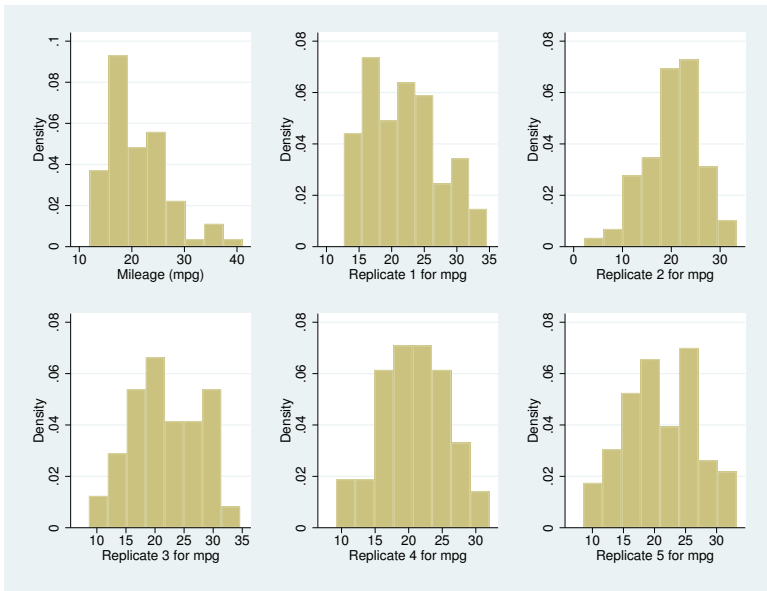
```
. quietly bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, normal(0,100)) prior({var}, igamma(10,10))
> saving(linregsim) rseed(16)
```

We can now use **bayesreps** to generate the replicated outcomes for variable **mpg**. These will be samples from the posterior predictive distribution of **mpg** conditioned on the observed set of explanatory variables, **weight**. Each replication sample will be of the same size, 74, as the original outcome **mpg**. Let's generate 5 replication samples and save them in the original dataset as new variables, **mpgprep1** through **mpgprep5**, specified as the stub **mpgprep\***.

```
. bayesreps mpgprep*, nreps(5) rseed(16)
Computing predictions ...
```

We can visually inspect the histograms of the replicated samples and compare them with the histogram for the observed mpg.

```
. qui histogram mpg, name(hist0) nodraw
. local histlist hist0
. forvalues i = 1/5 {
2.     qui histogram mpgrep'i', name(hist'i') nodraw
3.     local histlist 'histlist' hist'i'
4. }
. graph combine 'histlist'
```



The histogram of mpg (top, left) looks different from those of the replications. All of them cover the range of (10, 30), but the observed mpg is skewed to the left and has heavier tails. The normal model does not appear to capture the observed distribution well. After these initial checks, we proceed with a more quantitative assessment of model fit.

## Posterior predictive checks

A [posterior predictive check](#) is one of the main applications of Bayesian predictions. It starts with defining test statistics that represent different aspects of the outcome distribution. Then, these test statistics are computed using the observed and replicated outcomes, and their values are compared. For example, the mean, minimum, and maximum statistics can be used for assessing how well the model represents the outcome distribution with respect to its center and extremes.

We can simulate the mean, minimum, and maximum statistics using `bayespredict`, which supports the use of Mata functions to compute functions of simulated outcomes. Thus, we can use Mata functions `mean()`, `min()`, and `max()` to compute the desired statistics. We specify the argument `{_ysim}` with the functions to request statistics of the simulated outcomes (we can also use `{_resid}` for residuals). We save the prediction results in `mpgsim.dta`. See [\[BAYES\] bayespredict](#) for details about the specification.

```
. bayespredict (prmean:@mean({_ysim})) (prmin:@min({_ysim}))
> (prmax:@max({_ysim})), saving(mpgsim) rseed(16)
Computing predictions ...
file mpgsim.dta saved
file mpgsim.ster saved
```

We can now access the prediction results within other Bayesian postestimation commands such as `bayesstats summary` and `bayesstats ppvalues`.

Let’s compare the agreement for the mean, minimum, and maximum between the replicated data and observed data. The `bayesstats ppvalues` command makes such comparisons easy. It reports the proportion of cases when the simulated statistics are greater than or equal to the observed values of statistics, which is an estimate of the so-called **posterior predictive *p*-value**.

```
. bayesstats ppvalues {prmean} {prmin} {prmax} using mpgsim
Posterior predictive summary      MCMC sample size =      10,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
prmean	21.24042	.5016505	21.2973	.4511
prmin	8.372033	2.159442	12	.027
prmax	32.92524	1.802402	41	.0004

Note: P(T>=T\_obs) close to 0 or 1 indicates lack of fit.

The posterior predictive *p*-value is 0.45 for the mean statistic, 0.03 for the minimum, and less than 0.001 for the maximum. Our normal model captures the center of the distribution of `mpg` well but fails to capture the extreme values. The posterior predictive *p*-value for the maximum statistic is particularly small, which agrees with our earlier conclusion based on the histograms that the maximum values are not well represented by the model. If we believe that the extremely large observations of `mpg` are not aberrant outliers, we may need to look for a better-fitting likelihood model than the normal model.

As the final step, we remove the files generated by `bayesmh` and `bayespredict` because we no longer need them.

```
. erase linregsim.dta
. erase mpgsim.dta
. erase mpgsim.ster
```

See [\[BAYES\] bayespredict](#) and [\[BAYES\] bayesstats ppvalues](#) for more examples.

**Logistic regression model: A case of nonidentifiable parameters**

We use the heart disease dataset from the UCI Machine Learning Repository ([Lichman 2013](#)) and, in particular, we consider a subset of the Switzerland data created by William Steinbrunn, M.D. of University Hospital in Zurich, Switzerland, and by Matthias Pfisterer, M.D. of University Hospital in Basel, Switzerland. The dataset is named `heartswitz.dta` and contains 6 variables, of which `num` is the predicted attribute that takes values from 0 (no heart disease) to 4. We dichotomized `num` to create a new binary variable `disease` as an indicator for the presence of a heart disease.

```
. use https://www.stata-press.com/data/r16/heartswitz, clear
(Subset of Switzerland heart disease data from UCI Machine Learning Repository)

. describe
Contains data from https://www.stata-press.com/data/r16/heartswitz.dta
   obs:                123                Subset of Switzerland heart
                                         disease data from UCI Machine
                                         Learning Repository
   vars:                6                5 Feb 2018 16:55
                                         (_dta has notes)
```

variable name	storage type	display format	value label	variable label
age	byte	%9.0g		Age (in years)
male	byte	%9.0g	malelab	1 = male, 0 = female
isfbs	byte	%9.0g	fbslab	Indicator for fasting blood sugar > 120 mg/dl: 0 = no, 1 = yes
restecg	byte	%28.0g	ecglab	Resting electrocardiographic results (3 categories)
num	byte	%9.0g		Presence of heart disease: 0 = absent and 1,2,3,4 = present
disease	byte	%9.0g	dislab	Indicator for heart disease: 0 = absent, 1 = present (num>0)

Sorted by:

Our goal is to investigate the relationship between the presence of a heart disease and covariates `restecg`, `isfbs`, `age`, and `male`.

First, we fit a standard logistic regression model using the `logit` command.

```
. logit disease restecg isfbs age male
note: restecg != 0 predicts success perfectly
      restecg dropped and 17 obs not used
note: isfbs != 0 predicts success perfectly
      isfbs dropped and 3 obs not used
note: male != 1 predicts success perfectly
      male dropped and 2 obs not used

Iteration 0:  log likelihood = -4.2386144
Iteration 1:  log likelihood = -4.2358116
Iteration 2:  log likelihood = -4.2358076
Iteration 3:  log likelihood = -4.2358076

Logistic regression               Number of obs   =          26
                                LR chi2(1)          =           0.01
                                Prob > chi2          =          0.9403
                                Pseudo R2            =          0.0007

Log likelihood = -4.2358076
```

disease	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
restecg	0 (omitted)					
isfbs	0 (omitted)					
age	-.0097846	.1313502	-0.07	0.941	-.2672263	.2476572
male	0 (omitted)					
_cons	3.763893	7.423076	0.51	0.612	-10.78507	18.31285

We encounter collinearity and dropping of observations because of perfect prediction. As a result, the regression coefficients corresponding to `restecg`, `isfbs`, and `male` are essentially excluded from the model. The standard logistic analysis is limited because of the small size of the dataset.

Next we consider Bayesian analysis of the same data. We fit the same logistic regression model using `bayesmh` and apply fairly noninformative normal priors  $N(0, 1e4)$  for all regression parameters.

```
. set seed 14
. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,10000))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  disease ~ logit(xb_disease)

Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,10000)          (1)
```

---

```
(1) Parameters are elements of the linear form xb_disease.

Bayesian logistic regression      MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =       2,500
                                          MCMC sample size =     10,000
                                          Number of obs   =        48
                                          Acceptance rate =      .2661
                                          Efficiency:  min =     .01685
                                          avg         =     .02389
Log marginal-likelihood = -16.709588      max         =     .02966
```

---

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	81.22007	63.87998	4.29587	68.31417	2.518447	237.8033
isfbs	81.65967	60.07603	4.03945	70.37466	2.035696	229.4291
age	-.0191681	.1777758	.013695	-.0154955	-.3833187	.3242438
male	-53.69173	42.4866	2.50654	-44.93144	-154.439	.7090207
_cons	59.39037	43.5938	2.53139	51.31836	.1225503	161.2943

The estimated posterior means of `{disease:restecg}`, `{disease:isfbs}`, `{disease:male}`, and `{disease:_cons}` are fairly large, roughly on the same scale as the prior standard deviation of 100.

Indeed, if we decrease the standard deviation of the priors to 10, we observe that the scale of the estimates decreases by the same order of magnitude.

```
. set seed 14
. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,100))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  disease ~ logit(xb_disease)

Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,100) (1)
```

---

```
(1) Parameters are elements of the linear form xb_disease.

Bayesian logistic regression      MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =       2,500
                                         MCMC sample size =     10,000
                                         Number of obs   =        48
                                         Acceptance rate =      .3161
                                         Efficiency: min =    .02287
                                           avg =      .0331
                                           max =      .05204

Log marginal-likelihood = -12.418273
```

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	8.559131	6.71	.443681	7.447336	-.889714	23.93564
isfbs	6.322615	6.411998	.281084	5.504684	-3.85021	20.56641
age	.0526448	.1226056	.00718	.0468937	-.1734675	.3050607
male	-3.831954	5.31727	.279435	-3.048654	-15.77187	4.451594
_cons	5.624899	6.641158	.417961	5.181183	-6.408041	20.1234

We can, therefore, conclude that the regression parameters are highly sensitive to the choice of priors and their scale cannot be determined by the data alone; that is, it cannot be determined by the likelihood of the model. In other words, these model parameters are not identifiable from the likelihood alone. This conclusion is in agreement with the results of the `logit` command.

We may consider applying an informative prior. We can use information from other heart disease studies from [Lichman \(2013\)](#). For example, we use a subset of the Hungarian data created by Andras Janosi, M.D. of Hungarian Institute of Cardiology in Budapest, Hungary. `hearthungary.dta` contains the same attributes as in `heartswitz.dta` but from a Hungarian population.



We fit bayesmh with noninformative priors to hearthungary.dta and obtain the following posterior mean estimates for the regression parameters:

```
. use https://www.stata-press.com/data/r16/hearthungary
(Substet of Hungarian heart disease data from UCI Machine Learning Repository)

. set seed 14

. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:}, normal(0,1000))
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  disease ~ logit(xb_disease)

Prior:
  {disease:restecg isfbs age male _cons} ~ normal(0,1000)                                (1)
```

---

(1) Parameters are elements of the linear form xb\_disease.

Bayesian logistic regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	285
	Acceptance rate =	.2341
	Efficiency: min =	.03088
	avg =	.04524
	max =	.06362
Log marginal-likelihood =	-195.7454	

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	-.1076298	.2931371	.013664	-.1036111	-.6753464	.4471483
isfbs	1.182073	.541182	.030797	1.169921	.2267485	2.268314
age	.042955	.0170492	.000676	.0432923	.0103757	.0763747
male	1.488844	.3612114	.018399	1.484816	.7847398	2.244648
_cons	-3.866674	.8904101	.041022	-3.869567	-5.658726	-2.112237

With this additional information, we can form more informative priors for the 5 parameters of interest—we center {restecg} and {age} at 0, {disease:isfbs} and {disease:male} at 1, and {disease:\_cons} at -4, and we use a prior variance of 10 for all coefficients.

```
. use https://www.stata-press.com/data/r16/heartswitz
(Substet of Switzerland heart disease data from UCI Machine Learning Repository)

. set seed 14

. bayesmh disease restecg isfbs age male, likelihood(logit)
> prior({disease:restecg age}, normal( 0,10))
> prior({disease:isfbs male}, normal( 1,10))
> prior({disease:_cons}, normal(-4,10))
Burn-in ...
Simulation ...

Model summary
```

---

```
Likelihood:
  disease ~ logit(xb_disease)

Priors:
  {disease:restecg age} ~ normal(0,10)                (1)
  {disease:isfbs male} ~ normal(1,10)                 (1)
  {disease:_cons} ~ normal(-4,10)                     (1)
```

---

```
(1) Parameters are elements of the linear form xb_disease.

Bayesian logistic regression          MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in          =       2,500
                                          MCMC sample size =    10,000
                                          Number of obs    =       48
                                          Acceptance rate  =     .247
                                          Efficiency: min =    .03691
                                          avg             =    .05447
                                          max             =    .06737

Log marginal-likelihood = -11.021903
```

disease	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
restecg	1.74292	2.21888	.097001	1.385537	-2.065912	6.584702
isfbs	1.885653	2.792842	.145375	1.595679	-2.976167	7.976913
age	.1221246	.0698409	.002691	.1174274	-.0078114	.2706446
male	.2631	2.201574	.089281	.2667496	-4.125275	4.646742
_cons	-2.304595	2.706482	.115472	-2.256248	-7.785531	3.098357

We now obtain more reasonable results that also agree with the Hungarian results. For the final analysis, we may consider other heart disease datasets to verify the reasonableness of our prior specifications and to check the sensitivity of the parameters to other prior specifications.

## Ordered probit regression

Ordered probit and ordered logit regressions are appropriate for modeling ordinal response variables. You can perform Bayesian analysis of an ordinal outcome by specifying the `oprobit` or `ologit` likelihood function. In addition to regression coefficients in ordered models, `bayesmh` automatically introduces parameters representing the cutpoints for the linear predictor. The cutpoint parameters are declared as `{depname:_cut1}`, `{depname:_cut2}`, and so on, where `depname` is the name of the response variable.

In the next example, we consider the full auto dataset and model the ordinal variable `rep77`, the repair record, as a function of independent variables `foreign`, `length`, and `mpg`. The variable `rep77` has 5 levels, so the cutpoint parameters are `{rep77:_cut1}`, `{rep77:_cut2}`, `{rep77:_cut3}`, and `{rep77:_cut4}`. The independent variables are all positive, so it seems reasonable to use exponential prior for the cutpoint parameters. The exponential prior is controlled by a hyperparameter `{lambda}`. Based on the range of the independent predictors, we assign `{lambda}` a prior that is uniform in

the 10 to 40 range. We assign  $N(0, 1)$  prior for regression coefficients. To monitor the progress, we specify dots to request that bayesmh displays dots every 100 iterations and iteration numbers every 1,000 iterations.

```
. use https://www.stata-press.com/data/r16/fullauto
(Automobile Models)
. replace length = length/10
variable length was int now float
(74 real changes made)
. set seed 14
. bayesmh rep77 foreign length mpg, likelihood(oprobit)
> prior({rep77: foreign length mpg}, normal(0,1))
> prior({rep77:_cut1 _cut2 _cut3 _cut4}, exponential({lambda=30}))
> prior({lambda}, uniform(10,40)) block(lambda) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

```
Likelihood:
rep77 ~ oprobit(xb_rep77,{rep77:_cut1 ... _cut4})
Priors:
{rep77:foreign length mpg} ~ normal(0,1)
{rep77:_cut1 ... _cut4} ~ exponential({lambda})
Hyperprior:
{lambda} ~ uniform(10,40)
```

(1) Parameters are elements of the linear form xb\_rep77.

Bayesian ordered probit regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	66
	Acceptance rate =	.3422
	Efficiency: min =	.02171
	avg =	.0355
	max =	.1136

Log marginal-likelihood = -102.82883

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
rep77						
foreign	1.338071	.3750768	.022296	1.343838	.6331308	2.086062
length	.3479392	.1193329	.00787	.3447806	.1277292	.5844067
mpg	.1048089	.0356498	.002114	.1022382	.0373581	.1761636
_cut1	7.204502	2.910222	.197522	7.223413	1.90771	13.07034
_cut2	8.290923	2.926149	.197229	8.258871	2.983281	14.16535
_cut3	9.584845	2.956191	.197144	9.497836	4.23589	15.52108
_cut4	10.97314	3.003014	.192244	10.89227	5.544563	17.06189
lambda	18.52477	7.252342	.215137	16.40147	10.21155	36.44309

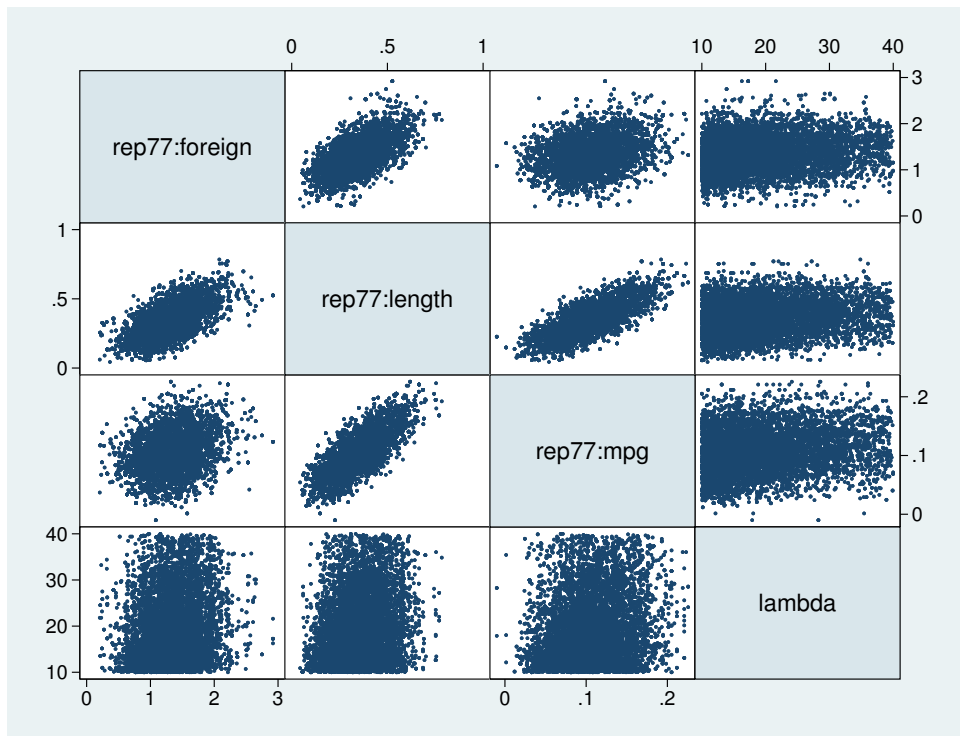
When we specify dots or dots(), bayesmh displays dots as simulation is performed. The burn-in and simulation iterations are displayed separately. During the adaptation period, iterations are displayed with a symbol a instead of a dot. This indicates the period during which the proposal distribution is still changing and thus may not be suitable for sampling from yet. Typically, adaptation is performed during the burn-in period, the iterations of which are discarded from the MCMC sample. You should pay closer attention to your results if you see adaptive iterations during the simulation period. This may happen, for example, if you increase adaptation(maxiter()) without increasing burnin()

correspondingly. In this case, you may need to perform additional checks to verify that the part of the MCMC sample corresponding to the adaptation period is similar to the rest of the sample.

Posterior credible intervals suggest that `foreign`, `length`, and `mpg` are among the explanatory factors for `rep77`. Based on MCSEs, their posterior mean estimates are fairly precise. The posterior mean estimates of cutpoints, as expected, are not as precise. The estimated posterior mean for `{lambda}` is 18.52.

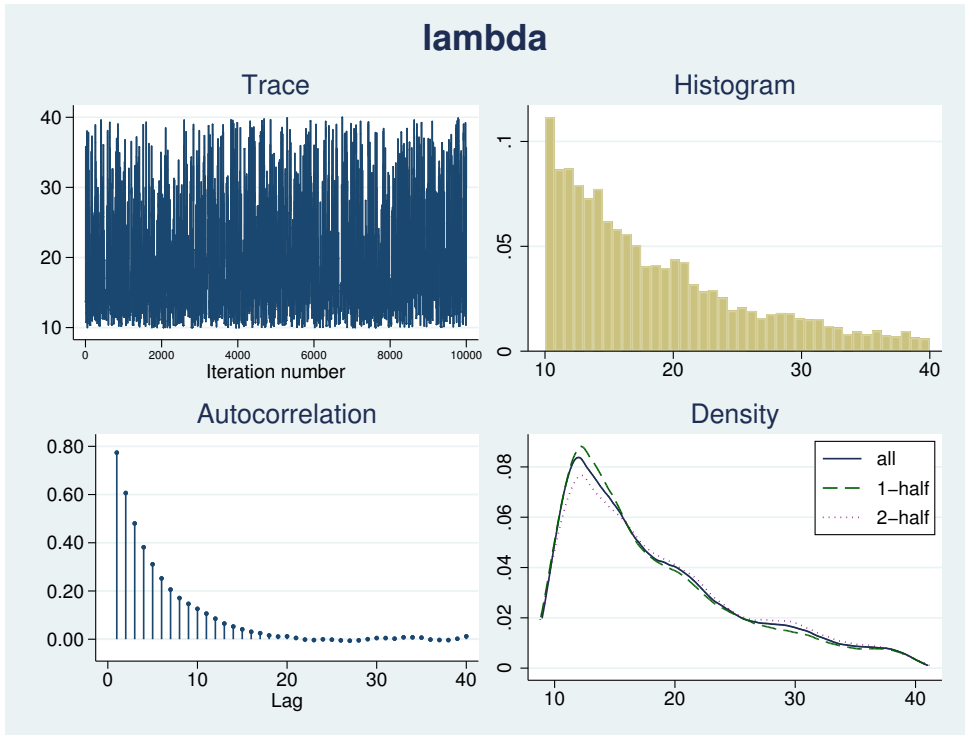
We placed the hyperparameter `{lambda}` in a separate block because we wanted to sample this nuisance parameter independently from the other model parameters. Based on the bivariate scatterplots, this parameter does appear to be independent of other model parameters a posteriori.

```
. bayesgraph matrix {rep77:foreign} {rep77:length} {rep77:mpg} {lambda}
```



As with any MCMC analysis, we should verify convergence of all of our parameters. Here we show diagnostic plots only for `{lambda}`.

```
. bayesgraph diagnostics {lambda}
```



The diagnostic plots for `{lambda}` do not cause any concern.

## Beta-binomial model

`bayesmh` is a regression command, which models the mean of the outcome distribution as a function of predictors. There are cases when we do not have any predictors and want to model the outcome distribution directly. For example, we may want to fit a Poisson distribution or a binomial distribution to our outcome. We can do this by specifying one of the four distributions supported by `bayesmh` in the `likelihood()` option: `dexponential()`, `dbernoulli()`, `dbinomial()`, or `dpoisson()`.

Let's revisit the example from *What is Bayesian analysis?* in [BAYES] **Intro**, originally from Hoff (2009, 3), of estimating the prevalence of a rare infectious disease in a small city. The outcome variable `y` is the number of infected subjects in a city of 20 subjects, and our data consist of only one observation, `y = 0`. We assume a binomial distribution for the outcome `y`, `Binom(20,  $\theta$ )`, where the infection probability  $\theta$  is a parameter of interest. Based on some previous studies, the model parameter  $\theta$  is assigned a `Beta(2, 20)` prior. For this model, the posterior distribution of  $\theta$  is known to be `Beta(2, 40)`.

To fit a binomial distribution to `y` using `bayesmh`, we specify the option `likelihood(dbinomial({theta}, 20))`. The infection probability  $\theta$  is represented by `{theta}`.

```
. set obs 1
number of observations (_N) was 0, now 1
. generate y = 0
. set seed 14
. bayesmh y, likelihood(dbinomial({theta},20))
> prior({theta}, beta(2,20)) initial({theta} 0.01)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  y ~ binomial({theta},20)
Prior:
  {theta} ~ beta(2,20)
```

---

Bayesian binomial model	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	1
	Acceptance rate =	.4527
Log marginal-likelihood = -1.1658052	Efficiency =	.1549

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
theta	.0467973	.0317862	.000808	.039931	.0051255	.1277823

The estimated posterior mean for `{theta}` is 0.0468, which is close to the theoretical value of  $2/(2 + 40) = 0.0476$  and is within the range of the MCSE of 0.0008.

## Multivariate regression

We consider a simple multivariate normal regression model without covariates. We use `auto.dta`, and we fit a multivariate normal distribution to variables `mpg`, `weight`, and `length`.

We rescale these variables to have approximately equal ranges. Equalizing the range of model variables is always recommended, because this makes the model computationally more stable.

```
. use https://www.stata-press.com/data/r16/auto, clear
(1978 Automobile Data)
. quietly replace weight = weight/1000
. quietly replace length = length/100
. quietly replace mpg = mpg/10
```

### ► Example 15: Default MH sampling with inverse-Wishart prior for the covariance

For a multivariate normal distribution, an inverse-Wishart prior is commonly used as a prior for the covariance matrix. Let's fit our multivariate model using `bayesmh`.

We specify the multivariate normal likelihood `likelihood(mvnormal({Sigma,m}))` for the three variables `mpg`, `weight`, and `length`, where `{Sigma,m}` is a matrix parameter for the covariance matrix. We use vague normal priors `normal(0,100)` for all three means of the variables. For a covariance matrix `{Sigma,m}`, which is of dimension three, we specify an inverse-Wishart prior with the identity scale matrix. We also specify the mean parameters and the covariance parameter in two separate blocks. To monitor the simulation process, we specify `dots`.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
```

Model summary

```
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})

Priors:
  {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
  {Sigma,m} ~ iwishart(3,100,I(3))
```

Bayesian multivariate normal regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.3255
	Efficiency: min	=	.001396
	avg	=	.04166
	max	=	.1111
Log marginal-likelihood = -254.88899			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	2.13089	.0455363	.001763	2.129007	2.04435	2.223358
weight						
_cons	3.018691	.0671399	.00212	3.020777	2.880051	3.149828
length						
_cons	1.879233	.0210167	.00063	1.879951	1.837007	1.920619
Sigma_1_1	.1571554	.0038157	.000183	.1570586	.1499028	.1648159
Sigma_2_1	-.1864936	.0024051	.000343	-.1864259	-.1912537	-.18194
Sigma_3_1	-.0533863	.0033667	.000199	-.053342	-.0601722	-.0468986
Sigma_2_2	.3293518	.0044948	.001203	.329703	.3193904	.3366703
Sigma_3_2	.0894404	.0040487	.000471	.0894156	.0816045	.0976702
Sigma_3_3	.0329253	.002521	.00024	.0328027	.0285211	.0383005

Note: There is a high autocorrelation after 500 lags.

In this first run, we do not achieve good mixing of the MCMC chain. bayesmh issues a note about significant autocorrelation of the simulated parameters.

A closer inspection of the ESS table reveals very low sampling efficiencies for the elements of the covariance matrix {Sigma}.

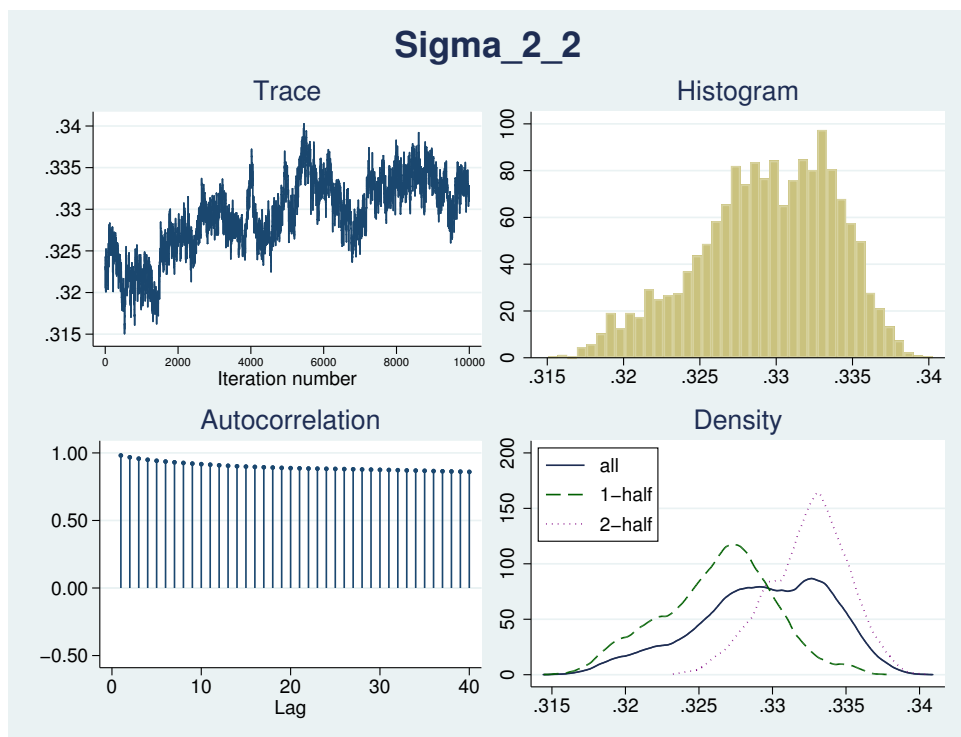
```
. bayesstats ess
```

```
Efficiency summaries      MCMC sample size =    10,000
                          Efficiency: min =    .001396
                              avg =    .04166
                              max =    .1111
```

	ESS	Corr. time	Efficiency
mpg			
_cons	667.48	14.98	0.0667
weight			
_cons	1002.92	9.97	0.1003
length			
_cons	1111.14	9.00	0.1111
Sigma_1_1	433.25	23.08	0.0433
Sigma_2_1	49.03	203.96	0.0049
Sigma_3_1	287.03	34.84	0.0287
Sigma_2_2	13.96	716.45	0.0014
Sigma_3_2	73.76	135.57	0.0074
Sigma_3_3	110.41	90.58	0.0110

For example, the diagnostic plots for {Sigma\_2\_2} provide visual confirmation of the convergence issues—very poorly mixing trace plot, high autocorrelation, and a bimodal posterior distribution.

```
. bayesgraph diagnostics Sigma_2_2
```





Here, we see a general problem associated with the simulation of covariance matrices. Random-walk MH algorithm is not well suited for sampling positive-definite matrices. This is why even an adaptive version of the MH algorithm, as implemented in bayesmh, may not achieve good mixing. ↴

➤ Example 16: Adaptation of MH sampling with inverse-Wishart prior for the covariance

Continuing example 15, we can specify longer adaptation and burn-in periods to improve convergence.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}) dots burnin(5000) adaptation(maxiter(50))
Burn-in 5000 aaaaaaaaa1000aaaaaaaa2000aaaaaaaa3000aaaaa.....4000.....5000
> done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

```
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
  {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
  {Sigma,m} ~ iwishart(3,100,I(3))
```

Bayesian multivariate normal regression	MCMC iterations	=	15,000
Random-walk Metropolis-Hastings sampling	Burn-in	=	5,000
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.2382
	Efficiency: min	=	.02927
	avg	=	.05053
	max	=	.07178
Log marginal-likelihood = -245.83844			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	2.13051	.0475691	.001809	2.13263	2.038676	2.220953
weight						
_cons	3.017943	.0626848	.00234	3.016794	2.898445	3.143252
length						
_cons	1.878912	.019905	.000769	1.878518	1.840311	1.918476
Sigma_1_1	.1711394	.0089943	.000419	.1706437	.1548036	.1898535
Sigma_2_1	-.1852432	.002432	.000126	-.1852973	-.1898398	-.1803992
Sigma_3_1	-.0517404	.0035831	.000201	-.051688	-.058747	-.0449874
Sigma_2_2	.3054418	.0144859	.000551	.3055426	.2783409	.3340654
Sigma_3_2	.0809091	.0057474	.000314	.080709	.0698331	.0924053
Sigma_3_3	.030056	.002622	.000153	.0299169	.0251627	.0355171

There is no note about high autocorrelation, and the average efficiency increases slightly from 4% to 5%.

Sampling efficiencies of the elements of the covariance matrix improved substantially.

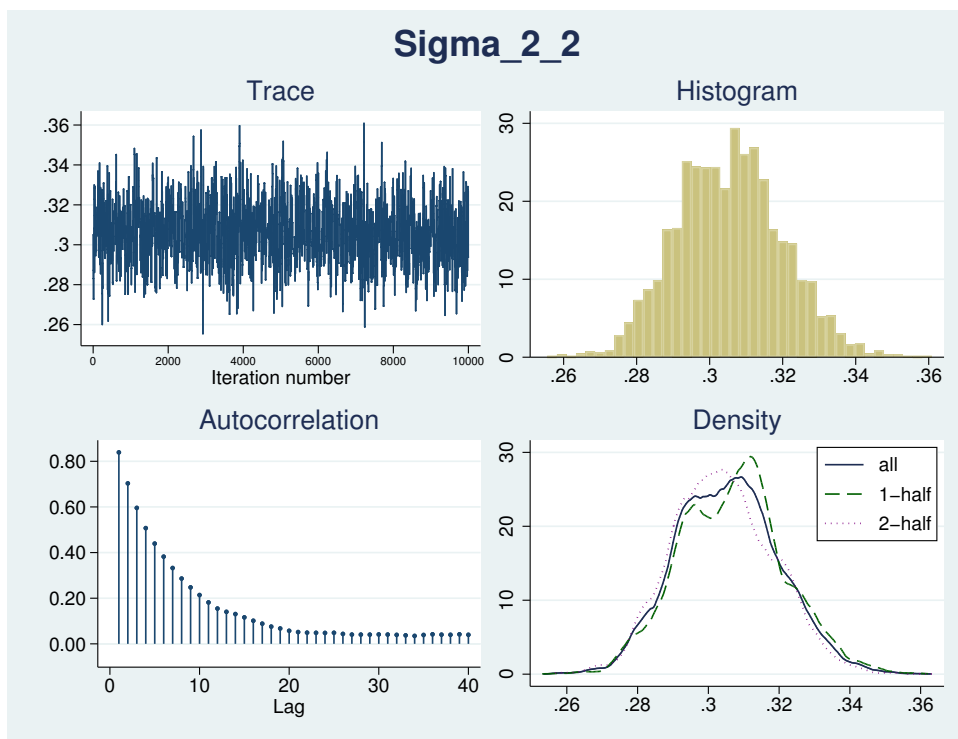
```
. bayesstats ess
```

```
Efficiency summaries      MCMC sample size =    10,000
                          Efficiency: min =     .02927
                              avg =     .05053
                              max =     .07178
```

	ESS	Corr. time	Efficiency
mpg			
_cons	691.54	14.46	0.0692
weight			
_cons	717.82	13.93	0.0718
length			
_cons	670.63	14.91	0.0671
Sigma_1_1	459.78	21.75	0.0460
Sigma_2_1	370.45	26.99	0.0370
Sigma_3_1	318.91	31.36	0.0319
Sigma_2_2	692.06	14.45	0.0692
Sigma_3_2	334.08	29.93	0.0334
Sigma_3_3	292.70	34.16	0.0293

The diagnostic plots for {Sigma\_2\_2} look much better.

```
. bayesgraph diagnostics Sigma_2_2
```



➤ Example 17: Gibbs sampling of a covariance matrix

Continuing [example 15](#), the convergence of the chain can be greatly improved if we use Gibbs sampling for the covariance matrix parameter. For a multivariate normal model, inverse Wishart is a conjugate prior, or more precisely semiconjugate prior, for the covariance matrix and thus Gibbs sampling is available. To request Gibbs sampling, we only need to add the `gibbs` suboption to the block specification of `{Sigma,m}`. The mean parameters are still updated by the random-walk MH algorithm.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:_cons} {weight:_cons} {length:_cons}, normal(0,100))
> prior({Sigma,m}, iwishart(3,100,I(3)))
> block({mpg:_cons} {weight:_cons} {length:_cons})
> block({Sigma,m}, gibbs) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaa.. done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

```
Likelihood:
  mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})
Priors:
  {mpg:_cons} ~ normal(0,100)
  {weight:_cons} ~ normal(0,100)
  {length:_cons} ~ normal(0,100)
  {Sigma,m} ~ iwishart(3,100,I(3))
```

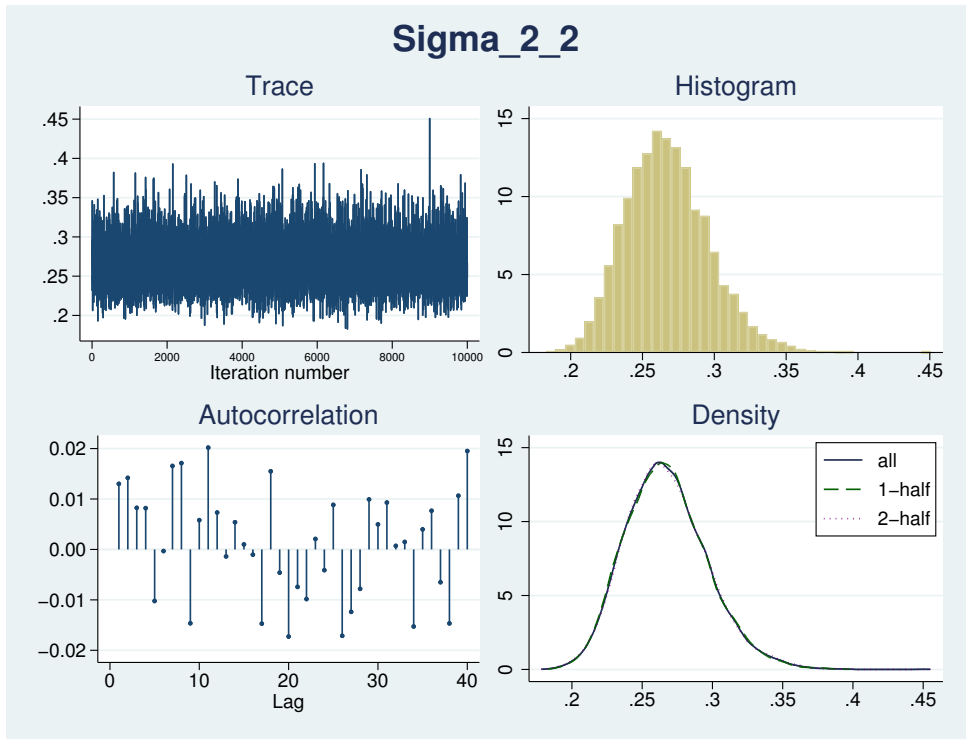
Bayesian multivariate normal regression	MCMC iterations	=	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.5942
	Efficiency: min	=	.06842
	avg	=	.6659
	max	=	.9781
Log marginal-likelihood = -240.48717			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	2.128801	.0457224	.00164	2.128105	2.041016	2.215
weight						
_cons	3.020533	.0609036	.002328	3.021561	2.908383	3.143715
length						
_cons	1.880409	.0197061	.000725	1.881133	1.843106	1.918875
Sigma_1_1	.150733	.0164464	.000166	.1495231	.1219304	.1869429
Sigma_2_1	-.1571622	.0196803	.000201	-.156005	-.1995812	-.1224243
Sigma_3_1	-.0443725	.0060229	.000061	-.0439466	-.0571876	-.0338685
Sigma_2_2	.2673525	.029205	.0003	.2654589	.2163041	.3305366
Sigma_3_2	.0708095	.0085435	.000087	.0702492	.0557448	.0893794
Sigma_3_3	.0273506	.0029932	.000031	.0271362	.0220723	.0337994

Compared with [example 15](#), the results improved substantially. Compared with [example 16](#), the minimum efficiency increases from about 3% to 7% and the average efficiency from 5% to 67%. MCSEs of posterior mean estimates, particularly for elements of `{Sigma}`, are lower.

The diagnostic plots, for example, for `Sigma_2_2` also indicate a very good convergence.

```
. bayesgraph diagnostics Sigma_2_2
```



➤ Example 18: Gibbs sampling of a covariance matrix with the Jeffreys prior

In this example, we perform a sensitivity analysis of the model by replacing the inverse-Wishart prior for the covariance matrix with a Jeffreys prior.

```
. set seed 14
. bayesmh (mpg) (weight) (length), likelihood(mvnormal({Sigma,m}))
> prior({mpg:} {weight:} {length:}, normal(0,100))
> prior({Sigma,m}, jeffreys(3))
> block({mpg:} {weight:} {length:})
> block({Sigma,m}, gibbs) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

Likelihood:		mpg weight length ~ mvnormal(3,{mpg:},{weight:},{length:},{Sigma,m})	
Priors:		{mpg:_cons} ~ normal(0,100)	
		{weight:_cons} ~ normal(0,100)	
		{length:_cons} ~ normal(0,100)	
		{Sigma,m} ~ jeffreys(3)	
Bayesian multivariate normal regression		MCMC iterations =	12,500
Metropolis-Hastings and Gibbs sampling		Burn-in =	2,500
		MCMC sample size =	10,000
		Number of obs =	74
		Acceptance rate =	.6223
		Efficiency: min =	.08573
		avg =	.6886
Log marginal-likelihood = -42.728723		max =	1

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	2.130704	.0709095	.002185	2.129449	1.989191	2.267987
weight						
_cons	3.019323	.0950116	.003245	3.019384	2.834254	3.208017
length						
_cons	1.879658	.0271562	.000892	1.879859	1.827791	1.933834
Sigma_1_1	.3596673	.0628489	.000628	.3526325	.2575809	.5028854
Sigma_2_1	-.3905511	.0772356	.000772	-.3824458	-.5668251	-.2654059
Sigma_3_1	-.1103824	.0220164	.000223	-.1077659	-.1611913	-.0751177
Sigma_2_2	.6503219	.1141333	.001141	.6378476	.466738	.9140429
Sigma_3_2	.1763159	.0318394	.000323	.1725042	.1248434	.2507866
Sigma_3_3	.0533981	.0093631	.000095	.0522228	.0382405	.0748096

Note: Adaptation tolerance is not met in at least one of the blocks.

Compared with [example 17](#), the estimates of the means of the multivariate distribution do not change much, but the estimates of the elements of the covariance matrix do change. The estimates for {Sigma,m} obtained using the Jeffreys prior are approximately twice as big as the estimates obtained using the inverse-Wishart prior. If we compute correlation matrices corresponding to {Sigma,m} from the two models, they will be similar. This can be explained by the fact that both the Jeffreys prior and the inverse-Wishart prior with identity scale matrix are not informative for the correlation structure

because they only depend on the determinant and the trace of  $\{\text{Sigma}, m\}$  whereas the correlation structure is determined by the data alone.

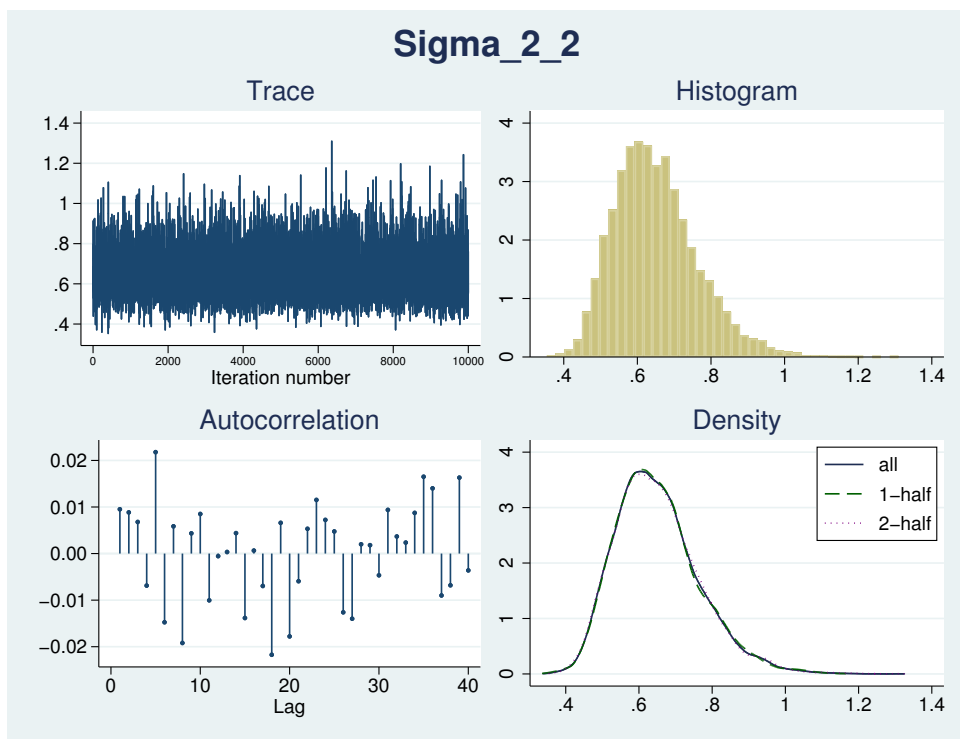
#### ❑ Technical note: Adaptation tolerance is not met

At the bottom of the table in the previous output, the note about the adaptation tolerance not being met in one of the blocks is displayed. Adaptation is part of MH sampling, so the note refers to the block of regression coefficients. This note does not necessarily indicate a problem. It simply notifies you that the default target acceptance rate as specified in `adaptation(tarate())` has not been reached within the tolerance specified in `adaptation(tolerance())`. The used default for the target acceptance rate corresponds to the theoretical asymptotically optimal acceptance rate of 0.44 for a block with one parameter and 0.234 for a block with multiple parameters. The rate is derived for a specific class of models and does not necessarily represent the optimal rate for all models. If your MCMC converged, you can safely ignore this note. Otherwise, you need to investigate your model further. One remedy is to increase the burn-in period, which automatically increases the adaptation period, or more specifically, the number of adaptive iterations as controlled by `adaptation(maxiter())`. For example, if we increase burn-in to 3,000 by specifying option `burnin(3000)` in the above example, we will meet the adaptation tolerance.

❑

The diagnostic plots of `Sigma_2_2` demonstrate excellent mixing properties.

```
. bayesgraph diagnostics Sigma_2_2
```



◀

Panel-data and multilevel models

Although the MH algorithm underlying **bayesmh** is not optimal for fitting Bayesian multilevel models, you can use it to fit some multilevel models that do not have too many random effects. Below we consider two-level random-intercept and random-coefficients models. A two-level random-effects model is also known as a panel-data model.

Two-level random-intercept model or panel-data model

Ruppert, Wand, and Carroll (2003) and Diggle et al. (2002) analyzed a longitudinal dataset consisting of **weight** measurements of 48 pigs on 9 successive **weeks**. Pigs were identified by the group variable **id**.

The following two-level model was considered:

$$\text{weight}_{ij} = \beta_0 + \beta_1 \text{week}_{ij} + u_j + \epsilon_{ij}$$

where  $u_j$  is the random effect for pig  $j$ ,  $j = 1, \dots, 48$ , and the counter  $i = 1, \dots, 9$  identifies the weeks.

We first use **mixed** to fit this model by using maximum likelihood for comparison purposes; see [ME] **mixed**.

```
. use https://www.stata-press.com/data/r16/pig, clear
(Longitudinal analysis of pig weights)

. mixed weight week || id:
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -1014.9268
Iteration 1:   log likelihood = -1014.9268

Computing standard errors:
Mixed-effects ML regression              Number of obs   =       432
Group variable: id                      Number of groups =        48
Obs per group:
                                         min =          9
                                         avg =         9.0
                                         max =          9
                                         Wald chi2(1)    =    25337.49
Log likelihood = -1014.9268              Prob > chi2      =      0.0000
```

weight	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974059	32.40	0.000	18.18472	20.52651

Random-effects Parameters		Estimate	Std. Err.	[95% Conf. Interval]	
id: Identity	var(_cons)	14.81751	3.124226	9.801716	22.40002
	var(Residual)	4.383264	.3163348	3.805112	5.04926

LR test vs. linear model: **chibar2**(01) = 472.65                      Prob >= **chibar2** = 0.0000

Consider the following Bayesian model for these data:

$$\begin{aligned}\text{weight}_{ij} &= \beta_0 + \beta_1 \text{week}_{ij} + u_j + \epsilon_{ij} = \beta_1 \text{week}_{ij} + \tau_j + \epsilon_{ij}, \\ \epsilon_{ij} &\sim \text{i.i.d. } N(0, \sigma_0^2) \\ \tau_j &\sim \text{i.i.d. } N(\beta_0, \sigma_{\text{id}}^2) \\ \beta_0 &\sim N(0, 100) \\ \beta_1 &\sim N(0, 100) \\ \sigma_0^2 &\sim \text{InvGamma}(0.001, 0.001) \\ \sigma_{\text{id}}^2 &\sim \text{InvGamma}(0.001, 0.001)\end{aligned}$$

The model has four main parameters of interest: regression coefficients  $\beta_0$  and  $\beta_1$  and variance components  $\sigma_0^2$  and  $\sigma_{\text{id}}^2$ .  $\beta_0$  is actually a hyperparameter in this example, because it is the mean parameter of the prior distribution for random effects  $\tau_j$ . The pig random effects  $\tau_j$  are considered nuisance parameters. We use normal priors for the regression coefficients and group levels identified by the `id` variable and inverse-gamma priors for the variance parameters. The chosen priors are fairly noninformative, so we would expect results to be similar to the frequentist results.

To fit this model using `bayesmh`, we need to include random effects for pig in our regression model. This can be done by adding factor levels of the `id` variable to the regression by using the factor-variable specification `i.id`. This specification, by default, will omit one of the `id` categories as a base category. In our Bayesian model, we need to keep all categories of `id`, so we use `fvset` to declare no base for the `id` variable.

```
. fvset base none id
```

In addition to two regression coefficients and two variance components, we have 48 random-effects parameters. As for other models, `bayesmh` will automatically create parameters of the regression function: `{weight:week}` for the regression coefficient of `week` and `{weight:1.id}`, `{weight:2.id}`, ..., `{weight:48.id}` for random effects. We do not include a constant in our regression function because it is modeled as a mean of random effects in their prior. So, we need to define the three remaining model parameters manually; we will use `{weight:_cons}` for the mean of random effects, `{var_id}` for the variance of random effects, and `{var_0}` for the error variance.

We will perform five simulations for the specified Bayesian model to illustrate some common difficulties in applying MH MCMC to multilevel models.

## ► Example 19: First simulation—default MH settings

In the first simulation, we use default simulation settings of the MH algorithm. We have many parameters in our model, so the simulation will take a few moments. For exploration purposes and to expedite results, here we use a smaller MCMC size of 5,000 instead of the default of 10,000. To monitor the progress of the simulation, we also specify `dots`.



```
. set seed 14
. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
>   prior({weight:i.id}, normal({weight:_cons},{var_id}))
>   prior({weight:_cons}, normal(0, 100))
>   prior({weight:week}, normal(0, 100))
>   prior({var_0}, igamma(0.001, 0.001))
>   prior({var_id}, igamma(0.001, 0.001))
>   mcmcsize(5000) dots
Burn-in 2500 aaaaaaaa.1000.....2000..... done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done

Model summary
```

---

Likelihood:  
weight ~ normal(xb\_weight,{var\_0})

Priors:  
{weight:i.id} ~ normal({weight:\_cons},{var\_id}) (1)  
{weight:week} ~ normal(0,100) (1)  
{var\_0} ~ igamma(0.001,0.001)  
{weight:\_cons} ~ normal(0,100)

Hyperprior:  
{var\_id} ~ igamma(0.001,0.001)

(1) Parameters are elements of the linear form xb\_weight.

Bayesian normal regression	MCMC iterations =	7,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	5,000
	Number of obs =	432
	Acceptance rate =	.2382
	Efficiency: min =	.00136
	avg =	.004915
	max =	.03084

Log marginal-likelihood = -1483.9819

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.263434	.0264724	.002955	6.262433	6.214032	6.31423
id						
1	16.24666	.2357628	.058097	16.2599	15.78635	16.67799
2	24.06862	.3243331	.06509	24.07464	23.37339	24.67859
(output omitted)						
47	29.73823	.3734104	.07144	29.71473	29.04301	30.48604
48	20.82722	.4258745	.160651	20.78619	20.13018	21.71069
var_0	9.218097	.5679745	.174024	9.181747	8.218479	10.38655
weight						
_cons	13.59053	.3519081	.028341	13.62244	12.88323	14.25594
var_id	12.49858	.3116721	.050076	12.50611	11.9335	13.12018

Note: There is a high autocorrelation after 500 lags.

bayesmh reports the presence of a high correlation after 500 lags. This and the low average efficiency of 0.005 may indicate problems with MCMC convergence for some of the parameters.

For convenience, we use `bayesstats summary` to show posterior summaries for parameters of interest only. Alternatively, you can specify the `noshow(i.id)` option with `bayesmh` to suppress the summaries for factor levels.

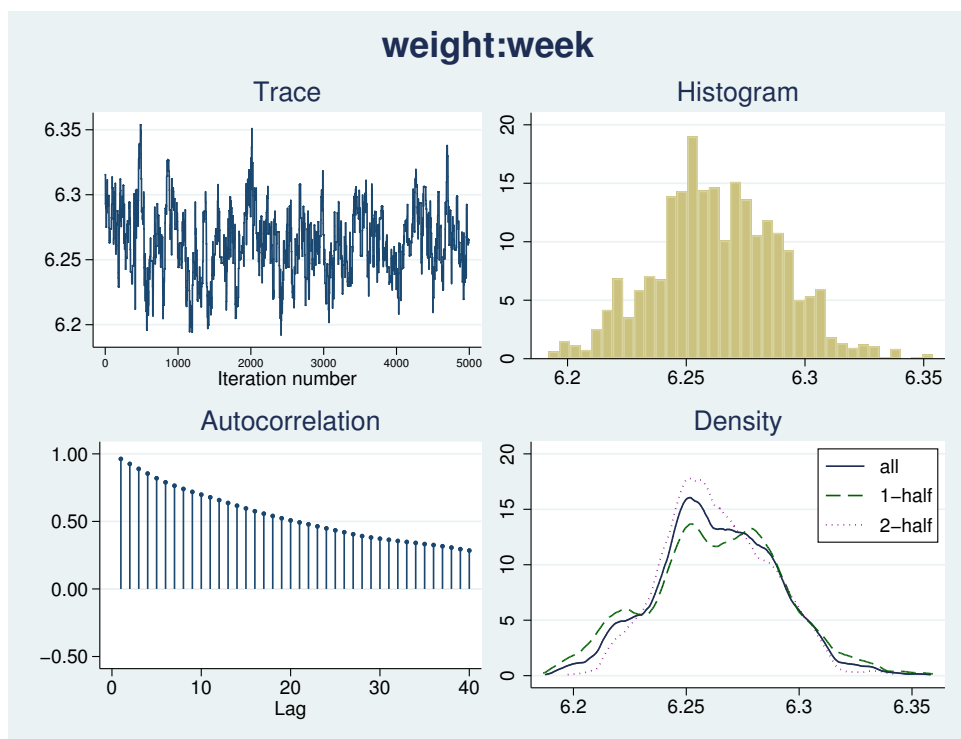
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
```

Posterior summary statistics					MCMC sample size = 5,000	
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.263434	.0264724	.002955	6.262433	6.214032	6.31423
_cons	13.59053	.3519081	.028341	13.62244	12.88323	14.25594
var_0	9.218097	.5679745	.174024	9.181747	8.218479	10.38655
var_id	12.49858	.3116721	.050076	12.50611	11.9335	13.12018

The posterior mean estimates for `{weight:week}` and `{weight:_cons}` are 6.26 and 13.59, respectively. The estimate for the residual variance `{var_0}` is 9.22 with the standard deviation of 0.57, and the estimate of the group-effect variance `{var_id}` is 12.5 with the standard deviation of 0.31.

Because of the low efficiencies, we should be suspicious of these results. If we look at diagnostic plots for, for example, `{weight:week}`,

```
. bayesgraph diagnostics {weight:week}
```



we see that the trace plot exhibits some trend and does not show good mixing and that the autocorrelation is relatively high after at least lag 40. Our MCMC does not seem to converge and thus we cannot trust the obtained results.



➤ Example 20: Second simulation—blocking of parameters

Continuing [example 19](#), we can improve efficiency of the MH algorithm by separating model parameters into blocks to be sampled independently. We consider a separate block for each model parameter with random-effects parameters sharing the same block. We also specify `nomodelsummary` to suppress the model summary and `notable` to suppress the table output of `bayesmh`.

```
. set seed 14
. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons},normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({var_0})
> block({var_id})
> block({weight:i.id})
> block({weight:week})
> block({weight:_cons})
> burnin(3000) mcmcsize(5000) dots notable nomodelsummary
Burn-in 3000 aaaaaaaaaa1000aaaaaaaaa2000aaaaaaaaa3000 done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done

Bayesian normal regression                                MCMC iterations =      8,000
Random-walk Metropolis-Hastings sampling                 Burn-in           =      3,000
                                                         MCMC sample size =      5,000
                                                         Number of obs     =       432
                                                         Acceptance rate    =       .4194
                                                         Efficiency: min    =     .001727
                                                         avg                =     .01731
                                                         max                =     .2403

Log marginal-likelihood = -1204.9586
```

Blocking certainly improved efficiencies: the average efficiency is now 0.017, but we still have a note about high autocorrelation.

We use `bayesstats summary` below to report summaries of only model parameters of interest.

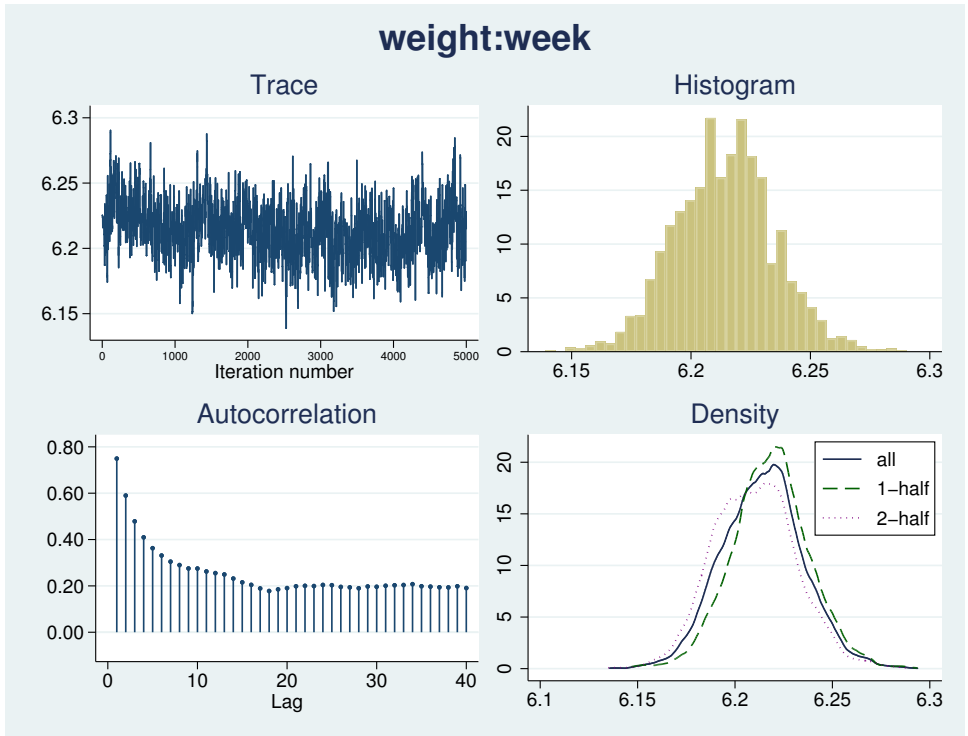
```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
```

Posterior summary statistics				MCMC sample size = 5,000		
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.214099	.020815	.002059	6.214429	6.174678	6.255888
_cons	19.28371	.552023	.015925	19.28177	18.2078	20.35016
var_0	4.183143	.2908152	.009833	4.167876	3.669035	4.828092
var_id	15.53468	3.251813	.112054	15.16295	10.46451	23.19296

Here our estimates of variance components change noticeably: `{var_0}` is 4.18 and `{var_id}` is 15.53.

The diagnostic plots for `{weight:week}` are much better, but the mixing of MCMC is still not great.

```
. bayesgraph diagnostics {weight:week}
```



◀

### ► Example 21: Third simulation—Gibbs sampling

The most efficient MCMC procedure for our Bayesian model is Gibbs sampling, which can be set up as follows. To request a Gibbs sampling for a block of model parameters, we must first define them in a separate `prior()` statement and then put them in a separate `block()` with the `gibbs` suboption.

```
. set seed 14

. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons},normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({var_0}, gibbs) block({var_id}, gibbs)
> block({weight:i.id}, gibbs) block({weight:week}, gibbs)
> block({weight:_cons},gibbs) mcmcsize(5000) dots notable nomodelsummary
Burn-in 2500 .....1000.....2000..... done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done

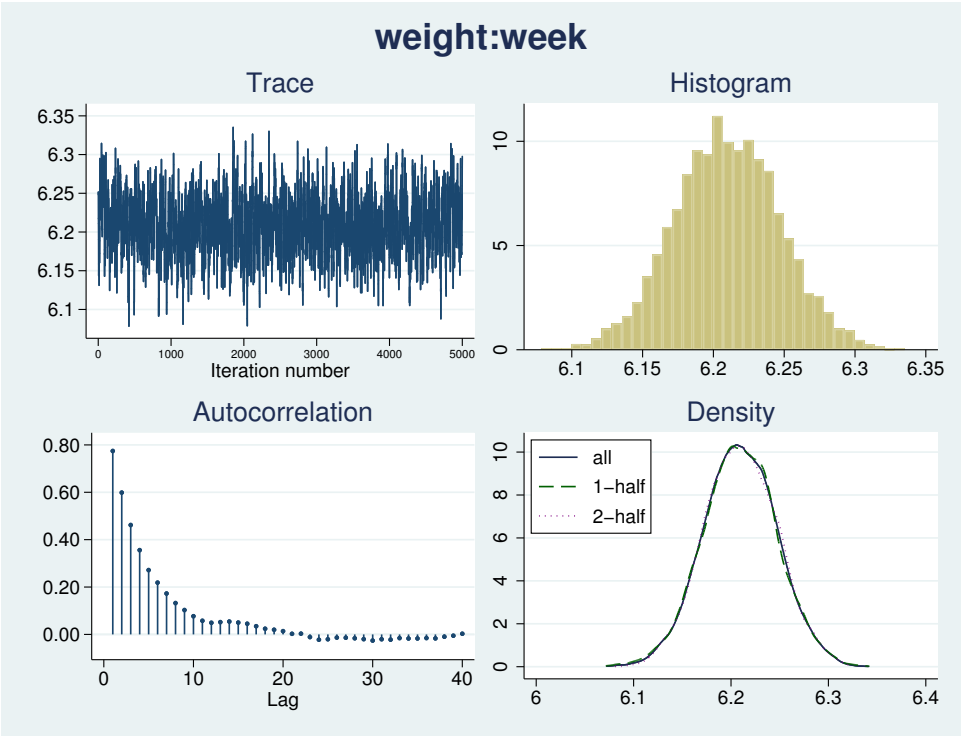
Bayesian normal regression                                MCMC iterations =      7,500
Gibbs sampling                                           Burn-in           =      2,500
                                                         MCMC sample size =    5,000
                                                         Number of obs    =     432
                                                         Acceptance rate  =        1
                                                         Efficiency: min  =     .123
                                                         avg              =     .6764
                                                         max              =     .857

Log marginal-likelihood = -1051.4228
```

There is no note about high autocorrelation in this run. The average efficiency increased dramatically to 0.68. It appears that our MCMC has now converged.

If we again inspect the diagnostic plots of, for example, {weight:week}, we will now see a very good mixing.

```
. bayesgraph diagnostics {weight:week}
```



We again use `bayesstats summary` to see posterior summaries of the model parameters of interest.

```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
Posterior summary statistics                                MCMC sample size =      5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
<b>weight</b>						
week	6.209425	.0373593	.001507	6.209439	6.135128	6.282676
_cons	19.29971	.6097913	.012916	19.2999	18.11953	20.47267
<b>var_0</b>	4.414173	.3194018	.004992	4.396302	3.828712	5.099535
<b>var_id</b>	15.85026	3.45786	.052824	15.44261	10.34387	23.6678

With Gibbs sampling, our estimates change only slightly. For example, the estimates of variance components are 4.41 for `{var_0}` and 15.85 for `{var_id}`.

All estimates are very close to the MLEs obtained [earlier](#) with the mixed command.

◀

## ► Example 22: Fourth simulation—splitting random-effects parameters

Gibbs sampling typically provides the most efficient sampling of parameters. Full Gibbs sampling is not always available; see, for example, [Mixed-effects logistic regression](#) below.

In the absence of Gibbs sampling for random effects, `block()`'s suboption `split` provides the next most efficient way of sampling the random-effects parameters in `bayesmh`. Taking into account conditional independence of individual random effects, random-effects parameters associated with levels of the grouping variable can be sampled sequentially (as separate blocks) instead of being sampled jointly from a high-dimensional proposal distribution (as in [example 20](#)).

For example, instead of using Gibbs sampling for the random effects (as in [example 21](#)), we use `block()`'s suboption `split` for the random-effects parameters `{weight:i.id}`.

```
. set seed 14
. bayesmh weight week i.id, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons}, normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({weight:_cons}, gibbs) block({weight:week}, gibbs)
> block({var_0}, gibbs) block({var_id}, gibbs)
> block({weight:i.id}, split)
> mcmcsize(5000) dots notable nomodelsummary
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done

Bayesian normal regression                                MCMC iterations =      7,500
Metropolis-Hastings and Gibbs sampling                    Burn-in      =      2,500
                                                           MCMC sample size =      5,000
                                                           Number of obs   =       432
                                                           Acceptance rate =     .4823
                                                           Efficiency: min =     .04123
                                                           avg            =     .1773
                                                           max            =     .7524

Log marginal-likelihood = -1050.2963
```

We use `bayesstats summary` to see posterior summaries of the model parameters of interest.

```
. bayesstats summary {weight:week _cons} {var_0} {var_id}
```

Posterior summary statistics				MCMC sample size = 5,000		
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight	6.206316	.0399631	.002783	6.206429	6.127974	6.28349
	19.31371	.6125276	.019648	19.31878	18.08646	20.52478
var_0	4.4213	.3205769	.006464	4.407209	3.825247	5.085138
var_id	15.74962	3.448178	.056218	15.32605	10.25279	23.57063

The estimated posterior means are close to those obtained with the full Gibbs sampler in [example 21](#), although the estimated MCMC standard errors are slightly higher. For example, the MCSE of `{var_0}` rises from 0.0050 to 0.0065, or about 30%.

The average sampling efficiency, 18%, is not as high as with the full Gibbs sampling in [example 21](#) but is still high enough for reliable estimation. The caveat with using the `split` option for sampling the random-effects parameters is a significant decrease in speed. When speed is an issue or when the number of random-effects parameters is large, the `reffects()` option may be a better alternative; see [example 23](#).



► **Example 23:** Fifth simulation—using the `reffects()` option

The `reffects()` option supported by `bayesmh` can be used for specifying the two-level random-intercept model considered in this series of examples. It allows for faster MCMC sampling of the parameters associated with a random-effects variable compared with `block()`'s suboption `split` (see [example 22](#)).

We modify the syntax from [example 22](#) as follows. We exclude `i.id` from the list of independent variables and add the `reffects(id)` option. We also omit the `block({weight:i.id}, split)` option because the blocking of the `{weight:i.id}` parameters is handled implicitly by the `reffects()` option.

```
. set seed 14

. bayesmh weight week, reffects(id) likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:_cons}, normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> block({weight:_cons}, gibbs) block({weight:week}, gibbs)
> block({var_0}, gibbs) block({var_id}, gibbs)
> mcmcsize(5000) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaa2000aaaaaa done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done
```

Model summary

Likelihood:  
weight ~ normal(xb\_weight,{var\_0})

Priors:  
{weight:i.id} ~ normal({weight:\_cons},{var\_id}) (1)  
{weight:week} ~ normal(0,100) (1)  
{var\_0} ~ igamma(0.001,0.001)  
{weight:\_cons} ~ normal(0,100)

Hyperprior:  
{var\_id} ~ igamma(0.001,0.001)

(1) Parameters are elements of the linear form xb\_weight.

Bayesian normal regression	MCMC iterations	=	7,500
Metropolis-Hastings and Gibbs sampling	Burn-in	=	2,500
	MCMC sample size	=	5,000
	Number of obs	=	432
	Acceptance rate	=	.8475
	Efficiency: min	=	.03221
	avg	=	.3708
	max	=	.77

Log marginal-likelihood = -1050.8235

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
weight						
week	6.209593	.040908	.003224	6.210008	6.127663	6.288345
var_0	4.412871	.3171205	.006625	4.406072	3.834571	5.067036
weight						
_cons	19.29717	.634793	.01903	19.28847	18.0127	20.53934
var_id	15.89952	3.549986	.057213	15.46713	10.36942	24.02605

Our estimates of the variance components do not change noticeably from those in [examples 21 and 22](#): {var\_0} is 4.41 and {var\_id} is 15.90.

Although the average efficiency, 0.37, of the displayed parameters is lower than the corresponding efficiency of the full Gibbs sampler in [example 21](#), the application of the `reffects()` option results in consuming about 35% less memory during simulation and a 25% improvement in speed. The real benefit of the `reffects()` option, however, becomes apparent for models with many random-effects levels and models for which full Gibbs samplers are not available; see [Mixed-effects logistic regression](#) below.



When we use option `reffects()`, `bayesmh` suppresses the estimates of random-effects parameters from the output. You can use the `showreffects()` or `show()` option to display them.



**Linear growth curve model—a random-coefficient model**

Continuing our pig data example from *Two-level random-intercept model or panel-data model*, we extend the random-intercept model to include random coefficients for `week` by using

$$\text{weight}_{ij} = \beta_0 + \beta_1 \text{week}_{ij} + u_{0j} + u_{1j} \text{week}_{ij} + \epsilon_{ij}$$

where  $u_{0j}$  is the random effect for pig and  $u_{1j}$  is the pig-specific random coefficient on `week` for  $j = 1, \dots, 48$  and  $i = 1, \dots, 9$ .

► **Example 24: Independent covariance structure for the random effects**

Let us first assume that the random effects  $u_{0j}$ ’s and  $u_{1j}$ ’s are independent. We can use `mixed` to fit this model by using maximum likelihood.

```
. use https://www.stata-press.com/data/r16/pig
(Longitudinal analysis of pig weights)

. mixed weight week || id: week
Performing EM optimization:
Performing gradient-based optimization:
Iteration 0:   log likelihood = -869.03825
Iteration 1:   log likelihood = -869.03825
Computing standard errors:
Mixed-effects ML regression              Number of obs   =       432
Group variable: id                      Number of groups =        48
Obs per group:
      min =          9
      avg =       9.0
      max =          9
Wald chi2(1)      =    4689.51
Prob > chi2       =      0.0000

Log likelihood = -869.03825
```

weight	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
week	6.209896	.0906819	68.48	0.000	6.032163	6.387629
_cons	19.35561	.3979159	48.64	0.000	18.57571	20.13551

Random-effects Parameters		Estimate	Std. Err.	[95% Conf. Interval]	
id: Independent	var(week)	.3680668	.0801181	.2402389	.5639103
	var(_cons)	6.756364	1.543503	4.317721	10.57235
	var(Residual)	1.598811	.1233988	1.374358	1.85992

LR test vs. linear model: `chi2(2) = 764.42` Prob > chi2 = 0.0000  
Note: LR test is conservative and provided only for reference.

Consider the following Bayesian model for these data:

$$\text{weight}_{ij} = \beta_0 + \beta_1 \text{week}_{ij} + u_{0j} + u_{1j} \text{week}_{ij} + \epsilon_{ij} = \tau_{0j} + \tau_{1j} \text{week}_{ij} + \epsilon_{ij},$$

$$\epsilon_{ij} \sim \text{i.i.d. } N(0, \sigma_0^2)$$

$$\tau_{0j} \sim \text{i.i.d. } N(\beta_0, \sigma_{\text{id}}^2)$$

$$\tau_{1j} \sim \text{i.i.d. } N(\beta_1, \sigma_{\text{week}}^2)$$

$$\beta_0 \sim N(0, 100)$$

$$\beta_1 \sim N(0, 100)$$

$$\sigma_0^2 \sim \text{InvGamma}(0.001, 0.001)$$

$$\sigma_{\text{id}}^2 \sim \text{InvGamma}(0.001, 0.001)$$

$$\sigma_{\text{week}}^2 \sim \text{InvGamma}(0.001, 0.001)$$

The model has five main parameters of interest: regression coefficients  $\beta_0$  and  $\beta_1$  and variance components  $\sigma_0^2$ ,  $\sigma_{\text{id}}^2$ , and  $\sigma_{\text{week}}^2$ .  $\beta_0$  and  $\beta_1$  are hyperparameters because they are specified as mean parameters of the prior distributions for random effects  $\tau_{0j}$  and  $\tau_{1j}$ , respectively. Random effects  $\tau_{0j}$  and  $\tau_{1j}$  are considered nuisance parameters. We again use normal priors for the regression coefficients and group levels identified by the `id` variable and their interactions with `week` and inverse-gamma priors for the variance parameters. We specify fairly noninformative priors.

To fit this model using `bayesmh`, we include random effects for `pig` and their interaction with `week` in our regression model. Following [example 21](#), we add factor levels of the `id` variable to the regression by using the factor-variable specification `i.id`. We include random coefficients on `week` as `i.id#c.week`. By default, the specification will omit one of the `id` categories as a base category. In our Bayesian model, we need to keep all categories of `id`:

```
. fvset base none id
```

We fit our model using `bayesmh`. Following [example 21](#), we perform blocking of parameters and use Gibbs sampling for the blocks. (We could have used the `reffects()` option as in [example 23](#) to include random intercepts, but we want to use Gibbs sampling in this example; thus we use the factor-variable specification instead.)

```
. set seed 14
. bayesmh weight i.id i.id#c.week, likelihood(normal({var_0})) noconstant
> prior({weight:i.id}, normal({weight:_cons},{var_id}))
> prior({weight:i.id#c.week}, normal({weight:week},{var_week}))
> prior({weight:_cons}, normal(0, 100))
> prior({weight:week}, normal(0, 100))
> prior({var_0}, igamma(0.001, 0.001))
> prior({var_id}, igamma(0.001, 0.001))
> prior({var_week}, igamma(0.001, 0.001))
> block({var_0}, gibbs)
> block({var_id}, gibbs)
> block({var_week}, gibbs)
> block({weight:i.id}, gibbs)
> block({weight:i.id#c.week}, gibbs)
> block({weight:week}, gibbs)
> block({weight:_cons}, gibbs)
> mcmcsize(5000) dots notable
Burn-in 2500 .....1000.....2000..... done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done

Model summary
-----
Likelihood:
  weight ~ normal(xb_weight,{var_0})

Priors:
  {weight:i.id} ~ normal({weight:_cons},{var_id})          (1)
  {weight:i.id#c.week} ~ normal({weight:week},{var_week})  (1)
  {var_0} ~ igamma(0.001,0.001)
  {weight:_cons week} ~ normal(0,100)

Hyperprior:
  {var_id var_week} ~ igamma(0.001,0.001)
-----

(1) Parameters are elements of the linear form xb_weight.

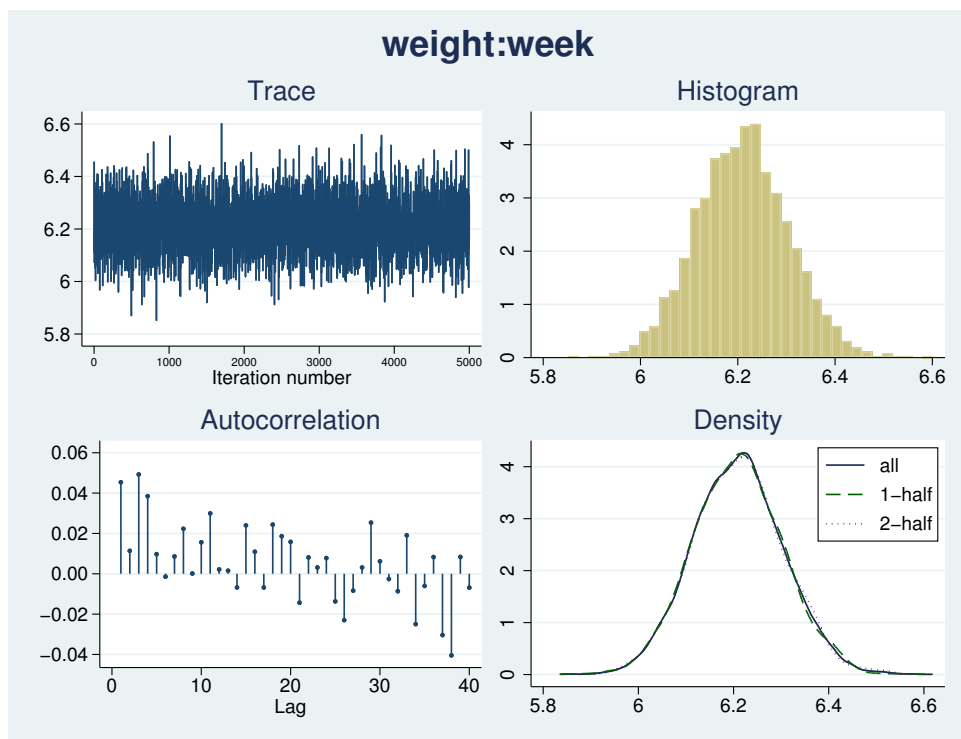
Bayesian normal regression      MCMC iterations =      7,500
Gibbs sampling                  Burn-in           =      2,500
                                MCMC sample size =    5,000
                                Number of obs      =       432
                                Acceptance rate     =         1
                                Efficiency:  min =    .08386
                                           avg =    .1582
                                           max =    .7758

Log marginal-likelihood = -929.94517
```

Our AR is good and efficiencies are high. We do not have a reason to suspect nonconvergence. Nevertheless, it is important to perform graphical convergence diagnostics to confirm this.

Let’s look at diagnostic plots. We show only diagnostic plots for the mean of random coefficients on week, but convergence should be established for all parameters before any inference can be made. We leave it to you to verify convergence of the remaining parameters.

```
. bayesgraph diagnostics {weight:week}
```



The diagnostic plots look good.

Our posterior mean estimates of the main model parameters are in agreement with maximum likelihood results from `mixed`, as is expected with noninformative priors.

```
. bayesstats summary {weight:week _cons} {var_0} {var_id} {var_week}
Posterior summary statistics                                MCMC sample size =      5,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
<b>weight</b>						
<b>week</b>	6.210054	.0948751	.001523	6.210372	6.029255	6.398015
<b>_cons</b>	19.32719	.4096827	.007805	19.32701	18.53177	20.14601
<b>var_0</b>	1.607193	.1224062	.002371	1.600899	1.384723	1.863646
<b>var_id</b>	7.253204	1.705803	.038343	7.034003	4.566251	11.32263
<b>var_week</b>	.3940417	.0886511	.001723	.3822614	.2545719	.607737



```
. set seed 14

. bayesmh weight i.id i.id#c.week, likelihood(normal({var_0})) noconstant
> prior({weight:i.id i.id#c.week},
>       mvnormal(2, {weight:_cons}, {weight:week}, {Sigma,m}))
> prior({weight:week _cons}, normal(0, 1e2))
> prior({var_0}, igamma(0.001,0.001))
> prior({Sigma,m}, iwishart(2,3,I(2)))
> block({var_0}, gibbs) block({Sigma,m}, gibbs)
> block({weight:_cons}) block({weight:week})
> block({weight:i.id}, reffects)
> block({weight:i.id#c.week}, reffects)
> noshow({weight:i.id i.id#c.week})
> mcmcsize(5000) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 5000 .....1000.....2000.....3000.....4000.....
> 5000 done
```

#### Model summary

##### Likelihood:

weight ~ normal(xb\_weight,{var\_0})

##### Priors:

```
{weight:i.id i.id#c.week} ~ mvnormal(2,{weight:_cons},{weight:week},{Sigma,m
                                }) (1)
                                {var_0} ~ igamma(0.001,0.001)
                                {weight:week _cons} ~ normal(0,1e2)
```

##### Hyperprior:

{Sigma,m} ~ iwishart(2,3,I(2))

(1) Parameters are elements of the linear form xb\_weight.

Bayesian normal regression	MCMC iterations =	7,500
Metropolis-Hastings and Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	5,000
	Number of obs =	432
	Acceptance rate =	.5581
	Efficiency: min =	.07112
	avg =	.1423
	max =	.2238

Log marginal-likelihood = -926.22043

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
var_0	1.607509	.1249066	.00435	1.601815	1.38134	1.860937
weight						
_cons	19.36808	.4017089	.021302	19.36764	18.52137	20.15876
week	6.201477	.0952501	.003317	6.199532	6.014793	6.389815
Sigma_1_1	6.850707	1.632765	.07773	6.60346	4.345719	10.66529
Sigma_2_1	-.0854197	.2652103	.010005	-.0803053	-.6326388	.4431884
Sigma_2_2	.400556	.0903881	.002702	.3889624	.260342	1.6140122

The average sampling efficiency is about 14% with no indications for convergence problems. The posterior mean estimates of the main model parameters are close to the maximum likelihood results from mixed. For example, the estimates of variance components  $\sigma_{id}^2$ ,  $\sigma_{21}$ , and  $\sigma_{week}^2$  are 6.85,  $-0.085$ , and 0.40, respectively, from bayesmh and 6.82,  $-0.098$ , and 0.37, respectively, from mixed.

Mixed-effects logistic regression

Here we revisit [example 1 \[ME\] melogit](#). The example analyzes data from the 1989 Bangladesh fertility survey ([Huq and Cleland 1990](#)). A logistic regression model applied to the response variable `c_use` uses fixed-effects variables `urban`, `age`, and `child*` and a random-effects variable, `district`, to account for the between-district variability.

A Bayesian analog of this two-level, random-intercept model using `bayesmh` is as follows. We use the `reffects()` option to specify the random-effects variable `district`. The corresponding random-effects parameters `{c_use:i.district}` are assigned a zero-mean normal prior distribution with variance `{district:var}`. A relatively weak `normal(0,100)` prior is applied to the fixed-effects parameters `{c_use:urban}`, `{c_use:age}`, `{c_use:child*}`, and `{c_use:_cons}`. The variance parameter `{district:var}` is assigned a noninformative `igamma(0.01,0.01)` prior, and a Gibbs sampler is used for it.

```
. use https://www.stata-press.com/data/r16/bangladesh
(Bangladesh Fertility Survey, 1989)

. set seed 14

. bayesmh c_use urban age child*, likelihood(logit) reffects(district)
> prior({c_use:i.district}, normal(0,{district:var}))
> prior({c_use:urban age child* _cons}, normal(0, 100))
> prior({district:var}, igamma(0.01,0.01))
> block({district:var}, gibbs) dots
Burn-in 2500 aaaaaaaaa1000aaaaaaaa2000aaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done

Model summary
-----
Likelihood:
    c_use ~ logit(xb_c_use)

Priors:
                                {c_use:i.district} ~ normal(0,{district:var}) (1)
    {c_use:urban age child1 child2 child3 _cons} ~ normal(0,100)           (1)
                                {district:var} ~ igamma(0.01,0.01)

(1) Parameters are elements of the linear form xb_c_use.

Bayesian logistic regression                MCMC iterations =      12,500
Metropolis–Hastings and Gibbs sampling      Burn-in         =       2,500
                                           MCMC sample size =    10,000
                                           Number of obs     =     1,934
                                           Acceptance rate   =     .4913
                                           Efficiency:      min =     .01728
                                           avg              =     .02523
                                           max              =     .04155

Log marginal-likelihood = -1240.2644
```

	Equal-tailed					
	Mean	Std. Dev.	MCSE	Median	[95% Cred. Interval]	
c_use						
urban	.7252685	.1260246	.009454	.7216279	.4789413	.9849255
age	-.0259076	.0076429	.000529	-.0259236	-.040793	-.0101205
child1	1.104812	.1540978	.008963	1.104046	.8012581	1.410451
child2	1.352477	.1890995	.014387	1.345373	.9832535	1.712931
child3	1.343504	.1793496	.012102	1.343257	.9941767	1.697041
_cons	-1.687957	.1420537	.008543	-1.683849	-1.964436	-1.405009
district						
var	.2380246	.0857548	.004207	.2269953	.1034288	.4357797

Although the average efficiency of 0.03 is not that high, there are no indications for convergence problems. (We can verify this by looking at convergence diagnostics using `bayesgraph` diagnostics.)

Our estimates of the main regression parameters are close to those obtained with the `melogit` command. The posterior mean estimate of variance parameter `{district:var}`, 0.24, is slightly larger than the corresponding estimate of 0.22 from `melogit`.

This model has a fairly large number of parameters, 67, and the logistic likelihood does not allow for efficient Gibbs sampling of regression parameters. If we do not use the `reffects()` option of `bayesmh` (or `block()`'s suboption `split` with `{c_use:i.district}`) and resort to the standard MH algorithm, we may have problems drawing a well-mixed MCMC sample.

For comparison, we show a standard `bayesmh` specification in which the `{c_use:i.district}` parameters are placed in a separate block without using the `reffects()` option. Statistically, the two model specifications are the same because they define one and the same posterior distribution. However, they use different MCMC sampling procedures. We are not interested in the estimates of random effects in this example, so we exclude the random-effects parameters `{c_use:i.district}` from the output table.

```
. set seed 14
. bayesmh c_use urban age child* ibn.district, likelihood(logit)
> prior({c_use:i.district}, normal(0,{district:var}))
> prior({c_use:urban age child* _cons}, normal(0, 100))
> prior({district:var}, igamma(0.01,0.01))
> block({district:var}, gibbs)
> block({c_use:i.district}) noshow({c_use:i.district}) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done

Model summary
```

---

```
Likelihood:
  c_use ~ logit(xb_c_use)

Priors:
                {c_use:i.district} ~ normal(0,{district:var})  (1)
  {c_use:urban age child1 child2 child3 _cons} ~ normal(0,100)  (1)
                {district:var} ~ igamma(0.01,0.01)
```

---

(1) Parameters are elements of the linear form `xb_c_use`.



Bayesian logistic regression

Metropolis-Hastings and Gibbs sampling

MCMC iterations = 12,500

Burn-in = 2,500

MCMC sample size = 10,000

Number of obs = 1,934

Acceptance rate = .53

Efficiency: min = .007367

avg = .0255

max = .04817

Log marginal-likelihood = -1362.0681

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
c_use	urban	.6929174	.119883	.013968	.6906259	.4664536	.9199054
	age	-.0280929	.0080467	.000375	-.0280689	-.0440295	-.0126579
	child1	1.158416	.1697389	.011534	1.155004	.839977	1.490753
	child2	1.442235	.1769685	.008064	1.439614	1.09767	1.796089
	child3	1.447863	.1928966	.012707	1.448637	1.065645	1.836695
	_cons	-2.348392	.14138	.010016	-2.350597	-2.621669	-2.069296
district							
	var	.7490145	.1557382	.014079	.7299348	.5026288	1.110885

Note: There is a high autocorrelation after 500 lags.

In this second run, we observe that the minimal sampling efficiency is less than 1% and that the MCMC convergence is questionable. For example, the reported MCMC standard error for {district:var} is about 0.014, or three times higher than the corresponding error of 0.004 in the previous run. The results from this last run are not trustworthy.

Bayesian analysis of change-point problem

Change-point problems deal with stochastic data, usually time-series data, which undergoes some abrupt change at some time point. It is of interest to localize the point of change and estimate the properties of the stochastic process before and after the change.

Here we analyze the British coal mining disaster data for the years 1851 to 1962 as given in table 5 in [Carlin, Gelfand, and Smith \(1992\)](#). The data are originally from [Maguire, Pearson, and Wynn \(1952\)](#) with updates from [Jarrett \(1979\)](#).

coal.dta contains 112 observations, and it includes the variables id, which records observation identifiers; count, which records the number of coal mining disasters involving 10 or more deaths; and year, which records the years corresponding to the disasters.

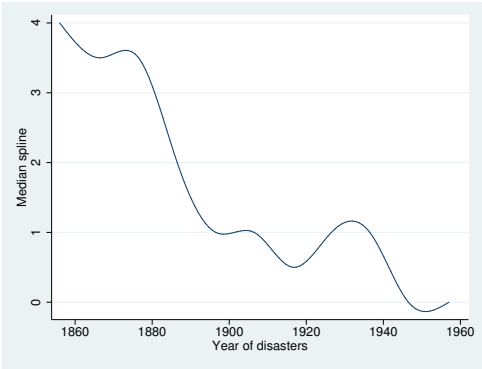
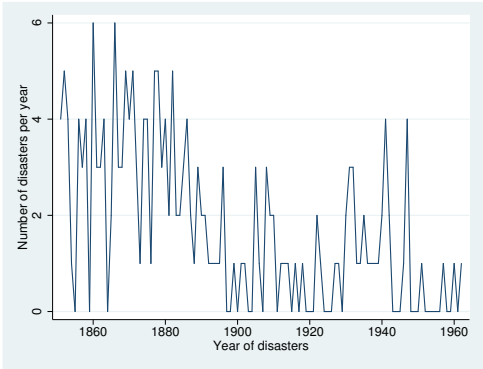
```
. use https://www.stata-press.com/data/r16/coal
(British coal-mining disaster data, 1851-1962)

. describe
Contains data from https://www.stata-press.com/data/r16/coal.dta
  obs:                112                British coal-mining disaster
                                         data, 1851-1962
  vars:                3                5 Feb 2018 18:03
                                         (_dta has notes)
```

variable name	storage type	display format	value label	variable label
id	int	%9.0g		Observation identifier
year	int	%9.0g		Year of disasters
count	byte	%9.0g		Number of disasters per year

Sorted by:

The figures below suggest a fairly abrupt decrease in the rate of disasters around the 1887–1895 period, possibly because of the decline in labor productivity in coal mining (Raftery and Akman 1986). The line plot of count versus year is shown in the left pane and its smoothed version in the right pane.



To find the change-point parameter (cp) in the rate of disasters, we apply the following Bayesian model with noninformative priors for the parameters (accounting for the restricted range of cp):

$$\begin{aligned} \text{counts}_i &\sim \text{Poisson}(\mu_1), \text{ if } \text{year}_i < \text{cp} \\ \text{counts}_i &\sim \text{Poisson}(\mu_2), \text{ if } \text{year}_i \geq \text{cp} \\ \mu_1 &\sim 1 \\ \mu_2 &\sim 1 \\ \text{cp} &\sim \text{Uniform}(1851, 1962) \end{aligned}$$

The model has three parameters:  $\mu_1$ ,  $\mu_2$ , and cp, which we will declare as {mu1}, {mu2}, and {cp} with bayesmh. One interesting feature of this model is the specification of a mixture distribution for count. To accommodate this, we specify the substitutable expression

$$(\{\text{mu1}\} * \text{sign}(\text{year} < \{\text{cp}\}) + \{\text{mu2}\} * \text{sign}(\text{year} \geq \{\text{cp}\}))$$

as the mean of a Poisson distribution dpoisson(). To ensure the feasibility of the initial state, we specify the desired initial values in option initial(). Because of high autocorrelation in the MCMC chain, we increase the MCMC size to achieve higher precision of our estimates. We change the default title to the title specific to our analysis. To monitor the progress of simulation, we request that bayesmh displays a dot every 500 iterations and an iteration number every 5,000 iterations.

```
. set seed 14
. bayesmh count,
> likelihood(dpoisson({mu1}*sign(year<{cp})+{mu2}*sign(year>={cp})))
> prior({mu1} {mu2}, flat)
> prior({cp}, uniform(1851,1962))
> initial({mu1} 1 {mu2} 1 {cp} 1906)
> mcmcsize(40000) title(Change-point analysis) dots(500, every(5000))
Burn-in 2500 a.... done
Simulation 40000 .....5000.....10000.....15000.....20000.....
> ..25000.....30000.....35000.....40000 done
```

Model summary

Likelihood:  
count ~ poisson({mu1}\*sign(year<{cp})+{mu2}\*sign(year>={cp}))

Priors:  
{mu1 mu2} ~ 1 (flat)  
{cp} ~ uniform(1851,1962)

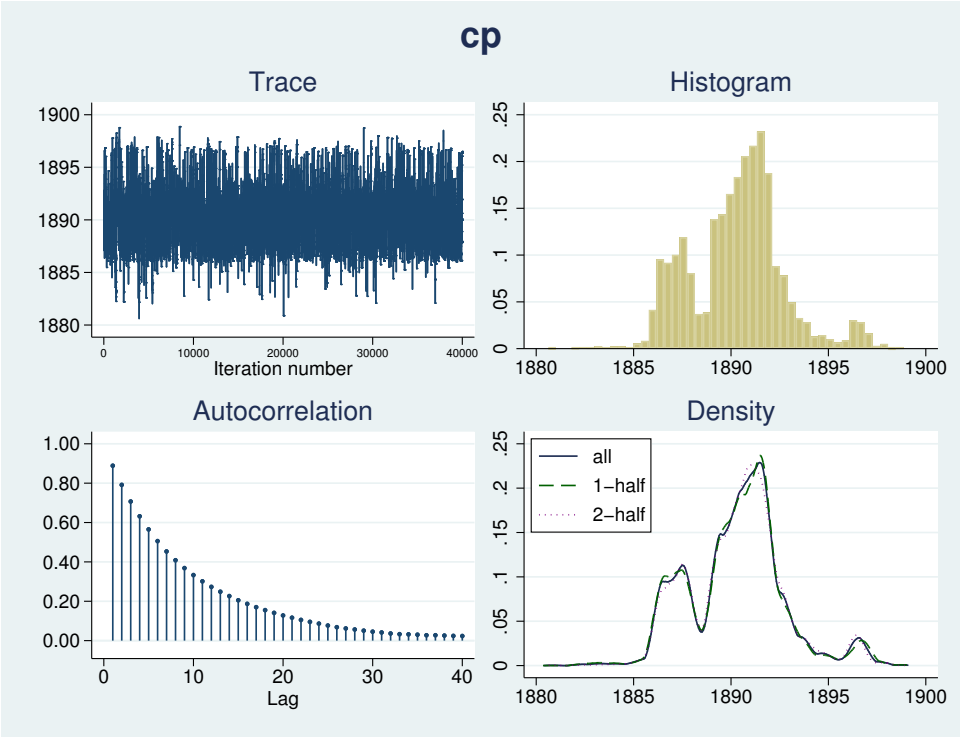
Change-point analysis	MCMC iterations	=	42,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	40,000
	Number of obs	=	112
	Acceptance rate	=	.215
	Efficiency: min	=	.04909
		avg	= .07177
		max	= .09142
Log marginal-likelihood = -173.39572			

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
cp	1890.309	2.43097	.05486	1890.523	1886.126	1896.411
mu1	3.151979	.2894379	.005291	3.137662	2.620379	3.741032
mu2	.934086	.1162233	.001922	.9286517	.7184804	1.175782

According to our results, the change occurred in the first half of 1890. The drop of the disaster rate was significant, from an estimated average of 3.2 to 0.9.

The diagnostic plots, for example, for {cp} do not indicate any convergence problems. (This is also true for other parameters.)

```
. bayesgraph diagnostics {cp}
```



The simulated marginal density of {cp} shown in the right bottom corner provides more details. Apart from the main peak, there are two smaller bumps around the years 1886 and 1896, which correspond to local peaks in the number of disasters at these years: 4 in 1886 and 3 in 1896.

We may be interested in estimating the ratio between the two means. We can use `bayesstats summary` to estimate this ratio.

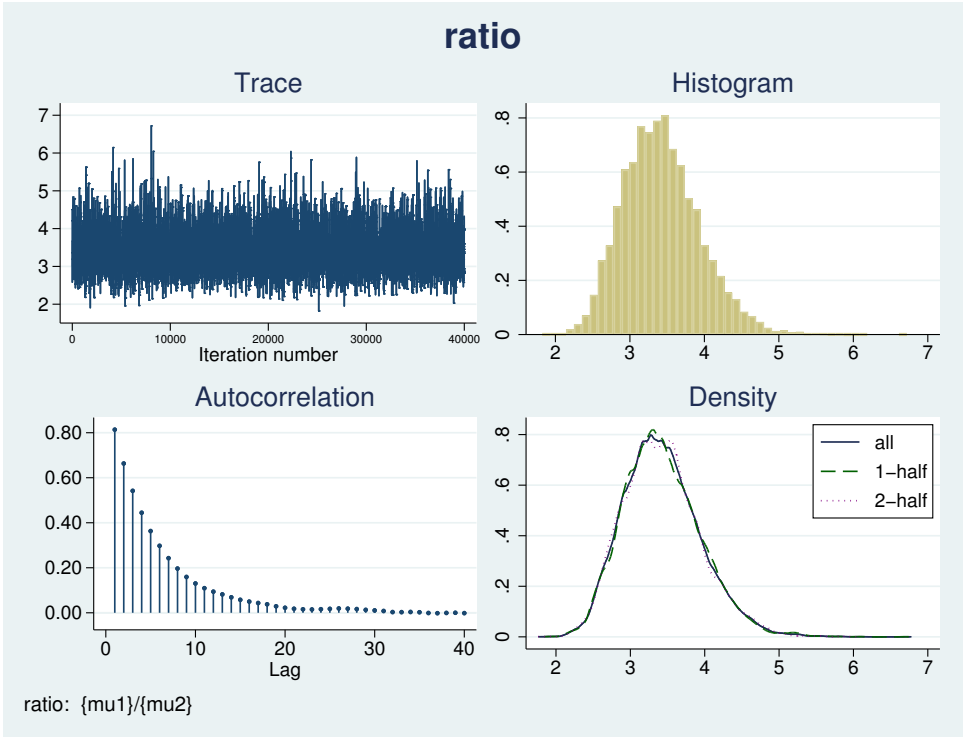
```
. bayesstats summary (ratio:{mu1}/{mu2})
Posterior summary statistics                                MCMC sample size =    40,000
    ratio : {mu1}/{mu2}
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
ratio	3.424565	.5169099	.008259	3.381721	2.541948	4.554931

The posterior mean estimate of the ratio and its 95% credible intervals confirm the change between the two means. After 1890, the mean number of disasters decreased by a factor of about 3.4 with a 95% credible range of [2.5, 4.6].

Remember that convergence must be verified not only for all model parameters but also for the functions of interest. The diagnostic plots for ratio look good.

```
. bayesgraph diagnostics (ratio:{mu1}/{mu2})
```



### Bioequivalence in a crossover trial

Balanced crossover designs are widely used in the pharmaceutical industry for testing the efficacy of new drugs. [Gelfand et al. \(1990\)](#) analyzed a two-treatment, two-period crossover trial comparing two Carbamazepine tablets. The data consist of log-concentration measurements and are originally described in [Maas et al. \(1987\)](#).

A random-effect two-treatment, two-period crossover design is given by

$$y_{i(jk)} = \mu + (-1)^{j-1} \frac{\phi}{2} + (-1)^{k-1} \frac{\pi}{2} + d_i + \epsilon_{i(jk)} = \mu_{i(jk)} + \epsilon_{i(jk)}$$

$$\epsilon_{i(jk)} \sim \text{i.i.d. } N(0, \sigma^2)$$

$$d_i \sim \text{i.i.d. } N(0, \tau^2)$$

where  $i = 1, \dots, n$  is the subject index,  $j = 1, 2$  is the treatment group, and  $k = 1, 2$  is the period.

`bioequiv.dta` has four main variables: subject identifier `id` from 1 to 10, treatment identifier `treat` containing values 1 or 2, period identifier `period` containing values 1 or 2, and outcome `y` measuring log concentration for the two tablets.

```
. use https://www.stata-press.com/data/r16/bioequiv
(Bioequivalent study of Carbamazepine tablets)

. describe
Contains data from https://www.stata-press.com/data/r16/bioequiv.dta
   obs:                20                      Bioequivalent study of
                                                Carbamazepine tablets
   vars:                5                      5 Feb 2018 23:45
                                                (_dta has notes)
```

---

variable name	storage type	display format	value label	variable label
obsid	byte	%9.0g		Observation identifier
id	byte	%9.0g		Subject identifier
treat	byte	%9.0g		Assigned treatment
period	byte	%9.0g		Period identifier
y	float	%9.0g		Log-concentration measurement

---

Sorted by: id period

Before fitting bayesmh, we request no base category for the id variable.

```
. fvset base none id
```

The outcome is assumed to be normally distributed with mean  $\mu_{i(jk)}$  and variance  $\sigma^2$ . To accommodate the specific structure of the regression function, we use a nonlinear specification of bayesmh. We specify the expression for the mean function  $\mu_{i(jk)}$  as a nonlinear expression following the outcome y. We use noninformative priors for parameters and separate parameters in blocks. To improve convergence, we increase our adaptation and burn-in periods. (The command may take some time to produce results, so we specify the dots() option.)

```
. set seed 14
. bayesmh y = ({mu}+(-1)^(treat-1)*{phi}/2+(-1)^(period-1)*{pi}/2+{y:i.id, nocons}),
> likelihood(normal({var}))
> prior({y:i.id}, normal(0,{tau}))
> prior({tau}, igamma(0.001,0.001))
> prior({var}, igamma(0.001,0.001))
> prior({mu} {phi} {pi}, normal(0,1e6))
> block({y:i.id}, split)
> block({tau}, gibbs) block({var}, gibbs)
> adaptation(every(200) maxiter(50)) burnin(10000) dots(250, every(2500))
Burn-in 10000 aaaaaaaaaa2500aaaaaaaa5000aaaaaaaa7500aaaaaaaa10000 done
Simulation 10000 .....2500.....5000.....7500.....10000 done
Model summary
```

---

Likelihood:  
y ~ normal(<expr1>,{var})

Priors:  
    {var} ~ igamma(0.001,0.001)  
    {mu phi pi} ~ normal(0,1e6)  
    {y:i.id} ~ normal(0,{tau})

Hyperprior:  
    {tau} ~ igamma(0.001,0.001)

Expression:  
expr1 : {mu}+(-1)^(treat-1)\*{phi}/2+(-1)^(period-1)\*{pi}/2+{y:i.id, nocons}

---

Bayesian normal regression	MCMC iterations	=	20,000
Metropolis-Hastings and Gibbs sampling	Burn-in	=	10,000
	MCMC sample size	=	10,000
	Number of obs	=	20
	Acceptance rate	=	.5131
	Efficiency: min	=	.01345
	avg	=	.02821
	max	=	.04365
Log marginal-likelihood = -25.692825			

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
y	id						
	1	.0668345	.0834954	.005428	.0645855	-.0879197	.2407731
	2	.1217473	.0895501	.005941	.1190309	-.037415	.308847
	3	.0561551	.0812912	.005154	.0525818	-.0971676	.2344846
	4	.0619807	.0827296	.005294	.0564789	-.0923602	.2365587
	5	.1701813	.09874	.006345	.1685315	-.0149722	.3676389
	6	-.1640241	.0917804	.005572	-.1690176	-.3443967	.0135562
	7	-.1191101	.0864379	.005291	-.1168358	-.2894083	.0400566
	8	-.0590061	.0803792	.004595	-.0572132	-.2217439	.0908653
	9	-.0779055	.0814977	.00481	-.0769495	-.2428321	.0816219
	10	-.014813	.0788845	.00452	-.0138628	-.1750312	.1463467
mu		1.43231	.0579197	.004993	1.434814	1.305574	1.545945
phi		-.0093502	.050824	.00257	-.0104379	-.1039488	.1010855
pi		-.1815055	.0542115	.003107	-.1821367	-.2963565	-.0702212
var		.0134664	.0087676	.000482	.0109334	.0042003	.0370388
tau		.0228884	.020285	.000971	.0182243	.0015547	.0725889

Sampling efficiencies look reasonable considering the number of model parameters. The diagnostic plots of the main model parameters (not shown here) look reasonable except there is a high autocorrelation in the MCMC for {mu}, so you may consider increasing the MCMC size or using thinning.

Parameter  $\theta = \exp(\phi)$  is commonly used as a measure of bioequivalence. Bioequivalence is declared whenever  $\theta$  lies in the interval  $[0.8, 1.2]$  with a high posterior probability.

We use `bayesstats summary` to calculate this probability and to also display other main parameters.

```
. bayesstats summary {mu} {phi} {pi} {tau} {var}
> (theta:exp({phi})) (equiv:exp({phi})>0.8 & exp({phi})<1.2)
Posterior summary statistics                                MCMC sample size =    10,000
    theta : exp({phi})
    equiv  : exp({phi})>0.8 & exp({phi})<1.2
```

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mu	mu	1.43231	.0579197	.004993	1.434814	1.305574	1.545945
	phi	-.0093502	.050824	.00257	-.0104379	-.1039488	.1010855
	pi	-.1815055	.0542115	.003107	-.1821367	-.2963565	-.0702212
	tau	.0228884	.020285	.000971	.0182243	.0015547	.0725889
	var	.0134664	.0087676	.000482	.0109334	.0042003	.0370388
	theta	.9919787	.0507755	.002569	.9896164	.9012714	1.106371
	equiv	.9982	.0423903	.000892	1	1	1

We obtain an estimate of 0.998 for the posterior probability of bioequivalence specified as an expression `equiv`. So we would conclude bioequivalence between the two tablets.

## Random-effects meta-analysis of clinical trials

In meta-analysis of clinical trials, one considers several distinct studies estimating an effect of interest. It is convenient to consider the true effect as varying randomly between the studies. A detailed description of the random-effects meta-analysis can be found in, for example, [Carlin \(1992\)](#). For traditional meta-analysis, see [\[META\] meta](#).

We illustrate Bayesian random-effects meta-analysis of  $2 \times 2$  tables for the beta-blockers dataset analyzed in [Carlin \(1992\)](#). These data are also analyzed in [Yusuf, Simon, and Ellenberg \(1987\)](#). The data summarize the results of 22 clinical trials of beta-blockers used as postmyocardial infarction treatment.

### ► Example 26: Normal–normal analysis

Here we follow the approach of [Carlin \(1992\)](#) for the normal–normal analysis of the beta-blockers data.

For our normal–normal analysis, we consider data in wide form and concentrate on modeling estimates of log odds-ratios from 22 studies.

```
. use https://www.stata-press.com/data/r16/betablockers_wide
(Beta-blockers data in wide form)

. describe
Contains data from https://www.stata-press.com/data/r16/betablockers_wide.dta
obs:                22      Beta-blockers data in wide form
vars:                7      5 Feb 2018 19:02
                        (_dta has notes)
```

variable name	storage type	display format	value label	variable label
study	byte	%9.0g		Study identifier
deaths0	int	%9.0g		Number of deaths in the control group
total0	int	%9.0g		Number of subjects in the control group
deaths1	int	%9.0g		Number of deaths in the treatment group
total1	int	%9.0g		Number of subjects in the treatment group
D	double	%10.0g		Log odds-ratio (based on empirical logits)
var	double	%10.0g		Squared standard error of log odds-ratio

Sorted by:

The estimates of log odds-ratios and their squared standard errors are recorded in variables `D` and `var`, respectively. They are computed from variables `deaths0`, `total0`, `deaths1`, and `total1` based on empirical logits; see [Carlin \(1992, eq. \(3\) and \(4\)\)](#). The `study` variable records study identifiers.

In a normal–normal model, we assume a random-effects model for estimates of log odds-ratios with normally distributed errors and normally distributed random effects. Specifically,

$$D_i = d + u_i + \epsilon_i = d_i + \epsilon_i$$

where  $\epsilon_i \sim N(0, \text{var}_i)$  and  $d_i \sim N(d, \sigma^2)$ . Errors  $\epsilon_i$ 's represent uncertainty about estimates of log odds-ratios in each study  $i$  and are assumed to have known study-specific variances,  $\text{var}_i$ 's. Random



effects  $d_i$ 's represent differences in estimates of log odds-ratios from study to study. The estimates of their mean and variance are of interest in meta-analysis:  $d$  estimates a true effect and  $\sigma^2$  estimates variation in estimating this effect across studies. Small values of  $\sigma^2$  imply that the estimates of a true effect agree among studies.

In Bayesian analysis, we additionally specify prior distributions for  $d$  and  $\sigma^2$ . Following [Carlin \(1992\)](#), we use noninformative priors for these parameters: normal with large variance for  $d$  and inverse gamma with very small degrees of freedom for  $\sigma^2$ .

$$d \sim N(0, 1000)$$

$$\sigma^2 \sim \text{InvGamma}(0.001, 0.001)$$

In our data, random effects  $d_i$  is represented by a factor variable `i.study`. We use all levels of study in our analysis, so we use `fvset` to request no base level for this variable.

```
. fvset base none study
```

We specify `normal()` likelihood with `bayesmh` and request observation-specific variances by specifying variable `var` as `normal()`'s variance argument. We follow the above model formulation for specifying prior distributions. To improve efficiency, we request that all parameters be placed in separate blocks and use Gibbs sampling for the mean parameter `{d}` and the variance parameter `{sig2}`. We also increase the burn-in period to 3,000 iterations and request more frequent adaptation by specifying the `adaptation(every(10))` option. The command will take a little longer to run, so we request that a dot be displayed every 500 iterations and an iteration number be displayed every 2,500 iterations to monitor the progress of the simulation.

```
. set seed 14
. bayesmh D i.study, likelihood(normal(var)) noconstant
> prior({D:i.study}, normal({d},{sig2}))
> prior({d}, normal(0,1000))
> prior({sig2}, igamma(0.001,0.001))
> block({D:i.study}, split)
> block({sig2}, gibbs)
> block({d}, gibbs)
> burnin(3000) adaptation(every(10)) dots(500, every(2500))
Burn-in 3000 aaaa2500a done
Simulation 10000 ....2500....5000....7500....10000 done
Model summary
```

---

```
Likelihood:
  D ~ normal(xb_D,var)

Prior:
  {D:i.study} ~ normal({d},{sig2})
  {d} ~ normal(0,1000)
  {sig2} ~ igamma(0.001,0.001)
```

---

(1) Parameters are elements of the linear form `xb_D`.

Bayesian normal regression	MCMC iterations =	13,000
Metropolis-Hastings and Gibbs sampling	Burn-in =	3,000
	MCMC sample size =	10,000
	Number of obs =	22
	Acceptance rate =	.5315
	Efficiency: min =	.01845
	avg =	.04462
Log marginal-likelihood = 14.38145	max =	.06842

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
D	study						
	1	-.2357346	.1380931	.005394	-.2396019	-.5018659	.0564967
	2	-.2701697	.135307	.006741	-.2585033	-.5760455	-.0174336
	3	-.2538771	.1376569	.005263	-.2495234	-.5436489	.0222503
	4	-.246526	.08904	.003506	-.2483908	-.4212739	-.0643877
	5	-.1969971	.12748	.006635	-.2072718	-.4149274	.1014951
	6	-.2527047	.1339466	.00647	-.2526702	-.5224128	.0229356
	7	-.3377723	.1100308	.006646	-.3283355	-.5829385	-.1548902
	8	-.2054826	.1130796	.005594	-.2121369	-.4051584	.0546629
	9	-.2666327	.1215781	.005263	-.2630645	-.5206763	-.0297599
	10	-.2803866	.0841634	.003593	-.2771339	-.4590086	-.1252279
	11	-.2354098	.1049351	.004449	-.237795	-.4360951	-.0191799
	12	-.202938	.1178808	.005967	-.209884	-.4105608	.0725293
	13	-.2714193	.1288598	.006394	-.263365	-.564746	-.023963
	14	-.1273999	.1468804	.009997	-.1553146	-.3495763	.2172828
	15	-.2518538	.1249082	.005184	-.2502685	-.5090334	-.0021013
	16	-.2245814	.1210757	.004998	-.231592	-.4488306	.0415657
	17	-.2043954	.1357651	.007347	-.2164064	-.4321717	.1044344
	18	-.2153688	.1423256	.006983	-.222428	-.4718119	.0991941
	19	-.2242526	.1360964	.006098	-.2300817	-.4938685	.075416
	20	-.2428998	.1151988	.005403	-.2424417	-.4723024	-.0126589
	21	-.2972177	.1281401	.006041	-.2862546	-.5946982	-.0770212
	22	-.2979427	.1266137	.005575	-.2885006	-.5953839	-.0816952
	d	-.2429052	.0611413	.004501	-.2426092	-.3623229	-.1261924
	sig2	.0166923	.020771	.001488	.0095773	.0007359	.0753652

Our posterior mean estimates `d` and `sig2` of mean  $d$  and variance  $\sigma^2$  are  $-0.24$  and  $0.017$ , respectively, with posterior standard deviations of  $0.06$  and  $0.02$ . The estimates are close to those reported by [Carlin \(1992\)](#). Considering the number of parameters, the AR and efficiency summaries look good.

We can obtain the efficiencies for the main parameters by using `bayesstats ess`.

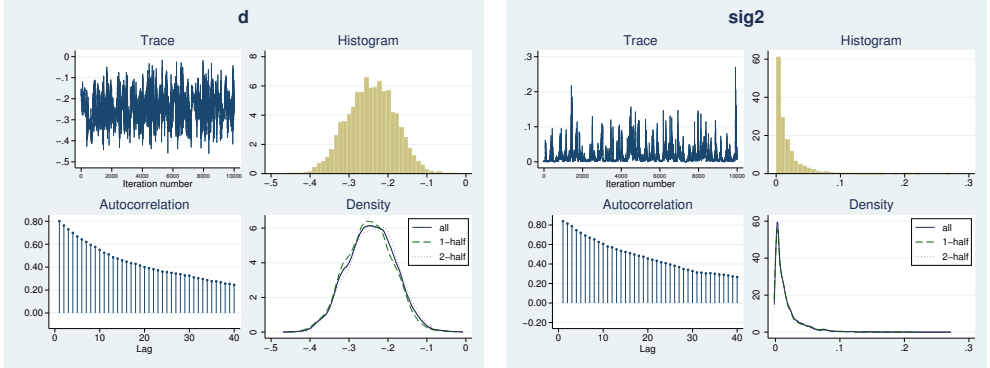
```
. bayesstats ess {d} {sig2}
Efficiency summaries      MCMC sample size =    10,000
                          Efficiency: min =     .01845
                          avg =     .01897
                          max =     .01949
```

	ESS	Corr. time	Efficiency
d	184.49	54.20	0.0184
sig2	194.88	51.31	0.0195

The efficiencies are acceptable, but based on the correlation times, the autocorrelation becomes small only after lag 50 or so. The precision of the mean and variance estimates is comparable to those based on 184 independent observations for the mean and 195 independent observations for the variance.

We explore convergence visually.

```
. bayesgraph diagnostics {d} {sig2}
```



The diagnostic plots look reasonable for both parameters, but autocorrelation is high. You may consider increasing the default MCMC size to obtain more precise estimates of posterior means.

◀

### ► Example 27: Binomial-normal model

There is an alternative but equivalent way of formulating the meta-analysis model from [example 26](#) as a binomial-normal model. Instead of modeling estimates of log odds-ratios directly, one can model probabilities of success (an event of interest) in each group.

Let  $p_i^T$  and  $p_i^C$  be the probabilities of success for the treatment and control groups in the  $i$ th trial. The random-effects meta-analysis model can be given as

$$\begin{aligned}\text{logit}(p_i^C) &= \mu_i \\ \text{logit}(p_i^T) &= \mu_i + d_i\end{aligned}$$

where  $\mu_i$  is log odds of success in the control group in study  $i$  and  $\mu_i + d_i$  is log odds of success in the treatment group.  $d_i$ 's are viewed as random effects and are assumed to be normally distributed as

$$d_i \sim \text{i.i.d. } N(d, \sigma^2)$$

where  $d$  is the population effect and  $\sigma^2$  is its variability across trials.

Suppose that we observe  $y_i^C$  successes out of  $n_i^C$  events in the control group and  $y_i^T$  successes out of  $n_i^T$  events in the treatment group from the  $i$ th trial. Then,

$$\begin{aligned}y_i^C &\sim \text{Binomial}(p_i^C, n_i^C) \\ y_i^T &\sim \text{Binomial}(p_i^T, n_i^T)\end{aligned}$$

The random effects are usually assumed to be normally distributed as

$$d_i \sim \text{i.i.d. } N(d, \sigma^2)$$

where  $d$  is the population effect and is the main parameter of interest in the model, and  $\sigma^2$  is its variability across trials.

We can rewrite the model above assuming the data are in long form as

$$\begin{aligned}\text{logit}(p_i) &= \mu_i + (T_i == 1) \times d_i \\ y_i &\sim \text{Binomial}(p_i, n_i) \\ d_i &\sim \text{i.i.d. } N(d, \sigma^2)\end{aligned}$$

where  $T_i$  is a binary treatment with  $T_i = 0$  for the control group and  $T_i = 1$  for the treatment group.

In Bayesian analysis, we additionally specify prior distributions for  $\mu_i$ ,  $d$ , and  $\sigma^2$ . We use noninformative priors.

$$\begin{aligned}\mu_i &\sim 1 \\ d &\sim N(0, 1000) \\ \sigma^2 &\sim \text{InvGamma}(0.001, 0.001)\end{aligned}$$

We continue our analysis of beta-blockers data. The analysis of these data using a binomial-normal model is also provided as an example in OpenBUGS (Thomas et al. 2006).

For this analysis, we use the beta-blockers data in long form.

```
. use https://www.stata-press.com/data/r16/betablockers_long
(Beta-blockers data in long form)
. describe
Contains data from https://www.stata-press.com/data/r16/betablockers_long.dta
obs:                44                Beta-blockers data in long form
vars:                4                5 Feb 2018 19:02
                                (_dta has notes)
```

variable name	storage type	display format	value label	variable label
study	byte	%9.0g		Study identifier
treat	byte	%9.0g	treatlab	Treatment group: 0 - control, 1 - treatment
deaths	int	%9.0g		Number of deaths in each group
total	int	%9.0g		Number of subjects in each group

Sorted by: study treat

Variable `treat` records the binary treatment: `treat==0` identifies the control group, and `treat==1` identifies the treatment group.

To relate to the notation of our model, we create variable `mu` to contain identifiers for each study. We also request that no base is set for our factor variables `mu` and `study`.

```
. generate mu = study
. fvset base none mu study
```

We use a `binomial()` likelihood model for the number of deaths. We split all parameters into separate blocks and request Gibbs sampling for `sig2` to improve efficiency of the algorithm.

```
. set seed 14

. bayesmh deaths i.mu 1.treat#i.study, likelihood(binomial(total)) noconstant
> prior({deaths:i.mu}, flat)
> prior({deaths:1.treat#i.study}, normal({d},{sig2}))
> prior({d}, normal(0,1000)) prior({sig2}, igamma(0.001,0.001))
> block({deaths:1.treat#i.study}, split)
> block({deaths:i.mu}, split) block({d}, gibbs)
> block({sig2}, gibbs) dots(500, every(2500))
Burn-in 2500 aaaa2500 done
Simulation 10000 ....2500....5000....7500....10000 done

Model summary
```

Likelihood:  
deaths ~ binlogit(xb\_deaths,total)

Priors:

{deaths:i.mu} ~ 1 (flat)

(1)

{deaths:1.treat#i.study} ~ normal({d},{sig2})

(1)

Hyperpriors:

{d} ~ normal(0,1000)

{sig2} ~ igamma(0.001,0.001)

(1) Parameters are elements of the linear form xb\_deaths.

Bayesian binomial regression	MCMC iterations	=	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	44
	Acceptance rate	=	.4633
	Efficiency: min	=	.01479
	avg	=	.092
	max	=	.228
Log marginal-likelihood =	-126.8754		

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
deaths	mu						
	1	-2.445566	.4440859	.009774	-2.428504	-3.386517	-1.650575
	2	-2.192517	.2372503	.005622	-2.189225	-2.67948	-1.751515
	3	-2.130301	.2682139	.006193	-2.123122	-2.677923	-1.615621
	4	-2.397429	.0783732	.002298	-2.396564	-2.55513	-2.244307
	5	-2.406291	.160123	.004778	-2.402657	-2.731128	-2.103221
	6	-2.235409	.3563328	.008201	-2.219613	-2.97514	-1.597156
	7	-1.719358	.0814695	.003238	-1.720512	-1.877615	-1.558778
	8	-2.112285	.1181128	.004557	-2.108343	-2.350633	-1.894469
	9	-1.966916	.1456821	.00391	-1.96278	-2.266818	-1.697274
	10	-2.246224	.0716096	.002504	-2.245524	-2.39558	-2.106513
	11	-2.315093	.1137358	.003641	-2.314937	-2.545779	-2.097822
	12	-1.466143	.1226934	.003342	-1.464731	-1.710775	-1.23385
	13	-2.999446	.207893	.004396	-2.992656	-3.419645	-2.599853
	14	-2.726241	.1262413	.006217	-2.718875	-2.992452	-2.488097
	15	-1.358942	.1558862	.003801	-1.3591	-1.672494	-1.054848
	16	-1.488891	.1430423	.003957	-1.488181	-1.772535	-1.220204
	17	-2.000612	.189858	.006088	-1.989813	-2.394403	-1.647906
	18	-2.970916	.2881977	.006512	-2.94776	-3.59226	-2.445288
	19	-3.431928	.3386751	.007093	-3.410703	-4.129594	-2.813783
	20	-1.487005	.1371778	.003604	-1.484245	-1.765015	-1.216753
	21	-2.138858	.1373135	.004474	-2.135716	-2.409188	-1.871238
	22	-2.923794	.1379264	.003928	-2.922266	-3.201064	-2.659156

treat#study						
1 1	-.2332275	.1389226	.005131	-.2365335	-.5125546	.0643276
1 2	-.2754821	.1376714	.005278	-.2639067	-.5865606	-.0155825
1 3	-.2521373	.1376143	.005031	-.2448604	-.5515139	.0168471
1 4	-.2441378	.0893076	.003315	-.2450735	-.4238717	-.0567493
1 5	-.1922897	.1322661	.006521	-.2075691	-.4254067	.1204152
1 6	-.2519257	.1398246	.005351	-.2490731	-.5571537	.0413351
1 7	-.3394292	.1060879	.007038	-.3293058	-.57003	-.1592483
1 8	-.2061079	.1133207	.005757	-.2172042	-.405318	.0483414
1 9	-.2681616	.1191742	.00505	-.2642165	-.5293988	-.0371976
1 10	-.2833565	.0857065	.003746	-.277193	-.4662505	-.1285165
1 11	-.2347279	.1053374	.004244	-.238179	-.4488999	-.0093842
1 12	-.1985932	.1140053	.005216	-.2092791	-.4048572	.0621012
1 13	-.2661801	.1313768	.005161	-.2591945	-.5575894	-.0043227
1 14	-.1199702	.1435854	.011808	-.1454409	-.345002	.2252269
1 15	-.2528203	.1187262	.004596	-.2474198	-.5094695	-.0194982
1 16	-.2250511	.1216752	.004596	-.2312945	-.4584372	.0489469
1 17	-.2032452	.1313935	.006245	-.2141859	-.4449455	.1008652
1 18	-.21541	.1365779	.005847	-.2247174	-.4765884	.0898225
1 19	-.2256096	.1423711	.005804	-.2297091	-.519874	.0853728
1 20	-.2384292	.116139	.004978	-.2397589	-.4725188	.0082366
1 21	-.3012616	.1250729	.007103	-.2866182	-.5961708	-.0823887
1 22	-.29952	.1259856	.006557	-.2836829	-.6027407	-.0829247
<hr/>						
d	-.2422989	.0600407	.004175	-.243193	-.3624938	-.1209386
sig2	.0170769	.0196724	.001529	.0104745	.0007195	.070371

Note: Adaptation tolerance is not met in at least one of the blocks.

This model has 22 more parameters than the model in [example 26](#). The posterior mean estimates `d` and `sig2` of mean  $d$  and variance  $\sigma^2$  are  $-0.24$  and  $0.017$ , respectively, with posterior standard deviations of  $0.06$  and  $0.02$ . The estimates of the mean and variance are again close to the ones reported by [Carlin \(1992\)](#).

Compared with [example 26](#), the efficiencies and other statistics for the main parameters are similar.

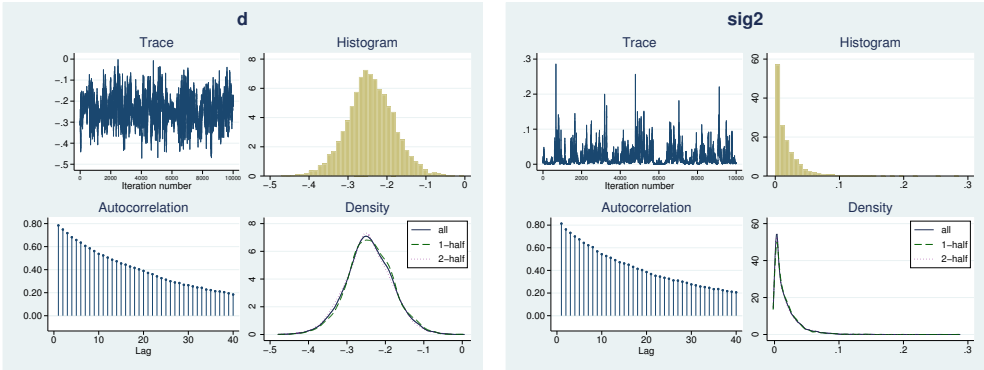
```
. bayesstats ess {d} {sig2}
```

```
Efficiency summaries      MCMC sample size =      10,000
                          Efficiency:  min =      .01656
                                   avg =      .01862
                                   max =      .02068
```

	ESS	Corr. time	Efficiency
<hr/>			
d	206.78	48.36	0.0207
sig2	165.58	60.39	0.0166

The diagnostic plots look similar to those shown in [example 26](#).

```
. bayesgraph diagnostics {d} {sig2}
```



4

## Item response theory

### ► Example 28: 1PL IRT model—Rasch model

If you are not familiar with IRT, see [\[IRT\] irt](#) for an introduction to IRT concepts and terminology. Here we revisit [example 1](#) of [\[IRT\] irt 1pl](#). The example analyzes student responses to nine test questions and uses an abridged version of the mathematics and science data from [De Boeck and Wilson \(2004\)](#). The goal of the analysis is to estimate the common discrimination of the questions (items) and their individual difficulties.

An alternative formulation of the one-parameter IRT model is the [Rasch \(1960\)](#) model with logit link; see, for example, [Methods and formulas](#) of [\[IRT\] irt 1pl](#). A typical IRT dataset consists of binary outcomes (success or failure) of  $J$  subjects, where each subject is tested on  $I$  items. Let the observation  $y_{ij}$  represent the binary outcome for item  $i$ , where  $i = 1, \dots, I$ , and subject  $j$ , where  $j = 1, \dots, J$ . Each item  $i$  is characterized by a level of difficulty  $b_i$ . The difficulties are not observed and must be estimated. Associated with each subject  $j$  is a latent trait level  $\theta_j$ , which characterizes the ability of the subject. The model likelihood has a generalized linear regression form

$$\text{logit}\{\Pr(y_{ij} = 1 | b_i, \theta_j)\} = a(\theta_j - b_i)$$

where  $a$  is a discrimination parameter. According to this likelihood model, the probability of success increases with the subject ability and decreases with item difficulty. The discrimination parameter  $a$  represents the slope of the item characteristic curves. The subject abilities are assumed to be standardized so that

$$\theta_j \sim \text{i.i.d. } N(0, 1)$$

The discrimination parameter  $a$  can be absorbed into  $\theta_j$  and  $b_i$  so that the model is reparameterized as

$$\text{logit}\{\Pr(y_{ij} = 1 | \tilde{b}_i, \tilde{\theta}_j)\} = \tilde{\theta}_j + \tilde{b}_i \tag{1}$$

$$\tilde{\theta}_j \sim \text{i.i.d. } N(0, \sigma^2)$$

where  $\sigma = a$  and  $\tilde{b}_i = -ab_i$ . In addition to the above, a Bayesian formulation of the model requires, prior specifications for parameters  $\sigma^2$  and  $\tilde{b}_i$ . In the following example, we use

$$\sigma^2 \sim \text{InvGamma}(0.01, 0.01)$$

$$\tilde{b}_i \sim N(0, 10)$$

To fit this model using `bayesmh`, we first need to reshape the data from [example 1](#) of [\[IRT\] irt 1pl](#) in long format so that the answers to the nine questions are represented by the response variable `y`, while the `item` and `id` variables encode the questions and students, respectively.

```
. use https://www.stata-press.com/data/r16/masc1, clear
(Data from De Boeck & Wilson (2004))
. generate id = _n
. quietly reshape long q, i(id) j(item)
. rename q y
```

The Rasch likelihood model can be specified with `bayesmh` using `y` as a dependent variable, `item` as an independent factor variable, and `id` as a random-effects variable. We suppress the base levels of `item` and `id` and use the `noconstant` option in the likelihood specification. The random-effects parameters `{y:i.id}` are assigned a zero-mean normal prior with variance `{var}` [ $\sigma^2$  in model specification (1)]. The parameter `{var}` is assigned a noninformative inverse-gamma prior with shape 0.01 and scale 0.01, whereas the parameters `{y:i.item}` [ $\tilde{b}_i$ 's in model (1)] are applied ad hoc informative normal(0,10) priors. Because there are many random-effects parameters `{y:i.id}`, we exclude them from the simulation results and the output table by specifying the `exclude()` option.

```
. fvset base none id item
. set seed 14
. bayesmh y i.item, noconstant reffects(id) likelihood(logit)
> prior({y:i.id}, normal(0, {var}))
> prior({y:i.item}, normal(0, 10))
> prior({var}, igamma(0.01,0.01))
> block({var}) block({y:i.item}, reffects) exclude({y:i.id}) dots
Burn-in 2500 aaaaaaaaaa1000aa.....2000..... done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done
Model summary
```

---

```
Likelihood:
  y ~ logit(xb_y)
```

```
Priors:
  {y:i.id} ~ normal(0,{var}) (1)
  {y:i.item} ~ normal(0,10) (1)
```

```
Hyperprior:
  {var} ~ igamma(0.01,0.01)
```

---

(1) Parameters are elements of the linear form `xb_y`.



Bayesian logistic regression  
Random-walk Metropolis-Hastings sampling

MCMC iterations = 12,500  
Burn-in = 2,500  
MCMC sample size = 10,000  
Number of obs = 7,200  
Acceptance rate = .3095  
Efficiency: min = .02018  
              avg = .09909  
              max = .1325

Log marginal-likelihood = .

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
y	item						
	1	.6026378	.0826471	.002651	.6031956	.4385031	.7627183
	2	.1033997	.0822957	.002612	.1058224	-.0577589	.2591329
	3	1.546171	.0983027	.002867	1.543848	1.352478	1.740203
	4	-.2710592	.0832457	.002601	-.2717436	-.4378639	-.1119697
	5	-1.411632	.0965821	.003242	-1.410272	-1.606338	-1.223846
	6	-.5902256	.0835119	.002599	-.5902611	-.7561024	-.4268245
	7	-1.127774	.0910435	.002649	-1.126278	-1.308778	-.9411566
	8	2.062963	.1089818	.002994	2.062155	1.858745	2.286346
	9	1.015927	.0884674	.002647	1.016285	.8401385	1.191785
var		.7281624	.0817864	.005757	.7256023	.5723933	.89159

In the simulation summary, `bayesmh` reports a modest average efficiency of about 10% with no indications for convergence problems. The log marginal-likelihood is reported as missing because we used the `exclude()` option. The Laplace–Metropolis approximation of the log marginal-likelihood requires that simulation results be available for all model parameters, including random-effects parameters.

To match the discrimination and question difficulty parameters of the `irt 1pl` command, we can apply the following transformation to the `bayesmh` model parameters. The common discrimination parameter equals the square-root of `{var}`, and the individual question difficulties equal the negative `{y:i.item}`’s parameters, normalized by their common discrimination. We can obtain estimates of these parameters using the `bayesstats summary` command.

```
. bayesstats summary (discr:sqrt({var}))
> (diff1:-{y:1bn.item}/sqrt({var}))
> (diff2:-{y:2.item}/sqrt({var}))
> (diff3:-{y:3.item}/sqrt({var}))
> (diff4:-{y:4.item}/sqrt({var}))
> (diff5:-{y:5.item}/sqrt({var}))
> (diff6:-{y:6.item}/sqrt({var}))
> (diff7:-{y:7.item}/sqrt({var}))
> (diff8:-{y:8.item}/sqrt({var}))
> (diff9:-{y:9.item}/sqrt({var})), nolegend
Posterior summary statistics MCMC sample size = 10,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
discr	.8519744	.0479822	.003395	.8518229	.7565668	.9442404
diff1	-.7091455	.102058	.00371	-.7096954	-.9100091	-.5076584
diff2	-.1218422	.0973904	.003249	-.1237917	-.3108689	.0687174
diff3	-1.81946	.1407559	.006296	-1.812969	-2.10733	-1.551773
diff4	.3189748	.0990609	.003198	.3169251	.1301887	.5176201
diff5	1.661228	.1364909	.006053	1.656482	1.407416	1.943332
diff6	.6945347	.1027624	.003673	.6933689	.498015	.9047016
diff7	1.32722	.123354	.004918	1.321498	1.093405	1.583459
diff8	-2.427933	.1721688	.008354	-2.417503	-2.785216	-2.116069
diff9	-1.195619	.1177288	.004806	-1.19854	-1.42661	-.9652485

We observe that the reported posterior means for the common discrimination and question difficulties are close to those obtained with `irt 1pl`, within the limits of the MCMC standard error of 0.003.

◀

In this example, we fit the Rasch model using the `reflects()` option and used transformation to estimate parameters of the corresponding 1PL IRT model. To avoid reparameterization, we could have fit the 1PL model directly using a nonlinear specification of `bayesmh`, as we demonstrate in [example 29](#) for the 2PL IRT model. The shortcoming of the nonlinear specification, which precludes the use of the `reflects()` option, is slower execution.

## ► Example 29: 2PL IRT model

A more comprehensive IRT model is the 2PL model introduced by [Birnbaum \(1968\)](#), which allows the discrimination and difficulty parameters to vary between items. For a detailed description and examples of the model, see [\[IRT\] irt 2pl](#).

A Bayesian formulation of the 2PL model allows the item-specific discrimination and difficulty parameters as well as the subject abilities to be modeled, either individually or as groups, using prior distributions.

The 2PL model likelihood has the following form,

$$\Pr(Y_{ij} = 1) = \frac{\exp\{a_i(\theta_j - b_i)\}}{1 + \exp\{a_i(\theta_j - b_i)\}}$$

where  $a_i$ 's and  $b_i$ 's are discrimination and difficulty parameters and  $\theta_j$ 's are subject abilities. This is a logistic regression model with probability of success modeled using the linear form  $a_i(\theta_j - b_i)$ . We assume that the probability of success increases with subject ability, which implies  $a_i > 0$ . Subject ability parameters are assumed independent and distributed according to the standard normal distribution

$$\theta_j \sim N(0, 1)$$

For Bayesian modeling, we additionally assume the following prior specifications:

$$\ln(a_i) \sim N(\mu_a, \sigma_a^2)$$

$$b_i \sim N(\mu_b, \sigma_b^2)$$

$$\mu_a, \mu_b \sim N(0, 1)$$

$$\sigma_a^2, \sigma_b^2 \sim \text{Gamma}(1, 1)$$

In the absence of prior knowledge about parameters  $a_i$ 's and  $b_i$ 's, we want to specify proper priors that are not subjective. Because  $a_i$ 's must be positive, a common choice is to assume that  $\ln(a_i)$ 's are normally distributed with mean  $\mu_a$  and variance  $\sigma_a^2$ . We assume that  $b_i$ 's are normally distributed with mean  $\mu_b$  and variance  $\sigma_b^2$ . Our prior assumption is that the questions in the study are fairly balanced in terms of discrimination and difficulty and we express this expectation by specifying  $N(0, 1)$  hyperpriors for  $\mu_a$  and  $\mu_b$ ; that is, we assume that  $\mu_a$  and  $\mu_b$  are not that different from zero. We also put a slight prior constraint on the variability of the discrimination and difficulty parameters by assigning a gamma distribution with shape 1 and scale 1 as hyperprior distributions for  $\sigma_a^2$  and  $\sigma_b^2$ . To demonstrate a Bayesian 2PL model, we use again the mathematics and science dataset `masc1`, reshaped in long format as in [example 28](#).

```
bayesmh y = ({discr:}*({subj:}-{diff:})), likelihood(logit)    ///  
  redefine(discr:i.item)                                       ///  
  redefine(diff:i.item)                                        ///  
  redefine(subj:i.id)                                          ///  
  prior({subj:i.id},    normal(0, 1))                          ///  
  prior({discr:i.item}, lognormal({mua}, {vara}))              ///  
  prior({diff:i.item},  normal({mub}, {varb}))                ///  
  prior({vara varb},    gamma(1, 1))                          ///  
  prior({mua mub},      normal(0, 1))                          ///  
  ...
```

To specify the 2PL model likelihood in `bayesmh`, we need to use a nonlinear specification to accommodate the varying coefficients  $a_i$ 's. For `masc1.dta`, we have 9 items, where  $i = 1, \dots, 9$ , and 800 subjects, where  $j = 1, \dots, 800$ . A straightforward nonlinear specification is `({discr:i.item}*({subj:i.id}-{diff:i.item}))`. Given the large number of subjects, it may be computationally prohibitive to use this substitutable expression. A more computationally efficient way is to use the `redefine()` option to specify the random effects associated with item discrimination, item difficulty, and student ability. For example, `redefine(subj:i.id)` introduces subject random-effects parameters, one for each subject, and represents the parameters  $\theta_j$ 's. Similarly, we use `redefine(discr:i.item)` and `redefine(diff:i.item)` to introduce the item-specific discrimination and difficulty parameters  $a_i$ 's and  $b_i$ 's, respectively. The probability of success is then modeled using the expression `({discr:}*({subj:}-{diff:}))`.

To achieve better sampling efficiency, we place the hyperparameters `{mua}`, `{mub}`, `{vara}`, and `{varb}` into separate blocks using the `block()`'s suboption `split`. We also initialize the discrimination and difficulty random effects with 1, because the default zeros result in an invalid initial state. We have many random-effects parameters `{subj:i.id}`, so we exclude them from the simulation results and the output table by specifying the `exclude()` option.

```

. fvset base none id item
. set seed 14
. bayesmh y = ({discr:}*({subj:}-{diff:})), likelihood(logit)
> redefine(discr:i.item) redefine(diff:i.item) redefine(subj:i.id)
> prior({subj:i.id}, normal(0, 1))
> prior({discr:i.item}, lognormal({mua}, {vara}))
> prior({diff:i.item}, normal({mub}, {varb}))
> prior({vara varb}, gamma(1, 1)) prior({mua mub}, normal(0, 1))
> block({vara varb mua mub}, split) init({discr:i.item} 1 {diff:i.item} 1)
> exclude({subj:i.id}) showreffects({discr:i.item} {diff:i.item}) dots
Burn-in 2500 aaaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 10000 .....1000.....2000.....3000.....4000.....
> 5000.....6000.....7000.....8000.....9000.....10000 done

```

Model summary

---

Likelihood:

y ~ logit(xb\_discr\*(xb\_subj-xb\_diff))

Priors:

{discr:i.item} ~ lognormal({mua},{vara}) (1)

{diff:i.item} ~ normal({mub},{varb}) (2)

{subj:i.id} ~ normal(0,1) (3)

Hyperpriors:

{vara varb} ~ gamma(1,1)

{mua mub} ~ normal(0,1)

---

(1) Parameters are elements of the linear form xb\_discr.

(2) Parameters are elements of the linear form xb\_diff.

(3) Parameters are elements of the linear form xb\_subj.

Bayesian logistic regression  
Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = .

MCMC iterations = 12,500  
Burn-in = 2,500  
MCMC sample size = 10,000  
Number of obs = 7,200  
Acceptance rate = .3657  
Efficiency: min = .01027  
              avg = .04499  
              max = .1762

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
discr	item						
	1	1.498385	.2400297	.017729	1.475542	1.077856	2.016152
	2	.666128	.1119997	.005344	.6649249	.4541802	.886375
	3	.9383295	.1477952	.01186	.9204028	.6765889	1.266951
	4	.7994094	.1238608	.005473	.7962891	.5688815	1.059357
	5	.9051624	.1482122	.011637	.9043871	.6396924	1.205136
	6	.958388	.142267	.009086	.9504636	.7001779	1.252891
	7	.4792205	.0899554	.008877	.4741658	.3261732	.6680243
	8	1.297704	.221223	.017263	1.295304	.8673528	1.758576
	9	.6670617	.1104876	.009091	.6625288	.4666832	.8896693
diff	item						
	1	-.4989148	.0830179	.005137	-.4993581	-.6742972	-.3519269
	2	-.1502021	.1239443	.003577	-.1458722	-.4059771	.0751602
	3	-1.710592	.2234976	.016935	-1.692729	-2.214283	-1.328619
	4	.3466566	.114595	.004388	.3434778	.1346476	.5714364
	5	1.605012	.2387544	.018704	1.57642	1.229628	2.138817
	6	.6442216	.1147523	.006181	.6396185	.4391379	.8736935
	7	2.190372	.4265088	.041367	2.130899	1.530794	3.148253
	8	-1.830477	.2345021	.019047	-1.795231	-2.420293	-1.46107
	9	-1.472878	.2480988	.01968	-1.446161	-2.045008	-1.072642
mua		-.157914	.1768629	.004552	-.1556469	-.5143239	.188628
vara		.2516064	.1808929	.007705	.1996404	.0597163	.7334286
mub		-.078763	.4360087	.010388	-.0791598	-.945727	.7879737
varb		1.953467	.8047981	.029965	1.806903	.8319104	3.947983

`bayesmh` reports an acceptable average efficiency of about 4%. A close inspection of the estimation table shows that the posterior mean estimates for item discrimination and difficulty are not much different from the MLE estimates obtained with the `irt 2pl` command; see [example 1](#) in [\[IRT\] irt 2pl](#).

## Stored results

bayesmh stores the following in `e()`:

### Scalars

<code>e(N)</code>	number of observations
<code>e(k)</code>	number of parameters
<code>e(k_sc)</code>	number of scalar parameters
<code>e(k_mat)</code>	number of matrix parameters
<code>e(n_eq)</code>	number of equations
<code>e(nchains)</code>	number of MCMC chains
<code>e(mcmcsize)</code>	MCMC sample size
<code>e(burnin)</code>	number of burn-in iterations
<code>e(mcmciter)</code>	total number of MCMC iterations
<code>e(thinning)</code>	thinning interval
<code>e(arate)</code>	overall AR
<code>e(eff_min)</code>	minimum efficiency
<code>e(eff_avg)</code>	average efficiency
<code>e(eff_max)</code>	maximum efficiency
<code>e(Rc_max)</code>	maximum Gelman–Rubin convergence statistic (only with <code>nchains()</code> )
<code>e(clevel)</code>	credible interval level
<code>e(hpd)</code>	1 if <code>hpd</code> is specified; 0 otherwise
<code>e(batch)</code>	batch length for batch-means calculations
<code>e(corrlag)</code>	maximum autocorrelation lag
<code>e(corrtol)</code>	autocorrelation tolerance
<code>e(dic)</code>	deviance information criterion
<code>e(lml_lm)</code>	log marginal-likelihood using Laplace–Metropolis method
<code>e(scale)</code>	initial multiplier for scale factor; <code>scale()</code>
<code>e(block#_gibbs)</code>	1 if Gibbs sampling is used in <code>#th</code> block, 0 otherwise
<code>e(block#_reffects)</code>	1 if the parameters in <code>#th</code> block are random effects, 0 otherwise
<code>e(block#_scale)</code>	<code>#th</code> block initial multiplier for scale factor
<code>e(block#_tarate)</code>	<code>#th</code> block target adaptation rate
<code>e(block#_tolerance)</code>	<code>#th</code> block adaptation tolerance
<code>e(adapt_every)</code>	adaptation iterations <code>adaptation(every())</code>
<code>e(adapt_maxiter)</code>	maximum number of adaptive iterations <code>adaptation(maxiter())</code>
<code>e(adapt_miniter)</code>	minimum number of adaptive iterations <code>adaptation(miniter())</code>
<code>e(adapt_alpha)</code>	adaptation parameter <code>adaptation(alpha())</code>
<code>e(adapt_beta)</code>	adaptation parameter <code>adaptation(beta())</code>
<code>e(adapt_gamma)</code>	adaptation parameter <code>adaptation(gamma())</code>
<code>e(adapt_tolerance)</code>	adaptation tolerance <code>adaptation(tolerance())</code>
<code>e(repeat)</code>	number of attempts used to find feasible initial values

### Macros

<code>e(cmd)</code>	<code>bayesmh</code>
<code>e(cmdline)</code>	command as typed
<code>e(method)</code>	sampling method
<code>e(depvars)</code>	names of dependent variables
<code>e(eqnames)</code>	names of equations
<code>e(likelihood)</code>	likelihood distribution (one equation)
<code>e(likelihood#)</code>	likelihood distribution for <code>#th</code> equation
<code>e(prior)</code>	prior distribution
<code>e(prior#)</code>	prior distribution, if more than one <code>prior()</code> is specified
<code>e(priorparams)</code>	parameter specification in <code>prior()</code>
<code>e(priorparams#)</code>	parameter specification from <code>#th prior()</code> , if more than one <code>prior()</code> is specified
<code>e(parnames)</code>	names of model parameters except <code>exclude()</code>
<code>e(postvars)</code>	variable names corresponding to model parameters in <code>e(parnames)</code>
<code>e(subexpr)</code>	substitutable expression
<code>e(subexpr#)</code>	substitutable expression, if more than one
<code>e(wtype)</code>	weight type (one equation)
<code>e(wtype#)</code>	weight type for <code>#th</code> equation
<code>e(wexp)</code>	weight expression (one equation)
<code>e(wexp#)</code>	weight expression for <code>#th</code> equation
<code>e(block#_names)</code>	parameter names from <code>#th</code> block

<code>e(exclude)</code>	names of excluded parameters
<code>e(filename)</code>	name of the file with simulation results
<code>e(scpars)</code>	scalar model parameters
<code>e(matparams)</code>	matrix model parameters
<code>e(pareqmap)</code>	model parameters in display order
<code>e(title)</code>	title in estimation output
<code>e(rngstate)</code>	random-number state at the time of simulation (only with single chain)
<code>e(rngstate#)</code>	random-number state for #th chain (only with <code>nchains()</code> )
<code>e(search)</code>	on, <code>repeat()</code> , or off

#### Matrices

<code>e(mean)</code>	posterior means
<code>e(sd)</code>	posterior standard deviations
<code>e(mcse)</code>	MCSE
<code>e(median)</code>	posterior medians
<code>e(cri)</code>	credible intervals
<code>e(Cov)</code>	variance–covariance matrix of parameters
<code>e(ess)</code>	effective sample sizes
<code>e(init)</code>	initial values vector
<code>e(dic_chains)</code>	deviance information criterion for each chain (only with <code>nchains()</code> )
<code>e(arate_chains)</code>	acceptance rate for each chain (only with <code>nchains()</code> )
<code>e(eff_min_chains)</code>	minimum efficiency for each chain (only with <code>nchains()</code> )
<code>e(eff_avg_chains)</code>	average efficiency for each chain (only with <code>nchains()</code> )
<code>e(eff_max_chains)</code>	maximum efficiency for each chain (only with <code>nchains()</code> )
<code>e(lml_lm_chains)</code>	log marginal-likelihood for each chain (only with <code>nchains()</code> )

#### Functions

<code>e(sample)</code>	mark estimation sample
------------------------	------------------------

## Methods and formulas

Methods and formulas are presented under the following headings:

*Adaptive MH algorithm*

*Adaptive MH algorithm for random effects*

*Gibbs sampling for some likelihood-prior and prior-hyperprior configurations*

*Likelihood-prior configurations*

*Prior-hyperprior configurations*

*Marginal likelihood*

## Adaptive MH algorithm

The `bayesmh` command implements an adaptive random-walk Metropolis–Hastings algorithm with optional blocking of parameters. Providing an efficient MH procedure for simulating from a general posterior distribution is a difficult task, and various adaptive methods have been proposed (Haario, Saksman, and Tamminen 2001; Giordani and Kohn 2010; Roberts and Rosenthal 2009; Andrieu and Thoms 2008). The essence of the problem is in choosing an optimal proposal covariance matrix and a scale for parameter updates. Below we describe the implemented adaptation algorithm, assuming one block of parameters. In the presence of multiple blocks, the adaptation is applied to each block independently. The `adaptation()` option of `bayesmh` controls all the tuning parameters for the adaptation algorithm.

Let  $\theta$  be a vector of  $d$  scalar model parameters. Let  $T_0$  be the length of a burn-in period (iterations that are discarded) as specified in `burnin()` and  $T$  be the size of the MCMC sample (iterations that are retained) as specified in `mcmcsize()`. The total number of MCMC iterations is then  $T_{\text{total}} = T_0 + (T - 1) \times \text{thinning}() + 1$ . Also, let  $\text{ALEN}$  be the length of the adaptation interval (option `adaptation(every())`) and  $\text{AMAX}$  be the maximum number of adaptations (option `adaptation(maxiter())`).

The steps of the adaptive MH algorithm are the following. At  $t = 0$ , we initialize  $\theta_t = \theta_0^f$ , where  $\theta_0^f$  is the initial feasible state, and we set adaptation counter  $k = 1$  and initialize  $\rho_0 = 2.38/\sqrt{d}$ , where  $d$  is the number of considered parameters.  $\Sigma_0$  is the identity matrix. For  $t = 1, \dots, T_{\text{total}}$ , do the following:

1. Generate proposal parameters:  $\theta_* = \theta_{t-1} + \mathbf{e}$ ,  $\mathbf{e} \sim N(\mathbf{0}, \rho_k^2 \Sigma_k)$ , where  $\rho_k$  and  $\Sigma_k$  are current values of the proposal scale and covariance for adaptation iteration  $k$ .
2. Calculate the acceptance probability using

$$\alpha(\theta_* | \theta_{t-1}) = \min \left\{ \frac{p(\theta_* | \mathbf{y})}{p(\theta_{t-1} | \mathbf{y})}, 1 \right\}$$

where  $p(\theta | \mathbf{y}) = f(\mathbf{y} | \theta)p(\theta)$  is the posterior distribution of  $\theta$  corresponding to the likelihood function  $f(\mathbf{y} | \theta)$  and prior  $p(\theta)$ .

3. Draw  $u \sim \text{Uniform}(0, 1)$  and set  $\theta_t = \theta_*$  if  $u < \alpha(\theta_* | \theta_{t-1})$  or  $\theta_t = \theta_{t-1}$ , otherwise.
4. Perform adaptive iteration  $k$ . This step is performed only if  $k \leq \text{AMAX}$  and  $t \bmod \text{ALEN} = 0$ . Update  $\rho_k$  according to (2), update  $\Sigma_k$  according to (3), and set  $k = k + 1$ .
5. Repeat steps 1–4. Note that the adaptation in step 4 is not performed at every MCMC iteration.

The output is the MCMC sequence  $\{\theta_t\}_{t=T_0+1}^{T_{\text{total}}}$  or  $\theta_1, \theta_{1+l}, \theta_{1+2l}, \dots$ , where  $l$  is the thinning interval as specified in the `thinning()` option.

If the parameter vector  $\theta$  is split into  $B$  blocks  $\theta^1, \theta^2, \dots, \theta^B$ , then steps 1 through 3 are repeated for each  $\theta^b$ ,  $b = 1, \dots, B$  sequentially. The adaptation in step 4 is then applied sequentially to each block  $b = 1, 2, \dots, B$ . See [Blocking of parameters](#) in [BAYES] [Intro](#) for details about blocking.

**Initialization.** We recommend that you carefully choose starting values for model parameters,  $\theta_0$ , to be within the domain of the posterior distribution; see [Specifying initial values](#). By default, for a single chain, MLEs are used as initial values, whenever available. If MLEs are not available, parameters with positive support are initialized with 1, probabilities are initialized with 0.5, and the remaining parameters are initialized with 0. Matrix parameters are initialized as identity matrices. If specified initial values  $\theta_0$  are within the domain of the posterior, then  $\theta_0^f = \theta_0$ . Otherwise, `bayesmh` performs 500 attempts (or as specified in `search(repeat())`) to find a feasible state  $\theta_0^f$ , which is used as the initial state in the algorithm. If the command cannot find feasible values, it exits with an error.

You can specify the `initrando` option to request random initial values for all model parameters. In this case, `bayesmh` generates random initial values from the corresponding prior distributions of the parameters, except for those that are assigned improper priors such as `flat` and `jeffreys()` or user-defined priors using the `density()` and `logdensity()` prior options. You must specify your own initial values for all model parameters for which random initial values cannot be generated.

With multiple chains, the initial values for the first chain are generated as described above and random initial values are generally generated from prior distributions for subsequent chains.

See [Specifying initial values](#) for details.

**Adaptation.** The adaptation step is performed as follows. At each adaptive iteration  $k$  of the  $t$ th MCMC iteration, the proposal covariance  $\Sigma_k$  and scale  $\rho_k$  are tuned to achieve an optimal AR. Some asymptotic results (for example, [Gelman, Gilks, and Roberts \[1997\]](#)) show that the optimal AR, hereafter referred to as a TAR, for a single model parameter is 0.44 and is 0.234 for a block of multiple parameters.

Adaptation is performed periodically after a constant number of iterations as specified by the `adaptation(every())` option. At least `adaptation(miniter())` adaptive iterations are performed not



to exceed `adaptation(maxiter())`. `bayesmh` does not perform adaptation if the absolute difference between the current AR and TAR is within the tolerance given by `adaptation(tolerance())`.

The `bayesmh` command allows you to control the calculation of AR through the `adaptation(alpha())` option with the default of 0.75, as follows,

$$\text{AR}_k = (1 - \alpha)\text{AR}_{k-1} + \alpha\widehat{\text{AR}}_k$$

where  $\widehat{\text{AR}}_k$  is the expected acceptance probability, which is computed as the average of the acceptance probabilities,  $\alpha(\theta_*|\theta_{t-1})$ , since the last adaptive iteration (for example, [Andrieu and Thoms \[2008\]](#)), and  $\text{AR}_0$  is defined as described in the `adaptation(tarate())` option. Choosing  $\alpha \in (0, 1)$  allows for smoother change in the current AR between adaptive iterations.

The tuning of the proposal scale  $\rho$  is based on results in [Gelman, Gilks, and Roberts \(1997\)](#), [Roberts and Rosenthal \(2001\)](#), and [Andrieu and Thoms \(2008\)](#). The initial  $\rho_0$  is set to  $2.38/\sqrt{d}$ , where  $d$  is the number of parameters in the considered block. Then,  $\rho_k$  is updated according to

$$\rho_k = \rho_{k-1} e^{\beta_k \{\Phi^{-1}(\text{AR}_k/2) - \Phi^{-1}(\text{TAR}/2)\}} \quad (2)$$

where  $\Phi(\cdot)$  is the standard normal cumulative distribution function and  $\beta_k$  is defined below.

The adaptation of the covariance matrix is performed when multiple parameters are in the block and is based on [Andrieu and Thoms \(2008\)](#). You may specify an initial proposal covariance matrix  $\Sigma_0$  in `covariance()` or use the identity matrix by default. Then, at time of adaptation  $k$ , the proposal covariance  $\Sigma_k$  is recomputed according to the formula

$$\Sigma_k = (1 - \beta_k)\Sigma_{k-1} + \beta_k\widehat{\Sigma}_k, \quad \beta_k = \frac{\beta_0}{k^\gamma} \quad (3)$$

where  $\widehat{\Sigma}_k = (\Theta_{t_k} - \mu_{k-1})(\Theta_{t_k} - \mu_{k-1})'/(t_k - t_{k-1})$  is the empirical covariance of the recent MCMC sample  $\Theta_{t_k} = \{\theta_s\}_{s=t_{k-1}}^{t_k}$  and  $t_{k-1}$  is the MCMC iteration corresponding to the adaptive iteration  $k-1$  or 0 if adaptation did not take place.  $\mu_k$  is defined as

$$\mu_k = \mu_{k-1} + \beta_k(\bar{\Theta}_{t_k} - \mu_{k-1}), \quad k > 1$$

and  $\mu_1 = \bar{\Theta}_{t_k}$ , where  $\bar{\Theta}_{t_k}$  is the sample mean of  $\Theta_{t_k}$ .

The constants  $\beta_0 \in [0, 1]$  and  $\gamma \in [0, 1]$  in (3) are specified in the options `adaptation(beta())` and `adaptation(gamma())`, respectively. The default values are 0.8 and 0, respectively. When  $\gamma > 0$ , we have a diminishing adaptation regime, which means that  $\Sigma_k$  is not changing much from one adaptive iteration to another. Random-walk Metropolis–Hastings algorithms with diminishing adaptation are shown to preserve the ergodicity of the Markov chain ([Roberts and Rosenthal 2007](#); [Andrieu and Moulines 2006](#); [Atchadé and Rosenthal 2005](#)).

The above algorithm is also used for discrete parameters, but discretization is used to obtain samples of discrete values. The default initial scale factor  $\rho_0$  is set to  $2.38/d$  for a block of  $d$  discrete parameters. The default TAR for discrete parameters with priors `bernoulli()` and `index()` is  $\max\{0.1353, 1/n_{\text{maxbins}}\}$ , where  $n_{\text{maxbins}}$  is the maximum number of discrete values a parameter can take among all the parameters specified in the same block. Blocks containing a mixture of continuous and discrete parameters are not allowed.

## Adaptive MH algorithm for random effects

Suppose that  $\mathbf{u}$  is a random-effects variable that takes discrete values  $1, \dots, m$ . For an independent sample  $Y = \{y_{ij}\}$ , where  $j = 1, \dots, m$  and where  $i = 1, \dots, n_j$ , we assume that  $\mathbf{u}$  takes value  $j$  for all  $y_{ij}$ , where  $i = 1, \dots, n_j$ . Consider a two-level Bayesian model that includes random-effect parameters  $\eta_j$ , where  $j = 1, \dots, m$ , one for each level of  $\mathbf{u}$ , and additional parameter vector  $\boldsymbol{\theta}$ . We assume that, with respect to the posterior distribution of the model, the random-effects parameters  $\eta_j$  are conditionally independent given  $\boldsymbol{\theta}$  and the data sample  $Y$ . The latter can be ensured the prior distribution of  $\eta_j$ 's satisfies the conditional independence condition

$$\pi(\eta_1, \dots, \eta_m | \boldsymbol{\theta}) = \prod_{j=1}^m \pi(\eta_j | \boldsymbol{\theta})$$

In this case, the posterior distribution admits the following factorization,

$$\Pr(\eta_1, \dots, \eta_m, \boldsymbol{\theta} | Y) = \pi(\boldsymbol{\theta}) \left\{ \prod_{j=1}^m \pi(\eta_j | \boldsymbol{\theta}) \prod_{i=1}^{n_j} \Pr(y_{ij} | \eta_j, \boldsymbol{\theta}) \right\}$$

where  $\pi(\boldsymbol{\theta})$  is the prior distribution of  $\boldsymbol{\theta}$ . This form of the posterior allows the parameters  $\eta_j$ 's to be placed in one block and steps 1, 2, and 3 of the adaptive MH algorithm to be performed for all of them simultaneously in a vector form, as if they were a single scalar parameter.

To request the random-effects MH algorithm in `bayesmh`, use `block`'s suboption `reffects`. The same algorithm is used if one specifies the `reffects()` option. A random-effects block of parameters has a default acceptance rate of 0.44, performs adaptation of the scale  $\rho_k$  according to (2), but uses a fixed identity matrix for the proposal covariance  $\Sigma_k$ .

## Gibbs sampling for some likelihood-prior and prior-hyperprior configurations

In some cases, when a block of parameters  $\boldsymbol{\theta}^b$  has a conjugate prior, or more appropriately, a semiconjugate prior, with respect to the respective likelihood distribution for this block, you can request Gibbs sampling instead of random-walk MH sampling. Then, steps 1 through 4 of the algorithm described in [Adaptive MH algorithm](#) are replaced with just one step of Gibbs sampling as follows:

1'. Simulate proposal parameters:  $\boldsymbol{\theta}_*^b \sim F_b(\boldsymbol{\theta}^b | \boldsymbol{\theta}_*^1, \dots, \boldsymbol{\theta}_*^{b-1}, \boldsymbol{\theta}_*^{b+1}, \dots, \boldsymbol{\theta}_*^B, \mathbf{y})$

Here  $F_b(\cdot | \cdot)$  is the full conditional distribution of  $\boldsymbol{\theta}^b$  with respect to the rest of the parameters.

Below we list the full conditional distributions for the likelihood-prior specifications for which `bayesmh` provides Gibbs sampling. All priors except Jeffreys priors are semiconjugate, meaning that full conditional distributions belong to the same family as the specified prior distributions for the chosen data model. This contrasts with a concept of conjugacy under which the posterior distribution of all parameters belongs to the same family as the joint prior distribution. All the combinations below assume prior independence; that is, all parameters are independent a priori. Thus their joint prior distribution is simply the product of the individual prior distributions.

## Likelihood-prior configurations

Let  $\mathbf{y} = (y_1, y_2, \dots, y_n)'$  be a data sample of size  $n$ . For multivariate data,  $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)' = \{y_{ij}\}_{i,j=1}^{n,d}$  is an  $n \times d$  data matrix.

1. **Normal–normal model:**  $\theta^b$  is a mean of a normal distribution of  $y_i$ 's with a variance  $\sigma^2$ ; mean and variance are independent a priori,

$$\begin{aligned} y_i | \theta^b, \sigma^2 &\sim N(\theta^b, \sigma^2), \quad i = 1, 2, \dots, n \\ \theta^b &\sim N(\mu_0, \tau_0^2) \\ \theta^b | \sigma^2, \mathbf{y} &\sim F_b = N(\mu_n, \tau_n^2) \end{aligned}$$

where  $\mu_0$  and  $\tau_0^2$  are hyperparameters (prior mean and prior variance) of a normal prior distribution for  $\theta^b$  and

$$\begin{aligned} \mu_n &= \left( \mu_0 \tau_0^{-2} + \sum y_i \sigma^{-2} \right) \tau_n^2 \\ \tau_n^2 &= (\tau_0^{-2} + n \sigma^{-2})^{-1} \end{aligned}$$

2. **Normal–normal regression:**  $\theta^b$  is a  $p_1 \times 1$  subvector of a  $p \times 1$  vector of regression coefficients  $\beta$  from a normal linear regression model for  $\mathbf{y}$  with an  $n \times p$  design matrix  $X = (\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_n)'$  and with a variance  $\sigma^2$ ; regression coefficients and variance are independent a priori,

$$\begin{aligned} y_i | \theta^b, \sigma^2 &\sim N(\mathbf{x}'_i \beta, \sigma^2), \quad i = 1, 2, \dots, n \\ \theta_k^b &\sim \text{i.i.d. } N(\beta_0, \tau_0^2), \quad k = 1, 2, \dots, p_1 \\ \theta^b | \sigma^2, \mathbf{y} &\sim F_b = \text{MVN}(\mu_n, \Lambda_n) \end{aligned}$$

where  $\beta_0$  and  $\tau_0^2$  are hyperparameters (prior regression coefficient and prior variance) of normal prior distributions for  $\theta_k^b$  and

$$\begin{aligned} \mu_n &= (\beta_0 \tau_0^{-2} + X_b' \mathbf{y} \sigma^{-2}) \Lambda_n \\ \Lambda_n &= (\tau_0^{-2} I_{p_1} + \sigma^{-2} X_b' X_b)^{-1} \end{aligned}$$

In the above,  $I_{p_1}$  is a  $p_1 \times p_1$  identity matrix, and  $X_b = (\mathbf{x}'_{1b}, \mathbf{x}'_{2b}, \dots, \mathbf{x}'_{nb})'$  is an  $n \times p_1$  submatrix of  $X$  corresponding to the regression coefficients  $\theta^b$ .

3. **Normal–inverse-gamma model:**  $\theta^b$  is a variance of a normal distribution of  $y_i$ 's with a mean  $\mu$ ; mean and variance are independent a priori,

$$\begin{aligned} y_i | \mu, \theta^b &\sim N(\mu, \theta^b), \quad i = 1, 2, \dots, n \\ \theta^b &\sim \text{InvGamma}(\alpha, \beta) \\ \theta^b | \mu, \mathbf{y} &\sim F_b = \text{InvGamma}(\alpha + n/2, \beta + \sum_{i=1}^n (y_i - \mu)^2 / 2) \end{aligned}$$

where  $\alpha$  and  $\beta$  are hyperparameters (prior shape and prior scale) of an inverse-gamma prior distribution for  $\theta^b$ .

4. **Multivariate-normal–multivariate-normal model:**  $\theta^b$  is a mean vector of a multivariate normal distribution of  $\mathbf{y}_s$  with a  $d \times d$  covariance matrix  $\Sigma$ ; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i | \theta^b, \Sigma &\sim \text{MVN}(\theta^b, \Sigma), \quad i = 1, 2, \dots, n \\ \theta^b &\sim \text{MVN}(\mu_0, \Lambda_0) \\ \theta^b | \Sigma, Y &\sim F_b = \text{MVN}(\mu_n, \Lambda_n) \end{aligned}$$

where  $\mu_0$  and  $\Lambda_0$  are hyperparameters (prior mean vector and prior covariance) of a multivariate normal prior distribution for  $\theta^b$  and

$$\begin{aligned} \mu_n &= \Lambda_n \Lambda_0^{-1} \mu_0 + \Lambda_n \Sigma^{-1} \left( \sum_{i=1}^n \mathbf{y}_i \right) \\ \Lambda_n &= (\Lambda_0^{-1} + n \Sigma^{-1})^{-1} \end{aligned}$$

5. **Multivariate-normal–inverse-Wishart model:**  $\Theta^b$  is a  $d \times d$  covariance matrix of a multivariate normal distribution of  $\mathbf{y}_s$  with a mean vector  $\mu$ ; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i | \mu, \Theta^b &\sim \text{MVN}(\mu, \Theta^b), \quad i = 1, 2, \dots, n \\ \Theta^b &\sim \text{InvWishart}(\nu, \Psi) \\ \Theta^b | \mu, Y &\sim F_b = \text{InvWishart}(n + \nu, \Psi + \sum_{i=1}^n (\mathbf{y}_i - \mu)(\mathbf{y}_i - \mu)') \end{aligned}$$

where  $\nu$  and  $\Psi$  are hyperparameters (prior degrees of freedom and prior scale matrix) of an inverse-Wishart prior distribution for  $\Theta^b$ .

6. **Multivariate-normal–Jeffreys model:**  $\Theta^b$  is a  $d \times d$  covariance matrix of a multivariate normal distribution of  $\mathbf{y}_s$  with a mean vector  $\mu$ ; mean and covariance are independent a priori,

$$\begin{aligned} \mathbf{y}_i | \mu, \Theta^b &\sim \text{MVN}(\mu, \Theta^b), \quad i = 1, 2, \dots, n \\ \Theta^b &\sim |\Theta^b|^{-\frac{d+1}{2}} \quad (\text{multivariate Jeffreys}) \\ \Theta^b | \mu, Y &\sim F_b = \text{InvWishart}(n - 1, \sum_{i=1}^n (\mathbf{y}_i - \mu)(\mathbf{y}_i - \mu)') \end{aligned}$$

where  $|\cdot|$  denotes the determinant of a matrix.

## Prior-hyperprior configurations

Suppose that a prior distribution of a parameter of interest  $\theta$  has hyperparameters  $\theta_h$  for which a prior distribution is specified. We refer to the former prior distribution as a hyperprior. You can also request Gibbs sampling for the following prior-hyperprior combinations.

We use  $\theta_h^b$  and  $\sigma_h^b$  to refer to the hyperparameters from the block  $b$ .

1. **Normal–normal model:**  $\theta_h^b$  is a mean of a normal prior distribution of  $\theta$  with a variance  $\sigma_h^2$ ; mean and variance are independent a priori,

$$\begin{aligned} \theta | \theta_h^b, \sigma_h^2 &\sim N(\theta_h^b, \sigma_h^2) \\ \theta_h^b &\sim N(\mu_0, \tau_0^2) \\ \theta_h^b | \sigma_h^2, \theta &\sim F_b = N(\mu_1, \tau_1^2) \end{aligned}$$

where  $\mu_0$  and  $\tau_0^2$  are the prior mean and prior variance of a normal hyperprior distribution for  $\theta_h^b$  and

$$\begin{aligned}\mu_1 &= (\mu_0 \tau_0^{-2} + \theta \sigma_h^{-2}) \tau_1^2 \\ \tau_1^2 &= (\tau_0^{-2} + \sigma_h^{-2})^{-1}\end{aligned}$$

2. **Normal–inverse-gamma model:**  $\theta_h^b$  is a variance of a normal prior distribution of  $\theta$  with a mean  $\mu_h$ ; mean and variance are independent a priori,

$$\begin{aligned}\theta | \mu_h, \theta_h^b &\sim N(\mu_h, \theta_h^b) \\ \theta_h^b &\sim \text{InvGamma}(\alpha, \beta) \\ \theta_h^b | \mu_h, \theta &\sim F_b = \text{InvGamma}(\alpha + 0.5, \beta + (\theta - \mu)^2/2)\end{aligned}$$

where  $\alpha$  and  $\beta$  are the prior shape and prior scale, respectively, of an inverse-gamma hyperprior distribution for  $\theta_h^b$ .

3. **Bernoulli–beta model:**  $\theta_h^b$  is a probability of success of a Bernoulli prior distribution of  $\theta$ ,

$$\begin{aligned}\theta | \theta_h^b &\sim \text{Bernoulli}(\theta_h^b) \\ \theta_h^b &\sim \text{Beta}(\alpha, \beta) \\ \theta_h^b | \theta &\sim F_b = \text{Beta}(\alpha + \theta, \beta + 1 - \theta)\end{aligned}$$

where  $\alpha$  and  $\beta$  are the prior shape and prior scale, respectively, of a beta hyperprior distribution for  $\theta_h^b$ .

4. **Poisson–gamma model:**  $\theta_h^b$  is a mean of a Poisson prior distribution of  $\theta$ ,

$$\begin{aligned}\theta | \theta_h^b &\sim \text{Poisson}(\theta_h^b) \\ \theta_h^b &\sim \text{Gamma}(\alpha, \beta) \\ \theta_h^b | \theta &\sim F_b = \text{Gamma}(\alpha + \theta, \beta/(\beta + 1))\end{aligned}$$

where  $\alpha$  and  $\beta$  are the prior shape and prior scale, respectively, of a gamma hyperprior distribution for  $\theta_h^b$ .

5. **Multivariate-normal–multivariate-normal model:**  $\theta_h^b$  is a mean vector of a multivariate normal prior distribution of  $\theta$  with a  $d \times d$  covariance matrix  $\Sigma_h$ ; mean and covariance are independent a priori,

$$\begin{aligned}\theta | \theta_h^b, \Sigma_h &\sim \text{MVN}(\theta_h^b, \Sigma_h) \\ \theta_h^b &\sim \text{MVN}(\mu_0, \Lambda_0) \\ \theta_h^b | \Sigma_h, \theta &\sim F_b = \text{MVN}(\mu_1, \Lambda_1)\end{aligned}$$

where  $\mu_0$  and  $\Lambda_0$  are the prior mean vector and prior covariance of a multivariate normal hyperprior distribution for  $\theta_h^b$  and

$$\begin{aligned}\mu_1 &= \Lambda_1 \Lambda_0^{-1} \mu_0 + \Lambda_1 \Sigma_h^{-1} \theta \\ \Lambda_1 &= (\Lambda_0^{-1} + \Sigma_h^{-1})^{-1}\end{aligned}$$

6. **Multivariate-normal-inverse-Wishart model:**  $\Theta_h^b$  is a  $d \times d$  covariance matrix of a multivariate normal prior distribution of  $\theta$  with a mean vector  $\mu_h$ ; mean and covariance are independent a priori,

$$\begin{aligned}\theta | \mu_h, \Theta_h^b &\sim \text{MVN}(\mu_h, \Theta_h^b) \\ \Theta_h^b &\sim \text{InvWishart}(\nu, \Psi) \\ \Theta_h^b | \mu_h, \theta &\sim F_b = \text{InvWishart}(\nu + 1, \Psi + (\theta - \mu_h)(\theta - \mu_h)')\end{aligned}$$

where  $\nu$  and  $\Psi$  are the prior degrees of freedom and prior scale matrix of an inverse-Wishart hyperprior distribution for  $\Theta_h^b$ .

## Marginal likelihood

The marginal likelihood is defined as

$$m(\mathbf{y}) = \int p(\mathbf{y}|\theta)\pi(\theta)d\theta$$

where  $p(\mathbf{y}|\theta)$  is the probability density of data  $\mathbf{y}$  given  $\theta$  and  $\pi(\theta)$  is the density of the prior distribution for  $\theta$ .

Marginal likelihood  $m(\mathbf{y})$ , being the denominator term in the posterior distribution, has a major role in Bayesian analysis. It is sometimes referred to as “model evidence”, and it is used as a goodness-of-fit criterion. For example, marginal likelihoods are used in calculating Bayes factors for the purpose of model comparison; see [Methods and formulas](#) in [\[BAYES\] bayesstats ic](#).

The simplest approximation to  $m(\mathbf{y})$  is provided by the Monte Carlo integration,

$$\hat{m}_p = \frac{1}{M} \sum_{s=1}^M p(\mathbf{y}|\theta_s)$$

where  $\{\theta_s\}_{s=1}^M$  is an independent sample from the prior distribution  $\pi(\theta)$ . This estimation is very inefficient, however, because of the high variance of the likelihood function. MCMC samples are not independent and cannot be used directly for calculating  $\hat{m}_p$ .

An improved estimation of the marginal likelihood can be obtained by using importance sampling. For a sample  $\{\theta_t\}_{t=1}^T$ , not necessarily independent, from the posterior distribution, the harmonic mean of the likelihood values,

$$\hat{m}_h = \left\{ \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}|\theta_t)^{-1} \right\}^{-1}$$

approximates  $m(\mathbf{y})$  ([Geweke 1989](#)).

Another method for estimating  $m(\mathbf{y})$  uses the Laplace approximation,

$$\hat{m}_l = (2\pi)^{p/2} |\tilde{H}|^{-1/2} p(\mathbf{y}|\tilde{\theta})\pi(\tilde{\theta})$$

where  $p$  is the number of parameters (or dimension of  $\theta$ ),  $\tilde{\theta}$  is the posterior mode, and  $\tilde{H}$  is the Hessian matrix of  $l(\theta) = p(\mathbf{y}|\theta)\pi(\theta)$  calculated at the mode  $\tilde{\theta}$ .

Using the fact that the posterior sample covariance matrix, which we denote as  $\hat{\Sigma}$ , is asymptotically equal to  $(-\tilde{H})^{-1}$ , [Raftery \(1996\)](#) proposed what he called the Laplace–Metropolis estimator (implemented by `bayesmh`):

$$\hat{m}_{lm} = (2\pi)^{p/2} |\hat{\Sigma}|^{1/2} p(\mathbf{y}|\tilde{\theta}) \pi(\tilde{\theta})$$

[Raftery \(1996\)](#) recommends that a robust and consistent estimator be used for the posterior covariance matrix.

Estimation of the log marginal-likelihood becomes unstable for high-dimensional models such as multilevel models and may result in a missing value.

With multiple chains, an average of the log-marginal-likelihood values over the chains is reported.

[Nicholas Constantine Metropolis](#) (1915–1999) was born in Chicago. He completed his PhD in experimental physics at the University of Chicago in 1941. In 1943, Metropolis moved to Los Alamos, where he spent much of his time working on computers and computational algorithms. He first worked with analog and then IBM punch card machines. Beginning in 1948, he helped design the MANIAC I computer, one of the first digital computers. He later oversaw the construction of the MANIAC II and MANIAC III. He collaborated with Stanislaw Ulam to develop the Monte Carlo method, and he coauthored a paper in 1953 introducing the Monte Carlo algorithm. The algorithm would later be extended to general cases by W. K. Hastings and would be known as the Metropolis–Hastings algorithm. In 1957, Metropolis returned to the University of Chicago, where he taught physics and helped found the Institute for Computer Research.

The American Physical Society elected Metropolis as a fellow in 1953 and created an award in his honor that recognizes extraordinary work in computational physics. Also, in 1984, the Institute of Electrical and Electronics Engineers (IEEE) awarded him the Computer Pioneer Award. In his late 70s, Metropolis appeared in a Woody Allen film, portraying a scientist.

[Wilfred Keith Hastings](#) (1930–2016) was born in Toronto, Ontario, Canada. He studied applied mathematics at the University of Toronto, obtaining his bachelors in 1953 and later working as a computer applications consultant. In this position, he was exposed to statistics and gained experience with simulations. In 1962, he obtained his PhD, also from the University of Toronto. His dissertation was on fiducial distributions, but after attending a statistics conference, he learned that people were abandoning the study of fiducial probability. Shortly after graduation, he joined the faculty at the University of Canterbury for two years and then worked at the research company Bell Labs for two years as well. In 1966, he became an associate professor at his alma mater, and three years later he published his work on the Markov chain Monte Carlo (MCMC) method. His publication on Monte Carlo sampling methods was an extension of the algorithm introduced in the 1953 publication by Nicholas Metropolis et al. The idea originated from his interactions and consultations with the chemistry department’s application of the Metropolis algorithm to estimating the energy of particles. Hastings’s publication was cited over 2,000 times and gave rise to the Metropolis–Hastings algorithm. After this publication, Hastings served as a professor at the University of Victoria for 21 years and conducted research with multiple grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

Harold Jeffreys (1891–1989) was born near Durham, England, and spent more than 75 years studying and working at the University of Cambridge, principally on theoretical and observational problems in geophysics, astronomy, mathematics, and statistics. He developed a systematic Bayesian approach to inference in his monograph *Theory of Probability*.

## References

- Andrieu, C., and É. Moulines. 2006. On the ergodicity properties of some adaptive MCMC algorithms. *Annals of Applied Probability* 16: 1462–1505.
- Andrieu, C., and J. Thoms. 2008. A tutorial on adaptive MCMC. *Statistics and Computing* 18: 343–373.
- Atchadé, Y. F., and J. S. Rosenthal. 2005. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli* 11: 815–828.
- Balov, N. 2016a. Bayesian binary item response theory models using bayesmh. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/01/18/bayesian-binary-item-response-theory-models-using-bayesmh/>.
- . 2016b. Fitting distributions using bayesmh. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/03/30/fitting-distributions-using-bayesmh/>.
- . 2016c. Gelman–Rubin convergence diagnostic using multiple chains. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/05/26/gelman-rubin-convergence-diagnostic-using-multiple-chains/>.
- Birnbaum, A. 1968. Some latent trait models and their use in inferring an examinee’s ability. In *Statistical Theories of Mental Test Scores*, ed. F. M. Lord and M. R. Novick, 395–479. Reading, MA: Addison–Wesley.
- Carlin, B. P., A. E. Gelfand, and A. F. M. Smith. 1992. Hierarchical Bayesian analysis of changepoint problems. *Journal of the Royal Statistical Society, Series C* 41: 389–405.
- Carlin, J. B. 1992. Meta-analysis for  $2 \times 2$  tables: A Bayesian approach. *Statistics in Medicine* 11: 141–158.
- De Boeck, P., and M. Wilson, ed. 2004. *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. New York: Springer.
- Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.
- Gelfand, A. E., S. E. Hills, A. Racine-Poon, and A. F. M. Smith. 1990. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association* 85: 972–985.
- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Gelman, A., W. R. Gilks, and G. O. Roberts. 1997. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Annals of Applied Probability* 7: 110–120.
- Geweke, J. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* 57: 1317–1339.
- Geyer, C. J. 2011. Introduction to Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, ed. S. P. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, 3–48. Boca Raton, FL: Chapman & Hall/CRC.
- Giordani, P., and R. J. Kohn. 2010. Adaptive independent Metropolis–Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics* 19: 243–259.
- Grant, R. L., B. Carpenter, D. C. Furr, and A. Gelman. 2017a. Introducing the StataStan interface for fast, complex Bayesian modeling using Stan. *Stata Journal* 17: 330–342.
- . 2017b. Fitting Bayesian item response models in Stata and Stan. *Stata Journal* 17: 343–357.
- Haario, H., E. Saksman, and J. Tamminen. 2001. An adaptive Metropolis algorithm. *Bernoulli* 7: 223–242.
- Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.
- Huber, C. 2016a. Introduction to Bayesian statistics, part 1: The basic concepts. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/11/01/introduction-to-bayesian-statistics-part-1-the-basic-concepts/>.
- . 2016b. Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/11/15/introduction-to-bayesian-statistics-part-2-mcmc-and-the-metropolis-hastings-algorithm/>.



- Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report)*. National Institute of Population Research and Training.
- Jarrett, R. G. 1979. A note on the intervals between coal-mining disasters. *Biometrika* 66: 191–193.
- Jeffreys, H. 1946. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London, Series A* 186: 453–461.
- Lichman, M. 2013. UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>.
- Maas, B., W. R. Garnett, I. M. Pellock, and T. J. Comstock. 1987. A comparative bioavailability study of Carbamazepine tablets and chewable formulation. *Therapeutic Drug Monitoring* 9: 28–33.
- Maguire, B. A., E. S. Pearson, and A. H. A. Wynn. 1952. The time intervals between industrial accidents. *Biometrika* 39: 168–180.
- Marchenko, Y. V. 2015. Bayesian modeling: Beyond Stata’s built-in models. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2015/05/26/bayesian-modeling-beyond-statas-built-in-models/>.
- Raftery, A. E. 1996. Hypothesis testing and model selection. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, 163–187. Boca Raton, FL: Chapman and Hall.
- Raftery, A. E., and V. E. Akman. 1986. Bayesian analysis of a Poisson process with a change-point. *Biometrika* 73: 85–89.
- Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests*. Copenhagen: Danish Institute of Educational Research.
- Roberts, G. O., and J. S. Rosenthal. 2001. Optimal scaling for various Metropolis–Hastings algorithms. *Statistical Science* 16: 351–367.
- . 2007. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability* 44: 458–475.
- . 2009. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics* 18: 349–367.
- Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression*. Cambridge: Cambridge University Press.
- Thomas, A., B. O’Hara, U. Ligges, and S. Sturtz. 2006. Making BUGS Open. *R News* 6: 12–17.
- Thompson, J. 2014. *Bayesian Analysis with Stata*. College Station, TX: Stata Press.
- Yusuf, S., R. Simon, and S. S. Ellenberg. 1987. Proceedings of the workshop on methodological issues in overviews of randomized clinical trials, May 1986. In *Statistics in Medicine*, vol. 6.
- Zellner, A. 1986. On assessing prior distributions and Bayesian regression analysis with  $g$ -prior distributions. In Vol. 6 of *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti (Studies in Bayesian Econometrics and Statistics)*, ed. P. K. Goel and A. Zellner, 233–343. Amsterdam: North-Holland.
- Zellner, A., and N. S. Revankar. 1969. Generalized production functions. *Review of Economic Studies* 36: 241–250.

## Also see

- [BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] **bayesmh evaluators** — User-defined evaluators with bayesmh
- [BAYES] **bayes** — Bayesian regression models using the bayes prefix
- [BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis
- [BAYES] **Bayesian estimation** — Bayesian estimation commands
- [BAYES] **Intro** — Introduction to Bayesian analysis
- [BAYES] **Glossary**

Description	Syntax	Options	Remarks and examples
Stored results	Reference	Also see	

Description

bayesmh provides two options, evaluator() and llevaluator(), that facilitate user-defined evaluators for fitting general Bayesian regression models. bayesmh, evaluator() accommodates log-posterior evaluators. bayesmh, llevaluator() accommodates log-likelihood evaluators, which are combined with built-in prior distributions to form the desired posterior density. For a catalog of built-in likelihood models and prior distributions, see [BAYES] bayesmh.

Syntax

Single-equation models

User-defined log-posterior evaluator

```
bayesmh depvar [indepvars] [if] [in] [weight], evaluator(evalspec) [options]
```

User-defined log-likelihood evaluator

```
bayesmh depvar [indepvars] [if] [in] [weight], llevaluator(evalspec)
prior(priorspec) [options]
```

Multiple-equations models

User-defined log-posterior evaluator

```
bayesmh (eqspecp) [(eqspecp) [...]] [if] [in] [weight], evaluator(evalspec)
[options]
```

User-defined log-likelihood evaluator

```
bayesmh (eqspecll) [(eqspecll) [...]] [if] [in] [weight], prior(priorspec)
[options]
```

The syntax of *eqspec* is

```
varspec [ , noconstant ]
```

The syntax of *eqspecll* for built-in likelihood models is

```
varspec , likelihood(modelspec) [ noconstant ]
```

The syntax of *eqspecll* for user-defined log-likelihood evaluators is

```
varspec , llevaluator(evalspec) [ noconstant ]
```

The syntax of *varspec* is one of the following:

for single outcome

```
[ eqname: ] depvar [ indepvars ]
```

for multiple outcomes with common regressors

```
depvars = [ indepvars ]
```

for multiple outcomes with outcome-specific regressors

```
( [ eqname1: ] depvar1 [ indepvars1 ] ) ( [ eqname2: ] depvar2 [ indepvars2 ] ) [ ... ]
```

The syntax of *evalspec* is

```
progrname , parameters(paramlist) [ extravars(varlist) passthruopts(string) ]
```

where *progrname* is the name of a Stata program that you write to evaluate the log-posterior density or the log-likelihood function (see [Program evaluators](#)), and *paramlist* is a list of model parameters:

```
paramdef [ paramdef [ ... ] ]
```

The syntax of *paramdef* is

```
{ [ eqname: ] param [ param [ ... ] ] [ , matrix ] }
```

where the parameter label *eqname* and parameter names *param* are valid Stata names. Model parameters are either scalars such as {*var*}, {*mean*}, and {*shape:alpha*} or matrices such as {*Sigma*, *matrix*} and {*Scale:V*, *matrix*}. For scalar parameters, you can use {*param*=#} in the above to specify an initial value. For example, you can specify {*var*=1}, {*mean*=1.267}, or {*shape:alpha*=3}. You can specify the multiple parameters with same equation as {*eq*:p1 p2 p3} or {*eq*: S1 S2, *matrix*}. Also see [Declaring model parameters](#) in [BAYES] **bayesmh**.

<i>options</i>	Description
* <b>evaluator</b> ( <i>evalspec</i> )	specify log-posterior evaluator; may not be combined with <code>llevauator()</code> and <code>prior()</code>
* <b>llevauator</b> ( <i>evalspec</i> )	specify log-likelihood evaluator; requires <code>prior()</code> and may not be combined with <code>evaluator()</code>
* <b>prior</b> ( <i>priorspec</i> )	prior for model parameters; required with log-likelihood evaluator and may be repeated
<b>likelihood</b> ( <i>modelspec</i> )	distribution for the likelihood model; allowed within an equation of a multiple-equations model only
<b>noconstant</b>	suppress constant term; not allowed with ordered models specified in <code>likelihood()</code> with multiple-equations models
<i>bayesmhopts</i>	any options of [BAYES] <b>bayesmh</b> except <code>likelihood()</code> and <code>prior()</code>

\*Option `evaluator()` is required for log-posterior evaluators, and options `llevauator()` and `prior()` are required for log-likelihood evaluators. With log-likelihood evaluators, `prior()` must be specified for all model parameters and may be repeated.

*indepvars* may contain factor variables; see [U] 11.4.3 **Factor variables**.

Only *fweights* are allowed; see [U] 11.1.6 **weight**.

## Options

**evaluator**(*evalspec*) specifies the name and the attributes of the log-posterior evaluator; see [Program evaluators](#) for details. This option may not be combined with `llevauator()` or `likelihood()`.

**llevauator**(*evalspec*) specifies the name and the attributes of the log-likelihood evaluator; see [Program evaluators](#) for details. This option may not be combined with `evaluator()` or `likelihood()` and requires the `prior()` option.

**prior**(*priorspec*); see [BAYES] **bayesmh**.

**likelihood**(*modelspec*); see [BAYES] **bayesmh**. This option is allowed within an equation of a multiple-equations model only.

**noconstant**; see [BAYES] **bayesmh**.

*bayesmhopts* specify any *options* of [BAYES] **bayesmh**, except `likelihood()` and `prior()`.

## Remarks and examples

Remarks are presented under the following headings:

- [Program evaluators](#)
- [Simple linear regression model](#)
- [Logistic regression model](#)
- [Multivariate normal regression model](#)
- [Cox proportional hazards regression](#)
- [Global macros](#)

## Program evaluators

If your likelihood model or prior distributions are particularly complex and cannot be represented by one of the predefined sets of distributions or by substitutable expressions provided with **bayesmh**, you can program these functions by writing your own evaluator program.

Evaluator programs can be used for programming the full posterior density by specifying the `evaluator()` option or only the likelihood portion of your Bayesian model by specifying the `l1evaluator()` option. For likelihood evaluators, `prior()` option(s) must be specified for all model parameters. Your program is expected to calculate and return an overall log-posterior or a log-likelihood density value.

It is allowed for the return values to match the log density up to an additive constant, in which case, however, some of the reported statistics such as DIC and log marginal-likelihood may not be applicable.

Your program evaluator *progrname* must be a Stata program; see [U] 18 Programming Stata. The program must follow the style below.

```
program progrname
  args lnden xb1 [xb2 ...] [modelparams]
  ... computations ...
  scalar 'lnden' = ...
end
```

Here *lnden* contains the name of a temporary scalar to be filled in with an overall log-posterior or log-likelihood value;

*xb#* contains the name of a temporary variable containing the linear predictor from the #th equation; and

*modelparams* is a list of names of scalars or matrices to contain the values of model parameters specified in suboption `parameters()` of `evaluator()` or `l1evaluator()`. For matrix parameters, the specified names will contain the names of temporary matrices containing current values. For scalar parameters, these are the names of temporary scalars containing current values. The order in which names are listed should correspond to the order in which model parameters are specified in `parameters()`.

Also see [Global macros](#) for a list of global macros available to the program evaluator.

After you write a program evaluator, you specify its name in the option `evaluator()` for log-posterior evaluators,

```
. bayesmh ..., evaluator(progrname, evalopts)
```

or option `l1evaluator()` for log-likelihood evaluators,

```
. bayesmh ..., l1evaluator(progrname, evalopts)
```

Evaluator options *evalopts* include `parameters()`, `extravars()`, and `passthruopts()`.

`parameters(paramlist)` specifies model parameters. Model parameters can be scalars or matrices. Each parameter must be specified in curly braces {}. Multiple parameters with the same equation names may be specified within one set of {}.

For example,

```
parameters({mu} {var:sig2} {S,matrix} {cov:Sigma, matrix} {prob:p1 p2})
```

specifies a scalar parameter with name *mu* without an equation label, a scalar parameter with name *sig2* and label *var*, a matrix parameter with name *S*, a matrix parameter with name *Sigma* and label *cov*, and two scalar parameters {*prob:p1*} and {*prob:p2*}.

`extravars(varlist)` specifies any variables in addition to dependent and independent variables that you may need in your program evaluator. Examples of such variables are offset variables, exposure variables for count-data models, and failure or censoring indicators for survival-time models. See [Cox proportional hazards regression](#) for an example.

`passthruopts(string)` specifies a list of options you may want to pass to your program evaluator. For example, these options may contain fixed values of model parameters and hyperparameters. See [Multivariate normal regression model](#) for an example.

bayesmh automatically creates parameters for regression coefficients: `{depname:varname}` for every `varname` in `indepvars`, and a constant parameter `{depname:_cons}` unless `noconstant` is specified. These parameters are used to form linear predictors used by the program evaluator. If you need to access values of the parameters in the evaluator, you can use `$MH_b`; see the log-posterior evaluator in [Cox proportional hazards regression](#) for an example. With multiple dependent variables, regression coefficients are defined for each dependent variable.

## Simple linear regression model

Suppose that we want to fit a Bayesian normal regression where we program the posterior distribution ourselves. The `normaljeffreys` program below computes the log-posterior density for the normal linear regression with flat priors for the coefficients and the Jeffreys prior for the variance parameter.

```
. program normaljeffreys
1.     version 16.0
2.     args lnp xb var
3.     /* compute log likelihood */
.     tempname sd
4.     scalar 'sd' = sqrt('var')
5.     tempvar lnfj
6.     quietly generate double 'lnfj'=lnnormalden($MH_y,'xb','sd')
>                                     if $MH_touse
7.     quietly summarize 'lnfj', meanonly
8.     if r(N) < $MH_n {
9.         scalar 'lnp' = .
10.        exit
11.    }
12.    tempname lnf
13.    scalar 'lnf' = r(sum)
14.    /* compute log prior */
.    tempname lnprior
15.    scalar 'lnprior' = -2*ln('sd')
16.    /* compute log posterior */
.    scalar 'lnp' = 'lnf' + 'lnprior'
17. end
```

The program accepts three parameters: a temporary name `'lnp'` of a scalar to contain the log-posterior value, a temporary name `'xb'` of the variable that contains the linear predictor, and a temporary name `'var'` of a scalar that contains the values of the variance parameter.

The first part of the program calculates the overall log likelihood of the normal regression. The second part of the program calculates the log of prior distributions of the parameters. Because the coefficients have flat prior distributions with densities of 1, their log is 0 and does not contribute to the overall prior. The only contribution is from the Jeffreys prior  $\ln(1/\sigma^2) = -2\ln(\sigma)$  for the variance  $\sigma^2$ . The third and final part of the program computes the values of the posterior density as the sum of the overall log likelihood and the log of the prior.

The substantial portion of this program is the computation of the overall log likelihood. The global macro `$MH_y` contains the name of the dependent variable, `$MH_touse` contains a temporary marker

variable identifying observations to be used in computations, and `$MH_n` contains the total number of observations in the sample identified by the `$MH_touse` variable.

We used the built-in function `lnnormalden()` to compute observation-specific log likelihood and used `summarize` to obtain the overall value. Whenever a temporary variable is needed for calculations, such as `'lnfj'` in our program, it is important to create it of type `double` to ensure the highest precision of the results. It is also important to perform computations using only the relevant subset of observations as identified by the marker variable stored in `$MH_touse`. This variable contains the value of 1 for observations to be used in the computations and 0 for the remaining observations. Missing values in used variables, `if`, and `in` affect this variable. After we compute the log-likelihood value, we should verify that the number of nonmissing observation-specific contributions to the log likelihood equals `$MH_n`. If it does not, the log-posterior value (or log-likelihood value in a log-likelihood evaluator) must be set to missing.

We can now specify the `normaljeffreys` evaluator in the `evaluator()` option of `bayesmh`. In addition to the regression coefficients, we have one extra parameter, the variance of the normal distribution, which we must specify in the `parameters()` suboption of `evaluator()`.

We use `auto.dta` to illustrate the command. We specify a simple regression of `mpg` on rescaled `weight`.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. quietly replace weight = weight/100

. set seed 14

. bayesmh mpg weight, evaluator(normaljeffreys, parameters({var}))
Burn-in ...
note: invalid initial state
Simulation ...
Model summary
```

---

```
Posterior:
  mpg ~ normaljeffreys(xb_mpg,{var})
```

---

Bayesian regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1433
	Efficiency: min =	.06246
	avg =	.06669
	max =	.07091
Log marginal-likelihood =	-198.247	

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
weight	-.6052218	.053604	.002075	-.6062666	-.7121237	-.4992178
_cons	39.56782	1.658124	.066344	39.54211	36.35645	42.89876
var	12.19046	2.008871	.075442	12.03002	8.831172	17.07787

The output of `bayesmh` with user-defined evaluators is the same as the output of `bayesmh` with built-in distributions, except the title and the model summary. The generic title `Bayesian regression` is used for all evaluators, but you can change it by specifying the `title()` option. The model summary provides the name of the posterior evaluator.

Following the command line, there is a note about invalid initial state. For program evaluators, bayesmh initializes all parameters with zeros, except for positive parameters used in prior specifications, which are initialized with ones. This may not be sensible for all parameters, such as the variance parameter in our example. We may consider using, for example, OLS estimates as initial values of the parameters.

```
. regress mpg weight
```

Source	SS	df	MS	Number of obs	=	74
Model	1591.99021	1	1591.99021	F(1, 72)	=	134.62
Residual	851.469254	72	11.8259619	Prob > F	=	0.0000
				R-squared	=	0.6515
				Adj R-squared	=	0.6467
Total	2443.45946	73	33.4720474	Root MSE	=	3.4389

mpg	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
weight	-.6008687	.0517878	-11.60	0.000	-.7041058 -.4976315
_cons	39.44028	1.614003	24.44	0.000	36.22283 42.65774

```
. display e(rmse)^2
11.825962
```

We specify initial values in the initial() option.

```
. set seed 14
. bayesmh mpg weight, evaluator(normaljeffreys, parameters({var}))
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary
```

---

Posterior:

```
mpg ~ normaljeffreys(xb_mpg,{var})
```

---

Bayesian regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.1668
	Efficiency: min	=	.04114
	avg	=	.04811
	max	=	.05938

Log marginal-likelihood = -198.14302

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
mpg					
weight	-.6025616	.0540995	.002667	-.6038729	-.7115221 -.5005915
_cons	39.50491	1.677906	.080156	39.45537	36.2433 43.14319
var	12.26586	2.117858	.086915	12.05298	8.827655 17.10703

We can compare our results with bayesmh that uses a built-in normal likelihood and flat and Jeffreys priors. To match the results, we must use the same initial values, because bayesmh has a different initialization logic for built-in distributions.



```
. set seed 14
. bayesmh mpg weight, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)
  {var} ~ jeffreys
```

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1668
	Efficiency: min =	.04114
	avg =	.04811
	max =	.05938

Log marginal-likelihood = -198.14302

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.6025616	.0540995	.002667	-.6038729	-.7115221	-.5005915
	_cons	39.50491	1.677906	.080156	39.45537	36.2433	43.14319
	var	12.26586	2.117858	.086915	12.05298	8.827655	17.10703

If your Bayesian model uses prior distributions that are supported by bayesmh but the likelihood model is not supported, you can write only the likelihood evaluator and use built-in prior distributions.

For example, we extract the portion of the normaljeffreys program computing the overall log likelihood into a separate program and call it normalreg.

```
. program normalreg
1.     version 16.0
2.     args lnf xb var
3.                                     /* compute log likelihood */
.     tempname sd
4.     scalar 'sd' = sqrt('var')
5.     tempvar lnfj
6.     quietly generate double 'lnfj' = lnnormalden($MH_y,'xb','sd')
>     if $MH_touse
7.     quietly summarize 'lnfj', meanonly
8.     if r(N) < $MH_n {
9.         scalar 'lnf' = .
10.        exit
11.    }
12.    scalar 'lnf' = r(sum)
13. end
```

We can now specify this program in the llevaluator() option and use prior() options to specify built-in flat priors for the coefficients and the Jeffreys prior for the variance.

```

. set seed 14

. bayesmh mpg weight, llevaluator(normalreg, parameters({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> initial({mpg:weight} -0.6 {mpg:_cons} 39 {var} 11.83)
Burn-in ...
Simulation ...
Model summary

```

---

```

Likelihood:
  mpg ~ normalreg(xb_mpg,{var})
Priors:
  {mpg:weight _cons} ~ 1 (flat)
  {var} ~ jeffreys

```

---

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1668
	Efficiency: min =	.04114
	avg =	.04811
	max =	.05938

Log marginal-likelihood = -198.14302

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	weight	-.6025616	.0540995	.002667	-.6038729	-.7115221	-.5005915
	_cons	39.50491	1.677906	.080156	39.45537	36.2433	43.14319
	var	12.26586	2.117858	.086915	12.05298	8.827655	17.10703

We obtain the same results as earlier.

### Logistic regression model

Some models, such as logistic regression, do not have any additional parameters except regression coefficients. Here we show how to use a program evaluator for fitting a Bayesian logistic regression model.

We start by creating a program for computing the log likelihood.

```

. program logitll
1.     version 16.0
2.     args lnf xb
3.     tempvar lnfj
4.     quietly generate `lnfj' = ln(invlogit( `xb'))
>     if $MH_y == 1 & $MH_touse
5.     quietly replace `lnfj' = ln(invlogit(-`xb'))
>     if $MH_y == 0 & $MH_touse
6.     quietly summarize `lnfj', meanonly
7.     if r(N) < $MH_n {
8.         scalar `lnf' = .
9.         exit
10.    }
11.    scalar `lnf' = r(sum)
12. end

```

The structure of our log-likelihood evaluator is similar to the one described in *Simple linear regression model*, except we have no extra parameters.

We continue with `auto.dta` and regress `foreign` on `mpg`. For simplicity, we assume a flat prior for the coefficients and use `bayesmh, llevaluator()` to fit this model.

```
. use https://www.stata-press.com/data/r16/auto, clear
(1978 Automobile Data)

. set seed 14

. bayesmh foreign mpg, llevaluator(logitll) prior({foreign:}, flat)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  foreign ~ logitll(xb_foreign)
Prior:
  {foreign:mpg _cons} ~ 1 (flat)                                     (1)
```

---

```
(1) Parameters are elements of the linear form xb_foreign.

Bayesian regression                                MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling           Burn-in           =       2,500
                                                    MCMC sample size =     10,000
                                                    Number of obs    =        74
                                                    Acceptance rate  =      .2216
                                                    Efficiency: min  =     .09293
                                                    avg             =     .09989
Log marginal-likelihood = -41.626028                max             =     .1068
```

---

foreign	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	.16716	.0545771	.00167	.1644019	.0669937	.2790017
_cons	-4.560637	1.261675	.041387	-4.503921	-7.107851	-2.207665

The results from the program-evaluator version match the results from bayesmh with a built-in logistic model.

```
. set seed 14
. bayesmh foreign mpg, likelihood(logit) prior({foreign:}, flat)
> initial({foreign:} 0)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
foreign ~ logit(xb\_foreign)

Prior:  
{foreign:mpg \_cons} ~ 1 (flat) (1)

---

(1) Parameters are elements of the linear form xb\_foreign.

Bayesian logistic regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2216
	Efficiency: min =	.09293
	avg =	.09989
	max =	.1068

Log marginal-likelihood = -41.626029

---

foreign	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	.16716	.0545771	.00167	.1644019	.0669937	.2790017
_cons	-4.560636	1.261675	.041387	-4.503921	-7.10785	-2.207665

Because we assumed a flat prior with the density of 1, the log prior is 0, so the log-posterior evaluator for this model is the same as the log-likelihood evaluator.

```
. set seed 14
. bayesmh foreign mpg, evaluator(logitll)
Burn-in ...
Simulation ...
Model summary
```

---

Posterior:  
foreign ~ logitll(xb\_foreign)

---

Bayesian regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2216
	Efficiency: min =	.09293
	avg =	.09989
	max =	.1068

Log marginal-likelihood = -41.626028

---

foreign	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	.16716	.0545771	.00167	.1644019	.0669937	.2790017
_cons	-4.560637	1.261675	.041387	-4.503921	-7.107851	-2.207665

## Multivariate normal regression model

Here we demonstrate how to write a program evaluator for a multivariate response. We consider a bivariate normal regression, and we again start with a log-likelihood evaluator. In this example, we also use Mata to speed up our computations.

```
. program mvnregll
1.     version 16.0
2.     args lnf xb1 xb2
3.     tempvar diff1 diff2
4.     quietly generate double `diff1' = $MH_y1 - `xb1' if $MH_touse
5.     quietly generate double `diff2' = $MH_y2 - `xb2' if $MH_touse
6.     local d $MH_yn
7.     local n $MH_n
8.     mata: st_numscalar("`lnf'", mvnll_mata(`d',`n',"`diff1'", "`diff2'"))
9. end

.
. mata:
----- mata (type end to exit) -----
: real scalar mvnll_mata(real scalar d, n, string scalar sdiff1, sdiff2)
> {
>     real matrix Diff
>     real scalar trace, lnf
>     real matrix Sigma
>
>     Sigma = st_matrix(st_global("MH_m1"))
>     st_view(Diff=.,.(sdiff1,sdiff2),st_global("MH_touse"))
>
>     /* compute log likelihood */
>     trace = trace(cross(cross(Diff',invsym(Sigma))',Diff'))
>     lnf = -0.5*n*(d*ln(2*pi())+ln(det(Sigma)))-0.5*trace
>
>     return(lnf)
> }
: end
```

---

The `mvnregll` program has three arguments: a scalar to store the log-likelihood values and two temporary variables containing linear predictors corresponding to each of the two dependent variables. It creates deviations `'diff1'` and `'diff2'` and passes them, along with other parameters, to the Mata function `mvnll_mata()` to compute the bivariate normal log-likelihood value.

The extra parameter in this model is a covariance matrix of a bivariate response. In [Simple linear regression model](#), we specified an extra parameter, variance, which was a scalar, as an additional argument of the evaluator. This is not allowed with matrix parameters. They should be accessed via globals `$MH_m1`, `$MH_m2`, and so on for each matrix model parameters in the order they are specified in `option parameters()`. In our example, we have only one matrix and we access it via `$MH_m1`. `$MH_m1` contains the temporary name of a matrix containing the current value of the covariance matrix parameter.

To demonstrate, we again use `auto.dta`. We rescale the variables to be used in our example to stabilize the results.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. replace weight = weight/100
variable weight was int now float
(74 real changes made)
. replace length = length/10
variable length was int now float
(74 real changes made)
```

We fit a bivariate normal regression of `mpg` and `weight` on `length`. We specify the extra covariance parameter as a matrix model parameter `{Sigma,m}` in suboption `parameters()` of `llevvaluator()`. We specify flat priors for the coefficients and an inverse-Wishart prior for the covariance matrix.

```
. set seed 14
. bayesmh mpg weight = length, llevaluator(mvnregll, parameters({Sigma,m}))
> prior({mpg:} {weight:}, flat)
> prior({Sigma,m}, iwishart(2,12,I(2))) mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg weight ~ mvnregll(xb_mpg,xb_weight,{Sigma,m})
Priors:
  {mpg:length _cons} ~ 1 (flat)
  {weight:length _cons} ~ 1 (flat)
  {Sigma,m} ~ iwishart(2,12,I(2))
```

---

```
(1) Parameters are elements of the linear form xb_mpg.
(2) Parameters are elements of the linear form xb_weight.
Bayesian regression                                MCMC iterations =      3,500
Random-walk Metropolis-Hastings sampling           Burn-in          =      2,500
                                                    MCMC sample size =      1,000
                                                    Number of obs    =        74
                                                    Acceptance rate  =      .1728
                                                    Efficiency:      min =      .02882
                                                    avg             =      .05012
                                                    max             =      .1275
```

Log marginal-likelihood = -415.01504

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
	_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight	length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
	_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1		11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1		-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2		5.830413	.9051206	.121931	5.630011	4.383648	8.000739

To reduce computation time, we used a smaller MCMC sample size of 1,000 in our example. In your analysis, you should always verify whether a smaller MCMC sample size results in precise enough estimates before using it for final results.

We can check our results against bayesmh using the built-in multivariate normal regression after adjusting the initial values.

```
. set seed 14
. bayesmh mpg weight = length, likelihood(mvnormal({Sigma,m}))
> prior({mpg:} {weight:}, flat)
> prior({Sigma,m}, iwishart(2,12,I(2)))
> mcmcsize(1000) initial({mpg:} {weight:} 0)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg weight ~ mvnormal(2,xb\_mpg,xb\_weight,{Sigma,m})

Priors:

```
{mpg:length _cons} ~ 1 (flat) (1)
{weight:length _cons} ~ 1 (flat) (2)
{Sigma,m} ~ iwishart(2,12,I(2))
```

---

(1) Parameters are elements of the linear form xb\_mpg.  
(2) Parameters are elements of the linear form xb\_weight.

Bayesian multivariate normal regression	MCMC iterations =	3,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	1,000
	Number of obs =	74
	Acceptance rate =	.1728
	Efficiency: min =	.02882
	avg =	.05012
	max =	.1275

Log marginal-likelihood = -415.01504

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
	_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight	length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
	_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1		11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1		-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2		5.830413	.9051206	.121931	5.630011	4.383648	8.000739

We obtain the same results.

Similarly, we can define the log-posterior evaluator. We already have the log-likelihood evaluator, which we can reuse in our log-posterior evaluator. The only additional portion is to compute the log of the inverse-Wishart prior density for the covariance parameter.

---

```

. program mvniWishart
1.     version 16.0
2.     args lnp xb1 xb2
3.     tempvar diff1 diff2
4.     quietly generate double 'diff1' = $MH_y1 - 'xb1' if $MH_touse
5.     quietly generate double 'diff2' = $MH_y2 - 'xb2' if $MH_touse
6.     local d $MH_yn
7.     local n $MH_n
8.     mata:
>         st_numscalar("`lnp'", mvniWish_mata('d','n',"diff1',"diff2'))
9. end

.
. mata:
----- mata (type end to exit) -----
: real scalar mvniWish_mata(real scalar d, n, string scalar sdiff1, sdiff2)
> {
>     real scalar lnf, lnprior
>     real matrix Sigma
>
>     /* compute log likelihood */
>     lnf = mvnll_mata(d,n,sdiff1,sdiff2)
>     /* compute log of inverse-Wishart prior for Sigma */
>     Sigma = st_matrix(st_global("MH_m1"))
>     lnprior = lniwishartden(12,I(2),Sigma)
>     return(lnf + lnprior)
> }
: end

```

---



The results of the log-posterior evaluator match our earlier results.

```
. set seed 14
. bayesmh mpg weight = length, evaluator(mvniWishart, parameters({Sigma,m}))
> mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
```

---

Posterior:  
mpg weight ~ mvniWishart(xb\_mpg,xb\_weight,{Sigma,m})

---

Bayesian regression	MCMC iterations =	3,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	1,000
	Number of obs =	74
	Acceptance rate =	.1728
	Efficiency: min =	.02882
	avg =	.05012
	max =	.1275

Log marginal-likelihood = -415.01504

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight						
length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1	11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1	-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2	5.830413	.9051206	.121931	5.630011	4.383648	8.000739

Sometimes, it may be useful to be able to pass options to our evaluators. For example, we used the identity  $I(2)$  matrix as a scale matrix of the inverse Wishart distribution. Suppose that we want to check the sensitivity of our results to other choices of the scale matrix. We can pass the name of a matrix we want to use in an option. In our example, we use the `vmatrix()` option to pass the name of the scale matrix. We later specify this option within suboption `passthruopts()` of the `evaluator()` option. The options passed this way are stored in the `$MH_passthruopts` global macro.

```
. program mvniWishartV
1.     version 16.0
2.     args lnp xb1 xb2
3.     tempvar diff1 diff2
4.     quietly generate double `diff1' = $MH_y1 - `xb1' if $MH_touse
5.     quietly generate double `diff2' = $MH_y2 - `xb2' if $MH_touse
6.     local d $MH_yn
7.     local n $MH_n
8.     local 0 , $MH_passthruopts
9.     syntax, vmatrix(string)
10.    mata: st_numscalar("`lnp'",
>         mvniWishV_mata(`d','n',"`diff1'", "`diff2'", "`vmatrix'"))
11. end
```

---

```

. mata:
_____ mata (type end to exit) _____
: real scalar mvniWishV_mata(real scalar d, n, string scalar sdiff1, sdiff2,
> vmat)
> {
>     real scalar lnf, lnprior
>     real matrix Sigma
>
>     /* compute log likelihood */
>     lnf = mvnll_mata(d,n,sdiff1,sdiff2)
>     /* compute log of inverse-Wishart prior for Sigma */
>     Sigma = st_matrix(st_global("MH_m1"))
>     lnprior = lniwishartden(12,st_matrix(vmat),Sigma)
>     return(lnf + lnprior)
> }
: end

```

---

We now define the scale matrix  $V$  (as the identity matrix to match our previous results) and specify `vmatrix(V)` in suboption `passthruopts()` of `evaluator()`.

```

. set seed 14
. matrix V = I(2)
. bayesmh mpg weight = length,
> evaluator(mvniWishartV, parameters({Sigma,m}) passthruopts(vmatrix(V)))
> mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary

```

---

```

Posterior:
  mpg weight ~ mvniWishartV(xb_mpg,xb_weight,{Sigma,m})

```

---

Bayesian regression	MCMC iterations =	3,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	1,000
	Number of obs =	74
	Acceptance rate =	.1728
	Efficiency: min =	.02882
	avg =	.05012
	max =	.1275
Log marginal-likelihood = -415.01504		

---

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	length	-2.040162	.2009062	.037423	-2.045437	-2.369287	-1.676332
	_cons	59.6706	3.816341	.705609	59.63619	52.54652	65.84583
weight	length	3.31773	.1461644	.026319	3.316183	3.008416	3.598753
	_cons	-32.19877	2.79005	.484962	-32.4154	-37.72904	-26.09976
Sigma_1_1		11.49666	1.682975	.149035	11.3523	8.691888	14.92026
Sigma_2_1		-2.33596	1.046729	.153957	-2.238129	-4.414118	-.6414916
Sigma_2_2		5.830413	.9051206	.121931	5.630011	4.383648	8.000739

---

The results are the same as before.

## Cox proportional hazards regression

Some evaluators may require additional variables, apart from the dependent and independent variables, for computation. For example, in a Cox proportional hazards model such variable is a failure or censoring indicator. The `coxphll` program below computes partial log likelihood for the Cox proportional hazards regression. The failure indicator will be passed to the evaluator as an extra variable in suboption `extravars()` of option `llevauator()` or option `evaluator()` and can be accessed from the global macro `$MH_extravars`.

```
. program coxphll
1.         version 16.0
2.         args lnf xb
3.         tempvar negt
4.         quietly generate double 'negt' = -$MH_y1
5.         local d "$MH_extravars"
6.         sort $MH_touse 'negt' 'd'
7.         tempvar B A sumd last L
8.         local byby "by $MH_touse 'negt' 'd'"
9.         quietly {
10.            gen double 'B' = sum(exp('xb')) if $MH_touse
11.            'byby': gen double 'A' = cond(_n==N, sum('xb'), .)
>                if 'd'==1 & $MH_touse
12.            'byby': gen 'sumd' = cond(_n==N, sum('d'), .) if $MH_touse
13.            'byby': gen byte 'last' = (_n==N & 'd' == 1) if $MH_touse
14.            gen double 'L' = 'A' - 'sumd'*ln('B') if 'last' & $MH_touse
15.            quietly count if $MH_touse & 'last'
16.            local n = r(N)
17.            summarize 'L' if 'last' & $MH_touse, meanonly
18.        }
19.        if r(N) < 'n' {
20.            scalar 'lnf' = .
21.            exit
22.        }
23.        scalar 'lnf' = r(sum)
24. end
```

We demonstrate the command using the survival-time cancer dataset. The survival-time variable is `studytime` and the failure indicator is `died`. The regressor of interest in this model is `age`. We use a fairly noninformative normal prior with a zero mean and a variance of 100 for the regression coefficient of `age`. (The constant in the Cox proportional hazards model is not likelihood-identifiable, so we omit it from this model with a noninformative prior.)

```
. use https://www.stata-press.com/data/r16/cancer, clear
(Patient Survival in Drug Trial)

. gsort -studytime died

. set seed 14

. bayesmh studytime age, llevaluator(coxphll, extravars(died))
> prior({studytime:}, normal(0,100)) noconstant mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  studytime ~ coxphll(xb_studytime)

Prior:
  {studytime:age} ~ normal(0,100)
```

---

(1)

(1) Parameter is an element of the linear form `xb_studytime`.

Bayesian regression

Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = -103.04797

MCMC iterations = 3,500

Burn-in = 2,500

MCMC sample size = 1,000

Number of obs = 48

Acceptance rate = .4066

Efficiency = .3568

studytime	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
age	.076705	.0330669	.001751	.077936	.0099328	.1454275

We specified the failure indicator died in suboption `extravars()` of `llevvaluator()`. We again used a smaller value for the MCMC sample size only to reduce computation time.

For the log-posterior evaluator, we add the log of the normal prior of the `age` coefficient to the log-likelihood value to obtain the final log-posterior value. We did not need to specify the loop in the log-prior computation in this example, but we did this to be general, in case more than one regressor is included in the model.

```
. program coxphnormal
1.     version 16.0
2.     args lnp xb
.     /* compute log likelihood */
.     tempname lnf
3.     scalar 'lnf' = .
4.     quietly coxphll 'lnf' 'xb'
.     /* compute log priors of regression coefficients */
.     tempname lnprior
5.     scalar 'lnprior' = 0
6.     forvalues i = 1/$MH_bn {
7.         scalar 'lnprior' = 'lnprior' + lnnormalden($MH_b[1,'i'], 10)
8.     }
9.     /* compute log posterior */
.     scalar 'lnp' = 'lnf' + 'lnprior'
10. end
```

As expected, we obtain the same results as previously.

```
. set seed 14
. bayesmh studytime age, evaluator(coxphnormal, extravars(died))
> noconstant mcmcsize(1000)
Burn-in ...
Simulation ...
Model summary
```

Posterior:  
studytime ~ coxphnormal(xb\_studytime)

Bayesian regression

Random-walk Metropolis-Hastings sampling

Log marginal-likelihood = -103.04797

MCMC iterations = 3,500

Burn-in = 2,500

MCMC sample size = 1,000

Number of obs = 48

Acceptance rate = .4066

Efficiency = .3568

studytime	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
age	.076705	.0330669	.001751	.077936	.0099328	.1454275

Global macros

Global macros	Description
\$MH_n	number of observations
\$MH_yn	number of dependent variables
\$MH_touse	variable containing 1 for the observations to be used; 0 otherwise
\$MH_w	variable containing weight associated with the observations
\$MH_extravars	<i>varlist</i> specified in <code>extravars()</code>
\$MH_passthruopts	options specified in <code>passthruopts()</code>
<i>One outcome</i>	
\$MH_y1	name of the dependent variable
\$MH_x1	name of the first independent variable
\$MH_x2	name of the second independent variable
...	
\$MH_xn	number of independent variables
\$MH_xb	name of a temporary variable containing the linear combination
<i>Multiple outcomes</i>	
\$MH_y1	name of the first dependent variable
\$MH_y2	name of the second dependent variable
...	
\$MH_y1x1	name of the first independent variable modeling y1
\$MH_y1x2	name of the second independent variable modeling y1
...	
\$MH_y1xn	number of independent variables modeling y1
\$MH_y1xb	name of a temporary variable containing the linear combination modeling y1
\$MH_y2x1	name of the first independent variable modeling y2
\$MH_y2x2	name of the second independent variable modeling y2
...	
\$MH_y2xn	number of independent variables modeling y2
\$MH_y2xb	name of a temporary variable containing the linear combination modeling y2
...	
<i>Scalar and matrix parameters</i>	
\$MH_b	name of a temporary vector of coefficients; stripes are properly named after the name of the coefficients
\$MH_bn	number of coefficients
\$MH_p	name of a temporary vector of additional scalar model parameters, if any; stripes are properly named
\$MH_pn	number of additional scalar model parameters
\$MH_m1	name of a temporary matrix of the first matrix parameter, if any
\$MH_m2	name of a temporary matrix of the second matrix parameter, if any
...	
\$MH_mn	number of matrix model parameters

## Stored results

In addition to the results stored by `bayesmh`, `bayesmh`, `evaluator()` and `bayesmh, lleveluator()` store the following in `e()`:

### Macros

<code>e(evaluator)</code>	program evaluator (one equation)
<code>e(evaluator#)</code>	program evaluator for the #th equation
<code>e(evalparams)</code>	evaluator parameters (one equation)
<code>e(evalparams#)</code>	evaluator parameters for the #th equation
<code>e(extravars)</code>	extra variables (one equation)
<code>e(extravars#)</code>	extra variables for the #th equation
<code>e(passthruopts)</code>	pass-through options (one equation)
<code>e(passthruopts#)</code>	pass-through options for the #th equation

## Reference

Marchenko, Y. V. 2015. Bayesian modeling: Beyond Stata's built-in models. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2015/05/26/bayesian-modeling-beyond-statas-built-in-models/>.

## Also see

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **Bayesian postestimation** — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

Postestimation commands

The following Bayesian postestimation commands are available after the `bayesmh` command ([BAYES] `bayesmh`) and the `bayes` prefix ([BAYES] `bayes`):

Command	Description
<code>bayesgraph</code>	graphical summaries and convergence diagnostics
<code>bayesstats grubin</code>	Gelman–Rubin convergence diagnostics
<code>bayesstats ess</code>	effective sample sizes and related statistics
<code>bayesstats ppvalues</code>	Bayesian predictive $p$ -values (available only after <code>bayesmh</code> )
<code>bayesstats summary</code>	Bayesian summary statistics for model parameters and their functions
<code>bayesstats ic</code>	Bayesian information criteria and Bayes factors
<code>bayestest model</code>	hypothesis testing using model posterior probabilities
<code>bayestest interval</code>	interval hypothesis testing
<code>bayespredict</code>	Bayesian predictions (available only after <code>bayesmh</code> )
<code>*estimates</code>	cataloging estimation results

\* `estimates` table and `estimates stats` are not appropriate with `bayesmh` and `bayes`: estimation results.

Remarks and examples

Remarks are presented under the following headings:

- Different ways of specifying model parameters*
- Specifying functions of model parameters*
- Storing estimation results after Bayesian estimation*
- Different ways of specifying predictions and their functions*

After estimation, you can use `bayesgraph` to check convergence of MCMC visually. If you simulated multiple chains, you can use `bayesstats grubin` to compute Gelman–Rubin convergence diagnostics. Once convergence is established, you can use `bayespredict` and `bayesstats ppvalues` to perform model checking after `bayesmh`. Once you are satisfied with the model, you can use `bayesstats summary` to obtain Bayesian summaries such as posterior means and standard deviations of model parameters and functions of model parameters; `bayesstats ess` to compute effective sample sizes and related statistics for model parameters and functions of model parameters; and `bayesstats ic` to compute Bayesian information criteria and Bayes factors for model parameters and their functions. You can use `bayestest model` to test hypotheses by comparing posterior probabilities of models. You can also use `bayestest interval` to test interval hypotheses about parameters and functions of parameters. After `bayesmh`, you can also use `bayespredict` to predict future outcome values.

For an overview example of postestimation commands, see *Overview example* in [BAYES] **Bayesian commands**.

## Different ways of specifying model parameters

Many Bayesian postestimation commands such as `bayesstats summary` and `bayesgraph` allow you to specify model parameters for which you want to see the results. To see results for all parameters, simply type a postestimation command without arguments after estimation using `bayesmh` or the `bayes` prefix, for example,

```
. bayesstats summary
```

or you could type

```
. bayesstats summary _all
```

To manually list all model parameters, type

```
. bayesstats summary {param1} {param2} ...
```

or

```
. bayesstats summary {param1 param2} ...
```

The only exception is the `bayesgraph` command when there is more than one model parameter. In that case, `bayesgraph` requires that you either specify `_all` to request all model parameters or specify the model parameters of interest.

You can refer to a single model parameter in the same way you define parameters in, say, the `bayesmh` command. For example, for a parameter with name `param` and no equation name, you can use `{param}`. For a parameter with name `param` and equation name `eqname`, you can use its full name `{eqname:name}`, where the equation name and the parameter name are separated with a colon. With postestimation commands, you can also omit the equation name when referring to the parameter with an equation name.

In the presence of more than one model parameter, you have several ways for referring to multiple parameters at once. If parameters have the same equation name, you can refer to all the parameters with that equation name as follows.

Suppose that you have three parameters with the same equation name `eqname`. Then the specification

```
. bayesstats summary {eqname:param1} {eqname:param2} {eqname:param3}
```

is the same as the specification

```
. bayesstats summary {eqname:}
```

or the specification

```
. bayesstats summary {eqname:param1 param2 param3}
```

The above specification is useful if we want to refer to a subset of parameters with the same equation name. For example, in the above, if we wanted to use only `param1` and `param2`, we could type

```
. bayesstats summary {eqname:param1 param2}
```

There is also a convenient way to refer to the parameters with the same name but different equation names. For example, typing

```
. bayesstats summary {eqname1:param} {eqname2:param}
```

is the same as simply typing

```
. bayesstats summary {param}
```



You can mix and match all the specifications above in one call to a postestimation command. You can also specify expressions of model parameters; see [Specifying functions of model parameters](#) for details.

Note that if `param` refers to a matrix model parameter, then the results will be provided for all elements of the matrix. For example, if `param` is the name of a  $2 \times 2$  matrix, then typing

```
. bayesstats summary {param}
```

implies the following:

```
. bayesstats summary {param_1_1} {param_1_2} {param_2_1} {param_2_2}
```

For multilevel models, there are various ways, *reref*, in which you can refer to individual random-effects parameters. Suppose that your model has random intercepts at the `id` level, which are labeled as `{U0[id]}` or `{U0}` for short. To refer to all random intercepts, you can use `{U0}`, `{U0[.]}`, and `{U0[id]}`. To refer to specific random intercepts, you can use `{U0[#]}`, where `#` refers to the `#`th element of the random-effects vector, or use `{U0[#.id]}`, where `#` refers to the `#`th level of the `id` variable. You can also refer to a subset *numlist* of random intercepts by using `{U0[numlist]}` or `{U0[(numlist).id]}`. For nested random effects, for example, `{UU0[id1>id2]}`, you can refer to all random effects as `{UU0}` or `{UU0[. ,.]}` and to subsets of random effects as `{UU0[numlist,numlist]}` or `{UU0[(numlist).id1,(numlist).id2]}`.

## Specifying functions of model parameters

You can use Bayesian postestimation commands to obtain results for functions or expressions of model parameters. Each expression must be specified in parentheses. An expression can be any Stata expression, but it may not include matrix model parameters. However, you may include individual elements of matrix model parameters. You may provide labels for your expressions.

For example, we can obtain results for the exponentiated parameter `{param}` as follows:

```
. bayesstats summary (exp({param}))
```

Note that we specified the expression in parentheses.

We can include a label, say, `myexp`, in the above by typing

```
. bayesstats summary (myexp: exp({param}))
```

We can specify multiple expressions by typing

```
. bayesstats summary (myexp: exp({param}) (sd: sqrt({var})))
```

If `param` is a matrix, we can specify expressions, including its elements, but not the matrix itself in the following:

```
. bayesstats summary (exp({param_1_1})) (exp({param_1_2})) ...
```

## Storing estimation results after Bayesian estimation

The `bayesmh` command and the `bayes` prefix store various `e()` results such as scalars, macros, and matrices in memory like any other estimation command. Unlike other estimation commands, these commands also save the resulting simulation dataset containing MCMC samples of parameters to disk. Many Bayesian postestimation commands such as `bayesstats summary` and `bayesstats ess` require access to this file. If you do not specify the `saving()` option with `bayesmh` or the `bayes` prefix, the commands save simulation results in a temporary Stata dataset. This file is being

replaced with the new simulation results each time `bayesmh` or the `bayes` prefix is run. To save your simulation results, you must specify the `saving()` option with `bayesmh` or the `bayes` prefix, in which case your simulation results are saved to the specified file in the specified location and will not be overridden by the next call to these commands.

You can specify the `saving()` option during estimation by typing

```
. bayesmh ..., likelihood() prior() ... saving()
```

or

```
. bayes, saving(): ...
```

or on replay by typing

```
. bayesmh, saving()
```

or

```
. bayes, saving()
```

As you can with other estimation commands, you can use `estimates store` to store Bayesian estimation results in memory and `estimates save` to save them to disk, but you must first use the `saving()` option with `bayesmh` or the `bayes` prefix to save simulation data in a permanent dataset. For example, type

```
. bayesmh ..., likelihood() prior() ... saving(bmh_simdata)
. estimates store model1
```

or, after `bayesmh` estimation, type

```
. bayesmh, saving(bmh_simdata)
. estimates store model1
```

Once you create a permanent dataset, it is your responsibility to erase it after it is no longer needed. `estimates drop` and `estimates clear` will drop estimation results only from memory; they will not erase the simulation files you saved.

```
. estimates drop model1
. erase bmh_simdata.dta
```

See [\[R\] estimates](#) for more information about commands managing estimation results. `estimates table` and `estimates stats` are not appropriate after `bayesmh` and the `bayes` prefix.

## Different ways of specifying predictions and their functions

After `bayesmh`, you can use the `bayespredict` command to simulate outcome variables, residuals, and other test quantities; see [\[BAYES\] bayespredict](#). Bayesian postestimation commands `bayesgraph`, `bayesstats summary`, `bayesstats ppvalues`, `bayesstats ess`, and `bayestest interval` can then be used to obtain graphs, posterior summaries, and so on for these prediction quantities.

In this section, we describe various specifications of prediction results with Bayesian postestimation commands mentioned above. We use `bayesstats summary` in our examples, but the same specifications may be used with other postestimation commands, except that `bayestest interval` allows only specifications containing individual observations.

Suppose that we use the `bayesmh` command to fit a model with two outcome variables.

```
. bayesmh y1 y2 = x1 x2, ... saving(mcmcfile)
```

We then use `bayespredict` to simulate samples for these two outcome variables and save them in a prediction dataset, `predfile.dta`.

```
. bayespredict {_ysim1} {_ysim2}, saving(predfile)
```

To access prediction results, all postestimation commands must specify the prediction dataset in the `using` specification. In fact, this is all postestimation commands need to produce results for the prediction quantities. (Technically, the auxiliary estimation file generated by `bayespredict`, for example, `predfile.ster`, must also exist.) That is, they do not rely on the estimation results or the simulation data from `bayesmh`.

When the prediction dataset contains simulated outcomes, in addition to accessing these outcomes (for instance, `{_ysim1}` and `{_ysim2}` in our example), postestimation commands may also access the residuals (`{_resid1}` and `{_resid2}`), expected values (`{_mu1}` and `{_mu2}`), and Stata expressions of simulated outcomes, residuals, and expected values. You can also call Mata functions within command specifications to compute functions of simulated outcomes, residuals, and expected values.

Let's calculate posterior summaries for all observations of the first outcome and for all residuals of the second outcome.

```
. bayesstats summary {_ysim1} {_resid2} using predfile
```

You can refer to a subset of predicted observations, say, from 1 to 10 for the observations and from 1 to 5 for the residuals.

```
. bayesstats summary {_ysim1[1/10]} {_resid2[1/5]} using predfile
```

You can compute expressions of individual simulated outcome observations and their residuals.

```
. bayesstats summary (exp({_ysim1[1]})) ({_resid2[1]}^2) using predfile
```

You can test whether the residual for the first observation of the second outcome variable is greater than zero by using `bayestest interval` to calculate the corresponding posterior probability.

```
. bayestest interval {_resid2[1]} using predfile, lower(0)
```

As we mentioned earlier, you can use Mata functions of predicted outcomes and residuals. These functions operate across observations. For example, to summarize the mean of the first simulated outcome and the variance of the second simulated outcome, type

```
. bayesstats summary (@mean({_ysim1})) (@variance({_ysim2})) using predfile
```

Instead of using the default labels for the computed quantities, you can specify your own. Below, we use `mean` and `var` to label the corresponding predictions.

```
. bayesstats summary (mean:@mean({_ysim1})) (var:@variance({_ysim2})) using predfile
```

You cannot specify Mata functions with `bayestest interval`, and, unlike `bayespredict`, you cannot specify Stata programs within the postestimation commands.

If you need to access individual values of the predicted quantity computed using a Mata function or specify an expression of this quantity, you need to compute and save this quantity with `bayespredict`.

Suppose that you wish to compute the sum of the two outcome variances. You simulate these variances by using `bayespredict` first.

```
. bayespredict (prvar1:@variance({_ysim1})) (prvar2:@variance({_ysim2})), ///
    saving(predfile)
```

In the above, we labeled the computed variances as `prvar1` and `prvar2`.

Then, you can call `bayesstats summary` to compute the sum of the predicted quantities.

```
. bayesstats summary ({prvar1} + {prvar2}) using predfile
```

Or you can obtain summaries of each predicted quantity.

```
. bayesstats summary {prvar1} {prvar2} using predfile
```

You can combine various specifications in one call to the postestimation command. For example, let's save the following prediction quantities with `bayespredict`.

```
. bayespredict {_ysim1} {_ysim2} (mean1:@mean({_ysim1})) ///
    (var2:@variance({_ysim2})), saving(predfile)
```

You can specify multiple prediction quantities in one call to `bayesstats summary` or other postestimation commands.

```
. bayesstats summary ({_ysim1}) ({_resid[1/5]}) ({mean1}) ///
    ({var2}) (mean2:@mean({_ysim2})) using predfile
```

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the bayes prefix

[BAYES] [bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm

[BAYES] [bayesmh evaluators](#) — User-defined evaluators with bayesmh

[BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis

[BAYES] [Intro](#) — Introduction to Bayesian analysis

[BAYES] [Glossary](#)

[U] [20 Estimation and postestimation commands](#)

[Description](#)  
[Options](#)  
[Also see](#)

[Quick start](#)  
[Remarks and examples](#)

[Menu](#)  
[Methods and formulas](#)

[Syntax](#)  
[References](#)

## Description

`bayesgraph` provides graphical summaries and convergence diagnostics for simulated posterior distributions (MCMC samples) of model parameters and functions of model parameters obtained after Bayesian estimation. Graphical summaries include trace plots, autocorrelation plots, and various distributional plots.

## Quick start

Trace plot, histogram, autocorrelation plot, and density plot for parameter `{p}`

```
bayesgraph diagnostics {p}
```

Add plots for parameter `{y:x1}`

```
bayesgraph diagnostics {p} {y:x1}
```

As above, but for all model parameters

```
bayesgraph diagnostics _all
```

As above, but for a function of model parameters `{y:x1}` and `{p}`

```
bayesgraph diagnostics ({y:x1}/{p})
```

Specify a blue trace plot line for all plots

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, traceopts(lcolor(blue))
```

Specify a blue trace plot line only for the second trace plot

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, trace2opts(lcolor(blue))
```

Trace plots for all parameters in a single graph

```
bayesgraph trace _all, byparm
```

Cumulative sum plot for parameter `{p}`

```
bayesgraph cusum {p}
```

Scatterplot matrix for parameters `{p}` and `{y:x1}`

```
bayesgraph matrix {p} {y:x1}
```

Autocorrelation plots for elements 1,1 and 2,1 of matrix parameter `{S}`

```
bayesgraph ac {S_1_1} {S_2_1}
```

Diagnostic plots for all parameters in the model and pause at least 3 seconds before displaying the next graph

```
bayesgraph diagnostics _all, sleep(3)
```

As above, but pause until the user presses any key

```
bayesgraph diagnostics _all, wait
```

As above, but close the current Graph window when the next graph is displayed

```
bayesgraph diagnostics _all, close
```

Histogram of the first 10 observations of the first simulated outcome plotted on one graph

```
bayespredict {_ysim}, saving(predres)
bayesgraph histogram {_ysim[1/10]} using predres, byparm
```

Density plot of the mean across observations of the simulated outcome labeled as mymean

```
bayesgraph kdensity (mymean: @mean({_ysim})) using predres
```

## Menu

Statistics > Bayesian analysis > Graphical summaries

## Syntax

Syntax is presented under the following headings:

*Graphical summaries for model parameters*

*Graphical summaries for predictions*

### Graphical summaries for model parameters

*Graphical summaries and convergence diagnostics for a single parameter*

```
bayesgraph graph scalar_param [ , singleopts ]
```

*Graphical summaries and convergence diagnostics for multiple parameters*

```
bayesgraph graph spec [spec ...] [ , multiopts ]
```

```
bayesgraph matrix spec spec [spec ...] [ , singleopts ]
```

*Graphical summaries and convergence diagnostics for all parameters*

```
bayesgraph graph _all [ , multiopts showeffects( (reref) ) ]
```

*scalar\_param* is a [scalar model parameter](#) specified as {param} or {eqname:param} or an expression *exprspec* of scalar model parameters. Matrix model parameters are not allowed, but you may refer to their individual elements.

*exprspec* is an optionally labeled expression of model parameters specified in parentheses:

```
( [exprlabel: ] expr )
```

*exprlabel* is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See [Specifying functions of model parameters](#) in [\[BAYES\] Bayesian postestimation](#) for examples.

*spec* is either *scalar\_param* or *exprspec*.

## Graphical summaries for predictions

Graphical summaries for an individual prediction

```
bayesgraph graph predspecsc using predfile [, singleopts]
```

Graphical summaries for multiple predictions

```
bayesgraph graph predspec [predspec ...] using predfile [, multiopts]
```

```
bayesgraph matrix predspec predspec [predspec ...] using predfile [, singleopts]
```

*predfile* is the name of the dataset created by **bayespredict** that contains prediction results.

*predspecsc* may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, `{label}`.

*predspec* is one of *yspec*, (*yexprspec*), or (*funcspec*). See *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation**.

*yspec* is `{ysimspec | residspec | muspec | label}`.

*ysimspec* is `{_ysim#}` or `{_ysim#[numlist]}`, where `{_ysim#}` refers to all observations of the *#*th simulated outcome and `{_ysim#[numlist]}` refers to the selected observations, *numlist*, of the *#*th simulated outcome. `{_ysim}` is a synonym for `{_ysim1}`.

*residspec* is `{_resid#}` or `{_resid#[numlist]}`, where `{_resid#}` refers to all residuals of the *#*th simulated outcome and `{_resid#[numlist]}` refers to the selected residuals, *numlist*, of the *#*th simulated outcome. `{_resid}` is a synonym for `{_resid1}`.

*muspec* is `{_mu#}` or `{_mu#[numlist]}`, where `{_mu#}` refers to all expected values of the *#*th outcome and `{_mu#[numlist]}` refers to the selected expected values, *numlist*, of the *#*th outcome. `{_mu}` is a synonym for `{_mu1}`.

*label* is the name of the function simulated using **bayespredict**.

With large datasets, specifications `{_ysim#}`, `{_resid#}`, and `{_mu#}` may use a lot of time and memory and should be avoided. See *Generating and saving simulated outcomes* in [BAYES] **bayespredict**.

*yexprspec* is `[exprlabel:]yexpr`, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, `{label}`.

*funcspec* is `[label:]@func(arg1 [, arg2])`, where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of `{_ysim#[#]}`, `{_resid#[#]}`, or `{_mu#[#]}`. *arg2* is primarily for use with user-defined Mata functions; see *Defining test statistics using Mata functions* in [BAYES] **bayespredict**.

<i>graph</i>	Description
<u>diagnostics</u>	multiple diagnostics in compact form
<u>trace</u>	trace plots
<u>ac</u>	autocorrelation plots
<u>histogram</u>	histograms
<u>kdensity</u>	density plots
<u>cusum</u>	cumulative sum plots
<u>matrix</u>	scatterplot matrix
baysgraph matrix requires at least two parameters. diagnostics, trace, ac, and cusum are not relevant for predictions.	

<i>singleopts</i>	Description
Chains	
<i>chainopts</i>	options controlling multiple chains
Options	
<code>skip(#)</code>	skip every # observations from the MCMC sample; default is <code>skip(0)</code>
<code>name(name, ...)</code>	specify name of graph
<code>saving(filename, ...)</code>	save graph in file
<i>graphopts</i>	graph-specific options

<i>multiopts</i>	Description
Chains	
<i>chainopts</i>	options controlling multiple chains
Options	
<code>byparm[ (<i>grbyparmopts</i>) ]</code>	specify the display of plots on one graph; default is separate graph for each plot; not allowed with graphs <code>diagnostics</code> and <code>matrix</code> or with options <code>combine()</code> and <code>bychain()</code>
<code>combine[ (<i>grcombineopts</i>) ]</code>	specify the display of plots on one graph; recommended when the number of parameters is large; not allowed with graphs <code>diagnostics</code> and <code>matrix</code> or with options <code>byparm()</code> and <code>bychain()</code>
<code>sleep(#)</code>	pause for # seconds between multiple graphs; default is <code>sleep(0)</code>
<code>wait</code>	pause until the —more— condition is cleared
<code>[no]close</code>	(do not) close Graph windows when the next graph is displayed with multiple graphs; default is <code>noclose</code>
<code>skip(#)</code>	skip every # observations from the MCMC sample; default is <code>skip(0)</code>
<code>name(namespec, ...)</code>	specify names of graphs
<code>saving(filespec, ...)</code>	save graphs in files
<code>graphopts(<i>graphopts</i>)</code>	control the look of all graphs; not allowed with <code>byparm()</code>
<code>graph#opts(<i>graphopts</i>)</code>	control the look of #th graph; not allowed with <code>byparm()</code>
<i>graphopts</i>	equivalent to <code>graphopts(<i>graphopts</i>)</code> ; only one may be specified



<i>chainopts</i>	Description
<code>chains(<u>_all</u>   <i>numlist</i>)</code>	specify which chains to plot; default is to plot the first 10 chains
<code>sepchains</code>	draw a separate graph for each chain; default is to overlay chains
<code>chainslegend</code>	show legend keys corresponding to chain numbers; not allowed with <code>graphs diagnostics</code> and <code>matrix</code> or with options <code>combine()</code> and <code>byparm()</code>
<code>bychain[ (<i>grbychainopts</i>) ]</code>	plot each chain as a subgraph on one graph; default is all chains overlayed on one graph; not allowed with <code>graphs diagnostics</code> and <code>matrix</code> or with options <code>combine()</code> and <code>byparm()</code>
<code>chainopts(<i>graphopts</i>)</code>	control the look of all chains
<code>chain#opts(<i>graphopts</i>)</code>	control the look of #th chain

Options *chainopts* are relevant only when option `nchains()` is used with [bayesmh](#) or the [bayes](#) prefix.

<i>graphopts</i>	Description
<i>diagnosticsopts</i>	options for <code>bayesgraph diagnostics</code>
<i>tslineopts</i>	options for <code>bayesgraph trace</code> and <code>bayesgraph csum</code>
<i>acopts</i>	options for <code>bayesgraph ac</code>
<i>histopts</i>	options for <code>bayesgraph histogram</code>
<i>kdensityopts</i>	options for <code>bayesgraph kdensity</code>
<i>grmatrixopts</i>	options for <code>bayesgraph matrix</code>

<i>diagnosticsopts</i>	Description
<code>traceopts(<i>tslineopts</i>)</code>	affect rendition of all trace plots
<code>trace#opts(<i>tslineopts</i>)</code>	affect rendition of #th trace plot
<code>acopts(<i>acopts</i>)</code>	affect rendition of all autocorrelation plots
<code>ac#opts(<i>acopts</i>)</code>	affect rendition of #th autocorrelation plot
<code>histopts(<i>histopts</i>)</code>	affect rendition of all histogram plots
<code>hist#opts(<i>histopts</i>)</code>	affect rendition of #th histogram plot
<code>kdensopts(<i>kdensityopts</i>)</code>	affect rendition of all density plots
<code>kdens#opts(<i>kdensityopts</i>)</code>	affect rendition of #th density plot
<i>grcombineopts</i>	any option documented in <a href="#">[G-2] graph combine</a>

<i>acopts</i>	Description
<code>ci</code>	plot autocorrelations with confidence intervals; not allowed with <code>byparm()</code>
<i>acopts</i>	any options other than <code>generate()</code> documented for the <code>ac</code> command in <a href="#">[TS] corrgram</a>

<i>kdensityopts</i>	Description
<i>kdensopts</i>	options for the overall kernel density plot
<code>show(<i>showspect</i>)</code>	show first-half density ( <code>first</code> ), second-half density ( <code>second</code> ), both, or none; default varies
<code>kdensfirst(<i>kdens1opts</i>)</code>	affect rendition of the first-half density plot
<code>kdenssecond(<i>kdens2opts</i>)</code>	affect rendition of the second-half density plot

## Options

### Chains

`chains(_all | numlist)` specifies which chains from the MCMC sample to plot. The default is to plot the first 10 chains. You can use `chains(_all)` to plot all chains.

`sepchains` specifies that a separate graph be drawn for each chain. This option is implied for `bayesgraph matrix` and may not be combined with `bychain()`.

`chainslegend` specifies that the graph be plotted with a legend showing keys corresponding to chain numbers. This option is not allowed with graphs `diagnostics` and `matrix` or with options `combine()` and `byparm()`.

`bychain[ (grbychainopts) ]` specifies that each chain be plotted as a subgraph on one graph. By default, all chains are displayed overlayed on one graph. This option is not allowed with graphs `diagnostics` and `matrix` or with options `combine()`, `byparm()`, and `sepchains`.

*grbychainopts* is any of the suboptions of `by()` documented in [G-3] *by\_option*.

`chainopts(graphopts)` and `chain#opts(graphopts)` control the look of chains. `chainopts()` controls the look of all chains but may be overridden for specific chains by using the `chain#opts()` option.

Chain-specific options are ignored if option `nchains()` is not specified with `bayesmh` or the `bayes` prefix.

### Options

`byparm[ (grbyparmopts) ]` specifies the display of all plots of parameters as subgraphs on one graph. By default, a separate graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with options `combine()` and `bychain()`. When many parameters or expressions are specified, this option may fail because of memory constraints. In that case, you may use option `combine()` instead.

*grbyparmopts* is any of the suboptions of `by()` documented in [G-3] *by\_option*.

`byparm()` allows *y* scales to differ for all graph types and forces *x* scales to be the same only for `bayesgraph trace` and `bayesgraph cusum`. Use `noyrescale` within `byparm()` to specify a common *y* axis, and use `xrescale` or `noxrescale` to change the default behavior for the *x* axis.

`byparm()` with `bayesgraph trace` and `bayesgraph cusum` defaults to displaying multiple plots in one column to accommodate the *x* axis with many iterations. Use `norowcoldefault` within `byparm()` to switch back to the default behavior of options `rows()` and `cols()` of the [G-3] *by\_option*.

`combine([grcombineopts])` specifies the display of all plots of parameters as subgraphs on one graph and is an alternative to `byparm()` with a large number of parameters. By default, a separate graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with option `byparm()`. It can be used in cases where a large number of parameters or expressions are specified and the `byparm()` option would cause an error because of memory constraints.

*grcombineopts* is any of the options documented in [G-2] [graph combine](#).

`sleep(#)` specifies pausing for # seconds before producing the next graph. This option is allowed only when multiple parameters are specified. This option may not be combined with `wait`, `combine()`, or `byparm()`.

`wait` causes `bayesgraph` to display `—more—` and pause until any key is pressed before producing the next graph. This option is allowed when multiple parameters are specified. This option may not be combined with `sleep()`, `combine()`, or `byparm()`. `wait` temporarily ignores the global setting that is specified using `set more off`.

`[no]close` specifies that, for multiple graphs, the Graph window be closed when the next graph is displayed. The default is `noclose` or to not close any Graph windows.

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`name(namespec[, replace])` specifies the name of the graph or multiple graphs. See [G-3] [name\\_option](#) for a single graph. If multiple graphs are produced, then the argument of `name()` is either a list of names or a *stub*, in which case graphs are named *stub1*, *stub2*, and so on. With multiple graphs, if `name()` is not specified and neither `sleep()` nor `wait` is specified, `name(Graph_#)`, `replace` is assumed, and thus the produced graphs may be replaced by subsequent `bayesgraph` commands.

The `replace` suboption causes existing graphs with the specified name or names to be replaced.

`saving(filespec[, replace])` specifies the filename or filenames to use to save the graph or multiple graphs to disk. See [G-3] [saving\\_option](#) for a single graph. If multiple graphs are produced, then the argument of `saving()` is either a list of filenames or a *stub*, in which case graphs are saved with filenames *stub1*, *stub2*, and so on.

The `replace` suboption specifies that the file (or files) may be replaced if it already exists.

`showeffects` and `showeffects(reref)` are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time.

`graphopts(graphopts)` and `graph#opts(graphopts)` affect the rendition of graphs. `graphopts()` affects the rendition of all graphs but may be overridden for specific graphs by using the `graph#opts()` option. The options specified within `graph#opts()` are specific for each type of graph.

The two specifications

`bayesgraph ..., graphopts(graphopts)`

and

`bayesgraph ..., graphopts`

are equivalent, but you may specify one or the other.

These options are not allowed with `byparm()` and when only one parameter is specified.

*graphopts* specifies options specific to each graph type.

*diagnosticopts* specifies options for use with `bayesgraph` diagnostics. See the corresponding table in the syntax diagram for a list of options.

*tslineopts* specifies options for use with `bayesgraph` trace and `bayesgraph` cusum. See the options of [TS] [tsline](#) except `by()`.

*acopts* specifies options for use with `bayesgraph` `ac`.

`ci` requests that the graph of autocorrelations with confidence intervals be plotted. By default, confidence intervals are not plotted. This option is not allowed with `byparm()`.

*acopts* specifies any options except `generate()` of the `ac` command in [TS] [corrgram](#).

*histopts* specifies options for use with `bayesgraph` histogram. See options of [R] [histogram](#) except `by()`.

*kdensityopts* specifies options for use with `bayesgraph` `kdensity`.

*kdensopts* specifies options for the overall kernel density plot. See the options documented in [R] [kdensity](#) except `generate()` and `at()`.

`show(showspec)` specifies which kernel density curves to plot. *showspec* is one of `first`, `second`, `both`, or `none`. If `show(first)` is specified, only the first-half density curve, obtained from the first half of an MCMC sample, is plotted. If `show(second)` is specified, only the second-half density curve, obtained from the second half of an MCMC sample, is plotted. `show(both)`, the default with `graph` [diagnostics](#), overlays both the first-half density curve and the second-half density curve with the overall kernel density curve. `show(none)`, the default with `graph` `kdensity`, shows only the overall kernel density curve.

`kdensfirst(kdens1opts)` specifies options of [G-2] [graph twoway kdensity](#) except `by()` to affect rendition of the first-half kernel density plot.

`kdenssecond(kdens2opts)` specifies options of [G-2] [graph twoway kdensity](#) except `by()` to affect rendition of the second-half kernel density plot.

*gmatrixopts* specifies options for use with `bayesgraph` `matrix`. See the options of [G-2] [graph matrix](#) except `by()`.

## Remarks and examples

Remarks are presented under the following headings:

*Using bayesgraph*

*Examples*

*Trace plots*

*Autocorrelation plots*

*Histogram plots*

*Kernel density plots*

*Cumulative sum plots*

*Bivariate scatterplots*

*Diagnostic plots*

*Functions of model parameters*

## Using bayesgraph

**bayesgraph** requires specifying at least one parameter with all graph types except matrix, which requires at least two parameters. To request graphs for all parameters, use `_all`.

When multiple graphs are produced, they are automatically stored in memory with names `Graph_#` and will all appear on the screen. After you are done reviewing the graphs, you can type

```
. graph close Graph_*
```

to close these graphs or type

```
. graph drop Graph_*
```

to close the graphs and drop them from memory.

If you would like to see only one graph at a time, you can specify option `close` to close the Graph window when the next graph is displayed. You can also use option `sleep()` or option `wait` to pause between the subsequent graphs. The `sleep(#)` option causes each graph to pause for # seconds. The `wait` option causes **bayesgraph** to wait until a key is pressed before producing the next graph.

You can combine separate graphs into one by specifying one of `byparm()` or `combine()`. These options are not allowed with diagnostics or matrix graphs. The `byparm()` option produces more compact graphs, but it may not be feasible with many parameters or expressions and large sizes of MCMC samples.

With multiple graphs, you can control the look of each individual graph with `graph#opts()`. Options common to all graphs may be specified in `graphopts()` or passed directly to the command as with single graphs.

With multiple chains, **bayesgraph** plots only the first 10 chains by default. If you have more than 10 chains, although only four chains are commonly used in practice, you can use the `chains(_all)` option to plot all the chains. You can also use the `chains()` option to handpick the chains you want to be plotted. For example, `chains(1/3 5)` will plot chains 1, 2, 3, and 5. If desired, you can see which plot corresponds to which chain by using the `chainslegend` option.

By default, the chains will be plotted overlaid on one graph. You can specify the `sepchains` option to plot each chain on a separate graph, in which case the graphs will be automatically stored in memory with names `Graph_#` and will all appear on the screen. Or, you can use the `bychain` option to plot each chain separately but one graph.

To control the look of an individual chain, you can use the `chain#opts()` options. For example, to change the line color to red for chain 2, you would specify the `chain2opts(lcolor(red))` option. To control the look of all chains, you can use the `chainopts()` option.

You can use `bayesgraph` to plot predicted quantities when you supply the prediction dataset generated by `bayespredict` in the `using` specification. Also see [Different ways of specifying predictions and their functions](#) in [BAYES] [Bayesian postestimation](#).

## Examples

We demonstrate the `bayesgraph` command using an example of Bayesian normal linear regression applied to `auto.dta`. We model the `mpg` variable using a normal distribution with unknown mean and variance. Our Bayesian model thus has two parameters, `{mpg:_cons}` and `{var}`, for which we need to specify prior distributions. We consider fairly noninformative prior distributions for these parameters:  $N(0, 1000)$  for the constant and inverse gamma with shape and scale of 0.1 for the variance. Because the specified prior distributions are independent and [semiconjugate](#) relative to the normal data distribution, we can use Gibbs sampling for both parameters instead of the default MH sampling. To illustrate, we will use Gibbs sampling for the variance and MH sampling (default) for the mean.

We use `bayesmh` to fit our model.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(0,1000))
> prior({var}, igamma(0.1,0.1)) block({var}, gibbs) rseed(14)
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
    mpg ~ normal({mpg:_cons},{var})

Priors:
    {mpg:_cons} ~ normal(0,1000)
    {var} ~ igamma(0.1,0.1)
```

Bayesian normal regression	MCMC iterations =	12,500
Metropolis-Hastings and Gibbs sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.7133
	Efficiency: min =	.2331
	avg =	.6166
	max =	1
Log marginal-likelihood =	-242.1155	

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
	_cons	21.29231	.6648867	.013771	21.29419	19.94367 22.56746
	var	34.2805	5.844213	.058442	33.6464	24.65882 47.5822

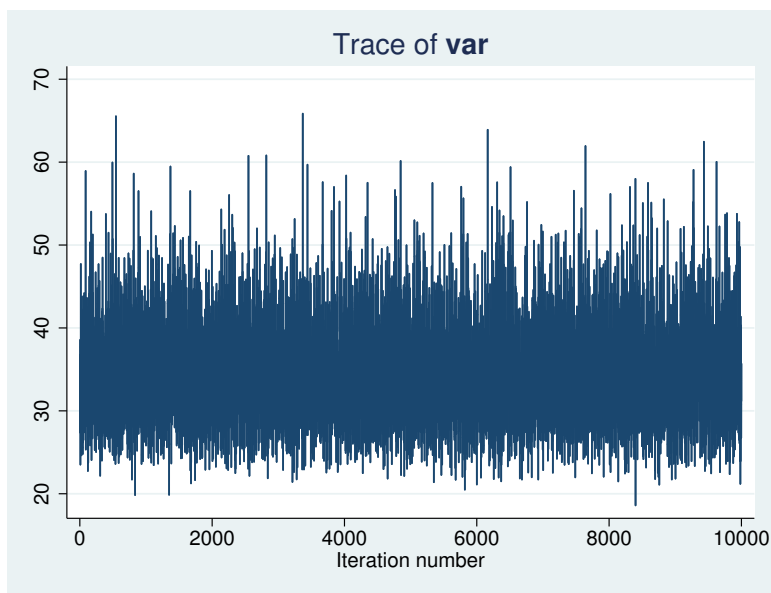
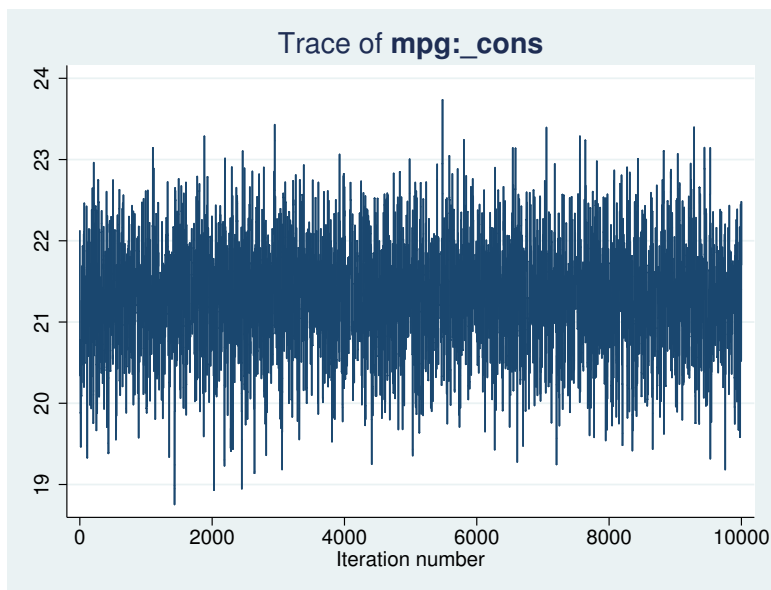
The MCMC simulation has a fairly high efficiency for the MH algorithm of 23% for the mean and an efficiency of 1 for the variance because of the Gibbs sampling. The output suggests no convergence problems. However, it is important to verify this and to also inspect various other graphical summaries of the parameters. This example demonstrates graphical summaries for a well-mixing MCMC chain that has converged and that generates samples from the posterior distribution of the model. For examples of poor-mixing MCMC chains, see [Convergence diagnostics in MCMC](#) in [BAYES] [Intro](#).

## Trace plots

We start with trace plots, which plot the values of the simulated parameters against the iteration number and connect consecutive values with a line. For a well-mixing parameter, the range of the parameter is traversed rapidly by the MCMC chain, which makes the drawn lines look almost vertical and dense. Sparseness and trends in the trace plot of a parameter suggest convergence problems.

Let's use `bayesgraph trace` to obtain trace plots for `{mpg:_cons}` and `{var}`. We specify `_all` to request both plots at once.

```
. bayesgraph trace _all
```

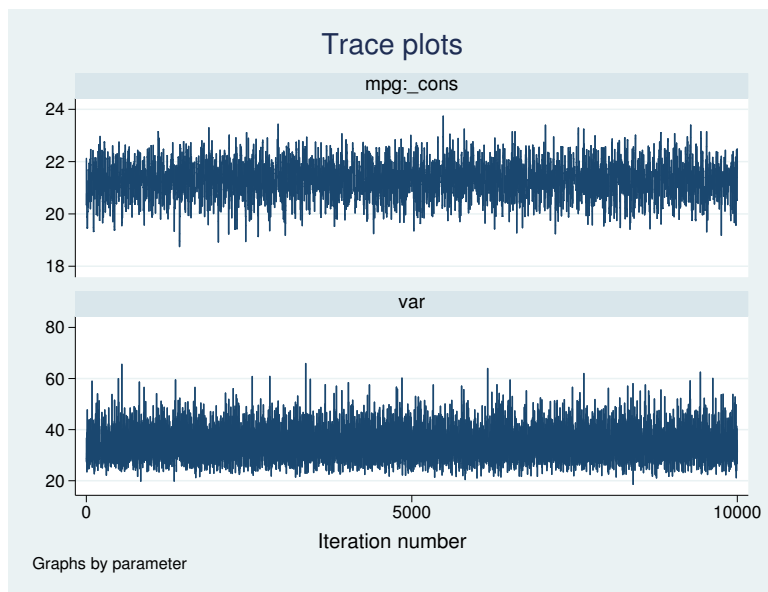


The mean parameter mixes very well and the variance parameter mixes perfectly.



Alternatively, we can use the `byparm()` option to plot results on one graph.

```
. bayesgraph trace _all, byparm
```



`bayesgraph trace` (as well as `bayesgraph cusum`) with option `byparm()` displays multiple plots in one column to accommodate an  $x$  axis with many iterations. You can specify `byparm(norowcoldefault)` to switch to the default behavior of options `rows()` and `cols()` documented in [G-3] *by\_option*.

Also see *Convergence diagnostics using multiple chains* in [BAYES] `bayesmh` for an example of trace plots with multiple chains.

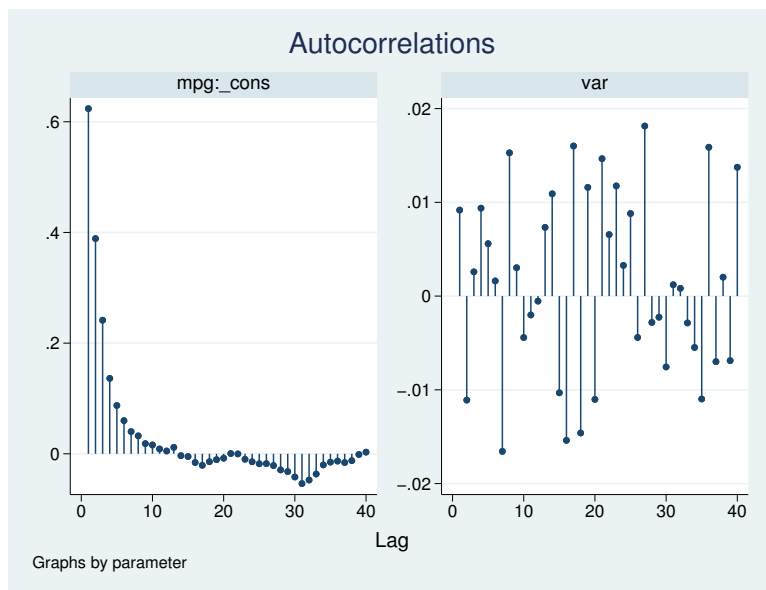
## Autocorrelation plots

The second graphical summary we demonstrate is an autocorrelation plot. This plot shows the degree of autocorrelation in an MCMC sample for a range of lags, starting from lag 0. At lag 0, the plotted value corresponds to the sample variance of MCMC.

Autocorrelation is usually present in any MCMC sample. Typically, autocorrelation starts from some positive value for lag 0 and decreases toward 0 as the lag index increases. For a well-mixing MCMC chain, autocorrelation dies off fairly rapidly.

For example, autocorrelation for {mpg: \_cons} becomes negligible after about lag 8 and is basically nonexistent for {var}.

```
. baysgraph ac _all, byparm
```



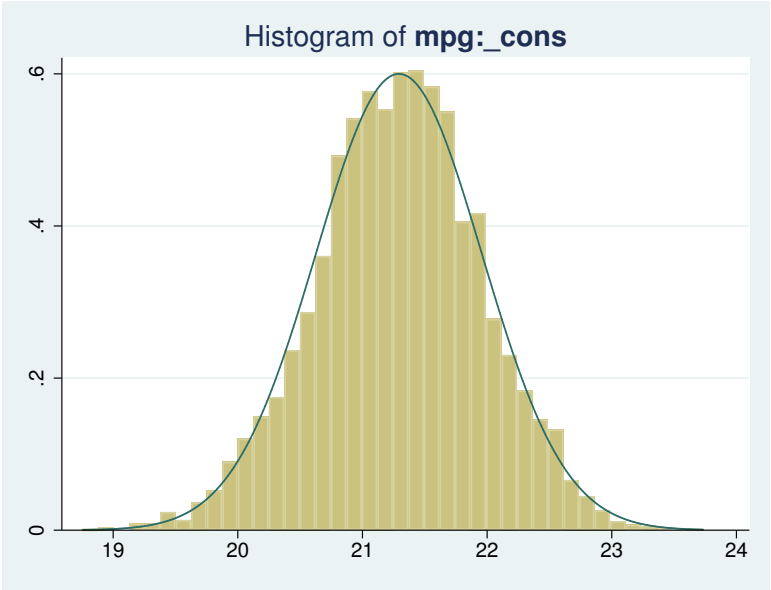
Autocorrelation lags are approximated by correlation times of parameters as reported by the `bayesstats ess` command; see [\[BAYES\] bayesstats ess](#) for details. Autocorrelation lags are also used to determine the batch size for the batch-means estimator of the MCMC standard errors; see [\[BAYES\] bayesstats summary](#).

## Histogram plots

Graphical posterior summaries such as histograms and kernel density estimates provide useful additions to the various numerical statistics (see [\[BAYES\] bayesstats summary](#)) for summarizing MCMC output. It is always a good practice to inspect the histogram and kernel density estimates of the marginal posterior distributions of parameters to ensure that these empirical distributions behave as expected. These plots can be used to compare the empirical posterior and the specified prior distributions to visualize the impact of the data.

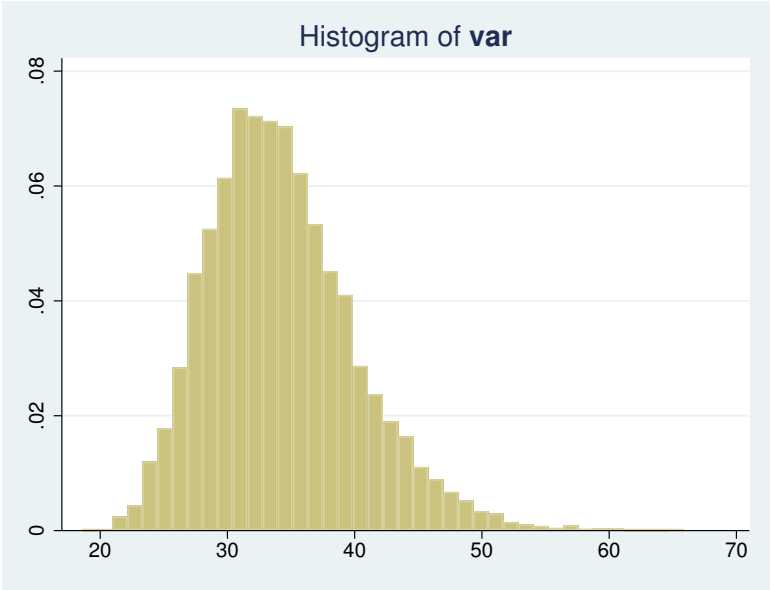
A histogram depicts the general shape of the marginal posterior distribution of a model parameter. Let's look at histograms of our parameters.

```
. bayesgraph histogram {mpg:_cons}, normal
```



The distribution of {mpg:\_cons} is in good agreement with the normal distribution. This is not surprising, because the specified conjugate normal prior implies that the marginal posterior for {mpg:\_cons} is a normal distribution. The unimodal histogram is also another confirmation that we have obtained a good simulation of the marginal posterior distribution of {mpg:\_cons}.

```
. bayesgraph histogram {var}
```



The histogram for `{var}` is also unimodal but is slightly skewed to the right. This is also in agreement with the specified prior because the marginal posterior for the variance is inverse gamma for the specified model.

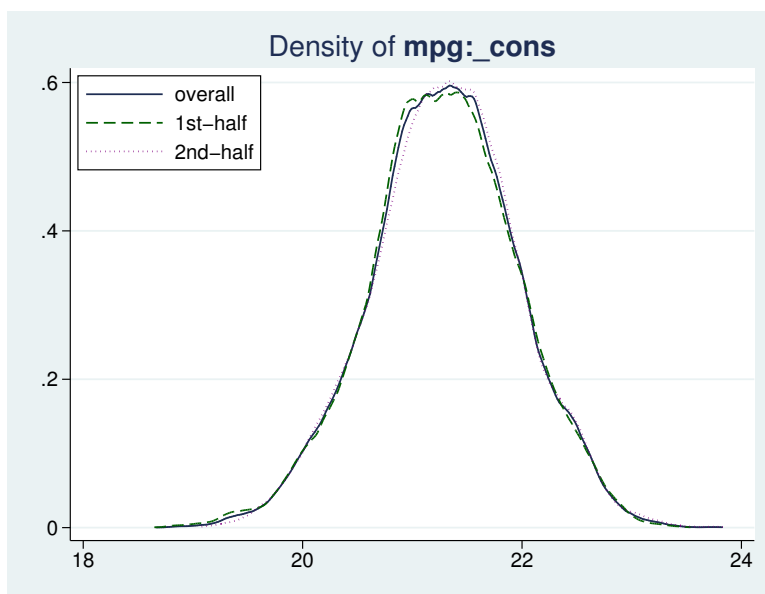
For examples of histograms for prediction quantities, see [example 4](#) and [example 7](#) in [\[BAYES\] bayespredict](#) and [example 1](#) and [example 3](#) in [\[BAYES\] bayesstats ppvalues](#).

## Kernel density plots

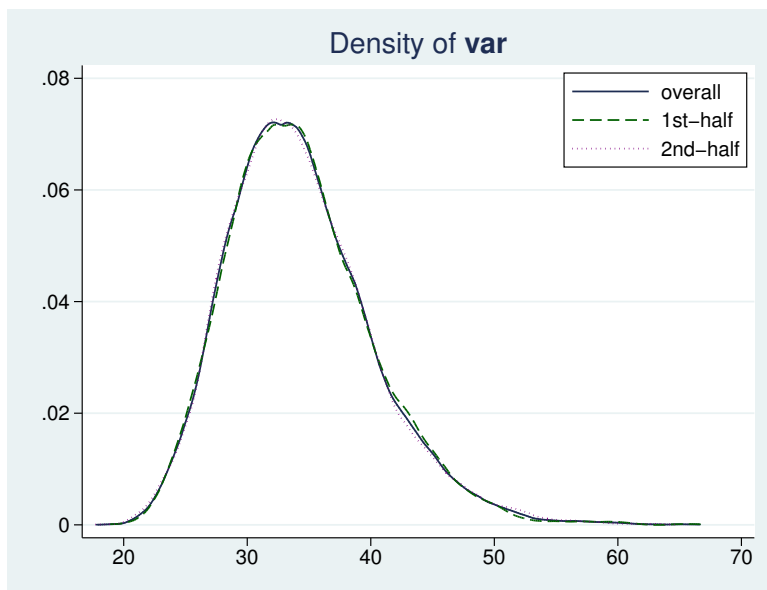
Kernel density plots provide alternative visualizations of the simulated marginal posterior distributions. They may be viewed as smoothed histograms. By default, the `bayesgraph kdensity` command shows an overall density of the entire MCMC sample. To explore convergence, the command provides the `show(both)` option, which additionally plots two density curves: the first-half density obtained using the first half of the MCMC sample and the second-half density obtained using the second half of the MCMC sample. If the chain has converged and mixes well, we expect the three density curves to be close to each other. Large discrepancies between the first-half curve and the second-half curve suggest convergence problems.

Let's look at the three kernel density curves for our two parameters.

```
. bayesgraph kdensity {mpg:_cons}, show(both)
```



```
. bayesgraph kdensity {var}, show(both)
```



Kernel density plots for `{mpg:_cons}` and `{var}` are similar in shape to the histograms' plots from the previous section. All three density curves are close to each other for both parameters.

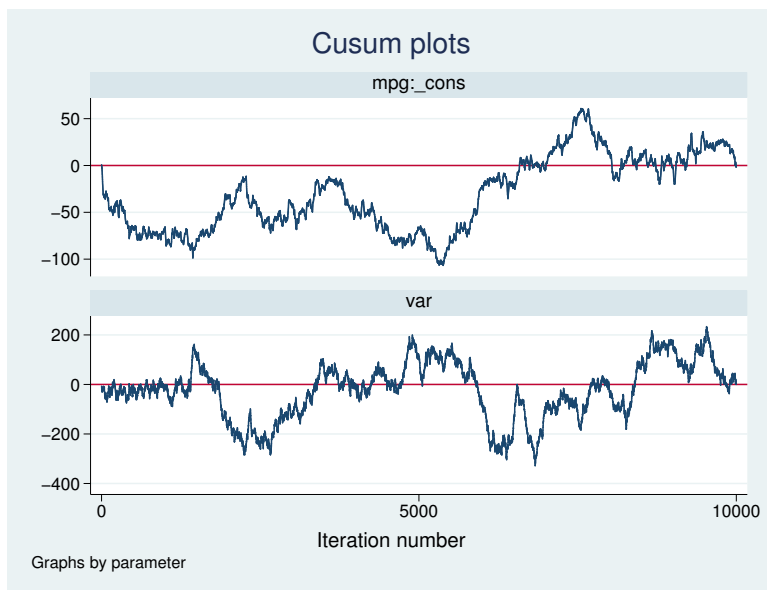
Also see [Convergence diagnostics using multiple chains](#) in [BAYES] [bayesmh](#) for an example of kernel density plots with multiple chains.

## Cumulative sum plots

Cumulative sum (cusum) plots are useful graphical summaries for detecting persistent trends in MCMC chains. All cusum plots start and end at 0 and may or may not cross the  $x$  axis. There is great variability in the looks of cusum plots, which make them difficult to interpret sometimes. Typically, if the cusum line never crosses the  $x$  axis, this may indicate a problem. See, for example, [Convergence diagnostics of MCMC](#) in [BAYES] [Intro](#) for a cusum plot demonstrating convergence problems.

By inspecting a cusum plot, we may detect an early drift in the simulated sample because of an insufficient burn-in period. In cases of pronounced persistent trends, the cusum curve may stay either in the positive or in the negative  $y$  plane. For a well-mixing parameter, the cusum curve typically crosses the  $x$  axis several times. This is the case for the cusum plots of `{mpg:_cons}` and `{var}`.

```
. bayesgraph cusum _all, byparm
```



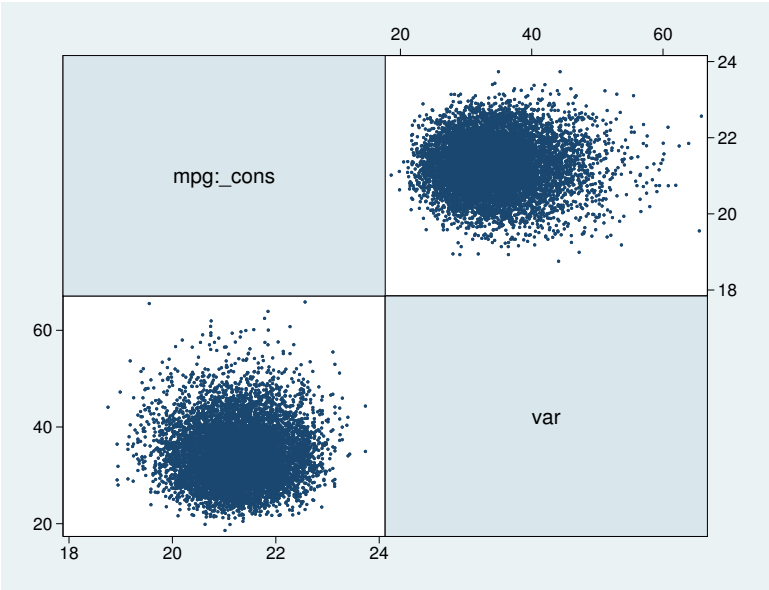
## Bivariate scatterplots

The `bayesgraph matrix` command draws bivariate scatterplots of model parameters based on MCMC samples. A bivariate scatterplot represents a joint sample posterior distribution for pairs of parameters. It may reveal correlation between parameters and characterize a general shape of a multivariate posterior distribution. For example, bivariate scatterplots are useful for detecting multimodal posterior distributions.

Typically, scatterplots depict clouds of points. Sparseness and irregularities in the scatterplots can be strong indications of nonconvergence of an MCMC. For a well-mixing chain, the scatterplots have an ellipsoidal form with an increasing concentration around the posterior mode.

This scatterplot of {mpg:\_cons} and {var} is an example of a well-behaved scatterplot.

```
. bayesgraph matrix {mpg:_cons} {var}
```

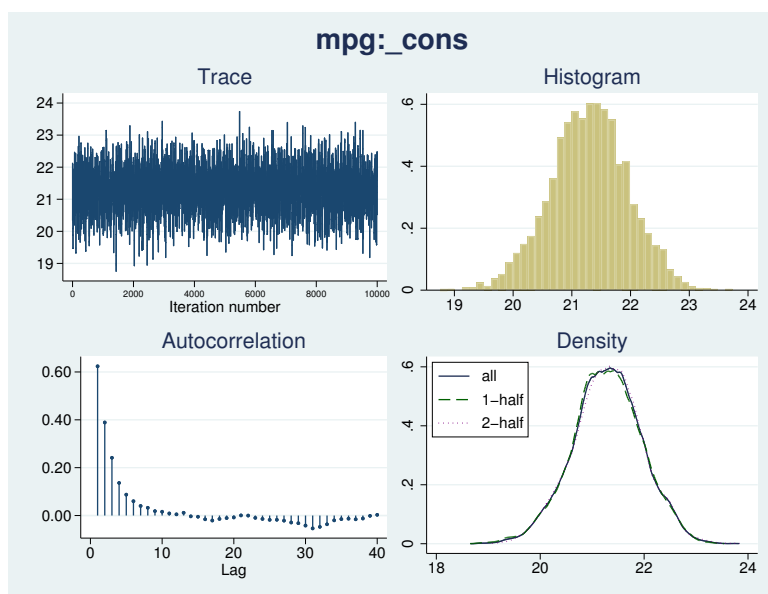


**Diagnostic plots**

Finally, we demonstrate the `bayesgraph diagnostics` command, which combines the trace, histogram, autocorrelation, and kernel density plots compactly on one graph. We already discussed the individual plots in the previous sections. Diagnostic plots are convenient for inspecting the overall behavior of a particular model parameter. We recommend that diagnostic plots for all parameters be inspected routinely as a part of the convergence-checking process.

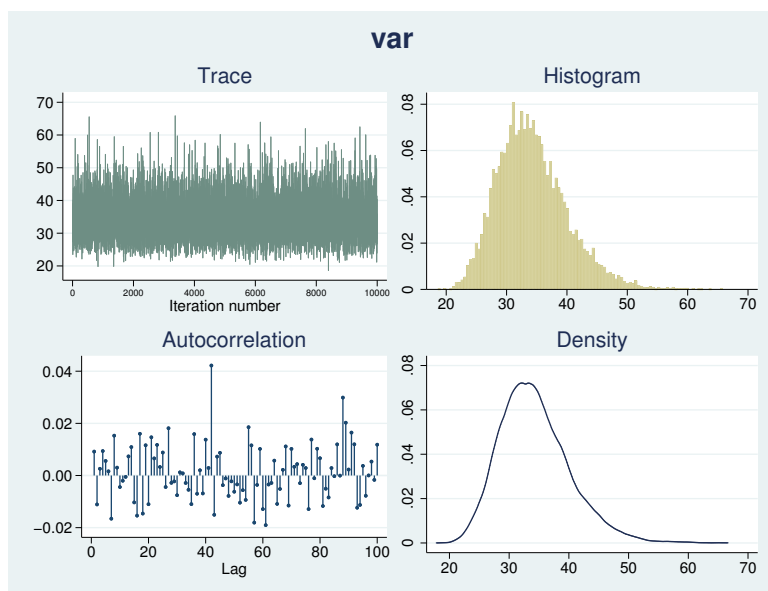
Let's obtain the diagnostic plot for {mpg:\_cons}.

```
. bayesgraph diagnostics {mpg:_cons}
```



In the diagnostics plot for {var}, let's also demonstrate the use of several options of the depicted plots.

```
. bayesgraph diagnostics {var}, traceopts(lwidth(0.2) lcolor(teal))
> acopts(lag(100)) histopts(bin(100)) kdensopts(show(none))
```





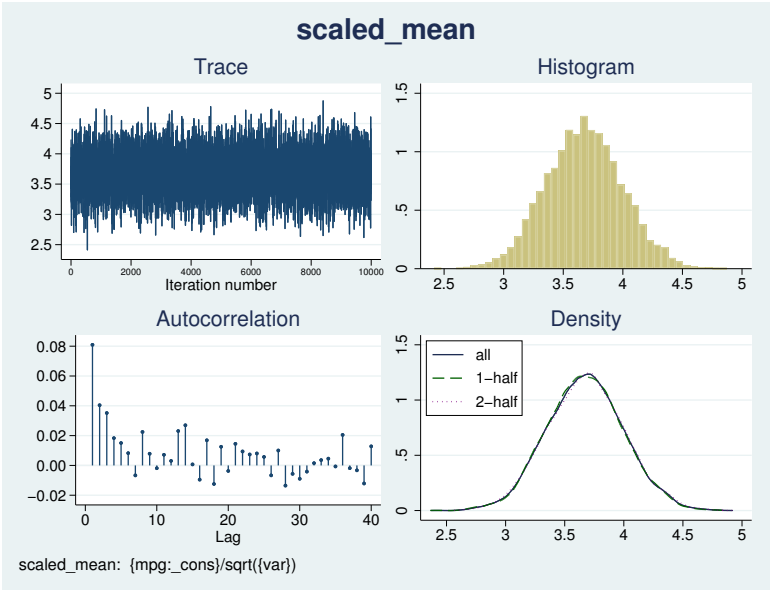
In the above, we changed the width and color of the trace line, the maximum lag for calculating the autocorrelation, the number of bins for the histogram, and requested that the two subsample kernel densities not be shown on the kernel density plot.

Also see *Convergence diagnostics using multiple chains* in [BAYES] **bayesmh** for an example of diagnostics plots with multiple chains.

**Functions of model parameters**

All **bayesgraph** subcommands can provide graphical summaries of functions of model parameters. Below we apply **bayesgraph** diagnostics to the expression `{mpg:_cons}/sqrt({var})`, which we label as `scaled_mean`.

```
. bayesgraph diagnostics (scaled_mean: {mpg:_cons}/sqrt({var}))
```



If you detect convergence problems in a function of parameters, you must inspect every parameter used in the expression individually. In fact, we recommend that you inspect all model parameters before you proceed with any postestimation analysis.

**Methods and formulas**

Let  $\theta$  be a scalar model parameter and  $\{\theta_t\}_{t=1}^T$  be an MCMC sample of size  $T$  drawn from the marginal posterior distribution of  $\theta$ .

The trace plot of  $\theta$  plots  $\theta_t$  against  $t$  with connecting lines for  $t = 1, \dots, T$ .

The autocorrelation plot of  $\theta$  shows the autocorrelation in the  $\{\theta_t\}_{t=1}^T$  sample for lags from 0 to the `lag(#)` option of the `ac` command.

The histogram and kernel density plots of  $\theta$  are drawn using the **histogram** and **kdensity** commands.

Yu and Mykland (1998) proposed a graphical procedure for assessing the convergence of individual parameters based on cumulative sums, also known as a cusum plot. The cusum plot for  $\theta$  plots  $S_t$  against  $t$  for  $t = 1, \dots, T$  and connects the successive points.  $S_t$  is the cumulative sum at time  $t$ :

$$S_t = \sum_{k=1}^t (\theta_k - \hat{\theta}), \quad \hat{\theta} = \frac{1}{T} \sum_{k=1}^T \theta_k$$

and  $S_0 = 0$ .

The scatterplot of two model parameters  $\theta^1$  and  $\theta^2$  plots points  $(\theta_t^1, \theta_t^2)$  for  $t = 1, \dots, T$ .

With multiple chains, the plots are produced separately for each chain.

## References

- Huber, C. 2016. Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/11/15/introduction-to-bayesian-statistics-part-2-mcmc-and-the-metropolis-hastings-algorithm/>.
- Yu, B., and P. Mykland. 1998. Looking at Markov samplers through cusum path plots: A simple diagnostic idea. *Statistics and Computing* 8: 275–286.

## Also see

- [BAYES] **bayes** — Bayesian regression models using the bayes prefix
- [BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm
- [BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] **bayesstats ess** — Effective sample sizes and related statistics
- [BAYES] **bayesstats summary** — Bayesian summary statistics
- [G-2] **graph matrix** — Matrix graphs
- [G-2] **graph twoway kdensity** — Kernel density plots
- [R] **histogram** — Histograms for continuous and categorical variables
- [R] **kdensity** — Univariate kernel density estimation
- [TS] **corrgram** — Tabulate and graph autocorrelations
- [TS] **tsline** — Time-series line plots

# Title

**bayesstats** — Bayesian statistics after Bayesian estimation

[Description](#)[Also see](#)

## Description

The following subcommands are available with `bayesstats` after `bayesmh` and the `bayes` prefix:

Command	Description
<code>bayesstats ess</code>	effective sample sizes and related statistics
<code>bayesstats summary</code>	Bayesian summary statistics for model parameters and their functions
<code>bayesstats ic</code>	Bayesian information criteria and Bayes factors
<code>bayesstats grubin</code>	Gelman–Rubin convergence diagnostics
<code>bayesstats ppvalues</code>	Bayesian predictive $p$ -values (available only after <code>bayesmh</code> )

## Also see

- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [BAYES] [Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

# Title

bayesstats ess — Effective sample sizes and related statistics

[Description](#)  
[Options](#)  
[Also see](#)

[Quick start](#)  
[Remarks and examples](#)

[Menu](#)  
[Stored results](#)

[Syntax](#)  
[Methods and formulas](#)

## Description

`bayesstats ess` calculates effective sample sizes (ESS), correlation times, and efficiencies for model parameters and functions of model parameters using current Bayesian estimation results.

## Quick start

Effective sample sizes for all model parameters after a Bayesian regression model

```
bayesstats ess
```

As above, but only for model parameters `{y:x1}` and `{var}`

```
bayesstats ess {y:x1} {var}
```

As above, but skip every 5 observations from the full MCMC sample

```
bayesstats ess {y:x1} {var}, skip(5)
```

Effective sample sizes for functions of scalar model parameters

```
bayesstats ess ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

As above, and include `{y:x1}` and `{var}`

```
bayesstats ess {y:x1} {var} ({y:x1}-{y:_cons}) (sd:sqrt({var}))
```

## Menu

Statistics > Bayesian analysis > Effective sample sizes

## Syntax

Syntax is presented under the following headings:

*Statistics for model parameters*

*Statistics for predictions*

### Statistics for model parameters

*Statistics for all model parameters*

```
bayesstats ess [ , options showreffects [ (reref) ] ]
```

```
bayesstats ess _all [ , options showreffects [ (reref) ] ]
```

*Statistics for selected model parameters*

```
bayesstats ess paramspec [ , options ]
```

*Statistics for expressions of model parameters*

```
bayesstats ess exprspec [ , options ]
```

*Full syntax*

```
bayesstats ess spec [ spec ... ] [ , options ]
```

*paramspec* can be one of the following:

*{eqname:param}* refers to a parameter *param* with equation name *eqname*;

*{eqname:}* refers to all model parameters with equation name *eqname*;

*{eqname:paramlist}* refers to parameters with names in *paramlist* and with equation name *eqname*;  
or

*{param}* refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See *Different ways of specifying model parameters* in [BAYES] **Bayesian postestimation** for examples.

*exprspec* is an optionally labeled expression of model parameters specified in parentheses:

```
( [ exprlabel : ] expr )
```

*exprlabel* is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **Bayesian postestimation** for examples.

*spec* is one of *paramspec* or *exprspec*.

## Statistics for predictions

*Statistics for simulated outcomes, residuals, and more*

```
bayesstats ess yspec [yspec ...] using predfile [, options]
```

*Statistics for expressions of simulated outcomes, residuals, and more*

```
bayesstats ess (yexprspec) [(yexprspec) ...] using predfile [, options]
```

*Statistics for Mata functions of simulated outcomes, residuals, and more*

```
bayesstats ess (funcspec) [(funcspec) ...] using predfile [, options]
```

*Full syntax*

```
bayesstats ess predspec [predspec ...] using predfile [, options]
```

*predfile* is the name of the dataset created by **bayespredict** that contains prediction results.

*yspec* is {*ysimspec* | *residspec* | *muspec* | *label*}.

*ysimspec* is {\_ysim#} or {\_ysim#[*numlist*]}, where {\_ysim#} refers to all observations of the #th simulated outcome and {\_ysim#[*numlist*] } refers to the selected observations, *numlist*, of the #th simulated outcome. {\_ysim} is a synonym for {\_ysim1}.

*residspec* is {\_resid#} or {\_resid#[*numlist*]}, where {\_resid#} refers to all residuals of the #th simulated outcome and {\_resid#[*numlist*] } refers to the selected residuals, *numlist*, of the #th simulated outcome. {\_resid} is a synonym for {\_resid1}.

*muspec* is {\_mu#} or {\_mu#[*numlist*]}, where {\_mu#} refers to all expected values of the #th outcome and {\_mu#[*numlist*] } refers to the selected expected values, *numlist*, of the #th outcome. {\_mu} is a synonym for {\_mu1}.

*label* is the name of the function simulated using **bayespredict**.

With large datasets, specifications {\_ysim#}, {\_resid#}, and {\_mu#} may use a lot of time and memory and should be avoided. See [Generating and saving simulated outcomes](#) in [BAYES] **bayespredict**.

*yexprspec* is [*exprlabel*:] *yexpr*, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, {\_ysim#[#]}; individual expected outcome values, {\_mu#[#]}; individual simulated residuals, {\_resid#[#]}; and other scalar predictions, {*label*}.

*funcspec* is [*label*:] @*func*(*arg1* [, *arg2*]), where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of {\_ysim#[#]}, {\_resid#[#]}, or {\_mu#[#]}. *arg2* is primarily for use with user-defined Mata functions; see [Defining test statistics using Mata functions](#) in [BAYES] **bayespredict**.

*predspec* is one of *yspec*, (*yexprspec*), or (*funcspec*). See [Different ways of specifying predictions and their functions](#) in [BAYES] **Bayesian postestimation**.

<i>options</i>	Description
Main	
* <code>chains(_all   <i>numlist</i>)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
* <code>sepchains</code>	compute results separately for each chain
<code>skip(#)</code>	skip every # observations from the MCMC sample; default is <code>skip(0)</code>
<code>nolegend</code>	suppress table legend
<code>display_options</code>	control spacing, line width, and base and empty cells
Advanced	
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Options `chains()` and `sepchains` are relevant only when option `nchains()` is used with `bayesmh` or the `bayes` prefix.

## Options

### Main

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`sepchains` specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option `chains()`. Option `sepchains` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`showreffects` and `showreffects(reref)` are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time.

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

`display_options`: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fwrapon(style)`, and `nolstretch`; see [R] [Estimation options](#).

### Advanced

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is `min{500, mcmcsizes()/2}`. The total autocorrelation is computed as the sum of

all lag- $k$  autocorrelation values for  $k$  from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag- $k$  autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the  $k$ th lag are discarded.

## Remarks and examples

Remarks are presented under the following headings:

*Effective sample size and MCMC sampling efficiency*  
*Using bayesstats ess*

### Effective sample size and MCMC sampling efficiency

It is well known that for a random sample of  $T$  independent subjects, the standard error of the sample mean estimator is proportional to  $1/\sqrt{T}$ . In Bayesian inference, it is of interest to estimate the standard error of the posterior mean estimator. The posterior mean of a parameter of interest is typically estimated as a sample mean from an MCMC sample obtained from the marginal posterior distribution of the parameter of interest. Observations from an MCMC sample are not independent and are usually positively correlated, which must be taken into account when computing the standard error. Thus the standard error of the posterior mean estimator is proportional to  $1/\sqrt{\text{ESS}}$ , where ESS is the effective sample size for the parameter of interest. Typically, ESS is less than  $T$ , the total number of observations in the MCMC sample. We can thus interpret the posterior mean estimate as a sample mean estimate from an independent sample of size ESS. In other words, the effective sample size is an estimate of the number of independent observations that the MCMC chain represents. We say that MCMC samples with higher ESS are more efficient.

Effective sample size is directly related to the convergence properties of an MCMC sample—very low ESS relative to  $T$  suggests nonconvergence. In the extreme case of a perfectly correlated MCMC observation, ESS is 1. It is thus a standard practice to assess the quality of an MCMC sample by inspecting ESS values for all involved model parameters. Note, however, that high ESS values are not generally sufficient for declaring convergence of MCMC because pseudoconvergence, which may occur when MCMC does not explore the entire distribution, may also lead to high ESS values.

### Using bayesstats ess

`bayesstats ess` reports effective sample sizes, correlation times, and efficiencies for model parameters and their functions using the current Bayesian estimation results. When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see *Different ways of specifying model parameters* in [BAYES] **Bayesian postestimation**. You can also obtain results for scalar functions of model parameters; see *Specifying functions of model parameters* in [BAYES] **Bayesian postestimation**. You can obtain the summaries for prediction quantities when you specify the prediction dataset in the `using` specification; see *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation** for how to specify prediction quantities within `bayesstats ess`.



Consider our analysis of `auto.dta` from [example 4](#) in [\[BAYES\] bayesmh](#) using the mean-only normal model for `mpg` with a noninformative prior.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. set seed 14

. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})

Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys
```

Bayesian normal regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.2668
	Efficiency: min	=	.09718
		avg	= .1021
		max	= .1071
Log marginal-likelihood	=	-234.645	

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

### ► Example 1: Effective sample sizes for all parameters

To compute effective sample sizes and other related statistics for all model parameters, we type `bayesstats ess` without arguments after the `bayesmh` command.

```
. bayesstats ess
Efficiency summaries      MCMC sample size =      10,000
                          Efficiency:  min =      .09718
                              avg =      .1021
                              max =      .1071
```

	ESS	Corr. time	Efficiency
mpg			
_cons	971.82	10.29	0.0972
var	1070.99	9.34	0.1071

The closer the ESS estimates are to the MCMC sample size, the better. Also, the lower the correlation times are and the higher the efficiencies are, the better. ESS estimates can be interpreted as follows. In a sample of 10,000 MCMC observations, we have only about 972 independent observations to obtain estimates for `{mpg:_cons}` and only about 1,071 independent observations to obtain estimates for `{var}`. Correlation times are the reciprocal of efficiencies. You can interpret them as an estimated lag after which autocorrelation in an MCMC sample is small. In our example, the estimated lag is roughly 10 for both parameters. In general, efficiencies above 10% are considered good for the MH algorithm. In our example, they are about 10% for both parameters.

Alternatively, we could have listed all parameters manually:

```
. bayesstats ess {mpg:_cons} {var}
(output omitted)
```

◀

### ► Example 2: Effective sample sizes for functions of model parameters

Similarly to other Bayesian postestimation commands, `bayesstats ess` accepts expressions to compute results for functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to compute effective sample sizes for the standard deviation of `mpg` in addition to the variance.

```
. bayesstats ess (sd:sqrt({var})) {var}
Efficiency summaries      MCMC sample size =      10,000
                          Efficiency:  min =      .1071
                              avg =      .1082
                              max =      .1094
```

	ESS	Corr. time	Efficiency
sd : sqrt({var})			
sd	1093.85	9.14	0.1094
var	1070.99	9.34	0.1071

ESS and efficiency are higher for the standard deviation than for the variance, which means that we need slightly more iterations to estimate `{var}` with the same precision as `sd`.

If we wanted, we could have suppressed the `sd` legend in the output above by specifying the `nolegend` option.



## Stored results

`bayesstats ess` stores the following in `r()`:

### Scalars

<code>r(mcmcsize)</code>	MCMC sample size used in the computation
<code>r(skip)</code>	number of MCMC observations to skip in the computation; every <code>r(skip)</code> observations are skipped
<code>r(corrlag)</code>	maximum autocorrelation lag
<code>r(corrtol)</code>	autocorrelation tolerance
<code>r(nchains)</code>	number of chains used in the computation

### Macros

<code>r(names)</code>	names of model parameters and expressions
<code>r(expr_#)</code>	#th expression
<code>r(exprnames)</code>	expression labels
<code>r(chains)</code>	chains used in the computation, if <code>chains()</code> is specified

### Matrices

<code>r(ess)</code>	matrix with effective sample sizes, correlation times, and efficiencies for parameters in <code>r(names)</code>
<code>r(ess_chain#)</code>	matrix <code>ess</code> for chain <code>#</code> , if <code>sepchains</code> is specified

## Methods and formulas

Let  $\theta$  be a scalar model parameter and  $\{\theta_t\}_{t=1}^T$  be an MCMC sample of size  $T$  drawn from the marginal posterior distribution of  $\theta$ . The effective sample size of the MCMC sample of  $\theta$  is given by

$$\text{ESS} = T / \left( 1 + 2 \sum_{k=1}^{\text{max\_lags}} \rho_k \right)$$

where  $\rho_k$  is the lag- $k$  autocorrelation of the MCMC sample, and `max_lags` is the maximum number less than or equal to  $\rho_{\text{lag}}$  such that for all  $k = 1, \dots, \text{max\_lags}$ ,  $|\rho_k| > \rho_{\text{tol}}$ , where  $\rho_{\text{lag}}$  and  $\rho_{\text{tol}}$  are specified in options `corrlag()` and `corrtol()` with the respective default values of 500 and 0.01.

The lag- $k$  autocorrelation is  $\rho_k = \gamma_k / \gamma_0$ , where

$$\gamma_k = \frac{1}{T} \sum_{t=1}^{T-k} (\theta_t - \hat{\theta})(\theta_{t+k} - \hat{\theta})$$

is the empirical autocovariance of lag  $k$ , and  $\gamma_0$  simplifies to the sample variance.  $\hat{\theta}$  is the posterior mean estimator.

Correlation time is defined as  $T/\text{ESS}$ , and efficiency is defined as the reciprocal of the correlation time,  $\text{ESS}/T$ . Because ESS is between 0 and  $T$ , inclusively, the efficiency is always between 0 and 1.

In the presence of multiple chains, the overall ESS is computed as the sum of the individual ESS statistics calculated using each chain independently. Correlation times and efficiencies are then computed using the overall ESS and the total MCMC sample size,  $M \times T$ , where  $M$  is the number of chains.

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the bayes prefix

[BAYES] [bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm

[BAYES] [Bayesian estimation](#) — Bayesian estimation commands

[BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[BAYES] [bayesstats summary](#) — Bayesian summary statistics

# Title

bayesstats grubin — Gelman–Rubin convergence diagnostics

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

## Description

`bayesstats grubin` calculates Gelman–Rubin convergence diagnostics for model parameters and functions of model parameters using current Bayesian estimation results containing at least two Markov chains.

## Quick start

Gelman–Rubin convergence diagnostics for all model parameters after a Bayesian regression model using four chains

```
bayes, nchains(4): regress y x1
bayesstats grubin
```

As above, but only for model parameters `{y:x1}` and `{sigma2}`

```
bayesstats grubin {y:x1} {sigma2}
```

Gelman–Rubin convergence diagnostics for functions of scalar model parameters

```
bayesstats grubin ({y:x1}--{y:_cons}) (sd:sqrt({sigma2}))
```

## Menu

Statistics > Bayesian analysis > Gelman–Rubin convergence diagnostics

# Syntax

Convergence statistics for all model parameters

```
bayesstats grubin [ , options showeffects[ (reref) ] ]
bayesstats grubin _all [ , options showeffects[ (reref) ] ]
```

Convergence statistics for selected model parameters

```
bayesstats grubin paramspec [ , options ]
```

Convergence statistics for functions of model parameters

```
bayesstats grubin exprspec [ , options ]
```

Full syntax

```
bayesstats grubin spec [ spec ... ] [ , options ]
```

*paramspec* can be one of the following:

- `{eqname:param}` refers to a parameter *param* with equation name *eqname*;
- `{eqname:}` refers to all model parameters with equation name *eqname*;
- `{eqname:paramlist}` refers to parameters with names in *paramlist* and with equation name *eqname*;
- or
- `{param}` refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See [Different ways of specifying model parameters](#) in [BAYES] Bayesian postestimation for examples.

*exprspec* is an optionally labeled expression of model parameters specified in parentheses:

```
( [ exprlabel : ] expr )
```

*exprlabel* is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See [Specifying functions of model parameters](#) in [BAYES] Bayesian postestimation for examples.

*spec* is one of *paramspec* or *exprspec*.

<i>options</i>	Description
<code>sort</code>	list parameters in descending order of their convergence statistics
<code>skip(#)</code>	skip every # observations from the MCMC sample; default is <code>skip(0)</code>
<code>nolegend</code>	suppress table legend
<code>display_options</code>	control spacing, line width, and base and empty cells

## Options

`sort` specifies that model parameters be listed in descending order of their Gelman–Rubin convergence statistics. This option is useful for models with many parameters, such as multilevel models, to more easily identify the set of parameters with large values of convergence statistics.

`showeffects` and `showeffects(reref)` are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time. If random-effects parameters are of interest in your study, you should use option `showeffects` to check their convergence diagnostics.

`skip(#)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

*display\_options*: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fwrapon(style)`, and `nolstretch`; see [R] [Estimation options](#).

## Remarks and examples

Remarks are presented under the following headings:

*Gelman–Rubin convergence diagnostic*  
*Using bayesstats grubin*

### Gelman–Rubin convergence diagnostic

The Gelman–Rubin convergence diagnostic,  $R_c$ , assesses MCMC convergence by analyzing differences between multiple Markov chains. The convergence is assessed by comparing the estimated between-chains and within-chain variances for each model parameter. Large differences between these variances indicate nonconvergence. See [Gelman and Rubin \(1992\)](#) and [Brooks and Gelman \(1998\)](#) for details.

Large values of  $R_c$  indicate nonconvergence of MCMC. Literature suggests that the values of this diagnostic should be less than 1.2 for all model parameters to declare MCMC convergence. In practice, a more stringent convergence rule,  $R_c < 1.1$ , is often used.

Gelman–Rubin diagnostic relies on a Student's  $t$  approximation of the marginal posterior distribution of a model parameter. When this assumption is suspect, it is recommended to transform the parameter such that its marginal posterior distribution is better approximated by a Student's  $t$  distribution before obtaining the diagnostic. For example, for the variance parameter, it is better to compute the diagnostic for the log variance.

## Using bayesstats grubin

The `bayesstats grubin` command computes the Gelman–Rubin convergence diagnostic for each model parameter using multiple MCMC samples or chains from a common posterior model. This command requires at least two chains. Multiple chains can be obtained by using the `nchains()` option with the `bayesmh` command ([BAYES] [bayesmh](#)) or with the `bayes` prefix ([BAYES] [bayes](#)). When you simulate multiple chains to assess convergence, it is important to use [overdispersed initial values](#) (Gelman and Rubin 1992, Brooks and Gelman 1998). See [Specifying initial values](#) in [BAYES] [bayesmh](#) and [Initial values](#) in [BAYES] [bayes](#) for details.

When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see [Different ways of specifying model parameters](#) in [BAYES] [Bayesian postestimation](#). You can also obtain results for scalar functions of model parameters; see [Specifying functions of model parameters](#) in [BAYES] [Bayesian postestimation](#). Also see [example 2](#).

For multilevel models, similarly to other Bayesian postestimation commands, `bayesstats grubin` does not report convergence statistics for the random-effects parameters by default. You can use the `showeffects` option to see them for all random-effects parameters or the `showeffects(reref)` option for a subset `reref` of random-effects parameters of interest. See [Multilevel models](#) in [BAYES] [bayes](#) for more information about MCMC convergence in multilevel models.

For models with many parameters such as multilevel models, you can use the `sort` option to list model parameters in descending order of their convergence statistics `Rc`. The parameters with the largest values of `Rc` will be listed first, making it easier to verify their convergence.

### ► Example 1: Convergence diagnostics for all parameters

Recall our analysis of `womenwage.dta` using the `bayes: regress` command from [example 1](#) in [BAYES] [bayes](#). We fit a linear regression model to the response variable `wage` with predictor `age`. Here we use option `nchains(3)` to simulate three Markov chains to formally check convergence of model parameters. To ensure reproducibility of multiple chains, we also specify the `rseed(15)` option. Specifying `set seed` is not sufficient for reproducibility with multiple chains; see [Reproducing results](#) in [BAYES] [bayesmh](#) for details.



```
. use https://www.stata-press.com/data/r16/womenwage
(Wages of women)
. bayes, nchains(3) rseed(15): regress wage age
Chain 1
  Burn-in ...
  Simulation ...
Chain 2
  Burn-in ...
  Simulation ...
Chain 3
  Burn-in ...
  Simulation ...
Model summary
-----
Likelihood:
  wage ~ regress(xb_wage,{sigma2})
Priors:
  {wage:age _cons} ~ normal(0,10000)
                  {sigma2} ~ igamma(.01,.01)                                     (1)
```

(1) Parameters are elements of the linear form xb\_wage.

Bayesian linear regression	Number of chains	=	3
Random-walk Metropolis-Hastings sampling	Per MCMC chain:		
	Iterations	=	12,500
	Burn-in	=	2,500
	Sample size	=	10,000
	Number of obs	=	488
	Avg acceptance rate	=	.3673
	Avg efficiency: min	=	.1409
	avg	=	.1735
	max	=	.2294
Avg log marginal-likelihood = -1810.1557	Max Gelman-Rubin Rc	=	1

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
wage						
age	.4003528	.0599411	.000922	.4002037	.2804134	.5188627
_cons	5.999502	1.769855	.026358	6.025288	2.571305	9.517341
sigma2	90.80977	5.822896	.070195	90.49567	79.92114	102.7621

Note: Default priors are used for model parameters.  
Note: Default initial values are used for multiple chains.

Compared with [example 1](#) in [\[BAYES\] bayes](#), the precision of the posterior means almost doubled with more chains, judging by the MCMC standard errors. For example, the MCSE estimate for {sigma2} drops from 0.12 to 0.07.

In the presence of multiple chains, the bayes prefix automatically reports in the header the maximum value of the Gelman–Rubin convergence statistics across all parameters. In practice, we want to see this value be close to 1; if it is less than 1.1, the chains are considered to have converged. This convergence rule is satisfied in our example.

To compute the Gelman–Rubin statistics for all model parameters, we type `bayesstats grubin` without arguments after the `bayes` prefix.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains      =          3
MCMC size, per chain =       10,000
Max Gelman-Rubin Rc   =       1.000323
```

	Rc
wage	
age	1.000062
_cons	1.000323
sigma2	1.000253

Convergence rule:  $Rc < 1.1$

Just like the `bayes` prefix, the `bayesstats grubin` command reports in the header the maximum value of  $Rc$  across all parameters. This is particularly useful as a quick convergence check for models with many parameters: if the maximum  $Rc$  is less than 1.2 or 1.1, then this convergence rule is satisfied by all parameters. In our example, the maximum  $Rc$  is 1.0003 and is less than 1.1, so the convergence criterion is met for all parameters.

The table reports the  $Rc$  estimates for each model parameter. As we already determined based on the maximum  $Rc$ , the convergence diagnostics for all model parameters are less than 1.1. This suggests that all chains have converged.

◀

## ► Example 2: Convergence diagnostics for functions of parameters

Continuing with [example 1](#), we can compute the Gelman–Rubin statistics for functions of parameters. Let's compute the convergence diagnostic for the log-transformed variance parameter `{sigma2}`.

```
. bayesstats grubin (lnsigma2: ln({sigma2}))
Gelman-Rubin convergence diagnostic
Number of chains      =          3
MCMC size, per chain =       10,000
Max Gelman-Rubin Rc   =       1.000268
```

    lnsigma2 : ln({sigma2})

	Rc
lnsigma2	1.000268

Convergence rule:  $Rc < 1.1$

Again, the convergence diagnostic for the log-transformed variance is less than 1.1 indicating no convergence problems with the transformed parameter. This also suggests that `{sigma2}` does not have convergence problems.

◀

In our examples, we used the default initial values provided by `bayes`: with multiple chains; see [Initial values](#) in [\[BAYES\]](#) `bayes`. To fully explore MCMC convergence, particularly when a posterior distribution is suspected to have multiple modes, you should use [overdispersed initial values](#). See

*Multiple chains using overdispersed initial values* in [BAYES] **bayesmh** for an example of how to specify overdispersed initial values.

Of course, it is important to explore convergence visually as well; see *Convergence diagnostics using multiple chains* in [BAYES] **bayesmh**.

## Stored results

**bayesstats grubin** stores the following in `r()`:

Scalars

<code>r(mcmcsize)</code>	MCMC sample size of each chain
<code>r(nchains)</code>	number of MCMC chains
<code>r(Rc_max)</code>	maximum convergence diagnostic

Matrices

<code>r(Rc)</code>	convergence diagnostics <code>Rc</code>
<code>r(t_df)</code>	degrees of freedom of a $t$ distribution
<code>r(B)</code>	between-chains variances
<code>r(W)</code>	within-chain variances
<code>r(V)</code>	total variances

## Methods and formulas

Suppose we have  $M$  chains of length  $T$ . For a model parameter  $\theta$ , let  $\{\theta_{jt}\}_{t=1}^T$  be the  $j$ th simulated chain drawn from the marginal posterior distribution of  $\theta$ ,  $j = 1, \dots, M$ . Let  $\hat{\theta}_j$  and  $\hat{s}_j^2$  be the respective sample posterior mean and variance of the  $m$ th chain, and let the overall sample posterior mean be  $\hat{\theta} = (1/M) \sum_{j=1}^M \hat{\theta}_j$ . The between-chains and within-chain variances are given by

$$B = \frac{T}{M-1} \sum_{j=1}^M (\hat{\theta}_j - \hat{\theta})^2$$

$$W = \frac{1}{M} \sum_{j=1}^M \hat{s}_j^2$$

When the chains are strongly stationary, that is, all chains draw samples from the target posterior distribution, the weighted average of  $W$  and  $B$

$$\hat{\sigma}^2 = \frac{T-1}{T} W + \frac{1}{T} B$$

is an unbiased estimator of the marginal posterior variance of  $\theta$ .

Gelman and Rubin (1992) approximate the target distribution of  $\theta$  by a Student’s  $t$  distribution with mean  $\hat{\theta}$  and scale  $\sqrt{\hat{V}}$ , where

$$\hat{V} = \frac{T-1}{T} W + \frac{M+1}{MT} B$$

They define the so-called “scale” reduction factor as the ratio of  $\hat{V}$  and  $\sigma^2 = \text{Var}(\theta)$ . They further estimate  $\sigma^2$  by  $W$  and use the ratio of  $\hat{V}$  and  $W$  as an estimator of the scale reduction factor, known as the potential scale reduction factor. If the  $M$  chains have converged to the target posterior distribution, then the potential scale reduction factor should be close to 1.

Brooks and Gelman (1998) propose the corrected estimator of the potential scale reduction factor,  $R_c$ , that accounts for sampling variability:

$$R_c = \sqrt{\frac{\hat{d} + 3}{\hat{d} + 1} \frac{\hat{V}}{W}}$$

where  $\hat{d}$  is the estimated degrees of freedom of the approximating Student's  $t$  distribution for  $\theta$

$$\hat{d} = \frac{2\hat{V}^2}{\widehat{\text{Var}}(\hat{V})}$$

and

$$\begin{aligned} \widehat{\text{Var}}(\hat{V}) = & \left( \frac{T-1}{T} \right)^2 \frac{1}{M} \widehat{\text{Var}}(\hat{s}_j^2) + \left( \frac{M+1}{MT} \right)^2 \frac{2}{M-1} B^2 \\ & + 2 \frac{(M+1)(T-1)}{M^2 T} \left\{ \widehat{\text{Cov}}(\hat{s}_j^2, \hat{\theta}_j^2) - 2\hat{\theta} \widehat{\text{Cov}}(\hat{s}_j^2, \hat{\theta}_j) \right\} \end{aligned}$$

$\widehat{\text{Var}}(\hat{s}_j^2)$  is the sample variance of  $\hat{s}_j^2$ 's,  $j = 1, \dots, M$ .  $\widehat{\text{Cov}}(\hat{s}_j^2, \hat{\theta}_j^2)$  and  $\widehat{\text{Cov}}(\hat{s}_j^2, \hat{\theta}_j)$  are the sample covariances of  $\hat{s}_j^2$ 's and  $\hat{\theta}_j^2$ 's and  $\hat{s}_j^2$ 's and  $\hat{\theta}_j$ 's, respectively.

Brooks and Gelman (1998) suggested to use the criterion  $R_c < 1.2$  for all model parameters to declare MCMC convergence. In practice, a more stringent convergence criterion,  $R_c < 1.1$ , is often used. If a convergence criterion is not met, longer chains or other means for improving the convergence are needed.

## References

- Balov, N. 2016. Gelman–Rubin convergence diagnostic using multiple chains. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/05/26/gelman-rubin-convergence-diagnostic-using-multiple-chains/>.
- Brooks, S. P., and A. Gelman. 1998. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics* 7: 434–455.
- Gelman, A., and D. B. Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical Science* 7: 457–472.

## Also see

- [BAYES] **bayes** — Bayesian regression models using the bayes prefix
- [BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm
- [BAYES] **Bayesian estimation** — Bayesian estimation commands
- [BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] **bayesstats summary** — Bayesian summary statistics

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

Description

`bayesstats ic` calculates and reports model-selection statistics, including the deviance information criterion (DIC), log marginal-likelihood, and Bayes factors (BFs), using current Bayesian estimation results. BFs can be displayed in the original metric or in the log metric. The command also provides two different methods to approximate marginal likelihood.

Quick start

Information criteria for previously saved estimation results `A` and `B` with `A` used as the base model by default

```
bayesstats ic A B
```

As above, but use `B` as the base model instead of `A`

```
bayesstats ic A B, basemodel(B)
```

Report BFs instead of the default log BFs

```
bayesstats ic A B, bayesfactor
```

Menu

Statistics > Bayesian analysis > Information criteria

# Syntax

```
bayesstats ic [namelist] [, options]
```

*namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

<i>options</i>	Description
<b>Main</b>	
<code>basemodel(name)</code>	specify a base or reference model; default is the first-listed model
<code>bayesfactor</code>	report BFs instead of the default log BFs
<code>diconly</code>	report only DIC
<code>*chains(_all   numlist)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
<code>*sepchains</code>	compute results separately for each chain
<b>Advanced</b>	
<code>marglmethod(method)</code>	specify marginal-likelihood approximation method; default is to use Laplace–Metropolis approximation, <code>lmetropolis</code> ; rarely used

\*Options `chains()` and `sepchains` are relevant only when option `nchains()` is used with `bayesmh` or the `bayes` prefix.

<i>method</i>	Description
<code>lmetropolis</code>	Laplace–Metropolis approximation; the default
<code>hmean</code>	harmonic-mean approximation

# Options

## Main

`basemodel(name)` specifies the name of the model to be used as a base or reference model when computing BFs. By default, the first-listed model is used as a base model.

`bayesfactor` specifies that BFs be reported instead of the default log BFs.

`diconly` specifies that only DIC be reported in the table and that the log marginal-likelihood and Bayes factors be omitted from the table. Options `basemodel()`, `basefactor`, and `marglmethod()` have no effect when the `diconly` option is specified.

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`sepchains` specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option `chains()`. Option `sepchains` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`marglmethod(method)` specifies a method for approximating the marginal likelihood. *method* is either `lmetropolis`, the default, for Laplace–Metropolis approximation or `hmean` for harmonic-mean approximation. This option is rarely used.

## Remarks and examples

Remarks are presented under the following headings:

*Bayesian information criteria*

*Bayes factors*

*Using bayesstats ic*

## Bayesian information criteria

Bayesian information criteria are used for selecting a model among a set of candidate models that best fits the data. Likelihood-based inference is known to be prone to overfitting the data. Indeed, it is often possible to increase the likelihood by simply including more parameters in a model. Bayesian information criteria address this problem by applying a penalty proportional to the complexity of the models to the likelihood.

Consider a finite set of Bayesian models  $M_1, \dots, M_r$ , which we want to compare with a base model  $M_b$ . All models  $M_j$ s are fit to the same dataset but may differ in their likelihood or prior specification.

Three commonly used information criteria are Akaike information criterion (AIC), Bayesian information criterion (BIC), and DIC. All three criteria are likelihood based and include a goodness-of-fit term proportional to the negative likelihood of the model and a penalty term proportional to the number of parameters in the model. Models with smaller values of these criteria are preferable.

The BIC, originally derived for the exponential family of distributions, is based on the assumption that the model has a flat, noninformative prior. In frequentist statistics, BIC is widely used as a variable-selection criterion, particularly in linear regression. In BIC, the penalty term is a product of the number of parameters in the model and the log of the sample size. The penalty of BIC thus increases not only with the number of parameters but also with the sample size. In the AIC, the penalty term is two times the number of parameters and does not depend on the sample size. As a result, BIC is more conservative than AIC and prefers simpler models. DIC is similar to AIC, but its penalty term is based on a complexity term that measures the difference between the expected log likelihood and the log likelihood at the posterior mean point. DIC is designed specifically for Bayesian estimation that involves MCMC simulations.

The limitation of all three criteria is that they either ignore prior distributions or assume that prior distributions are noninformative. They are thus not well suited for Bayesian sensitivity analysis, when models with the same parameters but different priors are being compared.

The `bayesstats ic` command reports DIC. See [R] [estat ic](#) after the corresponding maximum likelihood estimation command for values of AIC and BIC.

## Bayes factors

In Bayesian inference, BFs are preferred to model-selection criteria because, unlike BIC, AIC, and DIC, they incorporate the information about model priors. Taking into account prior information is essential for Bayesian sensitivity analysis, when models with the same parameters but different priors are being compared.

The BF of two models is just the ratio of their marginal likelihoods calculated using the same dataset. Unlike BIC, AIC, and DIC, BFs include all information about the specified Bayesian model. Thus BFs are not applicable to models with improper priors, whereas BIC, AIC, and DIC are still applicable because they ignore prior information. BFs, however, are often difficult to compute reliably because of the difficulty in computing marginal likelihoods.

BFs also require that posterior distributions be completely specified, including the normalizing constants. The latter is especially important in Bayesian estimation using MCMC simulations, when the normalizing constants are often omitted from the specification of a posterior distribution. The Bayesian estimation commands always simulate from a complete posterior distribution when you select one of the supported Bayesian models, but you need to make sure to include all normalizing constants with your posterior distribution when you are programming your own Bayesian model (see [BAYES] [bayesmh evaluators](#)) and would like to use BFs during postestimation.

Let  $BF_{jb}$ ,  $j = 1, \dots, r$ , be the BF of model  $M_j$  with respect to the base model  $M_b$ . All models  $M_j$  are fit to the same dataset; otherwise, BFs are meaningless. The `bayesstats ic` command calculates  $BF_{jb}$ 's and reports them in log metric or in absolute metric when the `bayesfactor` option is specified.

Jeffreys (1961) proposes the following interpretation of the values of  $BF_{jb}$  based on half-units of the log metric:

$\log_{10}(BF_{jb})$	$BF_{jb}$	Evidence against $M_b$
0 to 1/2	1 to 3.2	Bare mention
1/2 to 1	3.2 to 10	Substantial
1 to 2	10 to 100	Strong
>2	>100	Decisive

Kass and Raftery (1995) suggest using twice the natural logarithm of the BF to make it have the same scale as the DIC and likelihood-ratio test statistic. They suggest the following interpretation table:

$2 \log_e(BF_{jb})$	$BF_{jb}$	Evidence against $M_b$
0 to 2	1 to 3	Bare mention
2 to 6	3 to 20	Positive
6 to 10	20 to 150	Strong
>10	>150	Very strong

Typically, the worst-fitting model is chosen as a base model. If the base model happens to be better than the comparison model, the corresponding BF will be negative. In this case, you can apply results above to the absolute value of the BF.

BFs compute relative probabilities of how well each model fits the data compared with the base model. Being relative quantities, BFs cannot be used to measure goodness of fit of a particular model unless one assumes that the base model fits the data well. Some researchers view this as a limitation of BFs (Gelman et al. 2014). Kass and Raftery (1995), on the other hand, show that BFs can be



viewed as differences between predictive scores and thus can be used to measure success of different models at predicting the data.

BFs have several advantages over the more traditional, frequentist testing methods. For example, they do not have the limitation of the  $p$ -value approach to systematically reject the null hypothesis in large samples. BFs are also suitable for comparing both nonnested and nested models. Also see *Comparing Bayesian models* in [BAYES] **Intro** for more information about Bayesian model comparison.

A key element in computing BFs is calculating the marginal likelihood. Except for some rare cases, marginal likelihood does not have a closed form and needs to be approximated. A detailed review of different approximation methods is given by [Kass and Raftery \(1995\)](#). The default method implemented in `bayesstats ic` (and `bayesmh`) is the Laplace–Metropolis approximation ([Lewis and Raftery 1997](#)). The harmonic-mean approximation of the marginal likelihood is also available via the `marglmethod(hmean)` option, but we recommend that you use the default method. See *Methods and formulas* in [BAYES] **bayesmh** for technical details.

Using **bayesstats ic**

► Example 1

The `bayesstats ic` command provides several model-selection statistics that can be used to compare models. To illustrate the use of `bayesstats ic`, we consider `auto.dta`. We model the fuel-efficiency variable `mpg` using a normal distribution with fixed variance but unknown, random mean. There is only one random parameter in this model—{`mpg:_cons`}. We compare the models with three different prior distributions to find the best one among them. We fit the three models using `bayesmh` and save the corresponding estimation results as `uniform1`, `uniform2`, and `normal`.

First, for comparison purposes, let’s obtain the maximum likelihood estimate (MLE) of the mean of `mpg`, which is simply the sample mean in our example:

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. summarize mpg
```

Variable	Obs	Mean	Std. Dev.	Min	Max
mpg	74	21.2973	5.785503	12	41

The sample mean of `mpg` is roughly 21.3.

Next, we use `bayesmh` to fit our first model of interest. We fix the variance of the normal distribution to 30, which is close to the estimated variance of `mpg` of  $5.79^2 = 33.52$ .

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, uniform(-10, 10))
> initial({mpg:_cons} 2) saving(uniform1_simdata)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ uniform(-10,10)
```

---

```
Bayesian normal regression          MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in       =       2,500
                                          MCMC sample size =     10,000
                                          Number of obs   =        74
                                          Acceptance rate =      .4102
Log marginal-likelihood = -397.42978      Efficiency     =      .08018
```

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	9.965511	.0342812	.001211	9.975729	9.871825	9.998796

---

```
file uniform1_simdata.dta saved
. estimates store uniform1
```

In the first model, we deliberately chose a prior for `{mpg:_cons}`, `uniform(-10,10)`, that does not include the value of the sample mean. We thus expect this model to fit poorly. Because of the restricted domain of the specified uniform prior, we also needed to specify an initial value for `{mpg:_cons}` for MCMC to start from a point of positive posterior probability.

We also specified the `saving()` option to save the MCMC simulation dataset so that we could use `estimates store` to store our estimation results for future use. See [Storing estimation results after Bayesian estimation](#) in [BAYES] [Bayesian postestimation](#) for details.

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, uniform(10, 30))
> initial({mpg:_cons} 20) saving(uniform2_simdata)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ uniform(10,30)
```

---

Bayesian normal regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.4272
	Efficiency	=	.2414

---

```
Log marginal-likelihood = -237.08583
```

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_cons	21.31085	.6447073	.013123	21.31485	20.06381	22.57936

---

```
file uniform2_simdata.dta saved
. estimates store uniform2
```

In the second model, we used a uniform prior that included the value of the sample mean in its domain.

```
. set seed 14
. bayesmh mpg, likelihood(normal(30))
> prior({mpg:_cons}, normal(30)) saving(normal_simdata)
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
```

```
Prior:
  {mpg:_cons} ~ normal(30)
```

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4295
Log marginal-likelihood = -244.16624	Efficiency =	.2319

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_cons	21.01901	.6461194	.013417	21.01596	19.76637	22.3019

```
file normal_simdata.dta saved
```

```
. estimates store normal
```

In the third model, we used a normal prior with a variance fixed at 30. Note that we did not need to specify an initial value for {mpg:\_cons} in this model, because the domain of the normal distribution is the whole real line.

Both the `uniform2` and `normal` models yield estimates close to the MLE of 21.3. According to their credible intervals, the domain of the posterior distribution of {mpg:\_cons} is concentrated around MLE. For example, the 95% credible interval for the `uniform2` model is [20.06, 22.58].

Now, let's use `bayesstats ic` to compare the three models. We list all the models following the command name and use the `normal` model as a reference model.

```
. bayesstats ic uniform1 uniform2 normal, basemodel(normal)
Bayesian information criteria
```

	DIC	log(ML)	log(BF)
uniform1	785.8891	-397.4298	-153.2635
uniform2	471.1909	-237.0858	7.080404
normal	471.3905	-244.1662	.

```
Note: Marginal likelihood (ML) is computed
      using Laplace-Metropolis approximation.
```

The `uniform1` model performs worse than the other two models according to the log marginal-likelihood, `log(ML)`, and `DIC`—the `DIC` value is much larger, and the `log(ML)` value is much smaller for the `uniform1` model. The other two models have only slightly different values for `DIC` and `log(ML)`, according to which the `uniform2` model is preferable.

Although the `uniform2` and `normal` models have different prior distributions, they have almost identical posterior domain, that is, the range of values of {mpg:\_cons} where the posterior is strictly positive. As such, they will have the same values for `AIC` and `BIC`, and we will not be able to discriminate between the two models based on these information criteria.

The most decisive factor between the `uniform2` and `normal` models is the BF. The value of  $\log \text{BF}$ ,  $\log(\text{BF})$ , is 7.08, which provides very strong evidence in favor of the `uniform2` model.

We thus conclude that `uniform2` is the best model among the three considered models. This may be explained by the fact that the specified `uniform(10,30)` prior is in more agreement with the likelihood of the data than the specified `normal(0,30)` prior.

After your analysis, remember to erase the saved simulation datasets you no longer need. For example, we erase all of them by typing

```
. erase uniform1_simdata.dta
. erase uniform2_simdata.dta
. erase normal_simdata.dta
```

◀

## Stored results

`bayesstats ic` stores the following in `r()`:

### Scalars

<code>r(bayesfactor)</code>	1 if <code>bayesfactor</code> is specified, 0 otherwise
<code>r(nchains)</code>	number of chains used in the computation

### Macros

<code>r(names)</code>	names of estimation results used
<code>r(basemodel)</code>	name of the base or reference model
<code>r(marglmethod)</code>	method for approximating marginal likelihood: <code>lmetropolis</code> or <code>hmean</code>
<code>r(chains)</code>	chains used in the computation, if <code>chains()</code> is specified

### Matrices

<code>r(ic)</code>	matrix reporting DIC, $\log(\text{ML})$ , and $\log(\text{BF})$ or $\text{BF}$ , if <code>bayesfactor</code> is specified
<code>r(ic_chain#)</code>	matrix <code>ic</code> for chain #, if <code>sepchains</code> is specified

## Methods and formulas

DIC was introduced by Spiegelhalter et al. (2002) for Bayesian model selection using MCMC simulations. DIC is based on the deviance statistics

$$D(\theta) = -2 \{ \log f(\mathbf{y}; \theta) - \log f^*(\mathbf{y}; \theta^*) \}$$

where  $f(\cdot; \cdot)$  is the likelihood function of the model and  $f^*(\mathbf{y}; \theta^*)$  is the likelihood of the full model that fits data perfectly. Because  $f^*(\mathbf{y}; \theta^*)$  is constant across models fit to the same data, it is ignored in the actual calculation of DIC. Given an MCMC sample  $\{\theta_t\}_{t=1}^T$ , the expected deviance can be estimated by the sample average  $\bar{D}(\theta) = 1/T \sum_{t=1}^T D(\theta_t)$ . Similarly to AIC and BIC, DIC is a sum of two components: the goodness-of-fit term  $\bar{D}(\theta)$  and the model complexity term  $p_D$ :  $\text{DIC} = \bar{D}(\theta) + p_D$ . The complexity is defined as the difference between the expected deviance and the deviance at the sample posterior mean:  $p_D = \bar{D}(\theta) - D(\bar{\theta})$ . We thus have

$$\text{DIC} = D(\bar{\theta}) + 2p_D$$

Models with smaller values of DIC are preferred to models with larger values of DIC.

With multiple chains, the `bayesstats ic` command reports the average DIC and the average log marginal-likelihood computed over the chains. If the `sepchains` option is specified, these statistics are reported separately for each chain.

BFs were introduced by [Jeffreys \(1961\)](#). The BF of two models,  $M_1$  and  $M_2$ , is given by

$$\text{BF}_{12} = \frac{P(\mathbf{y}|M_1)}{P(\mathbf{y}|M_2)} = \frac{m_1(\mathbf{y})}{m_2(\mathbf{y})}$$

where  $m_1(\cdot)$  and  $m_2(\cdot)$  are the corresponding marginal likelihoods associated with models  $M_1$  and  $M_2$ . (See [Methods and formulas](#) in [\[BAYES\] bayesmh](#) for details about computing marginal likelihood.) BFs are defined only for proper marginal densities. Comparing models with improper priors is allowed as long as the resulting marginal densities are proper. The methodological importance of BFs comes from the fact that the so-called posterior odds is a product of prior odds and BF:

$$\frac{P(M_1|\mathbf{y})}{P(M_2|\mathbf{y})} = \frac{P(M_1)}{P(M_2)} \times \text{BF}_{12}$$

Therefore, if we assume that  $M_1$  and  $M_2$  are equally probable a priori, the posterior odds will be equal to the BF. We thus prefer model  $M_1$  if  $\text{BF}_{12} > 1$  and model  $M_2$  otherwise. In practice, because of the higher numerical stability, we often calculate BFs in the (natural) log metric and compare its value against 0.

$$\log \text{BF}_{12} = \log m_1(\mathbf{y}) - \log m_2(\mathbf{y})$$

With multiple chains, BFs are computed using the average log marginal-likelihoods. If the `sepchains` option is specified, BFs are calculated and reported separately for each chain.

## References

- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Jeffreys, H. 1961. *Theory of Probability*. 3rd ed. Oxford: Oxford University Press.
- Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.
- Lewis, S. M., and A. E. Raftery. 1997. Estimating Bayes factors via posterior simulation with the Laplace–Metropolis estimator. *Journal of the American Statistical Association* 92: 648–655.
- Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. Van Der Linde. 2002. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B* 64: 583–639.

## Also see

- [\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix
- [\[BAYES\] bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm
- [\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands
- [\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix
- [\[BAYES\] bayestest model](#) — Hypothesis testing using model posterior probabilities
- [\[R\] estimates](#) — Save and manipulate estimation results

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

## Description

`bayesstats ppvalues` performs posterior predictive checking of the goodness of fit of a Bayesian model. It computes [posterior predictive  \$p\$ -values](#) (PPPs) for functions of [replicated outcomes](#) produced by `bayespredict` after fitting a model using `bayesmh`. PPPs measure the agreement between replicated and observed data. PPPs close to 0 or 1 indicate lack of model fit. The command also reports other summary statistics related to [posterior predictive checking](#).

## Quick start

*Posterior predictive summaries of replicated outcomes*

Bayesian predictions for all outcome variables after fitting a two-equation Bayesian model using `bayesmh`

```
bayespredict {_ysim1} {_ysim2}, saving(prdata)
```

Posterior predictive summaries for the first replicated outcome

```
bayesstats ppvalues {_ysim} using prdata
```

Posterior predictive summaries for the simulated residuals of the first outcome

```
bayesstats ppvalues {_resid} using prdata
```

Posterior predictive summaries for both replicated outcomes

```
bayesstats ppvalues {_ysim1} {_ysim2} using prdata
```

Posterior predictive summaries for the first observation of the second replicated outcome squared

```
bayesstats ppvalues ({_ysim2[1]}^2) using prdata
```

*Posterior predictive summaries for test statistics of replicated outcomes*

Posterior predictive summaries for the maximum and minimum across observations of the second replicated outcome

```
bayesstats ppvalues (y2max:@max({_ysim2})) (y2min:@min({_ysim2})) ///
using prdata
```

Posterior predictive summaries for the maximum and minimum across observations of the residuals for the first outcome variable

```
bayesstats ppvalues (rmax:@max({_resid1})) (rmin:@min({_resid1})) ///
using prdata
```

## Menu

Statistics > Bayesian analysis > Posterior predictive p-values

## Syntax

*Posterior predictive summaries for replicated outcomes, residuals, and more*

```
bayesstats ppvalues yspec [yspec ...] using predfile [, options]
```

*Posterior predictive summaries for expressions of replicated outcomes, residuals, and more*

```
bayesstats ppvalues (yexprspec) [(yexprspec) ...] using predfile [, options]
```

*Posterior predictive summaries for Mata functions of replicated outcomes, residuals, and more*

```
bayesstats ppvalues (funccspec) [(funccspec) ...] using predfile [, options]
```

*Full syntax*

```
bayesstats ppvalues predspec [predspec ...] using predfile [, options]
```

*predfile* is the name of the dataset created by [bayespredict](#) that contains prediction results.

*yspec* is {*ysimspec* | *residspec* | *label*}.

*ysimspec* is {\_ysim#} or {\_ysim#[*numlist*]}, where {\_ysim#} refers to all observations of the #th replicated outcome and {\_ysim#[*numlist*]} refers to the selected observations, *numlist*, of the #th replicated outcome. {\_ysim} is a synonym for {\_ysim1}.

*residspec* is {\_resid#} or {\_resid#[*numlist*]}, where {\_resid#} refers to all residuals of the #th replicated outcome and {\_resid#[*numlist*]} refers to the selected residuals, *numlist*, of the #th replicated outcome. {\_resid} is a synonym for {\_resid1}.

*label* is the name of the function simulated using [bayespredict](#).

With large datasets, specifications {\_ysim#} and {\_resid#} may use a lot of time and memory and should be avoided. See [Generating and saving simulated outcomes](#) in [BAYES] [bayespredict](#).

*yexprspec* is [*exprlabel*:] *yexpr*, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, {\_ysim#[#]}; individual expected outcome values, {\_mu#[#]}; individual simulated residuals, {\_resid#[#]}; and other scalar predictions, {*label*}.

*funccspec* is [*label*:] @*func*(*arg1* [, *arg2*]), where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of {\_ysim[#]}, {\_resid[#]}, or {\_mu[#]}. *arg2* is primarily for use with user-defined Mata functions; see [Defining test statistics using Mata functions](#) in [BAYES] [bayespredict](#).

*predspec* is one of *yspec*, (*yexprspec*), or (*funccspec*). See [Different ways of specifying predictions and their functions](#) in [BAYES] [Bayesian postestimation](#).



options	Description
* chains(_all   <i>numlist</i> )	specify which chains to use for computation; default is chains(_all)
* sepchains	compute results separately for each chain
<u>nolegend</u>	suppress table legend

\*Options chains() and sepchains are relevant only when option nchains() is used with bayesmh.

## Options

chains(\_all | *numlist*) specifies which chains from the MCMC sample to use for computation. The default is chains(\_all) or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option chains() is relevant only when option nchains() is specified with bayesmh.

sepchains specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option chains(). Option sepchains is relevant only when option nchains() is specified with bayesmh.

nolegend suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

## Remarks and examples

Remarks are presented under the following headings:

- Posterior predictive checks*
- PPPs*
- Nonlinear effect of labor and capital on companies' output*

### Posterior predictive checks

Posterior predictive checks, or model checks, are graphical and quantitative methods for comparing observed and replicated outcomes to assess goodness of fit of a Bayesian model. See Box (1980), Zellner (1975), West (1986), Gelman, Meng, and Stern (1996), and Gelman and Rubin (1992) for historical remarks and more in-depth discussions.

Replicated outcomes are outcome values that are simulated from the posterior predictive distribution using the observed covariate data; see Overview of Bayesian predictions and Methods and formulas in [BAYES] bayespredict. The distribution of replicated outcomes or its various summaries are compared with those of the observed outcomes. If they are similar, the Bayesian model is considered to fit the observed data well.

One of the graphical model checks uses quantile–quantile plots to compare observed and replicated residuals. These plots reveal misspecifications of the error distribution of a model. Histograms are commonly used to compare the distributions of the observed and replicated outcomes. More formally, the so-called PPPs, which we describe in the next section, are used to quantify the discrepancy between the summaries of the observed and replicated data.

## PPPs

The notion of a PPP was introduced by [Rubin \(1984\)](#) as a Bayesian version of the classical  $p$ -value. The role of  $p$ -values in classical hypothesis testing is to quantify the discrepancy between the observed sample and population quantities. Test statistics, which are scalar functions of a sample, are commonly used as discrepancy measures. The  $p$ -value is defined as the probability to obtain a value of the test statistic as or more extreme than its observed value if the null hypothesis is true. This probability is computed with respect to the sampling distribution of the test statistic.

In a Bayesian setting, the discrepancy between the model and the observed data is measured by test quantities, which are scalar functions of a sample and model parameters. A test statistic is a special case of a test quantity that depends only on the sample. The distribution of a test quantity is defined with respect to the posterior predictive distribution of the replicated data and posterior distribution of model parameters. A PPP (or a Bayesian  $p$ -value or a Bayesian predictive  $p$ -value) is then defined as the probability that a test quantity for the replicated data could be as or more extreme than for the observed data. You can think of a PPP as a classical  $p$ -value averaged over the posterior distribution ([Meng 1994](#)). For more information about PPPs, see [Tsui and Weerahandi \(1989\)](#), [Gelman, Meng, and Stern \(1996\)](#), and [Gelman et al. \(2014\)](#), among others.

One of the advantages of PPPs over their classical counterparts is that they automatically handle nuisance parameters by averaging over the posterior distribution of all model parameters. In contrast, classical  $p$ -values are conditional on fixed model parameters, typically MLEs. Also, PPPs are not defined conditional on the null hypothesis being true and can be viewed simply as probabilities of model misfit. Values of PPPs close to zero or one indicate lack of fit. For a well-fitting model, the PPP should, ideally, be close to 0.5, although values between 0.05 and 0.95 are often considered acceptable in the literature ([Gelman et al. 2014](#), 150; [Congdon 2010](#), sec. 2.5.2).

One criticism of PPPs is that their distribution under the correct model specification is generally not uniform (for example, [Bayarri and Berger \[2000\]](#) and [Robins, van der Vaart, and Ventura \[2000\]](#)). The distribution tends to be more concentrated around 0.5 when the model is correct. [Gelman \(2013\)](#) argues that this property may be desirable in some cases and discusses the cases when it is not desirable. The author concludes that although it is difficult to provide general recommendations for how best to interpret PPPs, he suggests that they are still useful in practice to discover systematic discrepancies between the observed data and the fitted model.

When you check model fit, it is important to consider different test quantities that describe various aspects of the distribution of the replicated data. Certain distinctive aspects of the assumed model distribution such as symmetry and weight of the tails are commonly used as test quantities. For example, for assumed normal errors, it is appropriate to test the skewness and kurtosis of replicated residuals and compare them with the skewness and kurtosis of a normal distribution. When you use test quantities, [Gelman \(2013\)](#) suggests to use “caution in interpreting diagnostics that strongly depend on parameters or latent data”. In addition to test quantities, you can use PPPs to compare individual observations, that is, compare the sample of replicated outcomes for a particular observation with the corresponding observed outcome value.

## Nonlinear effect of labor and capital on companies’ output

In this example, we show an application of PPPs to assess goodness of fit of a Bayesian model. We adapt an example described in [Koop \(2003, sec. 5.9\)](#) about the effect of labor and capital on companies’ production. The dataset, `coutput.dta`, includes data for 123 companies with variables `output`, `labor`, and `capital`. The variables are scaled.

```
. use https://www.stata-press.com/data/r16/coutput
(Company output data)

. describe
Contains data from https://www.stata-press.com/data/r16/coutput.dta
  obs:                123                Company output data
  vars:                3                 22 Feb 2019 13:24
                                      (_dta has notes)
```

variable name	storage type	display format	value label	variable label
output	float	%9.0g		Output
labor	float	%9.0g		Labor
capital	float	%9.0g		Capital

Sorted by:

[Koop \(2003\)](#) proposes the following nonlinear model for describing companies’ output:

$$\text{output}_i = \alpha + (\beta_1 \text{labor}_i^\lambda + \beta_2 \text{capital}_i^\lambda)^{1/\lambda} + \epsilon_i, \epsilon_i \sim_{\text{i.i.d.}} N(0, \sigma^2)$$

A nonlinear model ( $\lambda > 1$ ) is expected to provide a better fit for the data than the linear model ( $\lambda = 1$ ). We explore this by using posterior predictive checks.

Without concrete prior knowledge about the parameters  $\alpha$ ,  $\beta_1$ ,  $\beta_2$ , and  $\lambda$ , we specify weakly informative priors for them. We use the  $N(0, 100)$  prior for the coefficients, which is noninformative because the variables are scaled to be in the  $(0, 2)$  range. We apply exponential prior,  $\text{exp}(1)$ , for  $\lambda$  because  $\lambda$  is a positive parameter with 1 being a highly probable value for it. Below is the full model specification using `bayesmh`:

```
. bayesmh output =
> ({alpha}+({beta1}*labor^{lambda}+{beta2}*capital^{lambda})^(1/{lambda})),
> likelihood(normal({sig2}))
> prior({alpha beta1 beta2}, normal(0,100))
> prior({lambda}, exp(1)) prior({sig2}, igamma(0.1,0.1))
> init({alpha beta1 beta2 lambda} 1)
> saving(coutput_mcmc) mcmcsize(5000) rseed(16)
Burn-in ...
Simulation ...

Model summary
```

---

```
Likelihood:
  output ~ normal(<expr1>,{sig2})

Priors:
      {sig2} ~ igamma(0.1,0.1)
{alpha beta1 beta2} ~ normal(0,100)
      {lambda} ~ exponential(1)

Expression:
  expr1 : {alpha}+({beta1}*labor^{lambda}+{beta2}*capital^{lambda})^
> (1/{lambda})
```

---

```

Bayesian normal regression          MCMC iterations =      7,500
Random-walk Metropolis-Hastings sampling  Burn-in       =      2,500
                                         MCMC sample size =    5,000
                                         Number of obs   =     123
                                         Acceptance rate =     .2176
                                         Efficiency: min =     .02226
                                         avg             =     .03045
Log marginal-likelihood = 6.9478788    max             =     .03524

```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
alpha	1.028072	.0549225	.004364	1.028156	.9300813	1.137604
beta1	.6838483	.0990207	.007467	.6736414	.5037749	.8903975
beta2	.9578192	.140161	.013285	.9413369	.7197326	1.25467
lambda	1.270644	.2435384	.020624	1.255252	.8222015	1.78508
sig2	.0390367	.0052749	.000397	.038495	.0301147	.0503593

```
file coutput_mcmc.dta saved
```

We generated an MCMC sample of size 5,000 with an average efficiency of about 3%. `bayesmh` estimated the posterior mean of  $\lambda$ , 1.3, to be larger than 1, which implies that labor and capital do have a nonlinear effect on companies' output.

Model assumptions can be assessed through residual analysis. We follow [Koop \(2003\)](#) and use PPPs to compare various aspects of the distribution of residuals simulated from the fitted model and observed residuals. By construction, the distribution of the simulated residuals is  $N(0, \sigma^2)$ .

## ► Example 1: PPPs for simple test statistics

One simple check is to compare the means and variances of the simulated residuals,  $\mathbf{r}^{\text{sim}}$ , with those of the observed residuals,  $\mathbf{r}^{\text{obs}}$ . Let

$$T_1(\mathbf{y}) = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \text{ and } T_2(\mathbf{y}) = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$$

denote the mean and variance test statistics. We want to compare  $T_1(\mathbf{r}^{\text{sim}})$  with  $T_1(\mathbf{r}^{\text{obs}})$  and  $T_2(\mathbf{r}^{\text{sim}})$  with  $T_2(\mathbf{r}^{\text{obs}})$ .

We first use `bayespredict` to generate MCMC samples of means and variances of simulated and observed residuals.

```

. bayespredict (mean:@mean({_resid})) (var:@variance({_resid})),
> saving(coutput_pred) rseed(16)

Computing predictions ...
file coutput_pred.dta saved
file coutput_pred.ster saved

```

We used built-in Mata functions `mean()` and `variance()` to compute the means and variances; the Mata function specification is designated with `@`. We specified `{_resid}` as the argument to these functions to compute the means and variances of the simulated residuals. We labeled the resulting means as `mean` and variances as `var`; we can use these labels later within `bayesstats ppvalues` to refer to these prediction results. And we saved the simulated results in the prediction dataset, `coutput_pred.dta`. As we discussed in [Prediction dataset](#) of [\[BAYES\] bayespredict](#), the generated prediction dataset includes, among other variables, the `mean` variable containing 5,000 means of simulated residuals,  $\{T_1(\mathbf{r}^{\text{sim},1}), T_1(\mathbf{r}^{\text{sim},2}), \dots, T_1(\mathbf{r}^{\text{sim},5000})\}$ , where  $\mathbf{r}^{\text{sim},t}$  is the column vector

containing 123 residuals simulated from the fitted model using the  $t$ th set of MCMC estimates of model parameters. (We saved our MCMC estimates of model parameters in `couput_mcmc.dta` with `bayesmh`.) Additionally, `bayespredict` generated 5,000 means of the observed residuals,  $\{T_1(\mathbf{r}^{\text{obs},1}), T_1(\mathbf{r}^{\text{obs},2}), \dots, T_1(\mathbf{r}^{\text{obs},5000})\}$ , and stored them in the `_obs_mean` variable in the prediction dataset. Similarly, `bayespredict` generated variances of simulated and observed residuals and saved them in variables `var` and `_obs_var` in the prediction dataset. See [BAYES] `bayespredict` for details.

We can now access the simulated means and variances within `bayesstats ppvalues`. For example, we specify `{mean}` to compute PPPs for the mean test statistic. We also specify the prediction dataset, `coutput_pred.dta`, containing the simulated means in the required using specification.

```
. bayesstats ppvalues {mean} using coutput_pred
Posterior predictive summary   MCMC sample size =      5,000
```

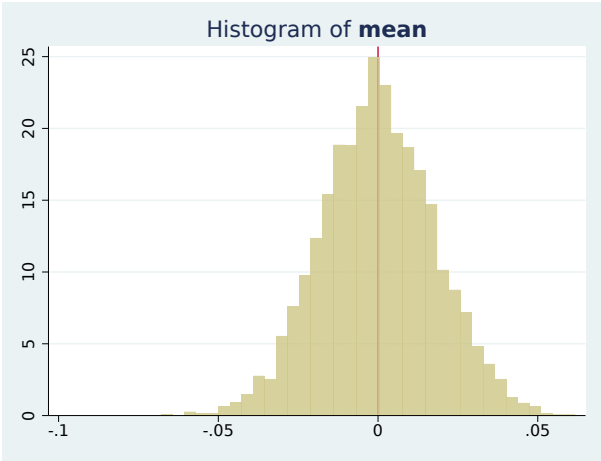
T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
mean	-.00007	.0177143	.0000147	.4978

Note: P(T>=T\_obs) close to 0 or 1 indicates lack of fit.

T and T\_obs denote the test statistics computed using the replicated data and observed data, respectively. In our example, T is  $T_1(\mathbf{r}^{\text{sim}})$  and T\_obs is  $T_1(\mathbf{r}^{\text{obs}})$ . The posterior mean estimate, Mean, of  $T_1(\mathbf{r}^{\text{sim}})$  from the MCMC sample of means of simulated residuals is  $-0.00007$ . The posterior mean estimate, E(T\_obs), of  $T_1(\mathbf{r}^{\text{obs}})$  from the MCMC sample of means of observed residuals is  $0.0000147$ . Both are close to zero. The estimated PPP is about 0.5, which indicates a perfect agreement between the two means. This  $p$ -value represents the proportion of times the mean of simulated residuals was greater than or equal to the mean of the observed residuals in the MCMC sample.

The PPP can be also visualized using a histogram with the reference line at the observed mean, which is essentially 0 in our example.

```
. bayesgraph histogram {mean} using coutput_pred, color(%75) xline(0)
```



The PPP is the area of the histogram to the right of the observed value, which is about 50% in our example.

As with the mean, we can compare the variances.

```
. bayesstats ppvalues {var} using coutput_pred
Posterior predictive summary   MCMC sample size =      5,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
var	.038952	.0073444	.03694	.5762

Note:  $P(T \geq T_{\text{obs}})$  close to 0 or 1 indicates lack of fit.

Posterior mean estimates of the variances of the simulated and observed residuals are similar and are close to the error variance `{sig2}` of 0.39, as estimated by `bayesmh`. The estimated PPP is 0.58 and again indicates very good agreement between the variances.

It is not surprising that the means and variances of simulated and observed residuals are in such good agreement. This tends to be true for many models in which the parameters directly model means and variances. In [example 3](#), we explore other discrepancy measures such as skewness and kurtosis.

◀

## ► Example 2: Specifying Mata functions directly with bayesstats ppvalues

In [example 1](#), we computed the mean and variance test statistics with `bayespredict`. Such specification is preferable with large datasets because it does not save a typically large sample of replicated outcomes. With moderate-sized datasets, you may save the replicated outcomes first and compute the functions within `bayesstats ppvalues`.

For example, here we simulate replicated outcomes using `bayespredict`. We replace the earlier `coutput_pred.dta` with new results.

```
. bayespredict {_ysim}, saving(coutput_pred, replace) rseed(16)
Computing predictions ...
file coutput_pred.dta saved
file coutput_pred.ster saved
```

In this case, the generated prediction dataset contains 5,000 MCMC replicates for each observation of our outcome output. That is, the dataset has 123 variables and 5,000 observations (and other auxiliary variables). We can now compute PPPs for any function of the replicated outcomes or their residuals.

We use the same specification of Mata functions with `bayesstats ppvalues` as we did with `bayespredict` in [example 1](#).

```
. bayesstats ppvalues (mean:@mean({_resid})) (var:@variance({_resid}))
> using coutput_pred
Posterior predictive summary   MCMC sample size =      5,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
mean	-.00007	.0177143	.0000147	.4978
var	.038952	.0073444	.03694	.5762

Note:  $P(T \geq T_{\text{obs}})$  close to 0 or 1 indicates lack of fit.

We obtain identical results to [example 1](#). Notice that we can combine various specifications in one call to `bayesstats ppvalues`.

`bayesstats ppvalues` used the replicated outcomes from `coutput_pred.dta` to compute the simulated residuals, which resulted in an intermediate sample of 5,000 MCMC residuals for each of the 123 observations. It then produced yet another intermediate sample of 5,000 means of the residuals over 123 observations. Finally, it used the sample of 5,000 means to compute the posterior predictive summaries as displayed in the output table. The command performed the same computations for the variances, `var`.

Using `bayespredict` to save the entire sample of replicated outcomes, whenever feasible, is convenient because you can explore various discrepancy measures without having to predefine them. However, there are two other advantages of the earlier specification, in addition to speed and storage efficiency. When you compute functions using `bayespredict`, you can specify expressions of these functions with `bayesstats ppvalues` (or other Bayesian postestimation commands). Also, you can compute your own functions within Stata programs and specify them with `bayespredict`, whereas the use of Stata programs is not allowed within `bayesstats ppvalues` and other Bayesian postestimation commands. But you can define your own Mata functions and use them with `bayesstats ppvalues`, as we demonstrate in the next example.

◀

### ► Example 3: PPPs for user-defined test statistics

Continuing with [example 2](#), we explore other discrepancy measures for the simulated and observed residuals. Given that we expect our residuals to be normally distributed when the model fits the data, we can explore their skewness and kurtosis.

Skewness and kurtosis are related to the third and fourth moments of a distribution. The skewness statistic measures the symmetry of a distribution about its mean. The kurtosis statistic measures the weight of the tails of a distribution. A normal distribution has skewness of 0 and kurtosis of 3.

There are no built-in Mata functions to compute these measures, so we need to define our own.

```
. mata:
_____ mata (type end to exit) _____
: real scalar skew(real colvector vresid) {
>     return (sqrt(length(vresid))*sum(vresid:^3)/(sum(vresid:^2)^1.5))
> }
: real scalar kurtosis(real colvector vresid) {
>     return (length(vresid)*sum(vresid:^4)/(sum(vresid:^2)^2) - 3)
> }
: end
```

Mata function `skew()` computes sample skewness, and `kurtosis()` computes sample kurtosis, but it subtracts 3 from the formula so that the kurtosis of a normally distributed sample is 0. Both functions accept a column vector of residuals as an argument and calculate and return the overall test statistic as a scalar.

We can now use these two functions to compute PPPs for skewness and kurtosis of residuals.

```
. bayesstats ppvalues (sy:@skew({_resid})) (ky:@kurtosis({_resid}))
> using coutput_pred
```

Posterior predictive summary MCMC sample size = 5,000

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
sy	.0014651	.3420932	.1763123	.3464
ky	-.0368386	.423227	-.3171961	.7304

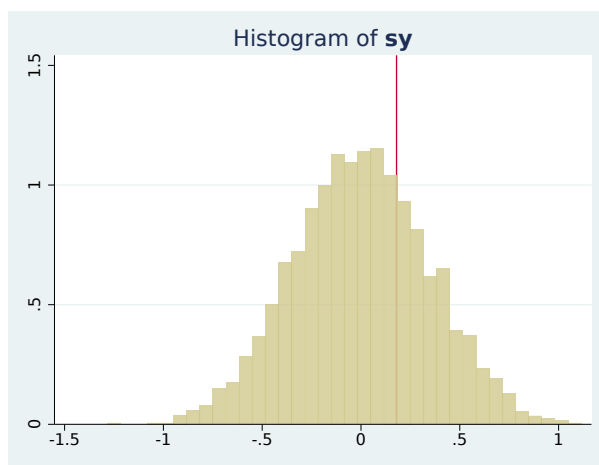
Note: P(T>=T\_obs) close to 0 or 1 indicates lack of fit.

The posterior mean estimates for the skewness and kurtosis of the observed residuals are not as close to zero as their counterparts simulated from the model. Nevertheless, according to the estimated PPPs of 0.35 for skewness and of 0.73 for kurtosis, the observed discrepancies can be explained by sampling variation. For instance, 35% of simulated skewnesses are greater than or equal to the observed skewnesses.

A PPP close to 0 or 1 indicates model misfit. Although there are no definitive recommendations, some literature suggests that PPPs less than 0.05 or larger than 0.95 be considered indicative of lack of fit (Gelman et al. 2014). However, it is important to consider PPPs in the context of your research question, such as whether the observed discrepancy is practically meaningful.

To visualize once again the PPP, we can plot the histogram of the simulated skewness with the reference line at the expected observed value of 0.18.

```
. bayesgraph histogram (sy:@skew({_resid})) using coutput_pred, xline(0.18)
> color(%70)
```



About 35% of the histogram area is on the right of the mean observed skewness of 0.18.

◀

In conclusion, our residual analysis revealed good agreement between the simulated and observed residuals with respect to several test statistics. Therefore, there do not appear to be any violations of the normality assumption for the error terms in the model.



We want to emphasize the importance of the choice of test statistics when assessing model fit. You should avoid using sufficient statistics such as sample mean and variance, which are usually well behaved because they are often directly modeled by parameters. Instead, you should focus on statistics that measure more specific distribution properties such as quantiles, skewness, kurtosis, maximum and minimum, and more.

## Stored results

`bayesstats ppvalues` stores the following in `r()`:

Scalars	
<code>r(mcmcsize)</code>	MCMC sample size used in the computation
<code>r(nchains)</code>	number of chains used in the computation
Macros	
<code>r(names)</code>	names of model parameters and expressions
<code>r(expr_#)</code>	#th expression
<code>r(exprnames)</code>	expression labels
<code>r(chains)</code>	chains used in the computation, if <code>chains()</code> is specified
Matrices	
<code>r(summary)</code>	matrix with predictive statistics for parameters in <code>r(names)</code>
<code>r(summary_chain#)</code>	matrix <code>summary</code> for chain #, if <code>sepchains</code> is specified

## Methods and formulas

See [Methods and formulas](#) of [\[BAYES\] bayespredict](#) for general definitions and for formulas related to replicated outcomes,  $\mathbf{y}^{\text{rep}}$ .

Let  $T_q(\mathbf{y}, \boldsymbol{\theta})$  be a test quantity. The PPP,  $q(T_q)$ , is defined as the probability that  $T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta})$  is greater than or equal to the observed  $T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})$  ([Rubin 1984](#), [Gelman et al. 2014](#)). Specifically,

$$\begin{aligned} q(T_q) &= \Pr \{ T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta}) \geq T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta}) | \mathbf{y}^{\text{obs}}, X^{\text{obs}} \} \\ &= \int \int 1_{T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta}) \geq T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})} p(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta} | \mathbf{y}^{\text{obs}}, X^{\text{obs}}) d\mathbf{y}^{\text{rep}} d\boldsymbol{\theta} \\ &= \int \int 1_{T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta}) \geq T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})} p(\mathbf{y}^{\text{rep}} | \boldsymbol{\theta}, X^{\text{obs}}) p(\boldsymbol{\theta} | \mathbf{y}^{\text{obs}}, X^{\text{obs}}) d\mathbf{y}^{\text{rep}} d\boldsymbol{\theta} \end{aligned}$$

and  $1_{(A)}$  is an indicator function of  $A$  being true.

In practice, the joint posterior distribution  $p(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta} | \mathbf{y}^{\text{obs}}, X^{\text{obs}})$  is not available. Instead, we have a simulated sample  $\{(\mathbf{y}^{\text{rep},1}, \boldsymbol{\theta}^1), (\mathbf{y}^{\text{rep},2}, \boldsymbol{\theta}^2), \dots, (\mathbf{y}^{\text{rep},T}, \boldsymbol{\theta}^T)\}$ , where  $T$  is the MCMC sample size. Then  $q(T_q)$  is estimated as

$$\widehat{q}(T_q) = \frac{1}{T} \sum_{t=1}^T 1_{T_q(\mathbf{y}^{\text{rep},t}, \boldsymbol{\theta}^t) \geq T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta}^t)}$$

bayesstats ppvalues reports  $\widehat{q}(T_q)$  in the output table and labels it  $P(T \geq T_{\text{obs}})$ . bayesstats ppvalues also reports the average observed test quantity,  $E(T_{\text{obs}})$ ,

$$\widehat{E}\{T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})\} = \frac{1}{T} \sum_{t=1}^T T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta}^t)$$

and the sample mean and standard deviation of the sample of replicated test quantities,  $\{T_q(\mathbf{y}^{\text{rep},1}, \boldsymbol{\theta}^1), T_q(\mathbf{y}^{\text{rep},2}, \boldsymbol{\theta}^2), \dots, T_q(\mathbf{y}^{\text{rep},T}, \boldsymbol{\theta}^T)\}$ .

For a special case of test statistics,  $T_q(\mathbf{y}, \boldsymbol{\theta}) = T_s(\mathbf{y})$ , the above formulas simplify correspondingly.

M. J. Bayarri (1956–2014) was born in Valencia, Spain. She received a bachelor's, master's, and doctorate degree in mathematics, all from the University of Valencia. She began as an assistant professor and then became a full professor at her alma mater.

Bayarri won a Fulbright fellowship to attend Carnegie Mellon University in 1984, which marked the beginning of routine visits to the United States. She became a visiting professor at Purdue University, an adjunct professor at Duke University, and leader of the research program at the Statistical and Applied Mathematical Sciences Institute (SAMSI). She coauthored books on Bayesian statistics and biostatistics, and coauthored numerous research articles, including some award-winning papers. Her main areas of research included selection models, weighted distributions, and Bayesian analysis of queuing systems.

Aside from her published contributions, she held multiple leadership roles. For example, Bayarri served as President of the International Society for Bayesian Analysis (ISBA) and as the principal investigator of Biostatnet, a network of biostatistical researchers. Her critical skills shined as Coordinating Editor of the *Journal of Statistical Planning and Inference* and as an award-winning food critic. In 1997, she was elected as a fellow of the American Statistical Association, and in 2009, she was elected as a fellow of the Institute of Mathematical Statistics.

## References

- Bayarri, M. J., and J. O. Berger. 2000. P values for composite null models. *Journal of the American Statistical Association* 95: 1127–1142.
- Box, G. E. P. 1980. Sampling and Bayes' inference in scientific modelling and robustness. *Journal of the Royal Statistical Society, Series A* 143: 383–404.
- Congdon, P. D. 2010. *Applied Bayesian Hierarchical Methods*. Boca Raton, FL: CRC Press.
- Gelman, A. 2013. Two simple examples for understanding posterior p-values whose distributions are far from uniform. *Electronic Journal of Statistics* 7: 2595–2602.
- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Gelman, A., X.-L. Meng, and H. Stern. 1996. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sinica* 6: 733–760.
- Gelman, A., and D. B. Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical Science* 7: 457–472.
- Koop, G. 2003. *Bayesian Econometrics*. Chichester, UK: Wiley.
- Meng, X.-L. 1994. Multiple-imputation inferences with uncongenial sources of input (with discussion). *Statistical Science* 9: 538–573.
- Robins, J. M., A. W. van der Vaart, and V. Ventura. 2000. Asymptotic distribution of p values in composite null models. *Journal of the American Statistical Association* 95: 1143–1156.

- Rubin, D. B. 1984. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *Annals of Statistics* 12: 1151–1172.
- Tsui, K.-W., and S. Weerahandi. 1989. Generalized p-values in significance testing of hypotheses in the presence of nuisance parameters. *Journal of the American Statistical Association* 84: 602–607.
- West, M. 1986. Bayesian model monitoring. *Journal of the Royal Statistical Society, Series B* 48: 70–78.
- Zellner, A. 1975. Bayesian analysis of regression error terms. *Journal of the American Statistical Association* 70: 138–144.

## Also see

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **bayespredict** — Bayesian predictions

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

## Description

`bayesstats summary` calculates and reports posterior summary statistics for model parameters and functions of model parameters using current Bayesian estimation results. Posterior summary statistics include posterior means, posterior standard deviations, MCMC standard errors (MCSE), posterior medians, and equal-tailed credible intervals or highest posterior density (HPD) credible intervals.

## Quick start

Posterior summaries for all model parameters after a Bayesian regression model

```
bayesstats summary
```

As above, but only for parameters `{y:x1}` and `{y:x2}`

```
bayesstats summary {y:x1} {y:x2}
```

Same as above

```
bayesstats summary {y:x1 x2}
```

Posterior summaries for elements 1,1 and 2,1 of matrix parameter `{S}`

```
bayesstats summary {S_1_1 S_2_1}
```

Posterior summaries for all elements of matrix parameter `{S}`

```
bayesstats summary {S}
```

Posterior summaries with HPD instead of equal-tailed credible intervals and with credible level of 90%

```
bayesstats summary, hpd clevel(90)
```

Posterior summaries with MCSE calculated using batch means

```
bayesstats summary, batch(100)
```

Posterior summaries for functions of scalar model parameters

```
bayesstats summary ({y:x1}--{y:_cons}) (sd:sqrt({var}))
```

Posterior summaries for the log-likelihood and log-posterior functions

```
bayesstats summary _loglikelihood _logposterior
```

Posterior summaries for selected model parameters and functions of model parameters and for log-likelihood and log-posterior functions using abbreviated syntax

```
bayesstats summary {var} ({y:x1}--{y:_cons}) _ll _lp
```

Posterior summaries of the simulated outcome

```
bayespredict {_ysim}, saving(predres)
bayesstats summary {_ysim} using predres
```

Posterior summaries of the mean across observations of the simulated outcome labeled as `mymean`  
`bayesstats summary (mymean: @mean({_ysim})) using predres`

## Menu

Statistics > Bayesian analysis > Summary statistics

## Syntax

Syntax is presented under the following headings:

[Summary statistics for model parameters](#)

[Summary statistics for predictions](#)

### Summary statistics for model parameters

*Summary statistics for all model parameters*

```
bayesstats summary [, options showreffects[(reref)]]
```

```
bayesstats summary _all [, options showreffects[(reref)]]
```

*Summary statistics for selected model parameters*

```
bayesstats summary paramspec [, options]
```

*Summary statistics for expressions of model parameters*

```
bayesstats summary exprspec [, options]
```

*Summary statistics of log-likelihood or log-posterior functions*

```
bayesstats summary _loglikelihood|_logposterior [, options]
```

*Full syntax*

```
bayesstats summary spec [spec ...] [, options]
```

*paramspec* can be one of the following:

`{eqname:param}` refers to a parameter *param* with equation name *eqname*;

`{eqname:}` refers to all model parameters with equation name *eqname*;

`{eqname:paramlist}` refers to parameters with names in *paramlist* and with equation name *eqname*;  
or

`{param}` refers to all parameters named *param* from all equations.

In the above, *param* can refer to a matrix name, in which case it will imply all elements of this matrix. See [Different ways of specifying model parameters](#) in [BAYES] [Bayesian postestimation](#) for examples.

*exprspec* is an optionally labeled expression of model parameters specified in parentheses:

`([exprlabel: ]expr)`

*exprlabel* is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See [Specifying functions of model parameters](#) in [BAYES] [Bayesian postestimation](#) for examples.

`_loglikelihood` and `_logposterior` also have respective synonyms `_ll` and `_lp`.

*spec* is one of [paramspec](#), [exprspec](#), `_loglikelihood` (or `_ll`), or `_logposterior` (or `_lp`).

## Summary statistics for predictions

*Summary statistics for simulated outcomes, residuals, and more*

`bayesstats summary yspec [yspec ...] using predfile [, options]`

*Summary statistics for expressions of simulated outcomes, residuals, and more*

`bayesstats summary (yexprspec) [(yexprspec) ...] using predfile [, options]`

*Summary statistics for Mata functions of simulated outcomes, residuals, and more*

`bayesstats summary (funspec) [(funspec) ...] using predfile [, options]`

*Full syntax*

`bayesstats summary predspec [predspec ...] using predfile [, options]`

*predfile* is the name of the dataset created by [bayespredict](#) that contains prediction results.

*yspec* is `{ysimspec | residspec | muspec | label}`.

*ysimspec* is `{_ysim#}` or `{_ysim#[numlist]}`, where `{_ysim#}` refers to all observations of the #th simulated outcome and `{_ysim#[numlist]}` refers to the selected observations, *numlist*, of the #th simulated outcome. `{_ysim}` is a synonym for `{_ysim1}`.

*residspec* is `{_resid#}` or `{_resid#[numlist]}`, where `{_resid#}` refers to all residuals of the #th simulated outcome and `{_resid#[numlist]}` refers to the selected residuals, *numlist*, of the #th simulated outcome. `{_resid}` is a synonym for `{_resid1}`.

*muspec* is `{_mu#}` or `{_mu#[numlist]}`, where `{_mu#}` refers to all expected values of the #th outcome and `{_mu#[numlist]}` refers to the selected expected values, *numlist*, of the #th outcome. `{_mu}` is a synonym for `{_mu1}`.

*label* is the name of the function simulated using [bayespredict](#).

With large datasets, specifications `{_ysim#}`, `{_resid#}`, and `{_mu#}` may use a lot of time and memory and should be avoided. See [Generating and saving simulated outcomes](#) in [BAYES] [bayespredict](#).

*yexprspec* is `[exprlabel: ]yexpr`, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, `{label}`.

*funcspec* is `[label:]@func(arg1[, arg2])`, where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of `{_ysim[#]}`, `{_resid[#]}`, or `{_mu[#]}`. *arg2* is primarily for use with user-defined Mata functions; see [Defining test statistics using Mata functions](#) in [BAYES] **bayespredict**.  
*predspec* is one of *yspec*, (*yexprspec*), or (*funcspec*). See [Different ways of specifying predictions and their functions](#) in [BAYES] **Bayesian postestimation**.

options	Description
Main	
<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>*chains(_all   numlist)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
<code>*sepchains</code>	compute results separately for each chain
<code>skip(#)</code>	skip every # observations from the MCMC sample; default is <code>skip(0)</code>
<code>nolegend</code>	suppress table legend
<code>display_options</code>	control spacing, line width, and base and empty cells
Advanced	
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Options `chains()` and `sepchains` are relevant only when option `nchains()` is used with **bayesmh** or the **bayes** prefix.

## Options

### Main

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is `clevel(95)` or as set by [BAYES] **set clevel**.

`hpd` displays the HPD credible intervals instead of the default equal-tailed credible intervals.

`batch(#)` specifies the length of the block for calculating batch means and MCSE using batch means. The default is `batch(0)`, which means no batch calculations. When `batch()` is not specified, MCSE is computed using effective sample sizes instead of batch means. Option `batch()` may not be combined with `corrlag()` or `corrtol()`.

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with **bayesmh** or the **bayes** prefix.

`sepchains` specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option `chains()`. Option `sepchains` is relevant only when option `nchains()` is specified with **bayesmh** or the **bayes** prefix.

`showeffects` and `showeffects(reref)` are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time.

`skip(#)` specifies that every `#` observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of `#+1`. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

*display\_options*: `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fwrapon(style)`, and `nolstretch`; see [\[R\] Estimation options](#).

#### Advanced

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is `min{500, mcmcsize()/2}`. The total autocorrelation is computed as the sum of all lag- $k$  autocorrelation values for  $k$  from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`. Options `corrlag()` and `batch()` may not be combined.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag- $k$  autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the  $k$ th lag are discarded. Options `corrtol()` and `batch()` may not be combined.

## Remarks and examples

Remarks are presented under the following headings:

[Introduction](#)

[Bayesian summaries for an auto data example](#)

## Introduction

`bayesstats summary` reports posterior summary statistics for model parameters and their functions using the current [Bayesian estimation](#) results. When typed without arguments, the command displays results for all model parameters. Alternatively, you can specify a subset of model parameters following the command name; see [Different ways of specifying model parameters](#) in [\[BAYES\] Bayesian postestimation](#). You can also obtain results for scalar functions of model parameters; see [Specifying functions of model parameters](#) in [\[BAYES\] Bayesian postestimation](#).

Sometimes, it may be useful to obtain posterior summaries of log-likelihood and log-posterior functions. This can be done by specifying `_loglikelihood` and `_logposterior` (or the respective synonyms `_ll` and `_lp`) following the command name.

You can also obtain the posterior summaries for prediction quantities when you specify the prediction dataset in the `using` specification; see [Different ways of specifying predictions and their functions](#) in [\[BAYES\] Bayesian postestimation](#) for how to specify prediction quantities with `bayesstats summary`.



`bayesstats summary` reports the following posterior summary statistics: posterior mean, posterior standard deviation, MCMC standard error, posterior median, and equal-tailed credible intervals or, if the `hpd` option is specified, HPD credible intervals. The default credible level is set to 95%, but you can change this by specifying the `clevel()` option. Equal-tailed and HPD intervals may produce very different results for asymmetric or highly skewed marginal posterior distributions. The HPD intervals are preferable in this situation.

You should not confuse the term “HPD interval” with the term “HPD region”. A  $\{100 \times (1 - \alpha)\}\%$  HPD interval is defined such that it contains  $\{100 \times (1 - \alpha)\}\%$  of the posterior density. A  $\{100 \times (1 - \alpha)\}\%$  HPD region also satisfies the condition that the density inside the region is never lower than that outside the region. For multimodal univariate marginal posterior distributions, the HPD regions may include unions of nonintersecting HPD intervals. For unimodal univariate marginal posterior distributions, HPD regions are indeed simply HPD intervals. The `bayesstats summary` command thus calculates HPD intervals assuming unimodal marginal posterior distributions (Chen and Shao 1999).

Some authors use the term “posterior intervals” instead of “credible intervals” and the term “central posterior intervals” instead of “equal-tailed credible intervals” (for example, Gelman et al. [2014]).

**Bayesian summaries for an auto data example**

Recall our analysis of `auto.dta` from [example 4](#) in [\[BAYES\] bayesmh](#) using the mean-only normal model for `mpg` with a noninformative prior.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys
```

---

Bayesian normal regression	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	74
	Acceptance rate	=	.2668
	Efficiency: min	=	.09718
	avg	=	.1021
	max	=	.1071

---

Log marginal-likelihood =	-234.645
---------------------------	----------

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

## ► Example 1: Summaries for all parameters

If we type `bayesstats summary` without arguments after the `bayesmh` command, we will obtain the same summary table as reported by `bayesmh`.

```
. bayesstats summary
```

Posterior summary statistics					MCMC sample size = 10,000	
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

The posterior mean of `{mpg:_cons}` is 21.29 and of `{var}` is 34.8. They are close to their respective frequentist analogs (the sample mean of `mpg` is 21.297, and the sample variance is 33.47), because we used a noninformative prior. Posterior standard deviations are 0.68 for `{mpg:_cons}` and 5.92 for `{var}`, and they are comparable to frequentist standard errors under this noninformative prior. The standard error estimates of the posterior means, MCSEs, are low. For example, MCSE is 0.022 for `{mpg:_cons}`. This means that the precision of our estimate is, up to one decimal point, 21.3 provided that MCMC converged. The posterior means and medians of `{mpg:_cons}` are close, which suggests that the posterior distribution for `{mpg:_cons}` may be symmetric. According to the credible intervals, we are 95% certain that the posterior mean of `{mpg:_cons}` is roughly between 20 and 23 and that the posterior mean of `{var}` is roughly between 25 and 48. We can infer from this that `{mpg:_cons}` is greater than, say, 15, and that `{var}` is greater than, say, 20, with a very high probability. (We can use [\[BAYES\] bayestest interval](#) to compute the actual probabilities.)

The above is also equivalent to typing

```
. bayesstats summary {mpg:_cons} {var}
(output omitted)
```

◀

## ► Example 2: Credible intervals

By default, `bayesstats summary` reports 95% equal-tailed credible intervals. We can change the default credible level by specifying the `clevel()` option.

```
. bayesstats summary, clevel(90)
```

Posterior summary statistics					MCMC sample size = 10,000	
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [90% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	20.18807	22.44172
var	34.76572	5.91534	.180754	34.18391	26.28517	44.81732

As expected, 90% credible intervals are more narrow.

To calculate and report HPD intervals, we specify the `hpd` option.

```
. bayesstats summary, hpd
```

Posterior summary statistics

MCMC sample size = 10,000

		Mean	Std. Dev.	MCSE	Median	HPD [95% Cred. Interval]	
mpg	_cons	21.29222	.6828864	.021906	21.27898	19.94985	22.54917
	var	34.76572	5.91534	.180754	34.18391	24.34876	46.12339

The posterior distribution of `{mpg:_cons}` is symmetric about the posterior mean; thus there is little difference between the 95% equal-tailed credible interval from [example 1](#) and this 95% HPD credible interval for `{mpg:_cons}`. The 95% HPD interval for `{var}` has a smaller width than the corresponding equal-tailed interval in [example 1](#).

➤ **Example 3: Batch-means estimator**

`bayesstats summary` provides two estimators for MCSE: effective-sample-size and batch-means. Estimation using effective sample sizes is the default. You can use the `batch(#)` option to request the batch-means estimator, where `#` is the batch size. The optimal batch size depends on the autocorrelation in the MCMC sample. For example, if we observe that the autocorrelation for the parameters of interest is negligible after lag 100, we can specify `batch(100)` to estimate MCSE.

In our example, autocorrelation dies out after about lag 10 (see, for example, [Autocorrelation plots](#) in [\[BAYES\] bayesgraph](#) and [example 1](#) in [\[BAYES\] bayesstats ess](#)), so we use 10 as our batch size:

```
. bayesstats summary, batch(10)
```

Posterior summary statistics

MCMC sample size = 10,000  
Batch size = 10

		Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	_cons	21.29222	.6828864	.015315	21.27898	19.99152	22.61904
	var	34.76572	5.91534	.135295	34.18391	24.9129	47.61286

Note: Mean and MCSE are estimated using batch means.

The batch-means MCSE estimates are somewhat smaller than those obtained by default using effective sample sizes.

Use caution when choosing the batch size for the batch-means method. For example, if you use the batch size of 1, you will obtain MCSE estimates under the assumption that the draws in the MCMC sample are independent, which is not true.

### Example 4: Subsampling or thinning the chain

You can reduce correlation between MCMC draws by thinning or subsampling the MCMC chain. You can use the `skip(#)` option to skip every # observations from the MCMC sample, which is equivalent to a thinning interval of # + 1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, `bayesstats summary` will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, `bayesstats summary` will skip every two observations in the sample and will use only observations 1, 4, 7, and so on in the computation. By default, no observations are skipped—`skip(0)`. Note that `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by `bayesmh`'s `thinning()` option. It discards only selected observations from the computation and leaves the original sample unmodified.

```
. bayesstats summary, skip(9)
note: skipping every 9 sample observations; using observations 1,11,21,...
Posterior summary statistics                                MCMC sample size =      1,000
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29554	.6813796	.029517	21.27907	19.98813	22.58582
var	34.7396	5.897313	.206269	33.91782	24.9554	48.11452

We selected to skip every 9 observations, which led to a significant reduction of the MCMC sample size and thus increased our standard deviations. In some cases, with larger MCMC sample sizes, subsampling may decrease standard deviations because of the decreased autocorrelation in the reduced MCMC sample.

### Example 5: Summaries for expressions of model parameters

`bayesstats summary` accepts expressions to provide summaries of functions of model parameters. For example, we can use expression `(sd:sqrt({var}))` with a label, `sd`, to summarize the standard deviation of `mpg` in addition to the variance.

```
. bayesstats summary (sd:sqrt({var})) {var}
Posterior summary statistics                                MCMC sample size =     10,000
sd : sqrt({var})
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
sd	5.87542	.4951654	.014972	5.846701	4.991282	6.900207
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

Expressions can also be used for calculating posterior probabilities, although this can be more easily done using `bayestest interval` (see [BAYES] [bayestest interval](#)). For illustration, let's verify the probability that `{var}` is within the endpoints of the reported credible interval, indeed 0.95.

```
. bayesstats summary (prob:{var}>24.913 & {var}<47.613)
Posterior summary statistics                                MCMC sample size =    10,000
      prob : {var}>24.913 & {var}<47.613
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
prob	.9502	.2175424	.005301	1	0	1



➤ Example 6: Summaries for log likelihood and log posterior

We can use reserved names `_loglikelihood` (or the synonym `_ll`) and `_logposterior` (or the synonym `_lp`) to obtain summaries of the log likelihood and log posterior for the simulated MCMC sample.

```
. bayesstats summary _ll _lp
Posterior summary statistics                                MCMC sample size =    10,000
      _ll : _loglikelihood
      _lp : _logposterior
```

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ll	-235.4162	.990654	.032232	-235.1379	-238.1236	-234.4345
_lp	-238.9507	1.037785	.034535	-238.6508	-241.7889	-237.9187



➤ Example 7: Summaries for predicted outcomes

We continue our series of examples by computing summaries for Bayesian predictions. Let's generate Bayesian predictions of `mpg` and summarize them.

We use `bayespredict` to simulate outcome values for `mpg` for the first 10 observations from the fitted `bayesmh` model. To use `bayespredict`, we must first save the simulation results from `bayesmh` in a Stata dataset, `autosim.dta`. We then use `bayespredict` to save the prediction results in the dataset `mpgreps.dta`.

```
. bayesmh, saving(autosim)
note: file autosim.dta saved
. bayespredict {_ysim[1/10]}, saving(mpgreps) rseed(16)
Computing predictions ...
file mpgreps.dta saved
file mpgreps.ster saved
```

We can now summarize the prediction results by using `bayesstats summary`. We specify the prediction quantity we wish to summarize, the simulated outcome `{_ysim}` in our example, and the prediction dataset, `mpgreps.dta`, which contains the prediction quantity, in the `using` specification.

```
. bayesstats summary {_ysim} using mpgreps
```

Posterior summary statistics					MCMC sample size = 10,000	
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ysim1_1	21.24878	6.018783	.062648	21.23939	9.444973	33.07051
_ysim1_2	21.2539	5.944206	.060415	21.21638	9.421932	32.90605
_ysim1_3	21.3256	5.910363	.061595	21.31499	9.801655	33.02746
_ysim1_4	21.40651	5.963456	.059479	21.45933	9.794156	33.39388
_ysim1_5	21.19781	5.926335	.061197	21.26437	9.759916	32.80291
_ysim1_6	21.34776	5.94413	.059441	21.32314	9.771529	33.30251
_ysim1_7	21.34043	5.898474	.058985	21.34119	9.821613	33.07709
_ysim1_8	21.25329	5.957051	.05886	21.26176	9.476474	32.96236
_ysim1_9	21.25284	5.866096	.05962	21.3052	9.714165	32.82636
_ysim1_10	21.3464	5.931401	.060853	21.30528	9.670334	33.10769

bayesstats summary reports posterior summaries for all simulated outcomes in the prediction dataset, mpgreps.dta. Estimated posterior means and standard deviations are similar to the corresponding observed values for mpg, 21.30 and 5.79, respectively.

We can specifically examine the first observation of the replicated sample, {\_ysim\_1}, and compare it with the observed value, mpg[1], of 22.

```
. bayesstats summary ({_ysim_1}>='mpg[1]') using mpgreps
```

Posterior summary statistics

MCMC sample size = 10,000

expr1 : \_ysim1\_1>=22

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
expr1	.4479	.497303	.004973	0	0 1

We find that 45% of the replicates of mpg[1] are greater than 22. The reported probability of 0.45 is known as the posterior predictive  $p$ -value and is used for goodness-of-fit checking; see [\[BAYES\] bayesstats ppvalues](#).



## Stored results

bayesstats summary stores the following in r():

### Scalars

r(mcmcsz)	MCMC sample size used in the computation
r(clevel)	credible interval level
r(hpd)	1 if hpd is specified, 0 otherwise
r(batch)	batch length for batch-means calculations
r(skip)	number of MCMC observations to skip in the computation; every r(skip) observations are skipped
r(corrlag)	maximum autocorrelation lag
r(corrtol)	autocorrelation tolerance
r(nchains)	number of chains used in the computation

### Macros

r(names)	names of model parameters and expressions
r(expr_#)	#th expression
r(exprnames)	expression labels
r(chains)	chains used in the computation, if chains() is specified

### Matrices

<code>r(summary)</code>	matrix with posterior summaries statistics for parameters in <code>r(names)</code>
<code>r(summary_chain#)</code>	matrix <code>summary</code> for chain <code>#</code> , if <code>sepchains</code> is specified

## Methods and formulas

Methods and formulas are presented under the following headings:

[Point estimates](#)  
[Credible intervals](#)

Most of the summary statistics employed in Bayesian analysis are based on the marginal posterior distributions of individual model parameters or functions of model parameters.

Let  $\theta$  be a scalar model parameter and  $\{\theta_t\}_{t=1}^T$  be an MCMC chain of size  $T$  drawn from the marginal posterior distribution of  $\theta$ . For a function  $g(\theta)$ , substitute  $\{\theta_t\}_{t=1}^T$  with  $\{g(\theta_t)\}_{t=1}^T$  in the formulas below. If  $\theta$  is a covariance matrix model parameter, the formulas below are applied to each element of the lower-diagonal portion of  $\theta$ .

### Point estimates

Marginal posterior moments are approximated using the Monte Carlo integration applied to the simulated samples  $\{\theta_t\}_{t=1}^T$ .

Sample posterior mean and sample posterior standard deviation are defined as follows,

$$\hat{\theta} = \frac{1}{T} \sum_{t=1}^T \theta_t, \quad \hat{s}^2 = \frac{1}{T-1} \sum_{t=1}^T (\theta_t - \hat{\theta})^2$$

where  $\hat{\theta}$  and  $\hat{s}^2$  are sample estimators of the population posterior mean  $E(\theta_t)$  and posterior variance  $\text{Var}(\theta_t)$ .

With multiple chains, the posterior mean and standard deviation are estimated using the combined sample of all chains or of those that are requested in the `chains()` option as follows. Let  $\{\theta_{jt}\}_{t=1}^T$  be the  $j$ th Markov chain,  $j = 1, \dots, M$ , with sample mean  $\hat{\theta}_j$  and variance  $\hat{s}_j^2$ . The overall sample posterior mean is

$$\hat{\theta} = \frac{1}{MT} \sum_{j=1}^M \sum_{t=1}^T \theta_{jt}$$

and equals the average of the sample means of individual chains. Let  $B$  and  $W$  be the respective between-chains and within-chain variances

$$B = \frac{T}{M-1} \sum_{j=1}^M (\hat{\theta}_j - \hat{\theta})^2, \quad W = \frac{1}{M} \sum_{j=1}^M \hat{s}_j^2$$

The estimator of the posterior variance is

$$\hat{s}^2 = \frac{T-1}{T} W + \frac{1}{T} B \tag{1}$$

When the chains are strongly stationary,  $\widehat{s}^2$  is an unbiased estimator of the marginal posterior variance of  $\theta$  (Gelman et al. 2014, sec. 11.4).

The precision of the sample posterior mean is evaluated by its standard error, also known as the Monte Carlo standard error (MCSE). Note that MCSE cannot be estimated using the classical formula for the standard error,  $\widehat{s}/\sqrt{T}$ , because of the dependence between  $\theta_t$ 's.

Let

$$\sigma^2 = \text{Var}(\theta_t) + 2 \sum_{k=1}^{\infty} \text{Cov}(\theta_t, \theta_{t+k})$$

Then,  $\sqrt{T} \times \text{MCSE}$  approaches  $\sigma$  asymptotically in  $T$ .

`bayesstats summary` provides two different approaches for estimating MCSE. Both approaches try to adjust for the existing autocorrelation in the MCMC sample. The first one uses the so-called effective sample size (ESS), and the second one uses batch means (Roberts 1996; Jones et al. 2006).

The ESS-based estimator for MCSE, the default in `bayesstats summary`, is given by

$$\text{MCSE}(\widehat{\theta}) = \widehat{s}/\sqrt{\text{ESS}}$$

ESS is defined as

$$\text{ESS} = T / (1 + 2 \sum_{k=1}^{\text{max\_lags}} \rho_k)$$

where  $\rho_k$  is the lag- $k$  autocorrelation, and `max_lags` is the maximum number less than or equal to  $\rho_{\text{lag}}$  such that for all  $k = 1, \dots, \text{max\_lags}$ ,  $|\rho_k| > \rho_{\text{tol}}$ , where  $\rho_{\text{lag}}$  and  $\rho_{\text{tol}}$  are specified in options `corrlag()` and `corrtol()` with the respective default values of 500 and 0.01.  $\rho_k$  is estimated as  $\gamma_k/\gamma_0$ , where

$$\gamma_k = \frac{1}{T} \sum_{t=1}^{T-k} (\theta_t - \widehat{\theta})(\theta_{t+k} - \widehat{\theta})$$

is the lag- $k$  empirical autocovariance.

With multiple chains, the overall ESS is given by the sum of the effective sample sizes of individual chains. The MCSE is then calculated using the formula

$$\text{MCSE}(\widehat{\theta}) = \widehat{s} / \sqrt{\sum_{j=1}^M \text{ESS}_j}$$

where  $\widehat{s}$  is computed using (1) and  $\text{ESS}_j$  is the effective sample size of the  $j$ th chain.

The batch-means estimator of MCSE is obtained as follows. For a given batch of length  $b$ , the initial MCMC chain is split into  $m$  batches of size  $b$ ,

$$\{\theta_{j'+1}, \dots, \theta_{j'+b}\} \{\theta_{j'+b+1}, \dots, \theta_{j'+2b}\} \dots \{\theta_{T-b+1}, \dots, \theta_T\}$$

where  $j' = T - m \times b$  and  $m$  batch means  $\widehat{\mu}_1, \dots, \widehat{\mu}_m$  are calculated as sample means of each batch.  $m$  is chosen as the maximum number such that  $m \times b \leq T$ . If  $m$  is not a divisor of  $T$ , the first  $T - m \times b$  observations of the sample are not used in the batch-means computation. The batch-means estimator of the posterior variance,  $\widehat{s}_{\text{batch}}^2$ , is based on the assumption that  $\widehat{\mu}_j$ s are much less correlated than the original sample draws.

The batch-means estimator of the posterior mean is

$$\widehat{\theta}_{\text{batch}} = \frac{1}{m} \sum_{j=1}^m \widehat{\mu}_j$$



We have  $\hat{\theta}_{\text{batch}} = \hat{\theta}$ , whenever  $m \times b = T$ . Under the assumption that the batch means are uncorrelated,  $\hat{s}_{\text{batch}}^2 = \{1/(m-1)\} \sum_{j=1}^m (\hat{\mu}_j - \hat{\theta}_{\text{batch}})^2$  can be used as an estimator of  $\sigma^2/b$ . This fact justifies the batch-means estimator of MCSE given by

$$\text{MCSE}_{\text{batch}}(\hat{\theta}) = \frac{\hat{s}_{\text{batch}}}{\sqrt{m}}$$

The accuracy of the batch-means estimator depends on the choice of the batch length  $b$ . The higher the autocorrelation in the original MCMC sample, the larger the batch length  $b$  should be, provided that the number of batches  $m$  does not become too small;  $\sqrt{T}$  is typically used as the maximum value for  $b$ . The batch length is commonly determined by inspecting the autocorrelation plot for  $\theta$ . Under certain assumptions, [Flegal and Jones \(2010\)](#) establish that an asymptotically optimal batch size is of order  $T^{1/3}$ .

With multiple chains, the batch-means estimator is calculated using the combined sample of all chains or of those that are requested in the `chains()` option.

### Credible intervals

Let  $\theta_{(1)}, \dots, \theta_{(T)}$  be an MCMC sample ordered from smallest to largest. Let  $(1 - \alpha)$  be a credible level. Then, a  $\{100 \times (1 - \alpha)\}\%$  equal-tailed credible interval is

$$(\theta_{([T\alpha/2])}, \theta_{([T(1-\alpha/2)])})$$

where  $[ ]$  in the above imply an integer number.

A  $\{100 \times (1 - \alpha)\}\%$  HPD interval is defined as the shortest interval among the  $\{100 \times (1 - \alpha)\}\%$  credible intervals  $(\theta_{(j)}, \theta_{(j+[T(1-\alpha)])})$ ,  $j = 1, \dots, T - [T(1 - \alpha)]$ .

With multiple chains, credible intervals are computed using the combined sample of all chains or of those requested with the `chains()` option; see [Brooks and Gelman \(1998, sec. 1.1\)](#).

### References

Brooks, S. P., and A. Gelman. 1998. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics* 7: 434–455.

Chen, M.-H., and Q.-M. Shao. 1999. Monte Carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics* 8: 69–92.

Flegal, J. M., and G. L. Jones. 2010. Batch means and spectral variance estimators in Markov chain Monte Carlo. *Annals of Statistics* 38: 1034–1070.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

Jones, G. L., M. Haran, B. S. Caffo, and R. Neath. 2006. Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association* 101: 1537–1547.

Roberts, G. O. 1996. Markov chain concepts related to sampling algorithms. In *Markov Chain Monte Carlo in Practice*, ed. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, 45–57. Boca Raton, FL: Chapman and Hall.

## Also see

- [BAYES] [bayes](#) — Bayesian regression models using the bayes prefix
- [BAYES] [bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm
- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] [bayesgraph](#) — Graphical summaries and convergence diagnostics
- [BAYES] [bayespredict](#) — Bayesian predictions
- [BAYES] [bayesstats ess](#) — Effective sample sizes and related statistics
- [BAYES] [bayesstats ppvalues](#) — Bayesian predictive p-values and other predictive summaries
- [BAYES] [bayestest interval](#) — Interval hypothesis testing

# Title

bayestest — Bayesian hypothesis testing

Description

Remarks and examples

Also see

## Description

`bayestest` provides two types of Bayesian hypothesis testing, interval hypothesis testing and model hypothesis testing, using current Bayesian estimation results.

`bayestest interval` performs interval hypothesis tests for model parameters and functions of model parameters; see [\[BAYES\] bayestest interval](#).

`bayestest model` tests hypotheses about models by computing posterior probabilities of the models; see [\[BAYES\] bayestest model](#).

## Remarks and examples

Bayesian hypothesis testing is fundamentally different from the conventional frequentist hypothesis testing using  $p$ -values. Frequentist hypothesis testing is based on the deterministic decision of whether to reject a null hypothesis against an alternative hypothesis based on the obtained  $p$ -value. Bayesian hypothesis testing is built upon a probabilistic formulation for a parameter of interest. For example, it can provide a probabilistic summary of how likely that parameter of interest belongs to some prespecified set of values. Also, Bayesian testing can assign a probability to a hypothesis of interest or model of interest given the observed data. This cannot be done in the frequentist testing. The ability to assign a probability to a hypothesis often provides a more natural interpretation of the results. For example, Bayesian hypothesis testing provides a direct answer to the following questions. How likely is it that the mean height of males is larger than six feet? What is the probability that a person is guilty versus being innocent? How likely is one model over the other model? Frequentist hypothesis testing cannot be used to answer these questions.

We consider two forms of Bayesian hypothesis testing: interval hypothesis testing and what we call model hypothesis testing.

The goal of interval hypothesis testing is to estimate the probability that a model parameter lies in a certain interval; see [\[BAYES\] bayestest interval](#) for details.

The goal of model hypothesis testing is to test hypotheses about models by computing probabilities of the specified models given the observed data; see [\[BAYES\] bayestest model](#) for details.

## Also see

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] bayestest interval](#) — Interval hypothesis testing

[\[BAYES\] bayestest model](#) — Hypothesis testing using model posterior probabilities

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">Reference</a>	<a href="#">Also see</a>		

## Description

`bayestest interval` performs interval hypothesis tests for model parameters and functions of model parameters using current Bayesian estimation results. `bayestest interval` reports mean estimates, standard deviations, and MCMC standard errors of posterior probabilities associated with an interval hypothesis.

## Quick start

Posterior probability of the hypothesis that  $45 < \{y\_cons\} < 50$

```
bayestest interval {y:_cons}, lower(45) upper(50)
```

As above, but skip every 5 observations from the full MCMC sample

```
bayestest interval {y:_cons}, lower(45) upper(50) skip(5)
```

Posterior probability of a hypothesis about a function of model parameter  $\{y:x1\}$

```
bayestest interval (OR:exp({y:x1})), lower(1.1) upper(1.5)
```

Posterior probability of hypotheses  $45 < \{y\_cons\} < 50$  and  $0 < \{var\} < 10$  tested independently

```
bayestest interval ({y:_cons}, lower(45) upper(50)) ///
({var}, lower(0) upper(10))
```

As above, but tested jointly

```
bayestest interval (({y:_cons}, lower(45) upper(50)) ///
({var}, lower(0) upper(10)), joint)
```

Posterior probability of the hypothesis  $\{mean\} = 2$  for discrete parameter  $\{mean\}$

```
bayestest interval ({mean}==2)
```

Posterior probability of the interval hypothesis  $0 \leq \{mean\} \leq 4$

```
bayestest interval {mean}, lower(0, inclusive) upper(4, inclusive)
```

Posterior probability that the first observation of the first simulated outcome is positive (after `bayesmh`)

```
bayespredict {_ysim}, saving(predres)
bayestest interval {_ysim[1]} using predres, lower(0)
```

Posterior probability that the predicted test statistic `chi2stat` is less than 1

```
bayespredict (chi2stat: @chi2stat({_ysim})), saving(predres)
bayestest interval {chi2stat} using predres, upper(1)
```

## Menu

Statistics > Bayesian analysis > Interval hypothesis testing

## Syntax

*Test one interval hypothesis about continuous or discrete parameter*

```
bayestest interval exspec [using predfile] [, luspec options]
```

*Test one point hypothesis about discrete parameter*

```
bayestest interval exspec==# [using predfile] [, options]
```

*Test multiple hypotheses separately*

```
bayestest interval (testspec) [(testspec) ...] [using predfile] [, options]
```

*Test multiple hypotheses jointly*

```
bayestest interval (jointspec) [using predfile] [, options]
```

*Full syntax*

```
bayestest interval (spec) [(spec) ...] [using predfile] [, options]
```

*exspec* is optionally labeled expression of model parameters, [*prlabel*:]*expr*, where *prlabel* is a valid Stata name (or *prob#* by default), and *expr* is a [scalar model parameter](#) or scalar expression (parentheses are optional) containing scalar model parameters. The expression *expr* may not contain variable names.

*predfile* is the name of the dataset created by [bayespredict](#) that contains prediction results. When you specify *using predfile*, *expr* may contain individual observations of simulated outcomes {\_ysim#[#]}, expected outcome values {\_mu#[#]}, simulated residuals {\_resid#[#]}, and their expressions as described in [Functions of simulated outcomes, expected values, and residuals](#) in Syntax of [BAYES] [bayespredict](#). *expr* may also contain {*label*}, which is the name of the function simulated using [BAYES] [bayespredict](#). See [Different ways of specifying predictions and their functions](#) in [BAYES] [Bayesian postestimation](#). *expr* may not contain model parameters when *using predfile* is specified.

*testspec* is *exspec*[, *luspec*] or *exspec*==# for discrete parameters only.

*jointspec* is [*prlabel*:](*testspec*) (*testspec*) ..., *joint*. The labels (if any) of *testspec* are ignored.

*spec* is one of *testspec* or *jointspec*. *spec* may not contain model parameters when *using predfile* is specified.

<i>luspec</i>	Null hypothesis
<code>lower(#)</code> <code>[upper(.)]</code>	$\theta > \#$
<code>lower(#, inclusive)</code> <code>[upper(.)]</code>	$\theta \geq \#$
<code>[lower(.)] upper(#)</code>	$\theta < \#$
<code>[lower(.)] upper(#, inclusive)</code>	$\theta \leq \#$
<code>lower(#<sub>l</sub>) upper(#<sub>u</sub>)</code>	$\#_l < \theta < \#_u$
<code>lower(#<sub>l</sub>) upper(#<sub>u</sub>, inclusive)</code>	$\#_l < \theta \leq \#_u$
<code>lower(#<sub>l</sub>, inclusive) upper(#<sub>u</sub>)</code>	$\#_l \leq \theta < \#_u$
<code>lower(#<sub>l</sub>, inclusive) upper(#<sub>u</sub>, inclusive)</code>	$\#_l \leq \theta \leq \#_u$

`lower(intspec)` and `upper(intspec)` specify the lower- and upper-interval values, respectively.  
*intspec* is `# [ , inclusive ]`

where `#` is the interval value, and suboption `inclusive` specifies that this value should be included in the interval, meaning a closed interval. Closed intervals make sense only for discrete parameters.  
*intspec* may also contain a dot (`.`), meaning negative infinity for `lower()` and positive infinity for `upper()`. Either option `lower(.)` or option `upper(.)` must be specified.

<i>options</i>	Description
<b>Main</b>	
<code>*chains(_all   numlist)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
<code>*sepchains</code>	compute results separately for each chain
<code>skip(#)</code>	skip every <code>#</code> observations from the MCMC sample; default is <code>skip(0)</code>
<code>nolegend</code>	suppress table legend
<b>Advanced</b>	
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Options `chains()` and `sepchains` are relevant only when option `nchains()` is used with `bayesmh` or the `bayes` prefix.

# Options

## Main

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`sepchains` specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option `chains()`. Option `sepchains` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

`skip(#)` specifies that every `#` observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(#)` is equivalent to a thinning interval of `#+1`. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will

skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`nolegend` suppresses the display of the table legend, which identifies the rows of the table with the expressions they represent.

#### Advanced

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is  $\min\{500, \text{mcmcsize}()/2\}$ . The total autocorrelation is computed as the sum of all lag- $k$  autocorrelation values for  $k$  from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag- $k$  autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the  $k$ th lag are discarded.

## Remarks and examples

Remarks are presented under the following headings:

[Introduction](#)

[Interval tests for continuous parameters](#)

[Interval tests for discrete parameters](#)

## Introduction

In this entry, we describe interval hypothesis testing, the goal of which is to estimate the probability that a model parameter lies in a certain interval. Interval hypothesis testing is inversely related to credible intervals. For example, if we have a 95% credible interval for  $\theta$  with endpoints  $U$  and  $L$ , then the probability of a hypothesis  $H_0: \theta \in [U, L]$  is 0.95. For hypothesis testing using model posterior probabilities, see [BAYES] [bayestest model](#).

In frequentist hypothesis testing, we often consider a point hypothesis such as  $H_0: \theta = \theta_0$  versus  $H_a: \theta \neq \theta_0$ . In Bayesian hypothesis testing, the probability  $P(\theta = \theta_0)$  is 0 whenever  $\theta$  has a continuous posterior distribution. A point hypothesis is relevant only to parameters with discrete posterior distributions. For continuous parameters, all hypotheses should be formulated as intervals. One possibility is to consider an interval hypothesis  $H_0: \theta \in (\theta_0 - \epsilon, \theta_0 + \epsilon)$ , where  $\epsilon$  is some small value.

Note that Bayesian hypothesis testing does not really need a distinction between the null and alternative hypotheses, in the sense that they are defined in a frequentist statistic. There is no need to “protect” the null hypothesis: if  $P\{H_0: \theta \in (a, b)\} = p$ , then  $P\{H_a: \theta \notin (a, b)\} = 1 - p$ . In what follows, when we refer to  $H_0$ , we imply a hypothesis of interest  $H_0: \theta \in \Theta$ , and when we refer to  $H_a$ , we imply the complement hypothesis  $H_a: \theta \in \Theta^c$ , where  $\Theta$  is a set of points from the domain of  $\theta$  and  $\Theta^c$  is its complement.

The `bayestest interval` command estimates the posterior probability of a null interval hypothesis  $H_0$  using the simulated posterior distributions of model parameters produced by Bayesian estimation. Essentially, `bayestest interval` reports posterior summaries for a dichotomous expression that represents  $H_0$ .

For example, suppose we would like to test the following hypothesis:  $H_0: \theta \in (a, b)$ . Then,

```
bayestest interval ({theta}, lower(a) upper(b))
```

is equivalent to

```
bayesstats summary ({theta} > a & {theta} < b)
```

`bayestest interval` reports the estimated posterior mean probability for  $H_0$ , which is not a  $p$ -value—as reported by classical frequentist tests—used to decide whether to reject  $H_0$  in favor of the alternative  $H_a$ . The  $p$ -value interpretation is based on the dichotomous problem formulation of  $H_0$  versus  $H_a$ , assuming that one of these two alternatives is actually true. The answer in the Bayesian context is a probability statement about  $\theta$  that is free of any deterministic presumptions. For example, if you estimate  $P(H_0)$  to be 0.15, you cannot ask whether this value is significant or whether you can reject the null hypothesis. Bayesian interpretation of this probability is that if you draw  $\theta$  from the specified prior distribution and update your knowledge about  $\theta$  based on the observed data, then there is a 15% chance that  $\theta$  will belong to the interval  $(a, b)$ . So the conclusion of Bayesian hypothesis testing is not an acceptance or rejection of the null hypothesis but an explicit probability statement about the tested hypothesis.

## Interval tests for continuous parameters

Let's continue our analysis of `auto.dta` from [example 4](#) in [\[BAYES\] bayesmh](#) using the mean-only normal model for `mpg` with a noninformative prior.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. set seed 14

. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, flat) prior({var}, jeffreys)
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})

Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys
```

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2668
	Efficiency: min =	.09718
	avg =	.1021
	max =	.1071
Log marginal-likelihood =	-234.645	

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286



➤ Example 1: Interval hypothesis and credible intervals

In the introduction, we commented on the inverse relationship that exists between interval hypothesis tests and credible intervals. Let’s verify this using `bayestest interval`. We are interested in a hypothesis  $H_0: \{mpg\_cons\} \in (19.992, 22.619)$ , where the specified numbers are the endpoints of the credible interval for `{mpg:_cons}` from the `bayesmh` output. To compute the posterior probability for this hypothesis, we specify the parameter following the command line and specify interval endpoints in `lower()` and `upper()`.

```
. bayestest interval {mpg:_cons}, lower(19.992) upper(22.619)
Interval tests      MCMC sample size =      10,000
      prob1 : 19.992 < {mpg:_cons} < 22.619
```

	Mean	Std. Dev.	MCSE
prob1	.9496	0.21878	.0053652

The estimated posterior probability is close to 0.95, as we expected, because we used the endpoints of the 95% credible intervals for `{mpg:_cons}`.

By default, `bayestest interval` labels probabilities as `prob#` (`prob1` in our example). You can specify your own label as long as you enclose the parameter in parentheses:

```
. bayestest interval (mean:{mpg:_cons}), lower(19.992) upper(22.619)
Interval tests      MCMC sample size =      10,000
      mean : 19.992 < {mpg:_cons} < 22.619
```

	Mean	Std. Dev.	MCSE
mean	.9496	0.21878	.0053652



➤ Example 2: Testing multiple hypotheses separately

Continuing [example 1](#), we can verify that the probability associated with the credible interval for `{var}` is also close to 0.95.

We can specify multiple hypotheses with `bayestest interval`, but we must enclose them in parentheses.

```
. bayestest interval ({mpg:_cons}, lower(19.992) upper(22.619))
>      ({var}, lower(24.913) upper(47.613))
Interval tests      MCMC sample size =      10,000
      prob1 : 19.992 < {mpg:_cons} < 22.619
      prob2 : 24.913 < {var} < 47.613
```

	Mean	Std. Dev.	MCSE
prob1	.9496	0.21878	.0053652
prob2	.9502	0.21754	.0053011

The estimated posterior probability `prob2` is also close to 0.95.



### ► Example 3: Testing multiple hypotheses jointly

We can perform joint tests of multiple hypotheses by enclosing hypothesis to be tested jointly in parentheses and by specifying suboption `joint`. Notice that each individual hypothesis must also be enclosed in parentheses.

```
. bayestest interval (({mpg:_cons}, lower(19.992) upper(22.619))
>                    ({var}, lower(24.913) upper(47.613)), joint)
Interval tests      MCMC sample size =    10,000
      prob1 : 19.992 < {mpg:_cons} < 22.619,
              24.913 < {var} < 47.613
```

	Mean	Std. Dev.	MCSE
prob1	.9034	0.29543	.0076789

The joint posterior probability of both `{mpg:_cons}` and `{var}` belonging to their respective intervals is 0.9 with a posterior variance of 0.3 and MCSE of 0.008.

◀

### ► Example 4: Full syntax

We can specify multiple separate hypotheses and hypotheses tested jointly in one call to `bayestest interval`.

```
. bayestest interval (({mpg:_cons}, lower(19.992) upper(22.619))
>                    ({var}, lower(24.913) upper(47.613)), joint)
>                    ({mpg:_cons}, lower(21))
>                    ({var}, upper(40))
Interval tests      MCMC sample size =    10,000
      prob1 : 19.992 < {mpg:_cons} < 22.619,
              24.913 < {var} < 47.613
      prob2 : {mpg:_cons} > 21
      prob3 : {var} < 40
```

	Mean	Std. Dev.	MCSE
prob1	.9034	0.29543	.0076789
prob2	.6505	0.47684	.015786
prob3	.8136	0.38945	.0110613

In addition to the joint hypothesis from the previous example, we specified two new separate interval hypotheses for testing `{mpg:_cons} > 21` and for testing `{var} < 40`. The estimated posterior probabilities for these hypotheses are 0.65 and 0.81, respectively.

◀

➤ Example 5: Point hypothesis for continuous parameters

As we discussed in [Introduction](#) above, point hypothesis for continuous parameters do not make sense, because the corresponding probability is 0:

```
. bayestest interval ({mpg:_cons}==21)
Interval tests      MCMC sample size =    10,000
      prob1 : {mpg:_cons}==21
```

	Mean	Std. Dev.	MCSE
prob1	0	0.00000	0

We can consider a small window around the value of interest and test an interval hypothesis instead:

```
. bayestest interval ({mpg:_cons}, lower(20.5) upper(21.5))
Interval tests      MCMC sample size =    10,000
      prob1 : 20.5 < {mpg:_cons} < 21.5
```

	Mean	Std. Dev.	MCSE
prob1	.4932	0.49998	.0138391

The probability that {mpg:\_cons} is between 20.5 and 21.5 is about 50%.

Note that the probability of a continuous parameter belonging to a closed interval or semiclosed interval is the same as that for the open interval. Below we use suboption `inclusive` within `lower()` and `upper()` to request the closed interval.

```
. bayestest interval ({mpg:_cons}, lower(20.5,inclusive) upper(21.5,inclusive))
Interval tests      MCMC sample size =    10,000
      prob1 : 20.5 <= {mpg:_cons} <= 21.5
```

	Mean	Std. Dev.	MCSE
prob1	.4932	0.49998	.0138391

We obtain the same results as above for the corresponding open interval.



➤ Example 6: Functions of parameters

We can test functions of model parameters. For example, let's compute the probability that the posterior standard deviation is greater than 6.

```
. bayestest interval (sd: sqrt({var}), lower(6))
Interval tests      MCMC sample size =    10,000
      sd : sqrt({var}) > 6
```

	Mean	Std. Dev.	MCSE
sd	.3793	0.48524	.0143883

The estimated probability is 0.38.



## Interval tests for discrete parameters

In this section, we demonstrate how to perform hypothesis testing for a discrete parameter.

First, we simulate data from the Poisson distribution with a mean of 2.

```
. clear
. set seed 12345
. set obs 20
number of observations (_N) was 0, now 20
. generate double y = rpoisson(2)
```

We fit a Bayesian Poisson model to the data and specify a discrete prior for the mean  $P(\mu = k) = 0.25$  for  $k = 1, 2, 3, 4$ .

```
. set seed 14
. bayesmh y, likelihood(dpoisson({mu}))
> prior({mu}, index(0.25,0.25,0.25,0.25)) initial({mu} 2)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  y ~ poisson({mu})
```

```
Prior:
  {mu} ~ index(0.25,0.25,0.25,0.25)
```

---

Bayesian Poisson model	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	20
	Acceptance rate	=	.2552
Log marginal-likelihood = -31.58903	Efficiency	=	.4428

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mu	2.0014	.1039188	.001562	2	2	2

---

➤ Example 7: Point hypotheses for discrete parameters

We can compute probabilities for each of the four discrete values of  $\{\mu\}$ .

```
. bayestest interval ({mu}==1) ({mu}==2) ({mu}==3) ({mu}==4)
Interval tests      MCMC sample size =      10,000
  prob1 : {mu}==1
  prob2 : {mu}==2
  prob3 : {mu}==3
  prob4 : {mu}==4
```

	Mean	Std. Dev.	MCSE
prob1	.0047	0.06840	.0013918
prob2	.9892	0.10337	.0027909
prob3	.0061	0.07787	.0017691
prob4	0	0.00000	0

The posterior probability that  $\{\mu\}$  equals 2 is 0.99.



➤ Example 8: Interval hypotheses for discrete parameters

As we can with continuous parameters, we can test interval hypotheses for discrete parameters. For example, we can compute the probability of whether  $\{\mu\}$  is between 2 and 4.

```
. bayestest interval {mu}, lower(2) upper(4)
Interval tests      MCMC sample size =      10,000
  prob1 : 2 < {mu} < 4
```

	Mean	Std. Dev.	MCSE
prob1	.0061	0.07787	.0017691

The estimated probability is very small.

Note that unlike hypotheses for continuous parameters, hypotheses including open intervals and closed or semiclosed intervals for discrete parameters may have different probabilities.

```
. bayestest interval {mu}, lower(2, inclusive) upper(4, inclusive)
Interval tests      MCMC sample size =      10,000
  prob1 : 2 <= {mu} <= 4
```

	Mean	Std. Dev.	MCSE
prob1	.9953	0.06840	.0013918

The estimated posterior probability that  $\{\mu\}$  is between 2 and 4, inclusively, is drastically different compared with the results for the corresponding open interval.



## Stored results

`bayestest interval` stores the following in `r()`:

### Scalars

<code>r(mcmcsize)</code>	MCMC sample size used in the computation
<code>r(skip)</code>	number of MCMC observations to skip in the computation; every <code>r(skip)</code> observations are skipped
<code>r(corrlag)</code>	maximum autocorrelation lag
<code>r(corrtol)</code>	autocorrelation tolerance
<code>r(nchains)</code>	number of chains used in the computation

### Macros

<code>r(names)</code>	names of probability expressions
<code>r(expr_#)</code>	#th probability expression
<code>r(chains)</code>	chains used in the computation, if <code>chains()</code> is specified

### Matrices

<code>r(summary)</code>	test results for parameters in <code>r(names)</code>
<code>r(summary_chain#)</code>	matrix <code>summary</code> for chain #, if <code>sepchains</code> is specified

## Methods and formulas

Let  $\theta$  be a model parameter and  $\{\theta_t\}_{t=1}^T$  be an MCMC sample of size  $T$  drawn from the marginal posterior distribution of  $\theta$ . It is often of interest to test how likely it is that  $\theta$  belongs to a particular range of values. Note that testing a point null hypothesis such as  $H_0: \theta = \theta_0$  is usually of no interest for parameters with continuous posterior distributions, because the posterior probability  $P(H_0)$  is 0.

To perform an open-interval test of the form

$$H_0: \theta \in (a, b) \text{ versus } H_a: \theta \notin (a, b)$$

we estimate the posterior probability of  $H_0$  from the given MCMC sample. The `bayestest interval` command calculates the probability  $P(H_0)$  based on the simulated marginal posterior distribution of  $\theta$ . The estimate is given by the frequency of inclusion of  $\theta_t$ s in the test interval

$$\hat{P}(H_0) = \frac{1}{T} \sum_{t=1}^T 1_{\{\theta_t \in (a, b)\}} \quad (1)$$

where  $1_{\{A\}}$  is an indicator function and equals 1 if  $A$  is true and 0 otherwise.

When a model parameter  $\theta$  is discrete, the following closed- and semiclosed-interval tests may be of interest in addition to open-interval tests:

$$H_0: \theta = a \text{ versus } H_a: \theta \neq a$$

$$H_0: \theta \in [a, b] \text{ versus } H_a: \theta \notin [a, b]$$

$$H_0: \theta \in [a, b) \text{ versus } H_a: \theta \notin [a, b)$$

$$H_0: \theta \in (a, b] \text{ versus } H_a: \theta \notin (a, b]$$

The corresponding probabilities are calculated as follows:

$$\hat{P}(H_0) = \frac{1}{T} \sum_{t=1}^T 1_{\{\theta_t=a\}}$$

$$\hat{P}(H_0) = \frac{1}{T} \sum_{t=1}^T 1_{\{\theta_t \in [a,b]\}}$$

$$\hat{P}(H_0) = \frac{1}{T} \sum_{t=1}^T 1_{\{\theta_t \in (a,b)\}}$$

$$\hat{P}(H_0) = \frac{1}{T} \sum_{t=1}^T 1_{\{\theta_t \in [a,b)\}}$$

The probability of an alternative hypothesis is always given by  $P(H_a) = 1 - P(H_0)$ .

The formulas above can be modified to accommodate joint hypotheses tests by multiplying the indicator functions of the individual hypothesis statements. For example, for a joint hypothesis  $H_0: \theta_1 > a, \theta_2 < b$ , we would replace the indicator function with  $1_{\{\theta_{1t} > a\}} \times 1_{\{\theta_{2t} < b\}}$  in (1), where  $\{\theta_{1t}\}_{t=1}^T$  and  $\{\theta_{2t}\}_{t=1}^T$  are the corresponding MCMC samples for  $\theta_1$  and  $\theta_2$ .

With multiple chains, the `bayestest interval` command performs computation using all simulated chains or those specified in the `chains()` option. The calculations are the same as for `bayesstats summary` in the presence of multiple chains; see [Methods and formulas](#) in [BAYES] `bayesstats summary`.

## Reference

Huber, C. 2016. Introduction to Bayesian statistics, part 1: The basic concepts. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2016/11/01/introduction-to-bayesian-statistics-part-1-the-basic-concepts/>.

## Also see

- [BAYES] [bayes](#) — Bayesian regression models using the bayes prefix
- [BAYES] [bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm
- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] [bayespredict](#) — Bayesian predictions
- [BAYES] [bayesstats summary](#) — Bayesian summary statistics
- [BAYES] [bayestest model](#) — Hypothesis testing using model posterior probabilities

# Title

bayestest model — Hypothesis testing using model posterior probabilities

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">Also see</a>			

## Description

`bayestest model` computes posterior probabilities of Bayesian models fit using the `bayesmh` command or the `bayes` prefix. These posterior probabilities can be used to test hypotheses about model parameters. The command reports marginal likelihoods, prior probabilities, and posterior probabilities for all tested models.

## Quick start

Compute posterior probabilities of models corresponding to previously saved estimation results `M1` and `M2`

```
bayestest model M1 M2
```

As above, but specify prior probabilities for models

```
bayestest model M1 M2, prior(0.3 0.7)
```

## Menu

Statistics > Bayesian analysis > Hypothesis testing using model posterior probabilities



## Syntax

```
bayestest model [namelist] [ , options ]
```

where *namelist* is a name, a list of names, `_all`, or `*`. A name may be `.`, meaning the current (active) estimates. `_all` and `*` mean the same thing.

options	Description
Main	
<code>prior(numlist)</code>	specify prior probabilities for tested models; default is all models are equally likely
<code>*chains(_all   numlist)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
<code>*sepchains</code>	compute results separately for each chain
Advanced	
<code>marglmethod(method)</code>	specify marginal-likelihood approximation method; default is to use Laplace–Metropolis approximation, <code>lmetropolis</code> ; rarely used
*Options <code>chains()</code> and <code>sepchains</code> are relevant only when option <code>nchains()</code> is used with <code>bayesmh</code> or the <code>bayes</code> prefix.	
method	Description
<code>lmetropolis</code>	Laplace–Metropolis approximation; default
<code>hmean</code>	harmonic-mean approximation

## Options

Main

- `prior(numlist)` specifies prior probabilities for models. By default, all models are assumed to be equally likely. You may specify probabilities for all tested models, in which case the probabilities must sum to one. Alternatively, you may specify probabilities for all but the last model, in which case the sum of the specified probabilities must be less than one, and the probability for the last model is computed as one minus this sum.
- `chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.
- `sepchains` specifies that the results be computed separately for each chain. The default is to compute results using all chains as determined by option `chains()`. Option `sepchains` is relevant only when option `nchains()` is specified with `bayesmh` or the `bayes` prefix.

Advanced

- `marglmethod(method)` specifies a method for approximating the marginal likelihood. *method* is either `lmetropolis`, the default, for Laplace–Metropolis approximation or `hmean` for harmonic-mean approximation. This option is rarely used.

## Remarks and examples

Remarks are presented under the following headings:

[Introduction](#)

[Testing nested hypotheses](#)

[Comparing models with different priors](#)

## Introduction

In this entry, we describe hypothesis testing by computing model posterior probabilities, probabilities of Bayesian models given observed data. For interval hypothesis testing, see [\[BAYES\] bayestest interval](#).

The `bayestest model` command computes posterior probabilities for specified models. The computed probabilities can be used to compare which model is more likely among considered models given observed data. You can compare models that differ only in several covariates or models with completely different regression functions, such as linear and nonlinear models. You can compare models with different outcome distributions or with different prior distributions or both. The only requirements are that the considered models have proper posterior distributions and that the same data are used to fit the models. If MCMC is used to approximate posterior distributions, convergence of MCMC should also be verified before model comparison.

The results reported by `bayestest model` are related to Bayes factors; see [\[BAYES\] bayesstats ic](#) to compute Bayes factors.

To use `bayestest model`, you must store estimation results after each Bayesian model of interest. You can use `estimates store` (see [\[R\] estimates store](#)) to store estimation results after `bayesmh` or the `bayes` prefix, as you can with other estimation commands, provided you also saved simulation results from `bayesmh` or the `bayes` prefix using the `saving()` option. See [Storing estimation results after Bayesian estimation](#) in [\[BAYES\] Bayesian postestimation](#) for details.

## Testing nested hypotheses

Consider the following Bayesian regression model for `auto.dta`,

$$\text{mpg} = \beta_0 + \beta_1 \text{weight1} + \beta_2 \text{length1} + \epsilon$$

where `weight1` and `length1` are the original `weight` and `length` variables rescaled to have similar scale as `mpg`.

We assume that errors are normally distributed:  $\epsilon \sim \text{normal}(0, \sigma^2)$ . We also assume a noninformative Jeffreys prior for the parameters:  $(\beta, \sigma^2) \sim 1/\sigma^2$ . Suppose that we are interested in testing whether there is a relationship between mileage and weight and length of cars. We will consider four models: the mean-only model, the model with weight only, the model with length only, and the full model with both covariates.

In a frequentist setting, the four models correspond to the following hypotheses:  $H_0: \beta_1 = 0$ ,  $\beta_2 = 0$ ,  $H_0: \beta_1 = 0$ , and  $H_0: \beta_2 = 0$ . In a Bayesian setting, we cannot formulate point hypotheses for parameters with continuous distributions; see [\[BAYES\] bayestest interval](#) for examples. However, we can compute probabilities of how likely each of the four models is given the observed data.

Let's load `auto.dta` and generate rescaled versions of `weight` and `length`.

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)

. generate weight1 = weight/100
. generate length1 = length/10
```

Next, we fit the four models using `bayesmh`. We use the `saving()` option to save the simulation datasets so that we can store estimation results of each model for later use with `bayestest model`.

The first model we fit is the mean-only model. We store its estimation results as `meanonly`.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(meanonly_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal({mpg:\_cons},{var})

Priors:  
{mpg:\_cons} ~ 1 (flat)  
{var} ~ jeffreys

---

Bayesian normal regression	MCMC iterations =	13,500
Random-walk Metropolis-Hastings sampling	Burn-in =	3,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2627
	Efficiency: min =	.105
	avg =	.1064
	max =	.1078

Log marginal-likelihood = -234.64617

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29355	.6768607	.020887	21.28059	20.00132	22.61904
var	34.80707	5.963995	.181615	34.23247	24.9129	47.6883

---

```
file meanonly_simdata.dta saved
. estimates store meanonly
```

To accommodate the Jeffreys prior for the parameters, we specify suboption `flat` within the `prior()` option for coefficients to request the flat prior with the density of 1 and suboption `jeffreys` within `prior()` for the variance parameter to request a Jeffreys prior. We also specify a longer burn-in period to improve convergence of MCMC samples for all examples. (Remember to use `bayesgraph` to check convergence of MCMC.)

We fit the second model containing only covariate length1 and store its results as length:

```
. set seed 14
. bayesmh mpg length1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(length_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:length1 \_cons} ~ 1 (flat) (1)  
{var} ~ jeffreys

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	13,500
Random-walk Metropolis-Hastings sampling	Burn-in =	3,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2865
	Efficiency: min =	.0771
	avg =	.07938
	max =	.08286

Log marginal-likelihood = -198.7678

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg	length1	-2.069861	.1882345	.006539	-2.068094	-2.44718
	_cons	60.20346	3.562119	.127411	60.20927	53.34306
	var	12.88852	2.273808	.081887	12.62042	9.169482
					18.16685	

```
file length_simdata.dta saved
. estimates store length
```

We fit the third model containing only covariate weight1 and store its results as weight:

```
. set seed 14
. bayesmh mpg weight1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(weight_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
mpg ~ normal(xb\_mpg,{var})

Priors:  
{mpg:weight1 \_cons} ~ 1 (flat) (1)  
{var} ~ jeffreys

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	13,500
Random-walk Metropolis-Hastings sampling	Burn-in =	3,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.1735
	Efficiency: min =	.0463
	avg =	.06694
	max =	.07989

Log marginal-likelihood = -198.20751

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
	weight1	-.6014409	.0506121	.001791	-.6013071	-.6996976
	_cons	39.45934	1.574673	.057646	39.49735	36.31386
	var	12.13997	2.141741	.099534	11.87332	8.883221
						17.14041

```
file weight_simdata.dta saved
. estimates store weight
```

Finally, we fit the last model containing both covariates and store its results as full:

```
. set seed 14
. bayesmh mpg weight1 length1, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(full_simdata) burnin(3500)
note: adaptation option maxiter() changed to 35
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal(xb_mpg,{var})
Priors:
  {mpg:weight1 length1 _cons} ~ 1 (flat)
                                {var} ~ jeffreys
```

(1) Parameters are elements of the linear form xb\_mpg.

Bayesian normal regression	MCMC iterations =	13,500
Random-walk Metropolis-Hastings sampling	Burn-in =	3,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2323
	Efficiency: min =	.05455
	avg =	.06647
	max =	.08085

Log marginal-likelihood = -196.86195

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
weight1	-.3977027	.1580411	.005558	-.401646	-.6965175	-.0721332
length1	-.7599159	.5546754	.021944	-.7502182	-1.907818	.3106868
_cons	47.5913	6.132597	.262563	47.5656	35.89593	60.18002
var	11.81753	1.96315	.07608	11.59273	8.729182	16.14065

file full\_simdata.dta saved

. estimates store full

## ► Example 1: Computing posterior probabilities of models

We now use bayestest model to compute posterior probabilities of the four models.

```
. bayestest model meanonly length weight full
Bayesian model tests
```

	log(ML)	P(M)	P(M y)
meanonly	-234.6462	0.2500	0.0000
length	-198.7678	0.2500	0.1055
weight	-198.2075	0.2500	0.1848
full	-196.8619	0.2500	0.7097

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

The mean-only model is very unlikely compared with other models. The length and weight models are somewhat likely with the respective posterior probabilities of 0.11 and 0.18, and the full model has the highest posterior probability of 0.71.

➤ Example 2: Specifying prior probabilities of models

If we have some prior knowledge about each of the models, we can use the `prior()` option to specify prior probabilities for each model. For example, suppose that we have prior knowledge that the weight model is much more likely than the full model so that the prior probabilities are 0.1 for the mean-only model and the length model, 0.6 for the weight model, and only 0.2 for the full model.

```
. bayestest model meanonly length weight full, prior(0.1 0.1 0.6 0.2)
Bayesian model tests
```

	log(ML)	P(M)	P(M y)
meanonly	-234.6462	0.1000	0.0000
length	-198.7678	0.1000	0.0401
weight	-198.2075	0.6000	0.4210
full	-196.8619	0.2000	0.5389

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

Under the specified prior, posterior probabilities of the weight and full models are now more similar: 0.42 and 0.54, respectively, but the full model is still preferable.

The above is equivalent to the following prior specification:

```
. bayestest model meanonly length weight full, prior(0.1 0.1 0.6)
(output omitted)
```



Using our results, we conclude that `mpg` is related to both `weight` and `length` and would proceed with the full model.

After your analysis, remember to erase the saved simulation datasets you no longer need. For example, we erase all of them by typing

```
. erase meanonly_simdata.dta
. erase weight_simdata.dta
. erase length_simdata.dta
. erase full_simdata.dta
```

Comparing models with different priors

In the previous section, we used `bayestest model` to compare nested hypotheses about which covariates to include in the regression function. We can use `bayestest model` to compare models with not only different covariates but also different outcome distributions and priors for parameters.

We continue our analysis of `auto.dta`, but for simplicity, we now consider the mean-only model for `mpg`. Let's compare models with two slightly different informative priors. We use an informative normal-inverse-gamma prior for both models,

$$(\beta_0|\sigma^2) \sim N(\mu_0, \sigma^2/n_0)$$
$$\sigma^2 \sim \text{InvGamma}(\nu_0/2, \nu_0\sigma_0^2/2)$$

with  $\mu_0 = 25$ ,  $n_0 = 10$ , and  $\sigma_0^2 = 30$ , but we consider two different values for the degrees of freedom:  $\nu_0 = 5$  and  $\nu_0 = 1$ .

We use `bayesmh` to fit our models. Following the formulas, we specify a `normal()` prior for the constant `{mpg:_cons}` (mean parameter) and an inverse-gamma prior `igamma()` for the variance parameter `{var}`. We specify an expression for the variance of the normal prior distribution in parentheses.

We fit the first model with  $\nu_0 = 5$  and store its estimation results as `informative1`.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(25,{var}/10))
> prior({var}, igamma(2.5,75)) saving(inf1_simdata)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
    mpg ~ normal({mpg:_cons},{var})
Priors:
    {mpg:_cons} ~ normal(25,{var}/10)
    {var} ~ igamma(2.5,75)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2548
	Efficiency: min =	.09065
	avg =	.1049
	max =	.1192

```
Log marginal-likelihood = -238.55856
```

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
mpg					
_cons	21.71853	.6592655	.019091	21.69554	20.44644 23.04896
var	35.47405	5.823372	.193417	34.72454	25.84419 48.228

---

```
file inf1_simdata.dta saved
. estimates store informative1
```



We fit the second model with  $\nu_0 = 1$  and store its estimation results as `informative2`.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, normal(25,{var}/10))
> prior({var}, igamma(0.5,15)) saving(inf2_simdata)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
    mpg ~ normal({mpg:\_cons},{var})

Priors:  
    {mpg:\_cons} ~ normal(25,{var}/10)  
    {var} ~ igamma(0.5,15)

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2261
	Efficiency: min =	.0941
	avg =	.109
	max =	.1239

Log marginal-likelihood = -239.4049

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.7175	.6539814	.021319	21.7295	20.47311	23.02638
var	35.89504	6.288571	.178665	35.17056	25.86084	50.21624

---

```
file inf2_simdata.dta saved
. estimates store informative2
```

➤ Example 3: Comparing models with informative priors

We now use `bayestest model` to compare our models with two different informative priors.

```
. bayestest model informative1 informative2
Bayesian model tests
```

	log(ML)	P(M)	P(M y)
informative1	-238.5586	0.5000	0.6998
informative2	-239.4049	0.5000	0.3002

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

Assuming that both models are equally likely a priori, the posterior probability of the `informative1` stored results, 0.70, is much higher than the probability of the `informative2` stored results, 0.3.



### ► Example 4: Comparing a model with noninformative prior

A note of caution regarding comparing models with informative and noninformative priors—models with noninformative priors will often win because they are typically in most agreement with the observed data. For models with noninformative priors, most of the information about parameters is contained in a likelihood. As such, any model with an informative prior that is not in perfect agreement with the data will not fit data as well as a model with a noninformative prior.

For example, let's fit our constant-only model using a noninformative Jeffreys prior for the parameters.

```
. set seed 14
. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:}, flat) prior({var}, jeffreys)
> saving(jeffreys_simdata)
Burn-in ...
Simulation ...
Model summary
```

```
Likelihood:
  mpg ~ normal({mpg:_cons},{var})
```

```
Priors:
  {mpg:_cons} ~ 1 (flat)
  {var} ~ jeffreys
```

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.2668
	Efficiency: min =	.09718
	avg =	.1021
	max =	.1071
Log marginal-likelihood =	-234.645	

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
mpg						
_cons	21.29222	.6828864	.021906	21.27898	19.99152	22.61904
var	34.76572	5.91534	.180754	34.18391	24.9129	47.61286

```
file jeffreys_simdata.dta saved
. estimates store jeffreys
```

Let's now compare this model with our two informative models.

```
. bayestest model informative1 informative2 jeffreys
Bayesian model tests
```

	log(ML)	P(M)	P(M y)
informative1	-238.5586	0.3333	0.0194
informative2	-239.4049	0.3333	0.0083
jeffreys	-234.6450	0.3333	0.9723

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

The posterior probability of the Jeffreys model is 0.97.

Finally, at the end of our analysis, we erase all the simulation datasets we no longer need. We erase all of them by typing

```
. erase inf1_simdata.dta
. erase inf2_simdata.dta
. erase jeffreys_simdata.dta
```

## Stored results

`bayestest model` stores the following in `r()`:

### Macros

<code>r(names)</code>	names of estimation results used
<code>r(marglmethod)</code>	method for approximating marginal likelihood: <code>lmetropolis</code> or <code>hmean</code>
<code>r(chains)</code>	chains used in the computation, if <code>chains()</code> is specified

### Matrices

<code>r(test)</code>	test results for models in <code>r(names)</code>
<code>r(test_chain#)</code>	matrix <code>test</code> for chain <code>#</code> , if <code>sepchains</code> is specified

## Methods and formulas

Suppose we have  $r$  models  $M_j$  for  $j = 1, \dots, r$  with prior probabilities  $P(M_j)$  such that  $\sum_{j=1}^r p(M_j) = 1$ . Then, posterior probability for model  $J$  is

$$P(M_j|\mathbf{y}) = \frac{P(\mathbf{y}|M_j)P(M_j)}{P(\mathbf{y})}$$

where  $P(\mathbf{y}|M_j) = m_j(y)$  is the marginal likelihood of  $M_j$  with respect to  $\mathbf{y}$ , and  $P(\mathbf{y}) = \sum_{j=1}^r P(\mathbf{y}|M_j)P(M_j)$ . See *Methods and formulas* in [\[BAYES\] bayesmh](#) for details about computing marginal likelihood.

With multiple chains, the `bayestest model` command uses the averaged across chains log marginal-likelihood for calculations. If the `sepchains` option is specified, the results are calculated and reported separately for each chain.

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[BAYES\] bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] bayesstats ic](#) — Bayesian information criteria and Bayes factors

[\[BAYES\] bayesstats summary](#) — Bayesian summary statistics

[\[BAYES\] bayestest interval](#) — Interval hypothesis testing

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Options</a>	<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>
<a href="#">References</a>	<a href="#">Also see</a>		

## Description

`bayespredict` computes Bayesian predictions using current estimation results produced by the `bayesmh` command and saves them in a separate Stata dataset. Bayesian predictions include simulated outcomes, which are samples from the [posterior predictive distribution](#) of the fitted Bayesian model, and their functions. You can also compute posterior summaries of simulated outcomes and store them as new variables in the current dataset.

`bayesreps` generates a random subset of [MCMC replicates](#) of simulated outcomes from the entire MCMC sample and stores them as new variables in the current dataset. This command is useful for checking model fit.

## Quick start

### *Simulated outcomes*

Predictions for the first outcome variable after fitting a two-equation Bayesian model using [bayesmh](#)

```
bayespredict {_ysim}, saving(prdata)
```

As above, but for the second outcome variable, replacing `prdata.dta` with new prediction results

```
bayespredict {_ysim2}, saving(prdata, replace)
```

Predictions for the first outcome variable and observations 2 through 5

```
bayespredict {_ysim1[2/5]}, saving(prdata, replace)
```

### *Test statistics for simulated outcomes*

Maximums and minimums of simulated outcomes computed over observations for the first outcome variable

```
bayespredict (rmax:@max({_ysim1})) (rmin:@min({_ysim1})), ///
    saving(prdata, replace)
```

Maximums and minimums of residuals for the second outcome variable

```
bayespredict (rmax:@max({_resid2})) (rmin:@min({_resid2})), ///
    saving(prdata, replace)
```

*Posterior summaries of simulated outcomes*

Posterior means for the two outcomes stored in new variables `pmean1` and `pmean2` in the current dataset

```
bayespredict pmean1 pmean2, mean
```

As above, but calculating posterior medians and storing them in new variables `pmedian1` and `pmedian2` specified as a variable stub `pmedian*`

```
bayespredict pmedian*, median
```

95% credible intervals for the second outcome variable `y2`; the lower and upper bounds are stored in `cril2` and `criu2`, respectively

```
bayespredict cril2 criu2, cri outcome(y2)
```

*Simulate and save MCMC replicates of simulated outcomes*

Generate 10 MCMC replicates for the first outcome in the model, and store them as new variables `y1rep1`, ..., `y1rep10` in the current dataset

```
bayesreps y1rep*, nreps(10)
```

As above, but for the second outcome `y2` and storing the results in new variables `y2rep1`, ..., `y2rep10`

```
bayesreps y2rep*, nreps(10) outcome(y2)
```

## Menu

Statistics > Bayesian analysis > Predictions

## Syntax

Syntax is presented under the following headings:

*Compute predictions*

*Compute posterior summaries of simulated outcomes*

*Generate a subset of MCMC replicates of simulated outcomes*

## Compute predictions

*Prediction of selected outcome variables and observations*

```
bayespredict ysimspec [ysimspec ...] [if] [in], saving(filespec) [simopts]
```

*Functions of simulated outcomes, expected values, and residuals*

```
bayespredict (funccspec) [(funccspec) ...] [if] [in], saving(filespec) [simopts]
```

*ysimspec* is `{_ysim#}` or `{_ysim#[numlist]}`, where `{_ysim#}` refers to all observations of the *#*th simulated outcome and `{_ysim#[numlist]}` refers to the selected observations, *numlist*, of the *#*th simulated outcome. `{_ysim}` is a synonym for `{_ysim1}`. With large datasets, specification `{_ysim#}` may use a lot of time and memory and should be avoided. See [Generating and saving simulated outcomes](#).

*funccspec* is one of the following,

```
[label:]@func(arg1 [, arg2])
```

```
[label:]@userprog arg1 [arg2] [, extravars(varlist) passthruopts(string) ]
```

where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; *userprog* is a user-defined Stata program; and *arg1* and *arg2* are one of `{_ysim[#]}`, `{_resid[#]}`, or `{_mu[#]}`. `{_mu#}` refers to expected values, and `{_resid#}` refers to residuals for the *#*th outcome, where the latter is defined as the difference between `{_ysim#}` and `{_mu#}`. *arg2* is primarily for use with user-defined Mata functions; see [Defining test statistics using Mata functions](#).

## Compute posterior summaries of simulated outcomes

*Posterior mean of simulated outcomes*

```
bayespredict [type] newvarspec [if] [in], mean
[outcome(depvar) meanopts simopts]
```

*Posterior median or posterior standard deviation of simulated outcomes*

```
bayespredict [type] newvarspec [if] [in], median|std
[outcome(depvar) simopts]
```

*Credible intervals for simulated outcomes*

```
bayespredict [type] newvarl newvaru [if] [in], cri
[outcome(depvar) criopts simopts]
```

*newvarspec* is *newvar* for single-outcome models and *newvarlist* or *stub\** for multiple-outcome models.

Generate a subset of MCMC replicates of simulated outcomes

```
bayesreps [type] newrepspec [if] [in], nreps(#) [outcome(depvar) simopts]
```

*newrepspec* is *newvar* with `nreps(1)` for a single replicate and *stub\** with `nreps(#)`, where # is greater than 1, for multiple replicates.

<i>meanopts</i>	Description
Main	
<code>mcse(<i>newvar</i>)</code>	create <i>newvar</i> containing MCSEs
Advanced	
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

<i>simopts</i>	Description
Simulation	
<code>rseed(#)</code>	random-number seed
<code>*chains(_all   <i>numlist</i>)</code>	specify which chains to use for computation; default is <code>chains(_all)</code>
<code>dots</code>	display dots every 100 iterations and iteration numbers every 1,000 iterations
<code>dots(#[, every(#)])</code>	display dots as simulation is performed

\*Option `chains()` is relevant only when option `nchains()` is used with `bayesmh`.

<i>criopts</i>	Description
Main	
<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	calculate HPD credible intervals instead of the default equal-tailed credible intervals

Options

Options are presented under the following headings:

- Options for predictions*
- Options for posterior summaries*
- Options for bayesreps*

## Options for predictions

### Main

`saving(filename[, replace])` saves the requested predictions such as simulated outcomes and residuals in `filename.dta`. It also saves auxiliary estimation results in `filename.ster`, which is accessible by specifying `estimates use filename`. The `replace` option specifies to overwrite `filename.dta` and `filename.ster` if they exist. `saving()` is required when computing predictions. The results are saved only for the outcome variables, observations, and functions that are specified with `bayespredict`. See [Prediction dataset](#) for details.

`extravars(varlist)` is for use with user-defined Stata programs. It specifies any variables in addition to dependent and independent variables that you may need to calculate predictions. For example, such variables are offset variables and exposure variables for count-data models.

`passthruopts(string)` is for use with user-defined Stata programs. It specifies a list of options you may want to pass to your program when calculating predictions. For example, these options may contain fixed values of model parameters and hyperparameters.

### Simulation

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. With one chain, `rseed(#)` is equivalent to typing `set seed #` prior to calling `bayespredict`; see [\[R\] set seed](#). With multiple chains, you should use `rseed()` for reproducibility; see [Reproducing results](#) in [\[BAYES\] bayesmh](#).

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with [bayesmh](#).

`dots` and `dots(#)` specify to display dots during simulation. With multiple chains, these options affect all chains. `dots(#)` displays a dot every `#` iterations. If `dots(..., every(#))` is specified, then an iteration number is displayed every `#`th iteration instead of a dot. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100, every(1000))`.

## Options for posterior summaries

### Main

`mean` calculates posterior means of a simulated outcome variable and stores them as a new variable in the current dataset.

`median` calculates posterior medians of a simulated outcome variable and stores them as a new variable in the current dataset.

`std` calculates posterior standard deviations of a simulated outcome variable and stores them as a new variable in the current dataset.

`mean`, `median`, and `std` can compute results for all simulated outcome variables or for a specific one. To compute results for all simulated outcome variables, you specify `p` new variables, where `p` is the number of dependent variables. Alternatively, you can specify `stub*`, in which case these options will store the results in variables `stub1`, `stub2`, ..., `stubp`. To compute the results for a specific simulated outcome variable, you specify one new variable and, optionally, the outcome variable name in option `outcome()`; if you omit `outcome()`, the first outcome variable is assumed.



`cri` calculates credible intervals for a simulated outcome variable and stores the corresponding lower and upper bounds in two new variables in the current dataset. For multiple-outcome models, it computes the results for the outcome variable as specified in option `outcome()` or, by default, for the first outcome variable.

`outcome(depvar)` is for use with multiple-outcome models when computing posterior summaries of simulated outcomes. It specifies for which simulated outcome posterior summaries are to be calculated. `outcome()` should contain a name of the outcome (dependent) variable. The default is the first outcome variable. `outcome()` may not be combined with the *newvarlist* or *stub\** specification.

`mcse(newvar)` is for use in a combination with option `mean`. It adds *newvar* of storage type *type* containing MCSEs for the posterior means of a simulated outcome variable. If multiple variables are specified with `bayespredict`, *newvar* is used as a stub *newvar\**.

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is `clevel(95)` or as set by [\[BAYES\] set clevel](#). This option requires that `cri` also be specified.

`hpd` calculates the HPD credible intervals instead of the default equal-tailed credible intervals. This option requires that `cri` also be specified.

---

#### Simulation

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. With one chain, `rseed(#)` is equivalent to typing `set seed #` prior to calling `bayespredict`; see [\[R\] set seed](#). With multiple chains, you should use `rseed()` for reproducibility; see [Reproducing results](#) in [\[BAYES\] bayesmh](#).

`chains(_all | numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with [bayesmh](#).

`dots` and `dots(#)` specify to display dots during simulation. With multiple chains, these options affect all chains. `dots(#)` displays a dot every `#` iterations. If `dots(..., every(#))` is specified, then an iteration number is displayed every `#`th iteration instead of a dot. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100, every(1000))`.

---

#### Advanced

The advanced options are available only in a combination with option `mean`.

`batch(#)` specifies the length of the block for calculating batch means and MCSE using batch means. The default is `batch(0)`, which means no batch calculations. When `batch()` is not specified, MCSE is computed using effective sample sizes instead of batch means. Option `batch()` may not be combined with `corrlag()` or `corrtol()`.

`corrlag(#)` specifies the maximum autocorrelation lag used for calculating effective sample sizes. The default is `min{500, mcmcsizes()/2}`. The total autocorrelation is computed as the sum of all lag-*k* autocorrelation values for *k* from 0 to either `corrlag()` or the index at which the autocorrelation becomes less than `corrtol()` if the latter is less than `corrlag()`. Options `corrlag()` and `batch()` may not be combined.

`corrtol(#)` specifies the autocorrelation tolerance used for calculating effective sample sizes. The default is `corrtol(0.01)`. For a given model parameter, if the absolute value of the lag-*k*

autocorrelation is less than `corrtol()`, then all autocorrelation lags beyond the  $k$ th lag are discarded. Options `corrtol()` and `batch()` may not be combined.

## Options for bayesreps

### Main

`nreps(#)` specifies the number of MCMC replicates of simulated outcomes to be drawn at random from the entire sample of MCMC replicates. `#` must be an integer between 1 and the MCMC sample size, inclusively. The generated replicates are stored as new variables in the current dataset. For a single replicate, `nreps(1)`, you specify one new variable name. For multiple replicates, you specify a *stub\**, in which case the replicates will be stored in variables *stub1*, *stub2*, ..., *stubR*, where  $R$  is the number of replicates specified in `nreps()`.

`outcome(depvar)` is for use with multiple-outcomes models when generating MCMC replicates of simulated outcomes using `bayesreps`. It specifies for which simulated outcome MCMC replicates are to be generated. The default is to use the first outcome variable. You can specify other outcome (dependent) variable names in `outcome()`.

### Simulation

`rseed(#)` sets the random-number seed. This option can be used to reproduce results. With one chain, `rseed(#)` is equivalent to typing `set seed #` prior to calling `bayespredict`; see [R] [set seed](#). With multiple chains, you should use `rseed()` for reproducibility; see [Reproducing results](#) in [BAYES] [bayesmh](#).

`chains(_all|numlist)` specifies which chains from the MCMC sample to use for computation. The default is `chains(_all)` or to use all simulated chains. Using multiple chains, provided the chains have converged, generally improves MCMC summary statistics. Option `chains()` is relevant only when option `nchains()` is specified with [bayesmh](#).

`dots` and `dots(#)` specify to display dots during simulation. With multiple chains, these options affect all chains. `dots(#)` displays a dot every `#` iterations. If `dots(..., every(#))` is specified, then an iteration number is displayed every `#`th iteration instead of a dot. `dots(, every(#))` is equivalent to `dots(1, every(#))`. `dots` displays dots every 100 iterations and iteration numbers every 1,000 iterations; it is a synonym for `dots(100, every(1000))`.

## Remarks and examples

Remarks are presented under the following headings:

- Overview of Bayesian predictions*
  - Prior and posterior predictive distributions*
  - Simulated outcomes*
  - Posterior predictive checking and replicated outcomes*
- Using bayespredict and bayesreps*
  - Generating and saving simulated outcomes*
  - Defining test statistics using Mata functions*
  - User-defined Stata programs*
  - Posterior summaries of simulated outcomes*
  - Prediction dataset*

Examples are presented under the following headings:

- Bayesian predictions*
- Posterior predictive inference*
- Out-of-sample prediction*

## Overview of Bayesian predictions

Bayesian analysis rests on the assumptions that model parameters are random quantities distributed according to some prior beliefs and that the data, once observed, are fixed. The main goal of Bayesian inference is to estimate the posterior distribution of model parameters, which combines the prior beliefs with evidence from the observed data, and form inferences about these parameters. But what if we want to estimate a future outcome value? This is one of the goals of Bayesian prediction.

Bayesian predictions are useful in a wide range of applications. They can be used as optimal predictors in forecasting, optimal classifiers in classification problems, imputations for missing data, and more. They are also important for checking model goodness of fit.

Bayesian prediction differs from frequentist prediction. Prediction, in a frequentist sense, is a deterministic function of estimated model parameters. For example, in a linear regression, the linear predictor, which is a linear combination of estimated regression coefficients and observed covariates, is used to predict values of continuous outcomes. Bayesian predictions, on the other hand, are functions of simulated outcomes and are thus stochastic quantities. Simulated outcomes are new outcome values generated from the so-called posterior predictive distribution, which we describe next.

### Prior and posterior predictive distributions

Before the data  $\mathbf{y}$  are observed, the distribution of  $\mathbf{y}$  is

$$p(\mathbf{y}) = \int p(\mathbf{y}, \boldsymbol{\theta}) d\boldsymbol{\theta} = \int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \quad (1)$$

where  $p(\mathbf{y}|\boldsymbol{\theta})$  is the likelihood of  $\mathbf{y}$  given model parameters  $\boldsymbol{\theta}$  and  $p(\boldsymbol{\theta})$  is the prior distribution for  $\boldsymbol{\theta}$ .  $p(\mathbf{y})$  is the so-called prior predictive distribution, which is more commonly known as the marginal distribution of  $\mathbf{y}$ .

Suppose that  $\mathbf{y}^{\text{obs}}$  are observed data and  $\mathbf{y} = \mathbf{y}^{\text{new}}$  are new, unobserved (future) data. The posterior predictive distribution of  $\mathbf{y}^{\text{new}}$  is

$$p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}) = \int p(\mathbf{y}^{\text{new}}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})d\boldsymbol{\theta} \quad (2)$$

where  $p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})$  is the posterior distribution of  $\boldsymbol{\theta}$ . You can think of a posterior predictive distribution (2) as a prior predictive distribution (1) updated after observing the data  $\mathbf{y}^{\text{obs}}$ .

### Simulated outcomes

Like the posterior distribution of model parameters, the predictive distribution  $p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}})$  usually does not have a closed form and must be approximated. The goal of Bayesian prediction is to simulate data from  $p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}})$ . We will refer to these data as *simulated outcomes*,  $\mathbf{y}^{\text{sim}}$ .

Formula (2) provides a way of simulating new outcome values by using a two-step procedure. First, model parameters  $\boldsymbol{\theta}^*$  are simulated from their posterior distribution  $p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})$ . Then, the new outcome values  $\mathbf{y}^{\text{sim}}$  are simulated from the likelihood model  $p(\mathbf{y}^{\text{sim}}|\boldsymbol{\theta}^*)$  using the simulated model parameters from step 1. These two steps are repeated for a prespecified number of MCMC iterations,  $T$ . The result is an MCMC sample of simulated outcomes,  $(\mathbf{y}^{\text{sim},1}, \mathbf{y}^{\text{sim},2}, \dots, \mathbf{y}^{\text{sim},T})$ . This sample is used to estimate the posterior predictive distribution.

Thus, unlike classical prediction, which produces a single value for each observation, Bayesian prediction produces a sample of  $T$  simulated values for each observation. If you have  $n$  observations in the dataset, the result of a Bayesian prediction will be a  $T \times n$  matrix (for each outcome or dependent variable). Therefore, Bayesian predictions are often computed for a subset of observations or for various summaries over observations such as means, quantiles, minimum and maximum values, and so on. Sometimes, a smaller sample of  $R \ll T$  MCMC replicates of simulated outcomes is used to explore the posterior distribution of simulated outcomes. In other cases, posterior summaries over the MCMC replicates such as posterior means and medians of simulated outcomes may be of interest.

## Posterior predictive checking and replicated outcomes

In addition to predicting future observations, Bayesian prediction is useful for model checking. Model checking is accomplished by performing the so-called posterior predictive checks, which compare various characteristics of the posterior predictive distribution with those observed in the data.

The concept of replicated data or replicated outcomes arises in the context of posterior predictive checking for regression-type models. In a regression setting, the posterior predictive distribution also depends on the covariate-data matrix  $X$ ,  $p(\mathbf{y}^{\text{new}} | \mathbf{y}^{\text{obs}}) = p(\mathbf{y}^{\text{new}} | \mathbf{y}^{\text{obs}}, X)$ . The data matrix  $X$  may contain the observed values that were used to fit the Bayesian model,  $X^{\text{obs}}$ , or the new values,  $X^{\text{new}}$ . *Replicated outcomes* are outcomes simulated from the posterior predictive distribution,  $p(\mathbf{y}^{\text{new}} | \mathbf{y}^{\text{obs}}, X^{\text{obs}})$ , using the observed covariate data. In other words, the replicated outcomes are the outcomes we would observe if we repeated our experiment again. We will denote replicated outcomes as  $\mathbf{y}^{\text{rep}}$ .

Replicated outcomes are also known as in-sample predictions, whereas outcomes simulated using new covariate data,  $X^{\text{new}}$ , are known as out-of-sample predictions. In-sample predictions are useful for diagnostic checks. Out-of-sample predictions can be used for forecasting and model validation. In the latter case, the data are split into training and test subsamples: the training subsample is used to fit a Bayesian model, and the test subsample is used to assess prediction accuracy of the fitted model.

Posterior predictive checking is performed by comparing the distribution (or certain aspects of it) of the replicated data to that of the observed data. This can be done visually by examining histograms and quantile plots. More formally, discrepancy measures such as a mean, minimum, and maximum statistics computed for the replicated data and for the observed data can be compared using posterior predictive  $p$ -values; see [\[BAYES\] bayesstats ppvalues](#) for details.

It is important to realize the difference between MCMC diagnostic checks ([Convergence of MCMC](#) in [\[BAYES\] bayesmh](#)) and posterior predictive checks. The former examines the properties of MCMC sampling, whereas the latter inspects how well the specified Bayesian model describes the observed data. But these two types of checks are related—an ill-fitting model lowers the MCMC sampling efficiency and may even lead to nonconvergence of the MCMC algorithm.

For in-depth coverage of Bayesian predictions and posterior predictive inference, see [Meng \(1994\)](#), [West \(1986\)](#), [Tsui and Weerahandi \(1989\)](#), [Gelman, Meng, and Stern \(1996\)](#), [Gelman and Rubin \(1992\)](#), and [Gelman et al. \(2014\)](#), to name a few.

## Using bayespredict and bayesreps

`bayespredict` computes Bayesian predictions using current estimation results produced by the `bayesmh` command and saves them in a separate Stata dataset. Bayesian predictions include simulated outcomes, which are samples from the [posterior predictive distribution](#) of the fitted Bayesian model, and their functions. You can also compute posterior summaries of simulated outcomes and store them as new variables in the current dataset.

To compute Bayesian predictions, you must specify the `saving()` option with `bayespredict` to save the [prediction results](#); see [Generating and saving simulated outcomes](#). To compute posterior summaries, you must specify one or more new variable names and the corresponding option such as `mean` for posterior mean and `std` for posterior standard deviation; see [Posterior summaries of simulated outcomes](#).

`bayesreps` generates a random subset of [MCMC replicates](#) of simulated outcomes from the entire MCMC sample and stores them as new variables in the current dataset. This command is useful for checking model fit. The number of replicates is specified in the `nreps(#_reps)` option. With multiple replicates, you must specify a variable `stub*` with `bayesreps`, and the command will generate new variables `stub1`, `stub2`, ..., `stub#_reps` in the current dataset. For multiple-outcome models, the replicates are produced for one outcome at a time. The first outcome is the default, but you can specify a different outcome variable in the `outcome()` option.

Both `bayespredict` and `bayesreps` require that `bayesmh`'s MCMC simulation dataset be saved prior to their execution. You can save MCMC simulation results by specifying the `saving()` option with `bayesmh` during or after estimation; see [Storing estimation results after Bayesian estimation in \[BAYES\] Bayesian postestimation](#).

Both commands produce stochastic results. Use the `rseed()` option for reproducibility. Depending on the number of observations, the specified MCMC sample size, and model complexity, the computations may be time consuming. Options `dots` and `dots()` may be useful in this case to monitor the progress. They display a dot for each simulation performed.

`bayespredict` and `bayesreps` can be used to make in-sample or out-of-sample predictions; see [Description in \[R\] predict](#) for how to specify such predictions.

## Generating and saving simulated outcomes

Generating and saving simulated outcomes is the main usage of `bayespredict`, which requires the `saving()` option when generating simulated outcomes. The simplest specification is

```
. bayespredict {_ysim1}, saving(filename)
```

which generates the simulated values for the first outcome variable and saves them in `filename.dta`. You can also use `{_ysim}` as a synonym for `{_ysim1}`.

The above specification produces the prediction dataset `filename.dta`, which contains  $T$  observations and  $n$  variables, where  $T$  is the MCMC sample size used by `bayesmh` and  $n$  is the number of observations in the original dataset. That is, an MCMC sample of size  $T$  is generated for each observation of the outcome variable.

For example, if our dataset has 100 observations and we use an MCMC sample of size 10,000 during simulation, `bayespredict` will produce the prediction dataset `filename.dta` with 10,000 observations and 100 variables. This specification may not always be feasible, especially for large datasets, or even necessary.

You would rarely need to simulate and store all observations for all outcome variables. More likely, if you are performing model diagnostics, you may be interested only in several test statistics, which you can simulate without storing the simulated outcomes; see [Defining test statistics using Mata functions](#). Or you may be interested only in posterior summaries of simulated outcomes; see [Posterior summaries of simulated outcomes](#). Or you may need to explore only a small random subset of MCMC replicates of simulated outcomes, which you can obtain by using the `bayesreps` command. Or if you are interested in forecasting, you may need to simulate values for only a few new data points.

For example, suppose we want to simulate outcome values for 10 new observations only, which are stored in observations 101 through 110 in our original dataset. We can do this using

```
. bayespredict {_ysim1[101/110]}, saving(filename)
```

or, equivalently, using

```
. bayespredict {_ysim1} in 101/110, saving(filename)
```

The two specifications above are more efficient with respect to execution time and storage.

The full syntax of `bayespredict` for simulating all variables and all observations is

```
. bayespredict {_ysim1} {_ysim2} ..., saving(filename)
```

where you specify `{_ysim#}` for the `#`th outcome variable. The order of variables is determined by the order in which they were specified with `bayesmh`.

If you need to predict multiple outcomes, it may be more efficient with regard to storage to simulate them separately. Remember that the total number of variables in the prediction dataset may not exceed the current `c(maxvar)` setting. Because `bayespredict` stores additional variables, the number of specified outcome observations may not exceed `floor((c(maxvar)-3)/2)`; see [Prediction dataset](#).

By default, `bayespredict` computes out-of-sample predictions. This may sometimes lead to missing predicted observations, for instance, when some of the covariates contain missing values. In the context of `bayespredict` when simulating outcomes, residuals, and expected values, this implies that the prediction dataset may contain variables containing all missing observations. Recall that the variables in the prediction dataset correspond to the observations in the original dataset. In such cases, to reduce the size of the prediction dataset, you may consider restricting the prediction sample to the estimation sample, if `e(sample)`; or specifying a subset of observations using [numlist](#), for example, `_ysim[numlist]`; or specifying the subset of interest by using `if` and `in`.

## Defining test statistics using Mata functions

Instead of simulating all observations for your outcomes of interest, you may be interested in obtaining only some summary statistics such as sample means, medians, smallest and largest observations, and standard deviations calculated over these observations. This is commonly used when performing posterior predictive checks; see [Posterior predictive inference](#).

Test statistics are scalar functions of observed (or simulated) outcome values. Let  $\mathbf{y}$  be an outcome variable in a dataset of size  $n$  and let  $\mathbf{y}^{\text{sim}} = (y_1^{\text{sim}}, y_2^{\text{sim}}, \dots, y_n^{\text{sim}})^T$  denote one simulated outcome sample given as a column vector. A test statistic  $T(\mathbf{y}^{\text{sim}})$  summarizes the column vector  $\mathbf{y}^{\text{sim}}$  by a single number. For example, the mean statistic is defined as

$$T(\mathbf{y}^{\text{sim}}) = \frac{1}{n}(y_1^{\text{sim}} + \dots + y_n^{\text{sim}}) = \bar{\mathbf{y}}^{\text{sim}}$$

In `bayespredict`, test statistics can be defined using Mata functions or Stata programs. Here we focus on the specifications using Mata functions; see [User-defined Stata programs](#) for Stata programs. Note that if you need to compute a test quantity,  $T(\mathbf{y}, \boldsymbol{\theta})$ , that directly uses model parameters  $\boldsymbol{\theta}$ , you must use Stata programs.

`bayespredict` supports Mata functions that return a scalar and accept one or two column vectors as arguments. You can specify the following as the arguments to the Mata functions: simulated outcomes, `{_ysim#}`; simulated residuals, `{_resid#}`; and expected outcome values, `{_mu#}`. `{_resid#}` is defined as the difference between `{_ysim#}` and `{_mu#}`. (Specifications `{_resid#}` and `{_mu#}` are not available for ordinal models.) You can also use `{_ysim}`, `{_resid}`, and `{_mu}` as synonyms

for `{_ysim1}`, `{_resid1}`, and `{_mu1}`, respectively. If you used `if` or `in` with `bayespredict` to restrict the prediction sample or specified only a subset of observations, that is, `{_ysim[1/10]}`, the column vectors passed to Mata functions as arguments will contain only the available observations.

Suppose we want to produce an MCMC sample of means of the first simulated outcome. We can specify

```
. bayespredict (@mean({_ysim1}), saving(...)
```

Similarly, we can produce an MCMC sample of means for the residuals of the first simulated outcome

```
. bayespredict (resmean: @mean({_resid1}), saving(...)
```

In the above, we also labeled our prediction as `resmean`. We can use this label to refer to this prediction in other Bayesian postestimation commands such as `bayesstats ppvalues` and `bayesstats summary`. If we do not specify our own labels, the default labels will be used for each prediction. The default label is `arg1_func`, where `arg1` is the first function argument and `func` is the name of the function. For instance, in our first example, the default label `_ysim1_mean` will be used.

You will typically specify only one argument with most official Mata functions. The support of two arguments is provided primarily for calculating more complicated test statistics using user-defined Mata functions. For example, let's define a new Mata function that calculates the sum of squared Pearson residuals assuming a Poisson model.

We define a Mata function, `sumpresid()`, that calculates the squared Pearson residuals as the squared difference between the simulated outcome vector, `ysim`, and expected values, `mu`, divided by the variance, which is also `mu` for a Poisson model. The result is the sum of these squared standardized differences.

```
mata:
    real scalar sumpresid(real colvector ysim, real colvector mu) {
        return (sum((ysim-mu)^2:/mu))
    }
end
```

Then, we can call `bayespredict` with the following specification to compute the sum of squared Pearson residuals for the first outcome in the model:

```
. bayespredict (@sumpresid({_ysim1}, {_mu1})), saving(...)
```

Mata functions can be used only with one outcome at a time. That is, specifications that refer to two outcomes such as `@myprog({_ysim1}, {_ysim2})`, `@myprog({_ysim1}, {_mu2})`, or `@myprog({_ysim1}, {_resid2})` are not allowed.

Mata functions are preferable to Stata programs because of speed, but Stata programs provide more flexibility to compute complicated functions; see [User-defined Stata programs](#) below.

## User-defined Stata programs

Mata functions (see [Defining test statistics using Mata functions](#)) are more efficient and faster in computing simple test statistics and test quantities, but they have limitations. For example, you cannot access model parameters within Mata functions. You can within Stata programs. Although executing Stata programs may be much slower, they provide more flexibility for computing test quantities.



A Stata program must have the following format in order to be used by bayespredict:

```

program userprog
    version 16.0
    args res simvar1 [simvar2]
    ... computation ...
    scalar 'res' = ...
end

```

The first argument, *res*, contains the name of a temporary scalar to store the final result. The second argument, *simvar1*, and the third (optional) argument, *simvar2*, contain the names of temporary variables, which store the simulation results for the quantities specified as program arguments *arg1* and *arg2* with bayespredict:

```
. bayespredict ([label]: @userprog arg1 [arg2]), saving(...) ...
```

*arg1* and *arg2* may be one of {\_ysim#}, {\_mu#}, or {\_resid#}, but they should refer to the same outcome variable; that is, they must use the same #. *label* is the label for the computed prediction result that can be used later to refer to this result within other Bayesian postestimation commands such as bayesstats summary. If we do not specify our own label, the default label will be used for each prediction. The default label is *arg1\_userprog*, where *arg1* is the first program argument and *userprog* is the name of the program.

Recall the sumpresid() Mata function defined in the [previous](#) section. Below, we replicate the same computation but now using the Stata program.

```

program sumpresidprog
    version 16.0
    args sum ysim mu
    tempvar presid
    generate double 'presid' = ('ysim'-'mu')^2/'mu'
    summarize 'presid', meanonly
    scalar 'sum' = r(sum)
end

```

We can then call bayespredict with the following specification,

```
. bayespredict (@sumpresidprog {_ysim1} {_mu1}), saving(...)
```

to compute this statistic for the first outcome. Because we did not specify our own label in the above, the default label *\_ysim1\_sumpresidprog* will be used.

Generally, our Stata program should use a proper “touse” variable, which marks the prediction sample of bayespredict. Unlike Mata functions, the prediction results passed to Stata programs as arguments will contain all observations. However, the observations outside the prediction sample will contain missing values. Nevertheless, it is good practice to always use the touse variable in the calculations.

```

program sumpresidprog
    version 16.0
    args sum ysim mu
    local touse $BAYESPR_touse
    tempvar presid
    generate double 'presid' = ('ysim'-'mu')^2/'mu' if 'touse'
    summarize 'presid' if 'touse', meanonly
    scalar 'sum' = r(sum)
end

```

The global macro \$BAYESPR\_touse contains a temporary name of a binary variable that marks the prediction sample, which we now use in our calculations.



One flexibility of Stata programs is that we can access model parameters within them. In the above programs, we used precomputed expected values, `mu`. We can compute these values manually by using the simulated model parameters and observed variables.

```
program sumpresidprogmu
  version 16.0
  args sum ysim
  local touse $BAYESPR_touse
  local theta $BAYESPR_theta //<--New line
  tempvar xb mu //<--New line
  matrix score double 'xb' = 'theta' if 'touse' //<--New line
  qui generate double 'mu' = invlogit('xb') if 'touse' //<--New line
  tempvar presid
  generate double 'presid' = ('ysim'-'mu')^2/'mu' if 'touse'
  summarize 'presid' if 'touse', meanonly
  scalar 'sum' = r(sum)
end
```

To compute expected values, we need to compute the linear predictor. To compute the linear predictor, we need coefficient estimates. The coefficient estimates are provided in a temporary matrix (row vector) with the name stored in the global macro `$BAYESPR_theta`. The columns of this temporary matrix are labeled properly with the names of the corresponding predictors, so we can use `matrix score` (see [P] [matrix score](#)) to easily compute the linear predictor. We then use the inverse-logit function to compute expected values (probabilities) from the linear predictions. The rest of the program is the same as earlier.

We call the above program using the following `bayespredict` specification:

```
. bayespredict (@sumpresidprogmu {_ysim1}), saving(...)
```

See [example 8](#).

For some programs, you may need to pass additional variables or contents of command options. You can use `extravars()` and `passthruopts()` for that; see [Options for predictions](#).

You can access the following global macros from the Stata programs used with `bayespredict`.

Global macros	Description
<code>\$BAYESPR_theta</code>	name of a temporary matrix (row vector) of scalar parameters; stripes are properly named after the names of model parameters
<code>\$BAYESPR_matrix_mname</code>	name of a temporary matrix containing simulated matrix parameter <i>mname</i>
<code>\$BAYESPR_touse</code>	variable containing 1 for the observations to be used; 0 otherwise
<code>\$BAYESPR_extravars</code>	<i>varlist</i> specified in <code>extravars()</code>
<code>\$BAYESPR_passthruopts</code>	options specified in <code>passthruopts()</code>

**Posterior summaries of simulated outcomes**

In some applications, we may not need the actual simulated outcomes but rather their posterior summaries such as posterior means, medians, and standard deviations. For this purpose, `bayespredict` offers the `mean`, `median`, `std`, and `cri` options to compute posterior means, medians, standard deviations, and credible intervals. When you specify these options, the prediction results are stored in the specified new variables in the current dataset. You do not need to specify the `saving()` option in this case because the high-dimensional simulation outcomes are not saved, only their posterior summaries.

With `mean`, `median`, and `std`, you can compute results for one outcome variable at a time or for all outcome variables. In the first case, you specify a new variable name and the name of the outcome (dependent) variable in the `outcome()` option. If you omit `outcome()`, the first outcome variable will be used. To compute results for all outcome variables, you specify a new variable name for each outcome or `stub*`, in which case the new variables will be named `stub1`, `stub2`, and so on.

When you compute posterior means, you can also specify the `mcse(newvar)` option to compute their corresponding MCSEs. If posterior means are computed for multiple outcome variables, `newvar` is used as `stub*` to store MCSEs for each outcome in `newvar1`, `newvar2`, and so on.

With `cri`, you specify two new variable names to contain the lower and upper credible bounds. You can compute results only for one outcome variable at a time, which you specify in the `outcome()` option. If you omit this option, the first outcome variable is assumed. You can specify the `clevel()` option to change the default 95% credible level and the `hpd` option to calculate HPD credible intervals instead of the default equal-tailed intervals.

All computed results are stochastic. You should specify the `rseed()` option for reproducibility. Also see [Syntax](#) for other available simulation options, [simopts](#).

## Prediction dataset

`bayespredict` saves prediction results in a dataset `filename.dta` as specified in the `saving(filename)` option. In addition, `bayespredict` stores auxiliary estimation results, described in [Stored results](#), in `filename.ster`. This file is used by other postestimation commands such as `bayesstats summary` when summarizing the simulated prediction quantities.

The format of the `filename.dta` file is similar to the simulation dataset created by the `bayesmh` command. The first two variables are `_chain` and `_index`, which store the respective chain and MCMC iteration identifiers. Following are the variables containing simulated values for the  $\#_1$ th outcome variable and the  $\#_2$ th observation, `_ysim $\#_1$ _ $\#_2$` , if any, and the corresponding expected outcome values, `_mu $\#_1$ _ $\#_2$` . For any function of simulated outcomes or residuals specified with `bayespredict`, there are two variables in the dataset named `label` and `_obs_label`, where `label` is the specified function or program label. Variable `label` contains the MCMC sample of values of the function. Variable `_obs_label` contains the observed values of the function, which are computed by substituting the simulated outcome for the observed outcome variable in the function specification. This variable is consumed by [\[BAYES\] bayesstats pvalues](#). Finally, the `_frequency` variable is the last variable in the prediction dataset. It always contains one in the prediction dataset and is provided purely for the consistency with the simulation dataset, where it records the frequency of duplicate sets of model parameters.

If `bayespredict` is specified with  $p$  simulated outcomes, each with  $n$  observations, and with  $k$  functions or programs, then the prediction dataset will contain  $2pn + 2k + 3$  variables. The number of observations in the prediction dataset is determined by the MCMC sample size,  $T$ , used by `bayesmh`.

After your analysis, if you no longer need the prediction dataset, remember to remove both `filename.dta` and `filename.ster`.

## Bayesian predictions

Consider the rare infectious disease example from [Hoff \(2009\)](#) that we analyzed in [Beta-binomial model](#) of [\[BAYES\] bayesmh](#). A small random sample of 20 subjects from a city is checked for infection, and none is observed to be infected. The parameter of interest  $\theta$ ,  $\theta \in [0, 1]$ , is the proportion of infected individuals in the city. The outcome  $y$  is the number of infected subjects in the sample of 20. The sampling distribution for the outcome  $y$  is thus assumed to be binomial,  $y|\theta \sim \text{binomial}(20, \theta)$ .

Our observed data contain one observation that is zero because we did not observe any infected subjects in our sample. We can easily generate these data as follows:

```
. set obs 1
number of observations (_N) was 0, now 1
. generate byte y = 0
```

Following the examples in *Beta-binomial model* (except we are using a different random-number seed here), we assume a  $\text{beta}(2, 20)$  prior for  $\theta$  and use `bayesmh` to fit the resulting beta-binomial model.

```
. bayesmh y, likelihood(dbinomial({theta}, 20))
> prior({theta}, beta(2, 20)) saving(betabin_mcmc) rseed(16)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  y ~ binomial({theta},20)
Prior:
  {theta} ~ beta(2,20)
```

---

Bayesian binomial model	MCMC iterations	=	12,500
Random-walk Metropolis-Hastings sampling	Burn-in	=	2,500
	MCMC sample size	=	10,000
	Number of obs	=	1
	Acceptance rate	=	.4627
	Efficiency	=	.1446

Log marginal-likelihood = -1.1575104

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
theta	.0476128	.0320509	.000843	.0406464	.0057875	.1251631

---

```
file betabin_mcmc.dta saved
```

The posterior mean for `{theta}`, which is also the probability that a subject from a sample of 20 will be infected, is estimated to be 0.0476. Thus, we would expect  $20 \times 0.0476 = 0.952$  infected subjects in a sample of 20.

Let’s explore various Bayesian predictions for this beta-binomial model. The relevant examples are presented under the following headings:

- Example 1: Predicting the number of infected subjects*
- Example 2: Summarizing prediction results*
- Example 3: Expressions of individual prediction results*
- Example 4: Visualizing prediction results*
- Example 5: Posterior summaries of simulated outcomes*

➤ Example 1: Predicting the number of infected subjects

Let’s predict the number of infected subjects, our outcome, assuming the fitted beta-binomial model. To do this in a Bayesian framework, we need to simulate the outcome from its posterior predictive distribution. We can use `bayespredict` to do this.

To use `bayespredict`, we must first save our MCMC simulation results from `bayesmh` in a dataset, which we already did by specifying the `saving(betabin_mcmc)` option with `bayesmh`. If you forget to specify this option during estimation, you can always do it after by typing

```
. bayesmh, saving(betabin_mcmc)
```

We simulate the outcome by specifying `{_ysim}` with `bayespredict` and save the simulated data in `betabin_pred.dta`; the `saving()` option is required with `bayespredict` when simulating Bayesian predictions. Because the command uses simulation, we also specify the `rseed()` option for reproducibility.

```
. bayespredict {_ysim}, saving(betabin_pred) rseed(16)
Computing predictions ...
file betabin_pred.dta saved
file betabin_pred.ster saved
```

The computation may be time consuming, so the command displays `Computing predictions ...` to inform you that the computation is in progress. You may also specify the `dots` or `dots()` option to see the dots as simulations are performed.

In addition to saving prediction results in a Stata dataset, `bayespredict` also saves auxiliary estimation results in the `betabin_pred.ster` file. This file is used by other postestimation commands such as `bayesstats summary` when summarizing the simulated prediction quantities. Remember to remove this file in addition to your prediction dataset when you no longer need them.

The `bayespredict` command simulates  $T$  outcome values for each specified outcome and for each specified observation.  $T$  is the MCMC sample size used by `bayesmh`. The outcome values are simulated for each set of  $T$  MCMC estimates of model parameters generated by `bayesmh`. Our `bayespredict` specification `{_ysim}` is equivalent to `{_ysim1}` and refers to all observations of the first outcome. In our example, we have only one observation and one outcome, and the default MCMC sample size is 10,000. Thus, `betabin_pred.dta` contains one simulated variable, `_ysim1_1`, and 10,000 observations, in addition to other auxiliary variables such as `chain` and iteration number identifiers; see [Prediction dataset](#).

```
. describe using betabin_pred
Contains data
  obs:      10,000
  vars:      5
                                28 Mar 2019 16:44
```

variable name	storage type	display format	value label	variable label
<code>_chain</code>	int	%8.0g		Chain identifier
<code>_index</code>	int	%8.0g		Iteration number
<code>_ysim1_1</code>	double	%10.0g		Simulated y, obs. #1
<code>_mu1_1</code>	double	%10.0g		Expected values for y, obs. #1
<code>_frequency</code>	byte	%8.0g		Frequency weight

Sorted by:

In this dataset, `_ysim1_1` represents an MCMC sample of size 10,000 from the posterior predictive distribution of  $y$  for the first observation. If we had more observations, say, 100, the dataset would have contained 100 variables, `_ysim1_1`, `_ysim1_2`, ..., `_ysim1_100`, one for each observation. In the prediction dataset, the observations are MCMC replicates, and the variables are outcome values for each observation and each outcome from the data that were used to fit the model.



➤ Example 2: Summarizing prediction results

We can summarize our prediction results like any other Bayesian model parameter. For example, we can calculate standard posterior summaries for `{_ysim}` by using `bayesstats summary`.

```
. bayesstats summary {_ysim} using betabin_pred
```

Posterior summary statistics				MCMC sample size = 10,000		
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_ysim1_1	.9526	1.145899	.020218	1	0	4

The calculated posterior predictive mean is 0.95, which agrees with our earlier computation of  $20 \times 0.0476 = 0.952$  using the posterior mean estimate of  $\theta$ , 0.0476. Under our Bayesian model, we should expect to observe roughly 1 infected individual in a sample of 20, which is comparable with our observed data with no infected subjects.

Generally, we should be careful when using `{_ysim}` with Bayesian postestimation commands because it refers to all observations of the outcome variable. A better approach is to use a subset of observations, `{_ysim[numlist]}`, such as `{_ysim[1/10]}`. In our example, we have only one observation, so this specification is equivalent to specifying only the first observation, `{_ysim[1]}`.



➤ Example 3: Expressions of individual prediction results

We can compute posterior summaries for the expressions involving the individual values, `{_ysim[#]}`, where `#` refers to an observation. For instance, let's calculate the probability of observing 0 infected subjects in our sample of 20. Recall that our only observation records the number of observed infected subjects. We can estimate the probability that the outcome value is 0 as a proportion of 0 values of our simulated outcome in a sample of 10,000 MCMC replicates. We can do this by specifying the expression `{_ysim[1]}==0` in `bayesstats summary`.

```
. bayesstats summary (prob0:{_ysim[1]}==0) using betabin_pred
```

Posterior summary statistics				MCMC sample size = 10,000		
prob0 : _ysim1_1==0						
	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
prob0	.4479	.497303	.00708	0	0	1

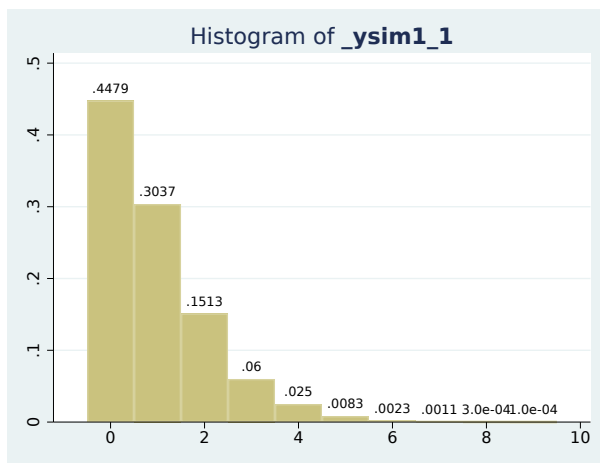
The posterior predictive mean (probability) for observing 0 infected subjects in the sample of 20 is 0.45, with a posterior predictive standard deviation of 0.5.



### ► Example 4: Visualizing prediction results

We can use graphical tools such as the histogram to summarize the posterior predictive distribution.

```
. bayesgraph histogram {_ysim[1]} using betabin_pred, discrete addlabels
```



The mass of the posterior predictive distribution for the number of infected subjects is concentrated on small numbers such as 0, 1, and 2 and thus agrees with what we observed in our sample.

### ► Example 5: Posterior summaries of simulated outcomes

We can compute the posterior mean of the simulated outcome and save it in the current dataset as a new variable.

```
. bayespredict pmean, mean rseed(16)
```

Computing predictions ...

```
. summarize pmean
```

Variable	Obs	Mean	Std. Dev.	Min	Max
pmean	1	.9526	.	.9526	.9526

The sample mean of `pmean` is an estimate of the posterior predictive mean of the outcome  $y$  and is the same as the one we obtained earlier by using `bayesstats summary`. Notice that we obtained the exact same values only because we used the same random-number seed, `rseed(16)`, with `bayespredict` when simulating the outcome `{_ysim}` and the posterior mean `pmean`.

If you need only posterior summaries of simulated outcomes, the above approach is preferable because it does not create a potentially large prediction dataset containing all MCMC replicates.

As the final step, we remove all the datasets created by `bayesmh` and `bayespredict` because we no longer need them, but you may choose to keep yours.

```
. erase betabin_mcmc.dta
. erase betabin_pred.dta
. erase betabin_pred.ster
```

Posterior predictive inference

To illustrate posterior predictive checking, we adapt an example described in [Gelman et al. \(2014, sec. 6.3\)](#). The example analyzes the speed of light measurements from the experiment performed by [Newcomb \(1891\)](#). Newcomb measured the time (in nanoseconds) it takes for light to travel 7,442 meters. `splight.dta` contains 66 independent measurements of the deviance of the travel time from 24,800 nanoseconds in variable `timedev`.

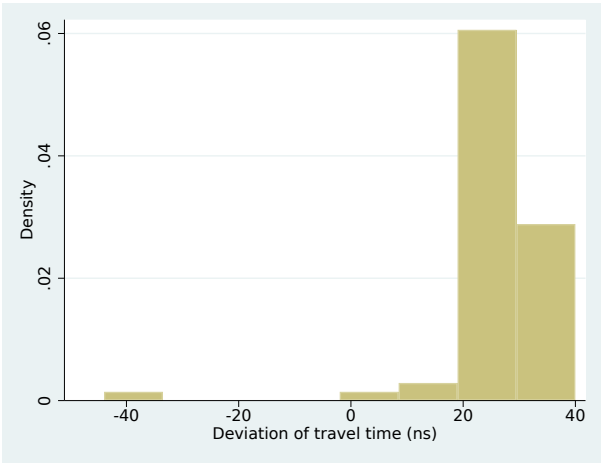
```
. use https://www.stata-press.com/data/r16/splight
(Newcomb's speed of light measurements)
. describe
Contains data from https://www.stata-press.com/data/r16/splight.dta
  obs:                66                Newcomb's speed of light
                                         measurements
  vars:                1                22 Feb 2019 13:24
                                         (_dta has notes)
```

variable name	storage type	display format	value label	variable label
timedev	byte	%9.0g		Deviation of travel time (ns)

Sorted by:

Let's look at the distribution of the data.

```
. histogram timedev
(bin=8, start=-44, width=10.5)
```



The data have several extreme observations in the left tail—the smallest observed `timedev` is `-44`, which is more than 6 standard deviations smaller than the sample mean.

To demonstrate posterior predictive checking, [Gelman et al. \(2014\)](#) intentionally used a simplified model for `timedev`, a normal model with unknown mean  $\mu$  and variance  $\sigma^2$ , which may not be a good fit given the presence of extreme observations. The authors chose a noninformative prior for the model parameters,  $(\mu, \sigma^2) \sim 1/\sigma^2$ , to achieve more objective analysis.

We fit the described model using `bayesmh` as follows:

```
. bayesmh timedev, likelihood(normal({sig2}))
> prior({timedev:_cons}, flat) prior({sig2}, jeffreys)
> mcmcsize(1000) rseed(16) saving(splight_mcmc)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
    timedev ~ normal({timedev:_cons},{sig2})
Priors:
    {timedev:_cons} ~ 1 (flat)
    {sig2} ~ jeffreys
```

---

Bayesian normal regression	MCMC iterations =	3,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	1,000
	Number of obs =	66
	Acceptance rate =	.2128
	Efficiency: min =	.104
	avg =	.1123
	max =	.1207

Log marginal-likelihood = -249.39408

---

	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]
timedev					
_cons	26.40191	1.306144	.128102	26.42451	23.57925 28.71792
sig2	118.8588	21.83563	1.98746	115.8515	81.03243 163.9617

file `splight_mcmc.dta` saved

The described prior is modeled in `bayesmh` by specifying the flat prior for `{timedev:_cons}`, the mean parameter of the normal model, and the Jeffreys prior for variance `{sig2}`. We requested a small MCMC sample of only 1,000. We also specified the `saving()` option to save MCMC estimates of model parameters, which is required to use `bayespredict` or `bayesreps`.

`bayesmh` reports a 95% equal-tailed credible interval of [23.6, 28.7] for `{timedev:_cons}`. The true deviance of the travel time of light is known to be 33.0 nanoseconds and is outside the reported credible interval. Clearly, our model does not produce an accurate estimate for the speed of light. The question is, Can we detect the misfit without the knowledge of the true value? We explore the answers to this question in the following examples:

- [Example 6: Goodness of fit using MCMC replicates of simulated outcomes](#)
- [Example 7: Test statistics as scalar functions of simulated outcomes](#)
- [Example 8: Test quantities via user-defined Stata programs](#)
- [Example 9: Working with a prediction dataset](#)

### ➤ Example 6: Goodness of fit using MCMC replicates of simulated outcomes

One way of checking goodness of fit is to compare the observed sample with the replication samples drawn from the posterior predictive distribution. Any systematic discrepancy between replicated and observed data will indicate misfit.

Let's start with visual inspection of the replicated data. We can use the `bayesreps` command to generate 20 MCMC replicates for the outcome `timedev`. Each replicate has 66 observations and is saved as a new variable in the dataset. We specify `tdrep*` as a variable stub for the replicate names.



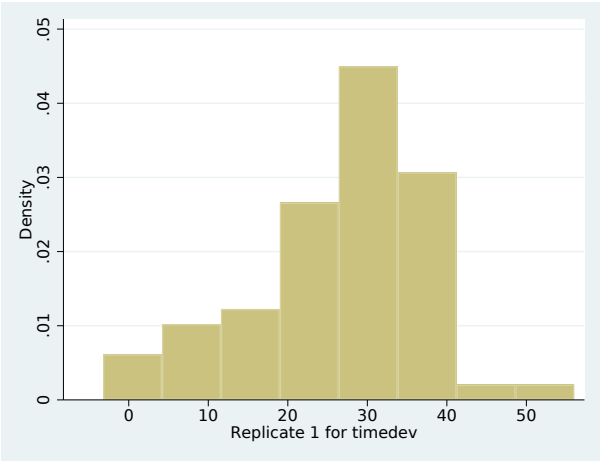
```
. bayesreps tdrep*, nreps(20) rseed(16)
Computing predictions ...
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
timedev	66	26.21212	10.74532	-44	40
tdrep1	66	26.10487	11.25766	-3.237112	56.03574
tdrep2	66	23.29179	11.30281	-12.58518	47.92589
tdrep3	66	26.21224	14.93573	-4.078057	61.37516
tdrep4	66	27.28245	11.60644	-2.502777	59.27184
tdrep5	66	27.74366	8.093924	5.70912	44.75622
tdrep6	66	26.15271	13.0279	-1.115488	53.42055
tdrep7	66	26.57665	10.11741	5.395408	46.71115
tdrep8	66	27.9395	12.06432	-4.903924	48.62425
tdrep9	66	25.54143	11.15095	1.560754	51.17381
tdrep10	66	28.11942	11.39326	1.364191	57.17214
tdrep11	66	24.18664	9.37403	7.49153	52.94038
tdrep12	66	25.87535	8.766691	10.5051	44.38683
tdrep13	66	27.49002	9.937486	4.89093	52.40338
tdrep14	66	26.17611	12.34034	-4.824428	51.16258
tdrep15	66	28.35187	10.58047	1.968471	50.73883
tdrep16	66	27.00237	11.44632	.7238956	52.77098
tdrep17	66	28.38859	11.1474	6.04494	63.34375
tdrep18	66	24.16652	9.289006	-.0226819	44.17939
tdrep19	66	24.9675	9.931602	6.675714	45.19432
tdrep20	66	27.69125	10.94969	1.289953	57.45961

The summary table shows that, compared with the observed data, the replicates have similar means and standard deviations but not the minimum and maximum values.

We can explore the entire distribution of a replicate. For example, we can produce the histogram for the first replicate and compare it with the earlier histogram of the observed data.

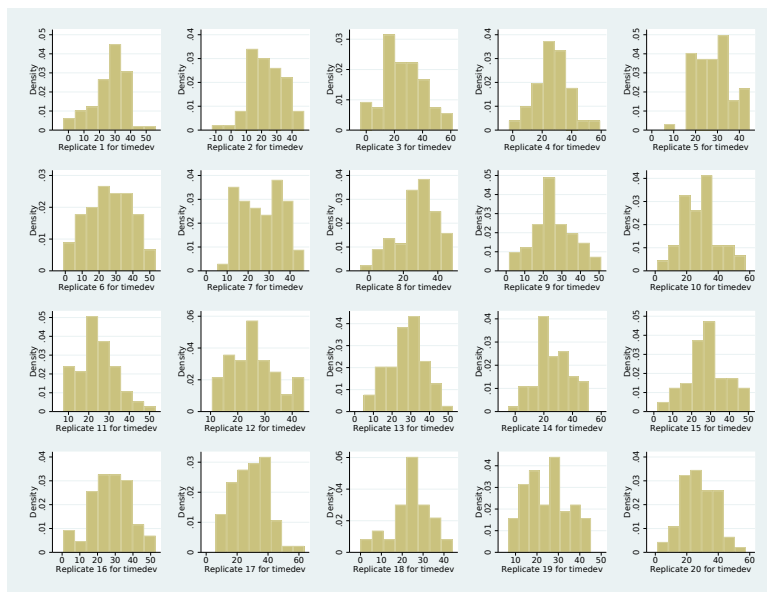
```
. histogram tdrep1
(bin=8, start=-3.237112, width=7.4091061)
```



The histograms look quite different. The replicate sample does not have the extreme negative values observed in the data.

With a few lines of code, we can produce histograms for all replicates and combine them on one graph.

```
. local histlist
. forvalues i = 1/20 {
2.     qui hist tdrep'i', name(hist'i') nodraw
3.     local histlist 'histlist' hist'i'
4. }
. graph combine 'histlist'
```



The histograms of all replicates look different from the observed data. The range for the replicated samples is about 0 to 50 with only a few negative values, which are smaller in magnitude than the negative values observed in the original data.



## ► Example 7: Test statistics as scalar functions of simulated outcomes

Gelman et al. (2014) suggest to use the smallest observation to measure the discrepancy between the observed and replicated data. That is, to compare the smallest values in the replicated samples with Newcomb's smallest observation of  $-44$ .

In addition to simulating outcome values, as we demonstrated in [example 1](#), we can use `bayespredict` to compute functions of simulated values that summarize the observations in a single statistic such as the minimum statistic. A function can be any Mata function that takes a column vector as an argument and returns a scalar. The result from `bayespredict` in this case is an MCMC sample of function values stored in the prediction dataset as a new variable.

Let's use `bayespredict` to produce an MCMC sample of the smallest observations (minimums) of the replicated data. Because we are not interested in individual observations, we can request that only

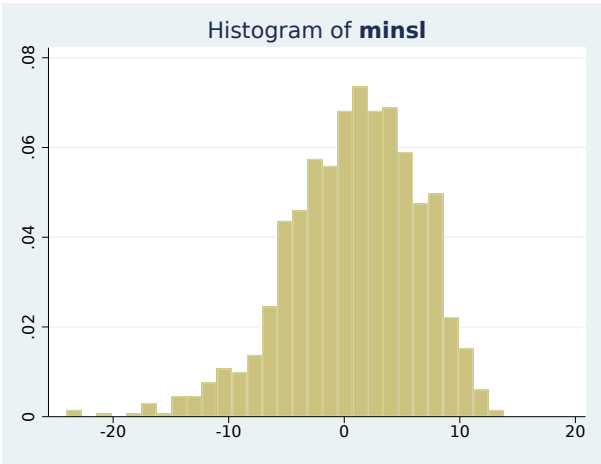
the smallest observation be simulated and stored by using the function specification `@min({_ysim})` with `bayespredict`.

```
. bayespredict (minsl:@min({_ysim})), saving(splight_pred) rseed(16)
Computing predictions ...
file splight_pred.dta saved
file splight_pred.ster saved
```

Per our specification, the command creates a new dataset, `splight_pred.dta`, that stores minimum statistics of the replicated data in the variable `minsl`. The prediction dataset has 1,000 observations, because 1,000 is the size of the MCMC sample simulated by `bayesmh`.

We can now use `{minsl}` within other Bayesian postestimation commands such as `bayesgraph` and `bayesstats summary` provided we supply the prediction dataset with the `using` specification. For example, let's draw the histogram of `{minsl}` using `bayesgraph histogram`.

```
. bayesgraph histogram {minsl} using splight_pred
```



The histogram provides the estimate of the posterior predictive distribution for the minimum statistic. The range of the histogram does not cover the observed minimum value of `-44`.

We can compare the posterior predictive distribution of the minimum statistic with the observed minimum value more formally by computing the posterior predictive *p*-value by using `bayesstats ppvalues`.

```
. bayesstats ppvalues {minsl} using splight_pred
Posterior predictive summary      MCMC sample size =      1,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
minsl	.8017725	5.590955	-44	1

Note:  $P(T \geq T_{\text{obs}})$  close to 0 or 1 indicates lack of fit.

The output table shows the posterior mean and standard deviation of `{minsl}`, the observed minimum value, `-44`, and the estimated posterior predictive *p*-value. The last is the probability that the replicated smallest value be greater or equal to the observed one. For a well-fitting model, the posterior predictive *p*-value should, ideally, be close to 0.5, although values between 0.05 and 0.95 are often considered acceptable in the literature (Gelman et al. 2014, 150). In our example, its estimate is essentially 1,

which indicates a strong misfit of the specified normal model. Therefore, if modeling of the tails of the outcome distribution is important, we should reconsider the normal likelihood model and find a better alternative.

◀

### ► Example 8: Test quantities via user-defined Stata programs

It is not sufficient to assess goodness of fit by examining just one test statistic. Different test statistics capture different aspects of the data. Which statistic to use depends on the research problem and the data characteristics you wish to account for. Generally, as pointed out by [Gelman et al. \(2014\)](#), for noninformative priors, [sufficient statistics](#) such as a sample mean and variance may not be good choices for checking model fit because they are typically modeled directly by the parameters of the likelihood function.

We demonstrated that our model does not model the minimum statistic well. Let's consider another aspect of `timedev`: symmetry with respect to the mean  $\mu$ .

Following [Gelman et al. \(2014\)](#), we define the following [test quantity](#) to measure asymmetry,

$$T(\text{timedev}, \mu) = |\text{timedev}_{(61)} - \mu| - |\text{timedev}_{(6)} - \mu|$$

where  $\text{timedev}_{(a)}$  defines the  $a$ th ordered value of `timedev` and  $(\text{timedev}_{(6)}, \text{timedev}_{(61)})$  represents about 90% of the distribution of `timedev`.

There is no predefined computation for the above statistic, so we need to write our own. For statistics that depend only on simulated outcome values, expected values, and residuals, we can write our own Mata functions or Stata programs. Mata functions are generally faster. For statistics that directly use model parameters, writing a Stata program is our only choice. Because the calculation of  $T(\text{timedev}, \mu)$  involves a model parameter,  $\mu$ , we must write a Stata program to calculate this statistic. Let's call our program `symstatprog`.

```

program symstatprog
    version 16.0
    args symout ysim
    tempname mu
    scalar 'mu' = $BAYESPR_theta[1,1]
    sort 'ysim'
    scalar 'symout' = abs('ysim'[61] - 'mu') - abs('ysim'[6] - 'mu')
end

```

The program has two input arguments, `symout` and `ysim`. The local macro `symout` contains the name of a temporary scalar for storing the final result. The local macro `ysim` contains the name of a temporary variable that stores the simulated outcome values of `timedev`. The global macro `$BAYESPR_theta` contains the name of a temporary matrix (row vector) that stores the current values of simulated model parameters, which are  $\mu$  and  $\sigma^2$  in our example. The parameters are stored in the same order they are displayed by `bayesmh`. Thus, in our example, the first element of this matrix corresponds to the mean,  $\mu$ . We use the earlier definition to compute the asymmetry test quantity and store it in the scalar `'symout'`.

We now call `bayespredict` to use the `symstatprog` program to compute the asymmetry test quantity for each set of simulated model parameters and label the prediction results as `symstat`. We replace our previously generated prediction dataset, `splight_pred.dta`, with these new prediction results.

```
. bayespredict (symstat:@symstatprog {_ysim}), saving(splight_pred, replace)
> rseed(16)
Computing predictions ...
file splight_pred.dta saved
file splight_pred.ster saved
```

We can use bayesstats ppvalues to test the goodness of fit for  $T(\text{timedev}, \mu)$ .

```
. bayesstats ppvalues {symstat} using splight_pred
Posterior predictive summary      MCMC sample size =      1,000
```

T	Mean	Std. Dev.	E(T_obs)	P(T>=T_obs)
symstat	.0953002	3.476211	3.196186	.235

Note: P(T>=T\_obs) close to 0 or 1 indicates lack of fit.

The posterior predictive  $p$ -value is estimated to be 0.235 and does not suggest model misfit with respect to  $T(\text{timedev}, \mu)$ .



► Example 9: Working with a prediction dataset

Sometimes, we may need to access the prediction results. For example, [Gelman et al. \(2014\)](#) provide a visual representation of the posterior predictive  $p$ -value by plotting the observed values of the asymmetry test quantity,  $T(\text{timedev}, \mu)$ , versus the replicated values,  $T(\text{timedev}^{\text{rep}}, \mu)$ . We can reproduce this graph as follows.

We start by loading the prediction dataset that contains our prediction results.

```
. use splight_pred, clear
. describe
Contains data from splight_pred.dta
  obs:      1,000
  vars:      5                               5 Mar 2019 12:29
```

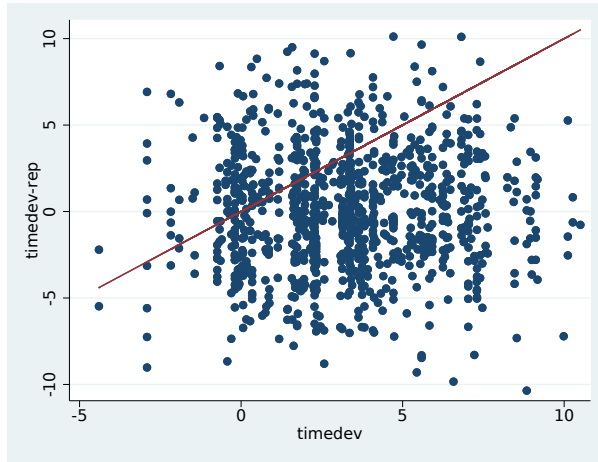
variable name	storage type	display format	value label	variable label
_chain	int	%8.0g		Chain identifier
_index	int	%8.0g		Iteration number
symstat	double	%10.0g		symstatprog {_ysim1}
_obs_symstat	double	%10.0g		Observed symstatprog {_ysim1}
_frequency	byte	%8.0g		Frequency weight

Sorted by:

Similarly to the [MCMC simulation dataset](#), variables `_chain` and `_index` record chain and index identifiers. Variable `symstat` contains the values for  $T(\text{timedev}^{\text{rep}}, \mu)$ , and variable `_obs_symstat` contains the values for  $T(\text{timedev}, \mu)$ . For consistency with the simulation dataset, the prediction dataset also contains the `_frequency` variable, but it is always one in the prediction dataset.

To visualize the posterior predictive  $p$ -value, we draw the scatterplot of `symstat` versus `_obs_symstat` overlaid with the diagonal line for `_obs_symstat` as the reference line.

```
. scatter symstat _obs_symstat || line _obs_symstat _obs_symstat,
> xtitle("timedev") ytitle("timedev-rep") legend(off)
```



The estimated posterior predictive  $p$ -value is the proportion of points above the diagonal line.

◀

In conclusion, although the normal model describes well the symmetry of the observed measurements, it fails to capture some of the smaller observations. It is possible that the experimental procedure was susceptible to aberrant measurements and a different model is needed to reflect this.

## Out-of-sample prediction

This section illustrates how `bayespredict` can be used as a classifier for binary outcomes.

### ► Example 10: Out-of-sample classification using predictive posterior means

We consider `titanic800.dta`, which contains the information of 800 passengers, who were on board the ocean liner *Titanic* when it sank. The dataset is a subset from a larger dataset published by Dawson (1995) at <https://www.amstat.org/publications/jse/v3n3/datasets.dawson.html>.

```

. use https://www.stata-press.com/data/r16/titanic800, clear
(Titanic passenger survival (Extract))

. describe
Contains data from https://www.stata-press.com/data/r16/titanic800.dta
   obs:                800                Titanic passenger survival
                                         (Extract)
   vars:                4                22 Feb 2019 13:24
                                         (_dta has notes)

```

variable name	storage type	display format	value label	variable label
class	byte	%9.0g	class	Class
adult	byte	%9.0g	age	Adult
male	byte	%9.0g	sex	Male
survived	byte	%9.0g	survived	Survived

Sorted by:

The binary variable `survived` records whether a passenger survived (`survived = 1`) or not (`survived = 0`). Passenger characteristics include the cabin type and class membership, `class` (first, second, third, or crew); the sex, `male`; and whether the passenger was an adult or a child, `adult`.

For illustration, we consider a simple logistic regression of `survived` on the categorical predictor `class` and binary predictors `male` and `adult`.

First, we randomly split the data into training and test subsamples. We use `split-sample` ([D] [split-sample](#)) to generate a variable, `sample`, that assigns 50% of the data to the training subsample (`sample = 1`) and the other 50% to the test subsample (`sample = 2`).

```

. split-sample, generate(sample) rseed(12345)

```

Second, we fit a Bayesian logistic regression using the training subsample of 400 passengers. We apply a Cauchy(0, 1) prior distribution for the coefficients. As a prerequisite for computing Bayesian predictions, we save the MCMC sample in `titanic_mcmc.dta`.

```
. bayesmh survived i.male i.adult ib1.class if sample==1, likelihood(logit)
> prior({survived:}, cauchy(0, 1)) saving(titanic_mcmc) rseed(16)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  survived ~ logit(xb_survived)

Prior:
  {survived:1.male 1.adult i.class _cons} ~ cauchy(0,1) (1)
```

---

```
(1) Parameters are elements of the linear form xb_survived.

Bayesian logistic regression      MCMC iterations =      12,500
Random-walk Metropolis-Hastings sampling  Burn-in      =       2,500
                                         MCMC sample size =     10,000
                                         Number of obs   =       400
                                         Acceptance rate =     .2054
                                         Efficiency: min =     .02887
                                         avg             =     .04189
                                         max             =     .05692

Log marginal-likelihood = -211.35694
```

survived	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
male						
male	-2.490095	.3330118	.019599	-2.498318	-3.13982	-1.844389
adult						
adult	-.5916052	.4910491	.024348	-.5666577	-1.551435	.3630116
class						
crew	-.6376593	.389797	.01675	-.6224151	-1.435266	.118966
second	-.5605325	.4214846	.017667	-.5507987	-1.423903	.2334895
third	-1.103689	.4064315	.021184	-1.106785	-1.923915	-.3518597
_cons	2.386679	.6342651	.034439	2.384843	1.188944	3.692627

file titanic\_mcmc.dta saved

All coefficients are negative, which means they are associated with lower survival probabilities compared with their respective baselines. For instance, adults were less likely to survive than children, and crew members and second- and third-class passengers were less likely to survive than the first-class passengers. The male passengers on board *Titanic* were especially unfortunate—the posterior mean estimate for the coefficient on `male` is  $-2.5$  with a 95% credible interval of  $[-3.1, -1.8]$ .

Let’s now compute out-of-sample predictions for the test subsample of the other 400 passengers. We use `bayespredict` with the `mean` option to calculate the posterior means of the simulated outcome for these passengers and store them as a new variable, `pmean`, in the current dataset.

```
. bayespredict pmean if sample==2, mean dots(100, every(1000)) rseed(16)
Computing predictions 10000 .....1000.....2000.....3000.....
> 4000.....5000.....6000.....7000.....8000.....9000.....
> 10000 done
```

The posterior means estimate the survival probabilities of the passengers and are, in fact, the optimal predictors with respect to the mean squared error (MSE). Let’s compute MSE for `pmean` to assess prediction accuracy of the model.



```
. generate err2 = (survived-pmean)^2
(400 missing values generated)
. summarize err2 if sample==2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
err2	400	.2567308	.2757182	.0189613	.7449417

Our model achieves an MSE of 0.26, but this number is difficult to interpret on its own, without any reference models.

Let’s compute the prediction accuracy of our model or how well our model predicted the outcome in the test subsample. We generate a new variable, `survived_logit`, to contain the binary outcome predicted from our Bayesian logistic model. We assign the predicted outcome to be 1 if `pmean` is greater than 0.5, and 0 otherwise. We then estimate the prediction accuracy as the proportion of matches between the observed `survived` and the predicted `survived_logit` in the test subsample.

```
. generate survived_logit = (pmean>0.5)
. generate pacc = (survived==survived_logit)
. summarize pacc if sample==2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
pacc	400	.62	.4859943	0	1

The prediction accuracy of our simple logistic model is about 0.62, which is not that high. Thus, a better prediction model should be considered for these data.



## Stored results

`bayespredict` stores the following in an estimation file, *filename.ster*, where *filename* is specified in the `saving(filename)` option.

Scalars	
<code>e(N)</code>	number of observations
<code>e(nchains)</code>	number of MCMC chains
<code>e(mcmcsize)</code>	MCMC sample size
Macros	
<code>e(cmd)</code>	<code>bayespredict</code>
<code>e(est_cmd)</code>	<code>bayesmh</code>
<code>e(cmdline)</code>	command as typed
<code>e(est_cmdline)</code>	estimation command as typed
<code>e(predfile)</code>	file containing prediction results
<code>e(mcmcfile)</code>	file containing simulation results
<code>e(predynames)</code>	names of simulated outcome observations, <code>_ysim#_#</code>
<code>e(predfnames)</code>	names of specified functions and programs
<code>e(predrngstate#)</code>	random-number state for #th chain for prediction
<code>e(rngstate)</code>	random-number state for simulation (only with single chain)
<code>e(rngstate#)</code>	random-number state for #th chain for simulation (only with <code>nchains()</code> )

## Methods and formulas

Methods and formulas are presented under the following headings:

- [Posterior predictive distribution](#)
- [MCMC sampling from posterior predictive distribution](#)
- [Residuals and expected values](#)

## Posterior predictive distribution

Recall from *Overview of Bayesian predictions* that the posterior predictive distribution of new data  $\mathbf{y}^{\text{new}}$  given observed data  $\mathbf{y}^{\text{obs}}$  is

$$\begin{aligned} p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}) &= \int p(\mathbf{y}^{\text{new}}, \boldsymbol{\theta}|\mathbf{y}^{\text{obs}})d\boldsymbol{\theta} \\ &= \int p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})d\boldsymbol{\theta} \\ &= \int p(\mathbf{y}^{\text{new}}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})d\boldsymbol{\theta} \end{aligned} \quad (3)$$

where we used the assumption of independence between  $\mathbf{y}^{\text{new}}$  and  $\mathbf{y}^{\text{obs}}$  given  $\boldsymbol{\theta}$  to arrive at the final expression.

Simulated outcomes,  $\mathbf{y}^{\text{sim}}$ , are the outcome values simulated from the posterior predictive distribution (3).

In a regression setting, posterior predictive distribution (3) also depends on the covariate data,

$$p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, X^{\text{new}}) = \int p(\mathbf{y}^{\text{new}}|\boldsymbol{\theta}, X^{\text{new}})p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}}, X^{\text{obs}})d\boldsymbol{\theta} \quad (4)$$

where  $X^{\text{new}}$  is the data matrix containing new covariate values and  $X^{\text{obs}}$  is the data matrix containing observed covariate values used to fit the model.

The concept of replicated outcomes or replicated data,  $\mathbf{y}^{\text{rep}}$ , arises in a regression setting when the data matrix used to generate new outcome values is the same as the observed data matrix used to fit the Bayesian model. That is,

$$p(\mathbf{y}^{\text{rep}}|\mathbf{y}^{\text{obs}}, X^{\text{obs}}) = \int p(\mathbf{y}^{\text{rep}}|\boldsymbol{\theta}, X^{\text{obs}})p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}}, X^{\text{obs}})d\boldsymbol{\theta} \quad (5)$$

In a regression setting, we use a general definition for the simulated outcome,  $\mathbf{y}^{\text{sim}}$ , as one generated either from (4) or (5).

Test quantities and test statistics are commonly used to check goodness of fit of a Bayesian model. A test quantity,  $T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta})$ , is a scalar function of replicated data  $\mathbf{y}^{\text{rep}}$  and model parameters  $\boldsymbol{\theta}$ . A test statistic,  $T_s(\mathbf{y}^{\text{rep}})$ , is a scalar function that depends only on the replicated data  $\mathbf{y}^{\text{rep}}$ . If the model fits the data well,  $T_q(\mathbf{y}^{\text{rep}}, \boldsymbol{\theta})$  should be close to  $T_q(\mathbf{y}^{\text{obs}}, \boldsymbol{\theta})$ , and, similarly,  $T_s(\mathbf{y}^{\text{rep}})$  should be close to  $T_s(\mathbf{y}^{\text{obs}})$ .

## MCMC sampling from posterior predictive distribution

Like the posterior distribution of model parameters, posterior predictive distributions (3), (4), and (5) usually do not have closed forms and must be approximated. In what follows, we will concentrate on the more general posterior predictive distribution  $p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, X^{\text{new}})$ , but the same principles apply to the other distributions by removing conditioning on covariate data in case of (3) and by replacing  $X^{\text{new}}$  with  $X^{\text{obs}}$  in case of (5).

The goal of Bayesian prediction is to simulate data from  $p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, X^{\text{new}})$ . Formula (4) underlies the following two-step iterative process for obtaining simulated outcomes from  $p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, X^{\text{new}})$ .

1. Draw a realization of model parameters,  $\theta^*$ , from their posterior distribution,  $p(\theta|\mathbf{y}^{\text{obs}}, X^{\text{obs}})$ .
2. Generate  $\mathbf{y}^{\text{sim}}$  from  $p(\mathbf{y}^{\text{new}}|\theta^*, X^{\text{new}})$ , the data distribution (likelihood) conditional on the parameters obtained in step 1.

Steps 1 and 2 are repeated to produce an MCMC sample of simulated outcomes,  $(\mathbf{y}^{\text{sim},1}, \mathbf{y}^{\text{sim},2}, \dots, \mathbf{y}^{\text{sim},T})$ , where  $T$  is the MCMC sample size. We can use this sample to estimate the posterior predictive distribution.

For step 1, `bayespredict` uses the MCMC sample of model parameters as produced by the `bayesmh` command. The main computation of `bayespredict` is the simulation of the outcome values from the respective likelihood model for each set of simulated model parameters from the MCMC sample. For an outcome variable with  $n$  observations, the result of a Bayesian prediction is a dataset containing  $T$  observations and  $n$  columns.

A function of simulated values is computed as follows:  $\{f(\mathbf{y}^{\text{sim},1}), f(\mathbf{y}^{\text{sim},2}), \dots, f(\mathbf{y}^{\text{sim},T})\}$ , where  $f(\cdot)$  is a function that operates on a column vector and returns a scalar. The resulting prediction dataset will contain a variable with  $T$  observations.

For a test statistic  $T_s(\mathbf{y}^{\text{rep}})$ , the following simulated sample is produced:

$\{T_s(\mathbf{y}^{\text{rep},1}), T_s(\mathbf{y}^{\text{rep},2}), \dots, T_s(\mathbf{y}^{\text{rep},T})\}$ . For a well-fitting model, the distribution of this sample should be concentrated around  $T_s(\mathbf{y}^{\text{obs}})$ .

For a test quantity  $T_q(\mathbf{y}^{\text{rep}}, \theta)$ , the following simulated sample is produced:

$\{T_q(\mathbf{y}^{\text{rep},1}, \theta^1), T_q(\mathbf{y}^{\text{rep},2}, \theta^2), \dots, T_q(\mathbf{y}^{\text{rep},T}, \theta^T)\}$ . For a well-fitting model, the distribution of this sample should be close to the distribution of  $\{T_q(\mathbf{y}^{\text{obs}}, \theta^1), T_q(\mathbf{y}^{\text{obs}}, \theta^2), \dots, T_q(\mathbf{y}^{\text{obs}}, \theta^T)\}$ .

## Residuals and expected values

Consider simulated outcome values  $\mathbf{y}_i^{\text{sim}}$  for an observation  $i = 1, 2, \dots, n$ , where  $\mathbf{y}_i^{\text{sim}} = (y_i^{\text{sim},1}, y_i^{\text{sim},2}, \dots, y_i^{\text{sim},T})^T$ . Let  $\hat{\boldsymbol{\mu}}_i = (\hat{\mu}_i^1, \hat{\mu}_i^2, \dots, \hat{\mu}_i^T)^T$ , where  $\hat{\mu}_i^t = E(y_i|\mathbf{x}_i, \theta^t)$  is the estimated expected value of  $y_i$  given covariate vector  $\mathbf{x}_i$  and simulated parameters  $\theta^t$ ,  $t = 1, 2, \dots, T$ . Let  $\mathbf{r}_i^{\text{sim}} = (r_i^{\text{sim},1}, r_i^{\text{sim},2}, \dots, r_i^{\text{sim},T})^T$  be simulated residuals for an observation  $i$ .

Simulated residuals are then defined as

$$\mathbf{r}_i^{\text{sim}} = \mathbf{y}_i^{\text{sim}} - \hat{\boldsymbol{\mu}}_i$$

Within `bayespredict`, you refer to  $\mathbf{y}_i^{\text{sim}}$  as `{_ysim_}` $_i$ ,  $\mathbf{r}_i^{\text{sim}}$  as `{_resid_}` $_i$ , and  $\hat{\boldsymbol{\mu}}_i$  as `{_mu_}` $_i$ . You can also use `{_ysim}`, `{_resid}`, and `{_mu}` to refer to all observations at once. With multiple outcomes, the above specifications correspond to the first outcome variable. For the  $\#$ th outcome variable, use `{_ysim#_}` $_i$ , `{_resid#_}` $_i$ , `{_mu#_}` $_i$ , `{_ysim#}`, `{_resid#}`, and `{_mu#}`, respectively.

Below are the definitions of  $\hat{\mu}_i^t$  for the likelihood models supported by `bayesmh`.

1. **Normal regression:**  $\hat{\mu}_i^t = \mathbf{x}_i\boldsymbol{\beta}^t$ .
2. **t-regression:**  $\hat{\mu}_i^t = \mathbf{x}_i\boldsymbol{\beta}^t$ .
3. **Lognormal regression:**  $\hat{\mu}_i^t = \exp(\mathbf{x}_i\boldsymbol{\beta}^t)$ .
4. **Exponential regression:**  $\hat{\mu}_i^t = \exp(\mathbf{x}_i\boldsymbol{\beta}^t)$ .

5. **Probit regression:**  $\hat{\mu}_i^t = \Phi(\mathbf{x}_i\beta^t)$ .
6. **Logistic regression:**  $\hat{\mu}_i^t = \text{invlogit}(\mathbf{x}_i\beta^t)$ .
7. **Binomial regression:**  $\hat{\mu}_i^t = n_{\text{trials}} \times \text{invlogit}(\mathbf{x}_i\beta^t)$ , where  $n_{\text{trials}}$  is the number of trials in binomial regression.
8. **Ordered probit regression:** `{_resid}` and `{_mu}` not supported.
9. **Ordered logistic regression:** `{_resid}` and `{_mu}` not supported.
10. **Poisson regression:**  $\hat{\mu}_i^t = \exp(\mathbf{x}_i\beta^t)$ .

Next are the definitions of  $\hat{\mu}_i^t$  for the distribution models `dexponential(beta)`, `dbernoulli(p)`, `dbinomial(n_trials, p)`, and `dpoisson(mu)`.

11. **Exponential distribution:**  $\hat{\mu}_i^t = \beta^t$ .
12. **Bernoulli distribution:**  $\hat{\mu}_i^t = p^t$ .
13. **Binomial distribution:**  $\hat{\mu}_i^t = n_{\text{trials}}p^t$ .
14. **Poisson distribution:**  $\hat{\mu}_i^t = \mu^t$ .

Typically, the expected values for the distribution models will be constant over observations unless the distribution parameters vary over the observations.

Raw residuals,  $\mathbf{r}_i^{\text{sim}}$ , may not always be the most appropriate for diagnostic purposes. For example, Pearson residuals are better suited for discrete outcome models such as binomial and Poisson regressions.

## References

- Dawson, R. J. M. 1995. The “Unusual Episode” data revisited. *Journal of Statistics Education* 3: 1–9.
- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.
- Gelman, A., X.-L. Meng, and H. Stern. 1996. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica Sanica* 6: 733–760.
- Gelman, A., and D. B. Rubin. 1992. Inference from iterative simulation using multiple sequences. *Statistical Science* 7: 457–472.
- Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.
- Meng, X.-L. 1994. Multiple-imputation inferences with uncongenial sources of input (with discussion). *Statistical Science* 9: 538–573.
- Newcomb, S. 1891. Measures of the velocity of light made under the direction of the Secretary of the Navy during the years 1880–1882. *Astronomical Papers* 2: 107–229.
- Tsui, K.-W., and S. Weerahandi. 1989. Generalized p-values in significance testing of hypotheses in the presence of nuisance parameters. *Journal of the American Statistical Association* 84: 602–607.
- West, M. 1986. Bayesian model monitoring. *Journal of the Royal Statistical Society, Series B* 48: 70–78.

## Also see

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **bayesgraph** — Graphical summaries and convergence diagnostics

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayesstats ppvalues** — Bayesian predictive p-values and other predictive summaries

[BAYES] **bayesstats summary** — Bayesian summary statistics

[BAYES] **bayestest interval** — Interval hypothesis testing

# Title

set clevel — Set default credible level

[Description](#)[Syntax](#)[Option](#)[Remarks and examples](#)[Also see](#)

## Description

`set clevel` specifies the default credible level for credible intervals for all Bayesian commands (see [\[BAYES\]](#) [Bayesian commands](#)) that report credible intervals. The initial value is 95, meaning 95% credible intervals.

## Syntax

```
set clevel # [ , permanently]
```

`#` is any number between 10.00 and 99.99 and may be specified with at most two digits after the decimal point.

## Option

`permanently` specifies that in addition to making the change right now, the `clevel` setting be remembered and become the default setting when you invoke Stata.

## Remarks and examples

To change the level of credible intervals reported by a particular command, you need not reset the default credible level. All commands that report credible intervals have a `clevel(#)` option. When you do not specify the option, the credible intervals are calculated for the default level set by `set clevel` or for 95% if you have not reset `set clevel`.

➤ **Example 1**

We use the `bayesmh` command to obtain the credible interval for the mean of `mpg`:

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. set seed 14
. bayesmh mpg, likelihood(normal(30)) prior({mpg:_cons}, flat)
Burn-in ...
Simulation ...
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4195
	Efficiency =	.2378

Log marginal-likelihood = -234.09275

---

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [95% Cred. Interval]	
_cons	21.30364	.6429995	.013186	21.30381	20.03481	22.5555

To obtain 90% credible intervals, we would type

```
. bayesmh, clevel(90)
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4195
	Efficiency =	.2378

Log marginal-likelihood = -234.09275

---

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [90% Cred. Interval]	
_cons	21.30364	.6429995	.013186	21.30381	20.24172	22.35158

or we could type

```
. set clevel 90
. bayesmh
Model summary
```

---

```
Likelihood:
  mpg ~ normal({mpg:_cons},30)
Prior:
  {mpg:_cons} ~ 1 (flat)
```

---

Bayesian normal regression	MCMC iterations =	12,500
Random-walk Metropolis-Hastings sampling	Burn-in =	2,500
	MCMC sample size =	10,000
	Number of obs =	74
	Acceptance rate =	.4195
	Efficiency =	.2378

---

```
Log marginal-likelihood = -234.09275
```

---

mpg	Mean	Std. Dev.	MCSE	Median	Equal-tailed [90% Cred. Interval]	
_cons	21.30364	.6429995	.013186	21.30381	20.24172	22.35158

---

If we opt for the second alternative, the next time that we fit a model, 90% credible intervals will be reported. If we wanted 95% credible intervals, we could specify `clevel(95)` on the estimation command, or we could reset the default by typing `set clevel 95`.

The current setting of `clevel()` is stored as the c-class value `c(clevel)`; see [P] [creturn](#). ◀

Also see

- [BAYES] [bayes](#) — Bayesian regression models using the bayes prefix
- [BAYES] [bayesmh](#) — Bayesian models using Metropolis–Hastings algorithm
- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [R] [query](#) — Display system parameters
- [P] [creturn](#) — Return c-class values



# Title

bayes: betareg — Bayesian beta regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: betareg` fits a Bayesian beta regression to a fractional outcome whose values are greater than 0 and less than 1; see [\[BAYES\] bayes](#) and [\[R\] betareg](#) for details.

## Quick start

Bayesian beta regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: betareg y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): betareg y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): betareg y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): betareg y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): betareg y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] betareg](#).

## Menu

Statistics > Fractional outcomes > Bayesian beta regression

# Syntax

```
bayes [ , bayesopts ] : betareg depvar indepvars [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

## Model

<u>noconstant</u>	suppress constant term
<u>scale</u> ( <i>varlist</i> [ , <u>noconstant</u> ])	specify independent variables for scale
<u>link</u> ( <i>linkname</i> )	specify link function for the conditional mean; default is <code>link(logit)</code>
<u>slink</u> ( <i>slinkname</i> )	specify link function for the conditional scale; default is <code>slink(log)</code>

## Reporting

<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 **Factor variables**.

*fweights* are allowed; see [U] 11.1.6 **weight**.

`bayes: betareg`, `level()` is equivalent to `bayes, clevel(): betareg`.

For a detailed description of *options*, see *Options* in [R] **betareg**.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{devar:indepvars}` for the main regression and `{scale:varlist}` for the scale equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] betareg](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] betareg](#) — Beta regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: binreg** — Bayesian generalized linear models: Extensions to the binomial family

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: binreg` fits a Bayesian binomial regression to a binary outcome, assuming different link functions; see [\[BAYES\] bayes](#) and [\[R\] binreg](#) for details.

## Quick start

Bayesian binomial regression of `y` on `x1` and `x2`, using the default logit link and using default normal priors for regression coefficients

```
bayes: binreg y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): binreg y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): binreg y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): binreg y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): binreg y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: binreg y x1 x2, or
```

Use the log link and report risk ratios

```
bayes: binreg y x1 x2, rr
```

Display coefficients instead of risk ratios

```
bayes, coefficients
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] binreg](#).

## Menu

Statistics > Generalized linear models > Bayesian GLM for the binomial family

Syntax

```
bayes [ , bayesopts ] : binreg depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
Model	
<u>noconstant</u>	suppress constant term
or	use logit link and report odds ratios
rr	use log link and report risk ratios
hr	use log-complement link and report health ratios
rd	use identity link and report risk differences
n( <i>#</i>   <i>varname</i> )	use <i>#</i> or <i>varname</i> for number of trials
<u>exposure</u> ( <i>varname</i> )	include ln( <i>varname</i> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
mu( <i>varname</i> )	use <i>varname</i> as the initial estimate for the mean of <i>depvar</i>
<u>init</u> ( <i>varname</i> )	synonym for mu( <i>varname</i> )
Reporting	
<u>coefficients</u>	report nonexponentiated coefficients
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> ( <i>#</i> )	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fwrights* are allowed; see [U] 11.1.6 weight.

bayes: binreg, level() is equivalent to bayes, clevel(): binreg.

For a detailed description of *options*, see *Options* in [R] binreg. binreg's option ml is implied with bayes: binreg.

<i>bayesopts</i>	Description
Priors	
* <u>normalprior</u> ( <i>#</i> )	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
prior( <i>priorspec</i> )	prior for model parameters; this option may be repeated
dryrun	show model summary without estimation
Simulation	
nchains( <i>#</i> )	number of chains; default is to simulate one chain
mcmcsize( <i>#</i> )	MCMC sample size; default is mcmcsize(10000)
burnin( <i>#</i> )	burn-in period; default is burnin(2500)
thinning( <i>#</i> )	thinning interval; default is thinning(1)
rseed( <i>#</i> )	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
Blocking	
* <u>blocksize</u> ( <i>#</i> )	maximum block size; default is blocksize(50)
block( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>coefficients</code>	report nonexponentiated coefficients
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, <i>every</i>(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] binreg](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Logistic regression with perfect predictors* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] binreg](#) — Generalized linear models: Extensions to the binomial family

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)



# Title

bayes: bipoibit — Bayesian bivariate probit regression

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: bipoibit` fits a Bayesian bivariate probit regression to two binary outcomes; see [\[BAYES\] bayes](#) and [\[R\] bipoibit](#) for details.

## Quick start

Bayesian bivariate probit regression of `y1` and `y2` on `x1` and `x2`, using default normal priors for regression coefficients and  $\text{atanh}$ -transformed correlation

```
bayes: bipoibit y1 y2 x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): bipoibit y1 y2 x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept of the dependent variable `y2`

```
bayes, prior({y2: x1 x2}, uniform(-10,10)) ///  
prior({y2:_cons}, normal(0,10)): bipoibit y1 y2 x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): bipoibit y1 y2 x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): bipoibit y1 y2 x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Bayesian seemingly unrelated bivariate probit regression using default priors

```
bayes: bipoibit (y1 = x1 x2 x3) (y2 = x1 x2)
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] bipoibit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Bivariate probit regression

Statistics > Binary outcomes > Bayesian regression > Seemingly unrelated bivariate probit

# Syntax

## Bayesian bivariate probit regression

```
bayes [ , bayesopts ] : biprobit depvar1 depvar2 [indepvars] [if] [in] [weight]
[ , options]
```

## Bayesian seemingly unrelated bivariate probit regression

```
bayes [ , bayesopts ] : biprobit equation1 equation2 [if] [in] [weight] [ , options]
```

where *equation*<sub>1</sub> and *equation*<sub>2</sub> are specified as

```
( [eqname: ] depvar [=] [indepvars] [ , noconstant offset(varname) ] )
```

<i>options</i>	Description
<b>Model</b>	
<u>noconstant</u>	suppress constant term
<u>offset1</u> ( <i>varname</i> )	offset variable for first equation
<u>offset2</u> ( <i>varname</i> )	offset variable for second equation
<b>Reporting</b>	
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)
<i>indepvars</i> may contain factor variables; see [U] 11.4.3 Factor variables.	
<i>depvar</i> <sub>1</sub> , <i>depvar</i> <sub>2</sub> , <i>depvar</i> , and <i>indepvars</i> may contain time-series operators; see [U] 11.4.4 Time-series varlists.	
<i>fweights</i> are allowed; see [U] 11.1.6 weight.	
bayes: biprobit, level() is equivalent to bayes, clevel(): biprobit.	
For a detailed description of <i>options</i> , see <i>Options</i> in [R] biprobit. Options noconstant, offset1(), and offset2() are not allowed with seemingly unrelated bivariate probit regression.	

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and atanh-transformed correlation; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

- \* **blocksize**(#) maximum block size; default is `blocksize(50)`
- block**(*paramref* [ , *blockopts* ]) specify a block of model parameters; this option may be repeated
- blocksummary** display block summary
- \* **noblocking** do not block parameters by default

## Initialization

- initial**(*initspec*) specify initial values for model parameters with a single chain
- init#**(*initspec*) specify initial values for #th chain; requires `nchains()`
- initall**(*initspec*) specify initial values for all chains; requires `nchains()`
- nomleinitial** suppress the use of maximum likelihood estimates as starting values
- initrandom** specify random initial values
- initsummary** display initial values used for simulation
- \* **noisily** display output from the estimation command during initialization

## Adaptation

- adaptation**(*adaptopts*) control the adaptive MCMC procedure
- scale**(#) initial multiplier for scale factor; default is `scale(2.38)`
- covariance**(*cov*) initial proposal covariance; default is the identity matrix

## Reporting

- clevel**(#) set credible interval level; default is `clevel(95)`
- hpd** display HPD credible intervals instead of the default equal-tailed credible intervals
- eform** [ (*string*) ] report exponentiated coefficients and, optionally, label as *string*
- batch**(#) specify length of block for batch-means calculations; default is `batch(0)`
- saving**(*filename* [ , *replace* ]) save simulation results to *filename.dta*
- nomodelsummary** suppress model summary
- chainsdetail** display detailed simulation summary for each chain
- [ *no* ] **dots** suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is `nodots`
- dots**(# [ , *every*(#) ]) display dots as simulation is performed
- [ *no* ] **show**(*paramref*) specify model parameters to be excluded from or included in the output
- notable** suppress estimation table
- noheader** suppress output header
- title**(*string*) display *string* as title above the table of parameter estimates
- display\_options** control spacing, line width, and base and empty cells

## Advanced

- search**(*search\_options*) control the search for feasible initial values
- corrlag**(#) specify maximum autocorrelation lag; default varies
- corrtol**(#) specify autocorrelation tolerance; default is `corrtol(0.01)`

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients  $\{depvar_1:indepvars\}$  and  $\{depvar_2:indepvars\}$  and atanh-transformed correlation  $\{athrho\}$ . Use the `dryrun` option to see the definitions of model parameters prior to estimation. For a detailed description of *bayesopts*, see *Options* in [BAYES] [bayes](#).

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] [bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [BAYES] [bayes](#). For details about the estimation command, see [R] [biprobit](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] [bayes](#).

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Also see

- [BAYES] [bayes](#) — Bayesian regression models using the `bayes` prefix
- [R] [biprobit](#) — Bivariate probit regression
- [BAYES] [Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix
- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis
- [BAYES] [Intro](#) — Introduction to Bayesian analysis
- [BAYES] [Glossary](#)

# Title

**bayes: clogit** — Bayesian conditional logistic regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: clogit** fits a Bayesian conditional logistic regression to matched case–control data; see [\[BAYES\] bayes](#) and [\[R\] clogit](#) for details.

## Quick start

Bayesian conditional logistic regression of *y* on *x1* and *x2*, using default normal priors for regression coefficients

```
bayes: clogit y x1 x2, group(id)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): clogit y x1 x2, group(id)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): clogit y x1 x2, group(id)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): clogit y x1 x2, group(id)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): clogit y x1 x2, group(id)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: clogit y x1 x2, group(id) or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] clogit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Conditional logistic regression

# Syntax

```
bayes [ , bayesopts ] : clogit depvar [indepvars] [if] [in] [weight] ,
      group(varname) [options]
```

<i>options</i>	Description
----------------	-------------

## Model

* <u>group</u> ( <i>varname</i> )	matched group variable
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1

## Reporting

<u>or</u>	report odds ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

\*group(*varname*) is required.

*indepvars* may contain factor variables; see [U] 11.4.3 **Factor variables**.

**fweights** are allowed; see [U] 11.1.6 **weight**. **fweights** are interpreted to apply to groups as a whole, not to individual observations. See *Use of weights* in [R] **clogit**.

**bayes: clogit**, `level()` is equivalent to **bayes**, `clevel()`: **clogit**.

For a detailed description of *options*, see *Options* in [R] **clogit**.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>or</code>	report odds ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , <i>replace</i> ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots(#[ , <i>every</i>(#) ])</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] [bayesmh](#).

*paramref* may contain factor variables; see [U] [11.4.3 Factor variables](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] [bayes](#).

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] clogit](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] clogit](#) — Conditional (fixed-effects) logistic regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)



# Title

**bayes: cloglog** — Bayesian complementary log-log regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: cloglog` fits a Bayesian complementary log-log regression to a binary outcome; see [\[BAYES\] bayes](#) and [\[R\] cloglog](#) for details.

## Quick start

Bayesian complementary log-log regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: cloglog y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): cloglog y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): cloglog y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): cloglog y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): cloglog y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display results as exponentiated coefficients

```
bayes: cloglog y x1 x2, eform
```

Display exponentiated coefficients on replay

```
bayes, eform
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] cloglog](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Complementary log-log regression

# Syntax

```
bayes [ , bayesopts ] : cloglog depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

## Model

<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<u>asis</u>	retain perfect predictor variables

## Reporting

<u>eform</u>	report exponentiated coefficients
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: cloglog, level() is equivalent to bayes, clevel(): cloglog.

For a detailed description of *options*, see *Options* in [R] cloglog.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of `bayesopts`, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] cloglog](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Logistic regression with perfect predictors* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] cloglog](#) — Complementary log-log regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: fracreg** — Bayesian fractional response regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: fracreg** fits a Bayesian fractional response regression to a fractional outcome whose values are greater than or equal to 0 and less than or equal to 1; see [\[BAYES\] bayes](#) and [\[R\] fracreg](#) for details.

## Quick start

Bayesian fractional probit regression of *y* on *x1* and *x2*, using default normal priors for regression coefficients

```
bayes: fracreg probit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): fracreg probit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): fracreg probit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): fracreg probit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): fracreg probit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Fit a fractional logistic regression and display results as odds ratios

```
bayes: fracreg logit y x1 x2, or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] fracreg](#).

## Menu

Statistics > Fractional outcomes > Bayesian fractional regression

# Syntax

Syntax for fractional probit regression

```
bayes [, bayesopts] : fracreg probit depvar [indepvars] [if] [in] [weight]
      [, options]
```

Syntax for fractional logistic regression

```
bayes [, bayesopts] : fracreg logit depvar [indepvars] [if] [in] [weight]
      [, options]
```

Syntax for fractional heteroskedastic probit regression

```
bayes [, bayesopts] : fracreg probit depvar [indepvars] [if] [in] [weight],
      het(varlist [, offset(varnameo)]) [options]
```

<i>options</i>	Description
Model	
<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
*het( <i>varlist</i> [, <u>offset</u> ( <i>varname</i> <sub>o</sub> )])	independent variables to model the variance and possible offset variable with fracreg probit
Reporting	
or	report odds ratios; only valid with fracreg logit
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

\* het() may be used only with fracreg probit to compute fractional heteroskedastic probit regression.

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: fracreg, level() is equivalent to bayes, clevel(): fracreg.

For a detailed description of *options*, see *Options* in [R] fracreg.

<i>bayesopts</i>	Description
Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

**Simulation**

<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(paramref)</code>	specify model parameters to be excluded from the simulation results

**Blocking**

<code>*blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(paramref[, blockopts])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
<code>*noblocking</code>	do not block parameters by default

**Initialization**

<code>initial(initspec)</code>	specify initial values for model parameters with a single chain
<code>init#(initspec)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(initspec)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(adaptopts)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(cov)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*or</code>	report odds ratio; only valid with <code>fracreg logit</code>
<code>eform(string)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every#])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

#### Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayermh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}` and, if option `het()` is specified, regression coefficients `{lnsigma:varlist}` for the log-standard deviation equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [R] `fracreg`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`.

## Stored results

See *Stored results* in [BAYES] `bayer`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayermh`.

## Also see

[BAYES] `bayer` — Bayesian regression models using the `bayer` prefix

[R] `fracreg` — Fractional response regression

[BAYES] `Bayesian postestimation` — Postestimation tools for `bayermh` and the `bayer` prefix

[BAYES] `Bayesian estimation` — Bayesian estimation commands

[BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis

[BAYES] `Intro` — Introduction to Bayesian analysis

[BAYES] `Glossary`



# Title

bayes: glm — Bayesian generalized linear models

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: glm` fits a Bayesian generalized linear model to outcomes of different types such as continuous, binary, count, and so on; see [\[BAYES\] bayes](#) and [\[R\] glm](#) for details.

## Quick start

Bayesian generalized linear model of `y` on `x1` and `x2`, using the Gaussian family and log link and using default normal priors for regression coefficients

```
bayes: glm y x1 x2, family(gaussian) link(log)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): glm y x1 x2, family(gaussian) link(log)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): ///  
glm y x1 x2, family(gaussian) link(log)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
glm y x1 x2, family(gaussian) link(log)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
glm y x1 x2, family(gaussian) link(log)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Fit a logit model and display results as odds ratios

```
bayes: glm z x1 x2, family(binomial) eform
```

Display odds ratios on replay

```
bayes, eform
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] glm](#).

## Menu

Statistics > Generalized linear models > Bayesian generalized linear models (GLM)

# Syntax

```
bayer [ , bayesopts ] : glm depcvar [ indepvars ] [ if ] [ in ] [ weight ] [ , options ]
```

<i>options</i>	Description
<b>Model</b>	
<u>family</u> ( <i>familyname</i> )	distribution of <i>depcvar</i> ; default is family(gaussian)
<u>link</u> ( <i>linkname</i> )	link function; default is canonical link for family() specified
<b>Model 2</b>	
<u>noconstant</u>	suppress constant term
<u>exposure</u> ( <i>varname</i> )	include ln( <i>varname</i> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<u>asis</u>	retain perfect predictor variables
<u>mu</u> ( <i>varname</i> )	use <i>varname</i> as the initial estimate for the mean of <i>depcvar</i>
<u>init</u> ( <i>varname</i> )	synonym for mu( <i>varname</i> )
<b>Reporting</b>	
<u>eform</u>	report exponentiated coefficients
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)
<i>indepvars</i> may contain factor variables; see [U] 11.4.3 Factor variables. <i>depcvar</i> and <i>indepvars</i> may contain time-series operators; see [U] 11.4.4 Time-series varlists. <i>fweights</i> are allowed; see [U] 11.1.6 weight. bayer: glm, level() is equivalent to bayer, clevel(): glm. For a detailed description of <i>options</i> , see <i>Options</i> in [R] glm.	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , <i>replace</i> ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[ , every(#) ]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] glm](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Generalized linear model* in [\[BAYES\] bayes](#).

`bayes: glm` does not estimate the scale parameter but uses a fixed value as provided by the `glm` command. If you want to fit a GLM and estimate the scale parameter, use `bayes: meglm` without specifying random effects.

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] glm](#) — Generalized linear models

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

bayes: gnbreg — Bayesian generalized negative binomial regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: gnbreg` fits a Bayesian generalized negative binomial regression to a nonnegative count outcome; see [\[BAYES\] bayes](#) and [\[R\] nbreg](#) for details.

## Quick start

Bayesian generalized negative binomial regression of `y` on `x1` and `x2`, using `z` to model the log-overdispersion and using default normal priors for regression coefficients and log-overdispersion parameter

```
bayes: gnbreg y x1 x2, lalpha(z)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): gnbreg y x1 x2, lalpha(z)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): gnbreg y x1 x2, lalpha(z)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): gnbreg y x1 x2, lalpha(z)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): gnbreg y x1 x2, lalpha(z)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: gnbreg y x1 x2, lalpha(z) irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] nbreg](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Generalized negative binomial regression

# Syntax

bayes [ , bayesopts ] : gnbreg depvar [ indepvars ] [ if ] [ in ] [ weight ] [ , options ]

options	Description
<b>Model</b>	
<u>noconstant</u>	suppress constant term
<u>lnalpha</u> (varlist)	dispersion model variables
<u>exposure</u> (varname <sub>e</sub> )	include ln(varname <sub>e</sub> ) in model with coefficient constrained to 1
<u>offset</u> (varname <sub>o</sub> )	include varname <sub>o</sub> in model with coefficient constrained to 1
<b>Reporting</b>	
<u>irr</u>	report incidence-rate ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: gnbreg, level() is equivalent to bayes, clevel(): gnbreg.

For a detailed description of *options*, see *Options for gnbreg* in [R] nbreg.

bayesopts	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-overdispersion parameter; default is normalprior(100)
<u>prior</u> (priorspec)	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> (paramref)	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> (paramref [ , blockopts ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{lnalpha: varlist}` for the log-dispersion equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] nbreg](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] nbreg](#) — Negative binomial regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)



# Title

**bayes: heckman** — Bayesian Heckman selection model

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: heckman` fits a Bayesian sample-selection linear regression to a partially observed continuous outcome; see [\[BAYES\] bayes](#) and [\[R\] heckman](#) for details.

## Quick start

Bayesian Heckman model of `y` on `x1` and `x2`, using `z1` and `z2` to model selection and using default normal priors for regression coefficients, log-standard-deviation, and atanh-correlation

```
bayes: heckman y x1 x2, select(z1 z2)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): heckman y x1 x2, select(z1 z2)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): heckman y x1 x2, select(z1 z2)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123):, ///  
heckman y x1 x2, select(z1 z2)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500):, ///  
heckman y x1 x2, select(z1 z2)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] heckman](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Heckman selection model

# Syntax

```
bayes [ , bayesopts ] : heckman depvar [indepvars] [if] [in] [weight] ,
    select( [depvars = ] varlists [ , noconstant offset(varnameo) ] ) [options]
```

<i>options</i>	Description
Model	
* <u>select</u> ()	specify selection equation: dependent and independent variables; whether to have constant term and offset variable
<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
Reporting	
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

\*`select()` is required.

The full specification is `select( [depvars = ] varlists [ , noconstant offset(varnameo) ] )`.

*indepvars* and *varlist*<sub>s</sub> may contain factor variables; see [U] 11.4.3 [Factor variables](#).

*depvar*, *indepvars*, *varlist*<sub>s</sub>, and *depvar*<sub>s</sub> may contain time-series operators; see [U] 11.4.4 [Time-series varlists](#).

*fweights* are allowed; see [U] 11.1.6 [weight](#).

`bayes: heckman`, `level()` is equivalent to `bayes, clevel(): heckman`.

For a detailed description of *options*, see [Options for Heckman selection model \(ML\)](#) and [Options for Heckman selection model \(two-step\)](#) in [R] [heckman](#).

<i>bayesopts</i>	Description
Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients, log-standard-deviation, and atanh-correlation; default is <code>normalprior(100)</code>
<i>prior</i> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<i>dryrun</i>	show model summary without estimation

## Simulation

<i>nchains</i> (#)	number of chains; default is to simulate one chain
<i>mcmcsize</i> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<i>burnin</i> (#)	burn-in period; default is <code>burnin(2500)</code>
<i>thinning</i> (#)	thinning interval; default is <code>thinning(1)</code>
<i>rseed</i> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <i>blocksize</i> (#)	maximum block size; default is <code>blocksize(50)</code>
<i>block</i> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<i>blocksummary</i>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{select: varlist_s}` for the selection equation, atanh-transformed correlation `{athrho}`, and log-standard deviation `{lnsigma}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] heckman](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Heckman selection model* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] heckman](#) — Heckman selection model

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: heckoprobit** — Bayesian ordered probit model with sample selection

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: heckoprobit` fits a Bayesian sample-selection ordered probit regression to a partially observed ordinal outcome; see [\[BAYES\] bayes](#) and [\[R\] heckoprobit](#) for details.

## Quick start

Bayesian sample-selection ordered probit regression of `y` on `x1` and `x2`, using `z1` and `z2` to model selection, and using default normal priors for regression coefficients and atanh-correlation and flat priors for cutpoints

```
bayes: heckoprobit y x1 x2, select(z1 z2)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): heckoprobit y x1 x2, select(z1 z2)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): heckoprobit y x1 x2, select(z1 z2)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123):, ///  
heckoprobit y x1 x2, select(z1 z2)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500):, ///  
heckoprobit y x1 x2, select(z1 z2)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] heckoprobit](#).

## Menu

Statistics > Ordinal outcomes > Bayesian regression > Ordered probit regression with sample selection

# Syntax

```
bayes [ , bayesopts ] : heckoprobit depvar indepvars [ if ] [ in ] [ weight ] ,
      select( [ depvars = ] varlists [ , noconstant offset(varnameo) ] ) [ options ]
```

<i>options</i>	Description
----------------	-------------

## Model

* <u>select</u> ()	specify selection equation: dependent and independent variables; whether to have constant term and offset variable
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1

## Reporting

<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

\* `select()` is required.

The full specification is `select( [ depvars = ] varlists [ , noconstant offset(varnameo) ] )`.

*indepvars* and *varlist<sub>s</sub>* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varlist<sub>s</sub>*, and *depvar<sub>s</sub>* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayes: heckoprobit`, `level()` is equivalent to `bayes, clevel(): heckoprobit`.

For a detailed description of *options*, see *Options* in [R] `heckoprobit`.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and atanh-correlation; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation

## Simulation

<code>nchains</code> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<code>rseed</code> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<code>block</code> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{select: varlist_s}` for the selection equation, atanh-transformed correlation `{athrho}`, and cutpoints `{cut1}`, `{cut2}`, and so on. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of `bayesopts`, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] [bayesmh](#). For remarks and examples specific to the bayes prefix, see [BAYES] [bayes](#). For details about the estimation command, see [R] [heckoprobit](#).

For a simple example of the bayes prefix, see *Introductory example* in [BAYES] [bayes](#). Also see *Heckman selection model* in [BAYES] [bayes](#).

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the bayes prefix

[R] [heckoprobit](#) — Ordered probit model with sample selection

[BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[BAYES] [Bayesian estimation](#) — Bayesian estimation commands

[BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis

[BAYES] [Intro](#) — Introduction to Bayesian analysis

[BAYES] [Glossary](#)



# Title

**bayes: heckprobit** — Bayesian probit model with sample selection

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: heckprobit` fits a Bayesian sample-selection probit regression to a partially observed binary outcome; see [\[BAYES\] bayes](#) and [\[R\] heckprobit](#) for details.

## Quick start

Bayesian sample-selection probit regression of `y` on `x1` and `x2`, using `z1` and `z2` to model selection and using default normal priors for regression coefficients and atanh-correlation

```
bayes: heckprobit y x1 x2, select(z1 z2)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): heckprobit y x1 x2, select(z1 z2)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): heckprobit y x1 x2, select(z1 z2)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123):, ///  
heckprobit y x1 x2, select(z1 z2)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500):, ///  
heckprobit y x1 x2, select(z1 z2)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] heckprobit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Probit model with sample selection

# Syntax

```
bayes [ , bayesopts ] : heckprobit depvar indepvars [ if ] [ in ] [ weight ] ,
      select( [ depvars = ] varlists [ , noconstant offset(varnameo) ] ) [ options ]
```

<i>options</i>	Description
----------------	-------------

## Model

* <u>select</u> ()	specify selection equation: dependent and independent variables; whether to have constant term and offset variable
<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1

## Reporting

<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

\*`select()` is required.

The full specification is `select( [ depvars = ] varlists [ , noconstant offset(varnameo) ] )`.

*indepvars* and *varlist<sub>s</sub>* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varlist<sub>s</sub>*, and *depvar<sub>s</sub>* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayes: heckprobit`, `level()` is equivalent to `bayes, clevel(): heckprobit`.

For a detailed description of *options*, see *Options* in [R] [heckprobit](#).

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and atanh-correlation; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{devar:indepvars}` for the main regression and `{select:varlist_s}` for the selection equation, and atanh-transformed correlation `{athrho}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] [bayesmh](#). For remarks and examples specific to the bayes prefix, see [BAYES] [bayes](#). For details about the estimation command, see [R] [heckprobit](#).

For a simple example of the bayes prefix, see *Introductory example* in [BAYES] [bayes](#). Also see *Heckman selection model* in [BAYES] [bayes](#).

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the bayes prefix

[R] [heckprobit](#) — Probit model with sample selection

[BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[BAYES] [Bayesian estimation](#) — Bayesian estimation commands

[BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis

[BAYES] [Intro](#) — Introduction to Bayesian analysis

[BAYES] [Glossary](#)

# Title

**bayes: hetoprobit** — Bayesian heteroskedastic ordered probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: hetoprobit` fits a Bayesian heteroskedastic ordered probit regression to an ordinal outcome; see [\[BAYES\] bayes](#) and [\[R\] hetoprobit](#) for details.

## Quick start

Bayesian heteroskedastic ordered probit regression of `y` on `x1` and `x2`, using `z1` to model the variance, and using default normal priors for regression coefficients and log-standard-deviation coefficients and flat priors for cutpoints

```
bayes: hetoprobit y x1 x2, het(z1)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): hetoprobit y x1 x2, het(z1)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): hetoprobit y x1 x2, het(z1)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
hetoprobit y x1 x2, het(z1)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
hetoprobit y x1 x2, het(z1)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] hetoprobit](#).

## Menu

Statistics > Ordinal outcomes > Bayesian regression > Heteroskedastic ordered probit regression

# Syntax

```
bayer [ , bayesopts ] : hetoprobbit depvar [ indepvars ] [ if ] [ in ] [ weight ] ,
    het(varlist [ , offset(varnameo) ]) [ options ]
```

<i>options</i>	Description
<b>Model</b>	
*het( <i>varlist</i> [...])	independent variables to model the variance and possible offset variable
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<b>Reporting</b>	
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)
<p>*het() is required. The full specification is het(<i>varlist</i> [ , <u>offset</u>(<i>varname<sub>o</sub></i>) ]).</p> <p><i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 <b>Factor variables</b>.</p> <p><i>depvar</i>, <i>indepvars</i>, and <i>varlist</i> may contain time-series operators; see [U] 11.4.4 <b>Time-series varlists</b>.</p> <p><i>fweights</i> are allowed; see [U] 11.1.6 <b>weight</b>.</p> <p>bayer: hetoprobbit, level() is equivalent to bayer, clevel(): hetoprobbit.</p> <p>For a detailed description of <i>options</i>, see <i>Options</i> in [R] <b>hetoprobbit</b>.</p>	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-standard-deviation coefficients; default is normalprior(100)
prior( <i>priorspec</i> )	prior for model parameters; this option may be repeated
dryrun	show model summary without estimation
<b>Simulation</b>	
nchains(#)	number of chains; default is to simulate one chain
mcmcsize(#)	MCMC sample size; default is mcmcsize(10000)
burnin(#)	burn-in period; default is burnin(2500)
thinning(#)	thinning interval; default is thinning(1)
rseed(#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
*blocksize(#)	maximum block size; default is blocksize(50)
block( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , <i>replace</i> ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[ , every(#) ]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}` for the main regression and `{lnsigma:varlist}` for the log-standard-deviation equation and cutpoints `{cut1}`, `{cut2}`, and so on. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of `bayesopts`, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] hetoprobit](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] hetoprobit](#) — Heteroskedastic ordered probit regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)



# Title

**bayes: hetprobit** — Bayesian heteroskedastic probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: hetprobit` fits a Bayesian heteroskedastic probit regression to a binary outcome; see [\[BAYES\] bayes](#) and [\[R\] hetprobit](#) for details.

## Quick start

Bayesian heteroskedastic probit regression of `y` on `x1` and `x2`, using `z1` to model the variance and using default normal priors for regression coefficients and log-variance coefficients

```
bayes: hetprobit y x1 x2, het(z1)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): hetprobit y x1 x2, het(z1)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): hetprobit y x1 x2, het(z1)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): hetprobit y x1 x2, het(z1)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): hetprobit y x1 x2, het(z1)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] hetprobit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Heteroskedastic probit regression

# Syntax

```
bayes [ , bayesopts ] : hetprobit depvar [ indepvars ] [ if ] [ in ] [ weight ] ,
    het(varlist [ , offset(varnameo) ]) [ options ]
```

<i>options</i>	Description
<b>Model</b>	
* <b>het</b> ( <i>varlist</i> [...])	independent variables to model the variance and possible offset variable
<b>noconstant</b>	suppress constant term
<b>offset</b> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<b>asis</b>	retain perfect predictor variables
<b>Reporting</b>	
<b>display_options</b>	control spacing, line width, and base and empty cells
<b>level</b> (#)	set credible level; default is level(95)
<p>*het() is required. The full specification is het(<i>varlist</i> [ , <i>offset</i>(<i>varname<sub>o</sub></i>) ]).</p> <p><i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 Factor variables.</p> <p><i>depvar</i>, <i>indepvars</i>, and <i>varlist</i> may contain time-series operators; see [U] 11.4.4 Time-series varlists.</p> <p><i>fweights</i> are allowed; see [U] 11.1.6 weight.</p> <p>bayes: hetprobit, level() is equivalent to bayes, clevel(): hetprobit.</p> <p>For a detailed description of <i>options</i>, see <i>Options</i> in [R] hetprobit.</p>	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <b>normalprior</b> (#)	specify standard deviation of default normal priors for regression coefficients and log-variance coefficients; default is normalprior(100)
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
<b>Simulation</b>	
<b>nchains</b> (#)	number of chains; default is to simulate one chain
<b>mcmcsize</b> (#)	MCMC sample size; default is mcmcsize(10000)
<b>burnin</b> (#)	burn-in period; default is burnin(2500)
<b>thinning</b> (#)	thinning interval; default is thinning(1)
<b>rseed</b> (#)	random-number seed
<b>exclude</b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <b>blocksize</b> (#)	maximum block size; default is blocksize(50)
<b>block</b> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b>blocksummary</b>	display block summary
* <b>noblocking</b>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , <i>replace</i> ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[ , every(#) ]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{lnsigma: varlist}` for the log-variance equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] hetprobit](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] hetprobit](#) — Heteroskedastic probit model

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: hetregress** — Bayesian heteroskedastic linear regression

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: hetregress` fits a Bayesian heteroskedastic linear regression to a continuous outcome; see [\[BAYES\] bayes](#) and [\[R\] hetregress](#) for details.

## Quick start

Bayesian heteroskedastic linear regression of `y` on `x1` and `x2`, using `z1` to model the variance and using default normal priors for regression coefficients and log-variance coefficients

```
bayes: hetregress y x1 x2, het(z1)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): hetregress y x1 x2, het(z1)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): hetregress y x1 x2, het(z1)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
hetregress y x1 x2, het(z1)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
hetregress y x1 x2, het(z1)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] hetregress](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Heteroskedastic linear regression

# Syntax

```
bayes [ , bayesopts ] : hetregress depvar [ indepvars ] [ if ] [ in ] [ weight ]
[ , options ]
```

<i>options</i>	Description
<b>Model</b>	
<b>het</b> ( <i>varlist</i> )	independent variables to model the variance
<b>noconstant</b>	suppress constant term
<b>Reporting</b>	
<b>display_options</b>	control spacing, line width, and base and empty cells
<b>level</b> (#)	set credible level; default is level(95)
<i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 Factor variables. <i>depvar</i> , <i>indepvars</i> , and <i>varlist</i> may contain time-series operators; see [U] 11.4.4 Time-series varlists. <i>fweights</i> are allowed; see [U] 11.1.6 weight. bayes: hetregress, level() is equivalent to bayes, clevel(): hetregress. For a detailed description of <i>options</i> , see <i>Options for maximum likelihood estimation</i> and <i>Options for two-step GLS estimation</i> in [R] hetregress.	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <b>normalprior</b> (#)	specify standard deviation of default normal priors for regression coefficients and log-variance coefficients; default is normalprior(100)
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
<b>Simulation</b>	
<b>nchains</b> (#)	number of chains; default is to simulate one chain
<b>mcmcsize</b> (#)	MCMC sample size; default is mcmcsize(10000)
<b>burnin</b> (#)	burn-in period; default is burnin(2500)
<b>thinning</b> (#)	thinning interval; default is thinning(1)
<b>rseed</b> (#)	random-number seed
<b>exclude</b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <b>blocksize</b> (#)	maximum block size; default is blocksize(50)
<b>block</b> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b>blocksummary</b>	display block summary
* <b>noblocking</b>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{lnsigma2: varlist}` for the log-variance equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] hetregress](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] hetregress](#) — Heteroskedastic linear regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)



# Title

**bayes: intreg** — Bayesian interval regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: intreg` fits a Bayesian interval regression to a continuous, interval-measured outcome; see [\[BAYES\] bayes](#) and [\[R\] intreg](#) for details.

## Quick start

Bayesian interval regression of `y_lower` and `y_upper` on `x1` and `x2`, using default normal priors for regression coefficients and log-variance

```
bayes: intreg y_lower y_upper x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): intreg y_lower y_upper x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y_lower: x1 x2}, uniform(-10,10)) ///  
prior({y_lower:_cons}, normal(0,10)): intreg y_lower y_upper x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
intreg y_lower y_upper x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
intreg y_lower y_upper x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] intreg](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Interval regression

# Syntax

```
bayes [ , bayesopts ] : intreg depvar1 depvar2 [indepvars] [if] [in] [weight]
[ , options ]
```

<i>options</i>	Description
<b>Model</b>	
<u>noconstant</u>	suppress constant term
<u>het</u> ( <i>varlist</i> [ , <u>noconstant</u> ])	independent variables to model the variance; use <u>noconstant</u> to suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<b>Reporting</b>	
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <u>level</u> (95)
<i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 Factor variables. <i>depvar</i> <sub>1</sub> , <i>depvar</i> <sub>2</sub> , <i>indepvars</i> , and <i>varlist</i> may contain time-series operators; see [U] 11.4.4 Time-series varlists. <i>fweights</i> are allowed; see [U] 11.1.6 weight. bayes: intreg, <u>level</u> () is equivalent to bayes, <u>clevel</u> (): intreg. For a detailed description of <i>options</i> , see <i>Options</i> in [R] intreg.	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-variance; default is <u>normalprior</u> (100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <u>mcmcsize</u> (10000)
<u>burnin</u> (#)	burn-in period; default is <u>burnin</u> (2500)
<u>thinning</u> (#)	thinning interval; default is <u>thinning</u> (1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <u>blocksize</u> (50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar1:indepvars}` and log-standard deviation `{lnsigma}` or, if option `het(varlist)` is specified, coefficients `{lnsigma:varlist}` of the log-standard-deviation equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] intreg](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] intreg](#) — Interval regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: logistic** — Bayesian logistic regression, reporting odds ratios

[Description](#)[Remarks and examples](#)[Also see](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Reference](#)

## Description

`bayes: logistic` fits a Bayesian logistic regression to a binary outcome; see [\[BAYES\]](#) [bayes](#) and [\[R\]](#) [logistic](#) for details.

## Quick start

Bayesian logistic regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: logistic y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): logistic y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): logistic y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): logistic y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcs(20000) burnin(5000) dots(500): logistic y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display coefficients instead of odds ratios

```
bayes: logistic y x1 x2, coef
```

Display coefficients on replay

```
bayes, coef
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[R\]](#) [logistic](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Logistic regression

# Syntax

```
bayaes [ , bayesopts ] : logistic depvar indepvars [ if ] [ in ] [ weight ] [ , options ]
```

<i>options</i>	Description
<b>Model</b>	
<code>noconstant</code>	suppress constant term
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1
<code>asis</code>	retain perfect predictor variables
<b>Reporting</b>	
<code>coef</code>	report estimated coefficients
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

bayaes: logistic, level() is equivalent to bayaes, clevel(): logistic.

For a detailed description of *options*, see Options in [R] logistic.

<i>bayesopts</i>	Description
<b>Priors</b>	
* <code>normalprior(#)</code>	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<code>prior(<i>priorspec</i>)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation
<b>Simulation</b>	
<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is mcmcsize(10000)
<code>burnin(#)</code>	burn-in period; default is burnin(2500)
<code>thinning(#)</code>	thinning interval; default is thinning(1)
<code>rseed(#)</code>	random-number seed
<code>exclude(<i>paramref</i>)</code>	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <code>blocksize(#)</code>	maximum block size; default is blocksize(50)
<code>block(<i>paramref</i> [ , <i>blockopts</i> ])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>coef</code>	report estimated coefficients
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] [bayesmh](#).

*paramref* may contain factor variables; see [U] [11.4.3 Factor variables](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] [bayes](#).

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] [bayesmh](#). For remarks and examples specific to the bayes prefix, see [BAYES] [bayes](#). For details about the estimation command, see [R] [logistic](#).

For a simple example of the bayes prefix, see *Introductory example* in [BAYES] [bayes](#). Also see *Logistic regression with perfect predictors* in [BAYES] [bayes](#).

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Reference

Balov, N. 2017. Bayesian logistic regression with Cauchy priors using the bayes prefix. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2017/09/08/bayesian-logistic-regression-with-cauchy-priors-using-the-bayes-prefix/>.

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the bayes prefix

[R] [logistic](#) — Logistic regression, reporting odds ratios

[BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[BAYES] [Bayesian estimation](#) — Bayesian estimation commands

[BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis

[BAYES] [Intro](#) — Introduction to Bayesian analysis

[BAYES] [Glossary](#)



# Title

**bayes: logit** — Bayesian logistic regression, reporting coefficients

[Description](#)[Remarks and examples](#)[Also see](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Reference](#)

## Description

`bayes: logit` fits a Bayesian logistic regression to a binary outcome; see [\[BAYES\]](#) [bayes](#) and [\[R\]](#) [logit](#) for details.

## Quick start

Bayesian logistic regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: logit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): logit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): logit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): logit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): logit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: logit y x1 x2, or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[R\]](#) [logit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Logistic regression

# Syntax

```
bayes [ , bayesopts ] : logit devar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
<b>Model</b>	
<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<u>asis</u>	retain perfect predictor variables
<b>Reporting</b>	
<u>or</u>	report odds ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

*devar* and *indepvars* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).

*fweights* are allowed; see [\[U\] 11.1.6 weight](#).

bayes: `logit`, `level()` is equivalent to bayes, `clevel()`: `logit`.

For a detailed description of *options*, see [Options](#) in [\[R\] logit](#).

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>or</code>	report odds ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] [bayesmh](#).

*paramref* may contain factor variables; see [U] [11.4.3 Factor variables](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] [bayes](#).

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] [bayesmh](#). For remarks and examples specific to the bayes prefix, see [BAYES] [bayes](#). For details about the estimation command, see [R] [logit](#).

For a simple example of the bayes prefix, see *Introductory example* in [BAYES] [bayes](#). Also see *Logistic regression with perfect predictors* in [BAYES] [bayes](#).

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Reference

Balov, N. 2017. Bayesian logistic regression with Cauchy priors using the bayes prefix. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2017/09/08/bayesian-logistic-regression-with-cauchy-priors-using-the-bayes-prefix/>.

## Also see

- [BAYES] [bayes](#) — Bayesian regression models using the bayes prefix
- [R] [logit](#) — Logistic regression, reporting coefficients
- [BAYES] [Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix
- [BAYES] [Bayesian estimation](#) — Bayesian estimation commands
- [BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis
- [BAYES] [Intro](#) — Introduction to Bayesian analysis
- [BAYES] [Glossary](#)

# Title

**bayes: mecloglog** — Bayesian multilevel complementary log-log regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: mecloglog` fits a Bayesian multilevel complementary log-log regression to a binary outcome; see [\[BAYES\] bayes](#) and [\[ME\] mecloglog](#) for details.

## Quick start

Bayesian two-level complementary log-log regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma prior for the variance of random intercepts

```
bayes: mecloglog y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mecloglog y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): mecloglog y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mecloglog y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): mecloglog y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display results as exponentiated coefficients

```
bayes: mecloglog y x1 x2 || id: , eform
```

Display exponentiated coefficients on replay

```
bayes, eform
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] mecloglog](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Complementary log-log regression

## Syntax

```
bayes [ , bayesopts ] : mecloglog depvar fe_equation
[ [ re_equation ] [ [ re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar : R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1
<code>asis</code>	retain perfect predictor variables

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Model	
<code>binomial(<i>varname</i>   #)</code>	set binomial trials if data are in binomial form
Reporting	
<code>eform</code>	report exponentiated coefficients
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 **Factor variables**.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 **Time-series varlists**.

*fweights* are allowed; see [U] 11.1.6 **weight**.

**bayes: mecloglog**, `level()` is equivalent to **bayes**, `clevel()`: **mecloglog**.

For a detailed description of *options*, see *Options* in [ME] **mecloglog**.

<i>bayesopts</i>	Description
<b>Priors</b>	
* <b><u>normalprior</u></b> (#)	specify standard deviation of default normal priors for regression coefficients; default is <b>normalprior</b> (100)
* <b><u>igammaprior</u></b> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <b>igammaprior</b> (0.01 0.01)
* <b><u>iwishartprior</u></b> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
<b>Simulation</b>	
<b><u>nchains</u></b> (#)	number of chains; default is to simulate one chain
<b><u>mcmcsize</u></b> (#)	MCMC sample size; default is <b>mcmcsize</b> (10000)
<b><u>burnin</u></b> (#)	burn-in period; default is <b>burnin</b> (2500)
<b><u>thinning</u></b> (#)	thinning interval; default is <b>thinning</b> (1)
<b><u>rseed</u></b> (#)	random-number seed
<b><u>exclude</u></b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b><u>restubs</u></b> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <b><u>blocksize</u></b> (#)	maximum block size; default is <b>blocksize</b> (50)
<b><u>block</u></b> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b><u>blocksummary</u></b>	display block summary
* <b><u>noblocking</u></b>	do not block parameters by default
<b>Initialization</b>	
<b><u>initial</u></b> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<b><u>init#</u></b> ( <i>initspec</i> )	specify initial values for #th chain; requires <b>nchains</b> ()
<b><u>initall</u></b> ( <i>initspec</i> )	specify initial values for all chains; requires <b>nchains</b> ()
<b><u>nomleinitial</u></b>	suppress the use of maximum likelihood estimates as starting values
<b><u>initrandom</u></b>	specify random initial values
<b><u>initsummary</u></b>	display initial values used for simulation
* <b><u>noisily</u></b>	display output from the estimation command during initialization
<b>Adaptation</b>	
<b><u>adaptation</u></b> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<b><u>scale</u></b> (#)	initial multiplier for scale factor; default is <b>scale</b> (2.38)
<b><u>covariance</u></b> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>mecloglog</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bays` prefix; other options are common between `bays` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how *renames* and *restub* are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.



## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[ME\] mecloglog](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). For multilevel examples, see *Multilevel models* in [\[BAYES\] bayes](#). Also see *Crossed-effects model* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[ME\] mecloglog](#) — Multilevel mixed-effects complementary log-log regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: meglm** — Bayesian multilevel generalized linear model

[Description](#)  
[Remarks and examples](#)

[Quick start](#)  
[Stored results](#)

[Menu](#)  
[Methods and formulas](#)

[Syntax](#)  
[Also see](#)

## Description

**bayes: meglm** fits a Bayesian multilevel generalized linear model to outcomes of different types such as continuous, binary, count, and so on; see [\[BAYES\] bayes](#) and [\[ME\] meglm](#) for details.

## Quick start

Bayesian two-level generalized linear model of *y* on *x1* and *x2* with random intercepts by *id*, using the Gaussian family and log link, and using default normal priors for regression coefficients and default inverse-gamma prior for the variance of random intercepts

```
bayes: meglm y x1 x2 || id:, family(gaussian) link(log)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): meglm y x1 x2 || id:, family(gaussian) link(log)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): ///  
meglm y x1 x2 || id:, family(gaussian) link(log)
```

Save simulation results to *simdata.dta*, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
meglm y x1 x2 || id:, family(gaussian) link(log)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): ///  
meglm y x1 x2 || id:, family(gaussian) link(log)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Fit a logit model and display results as odds ratios

```
bayes: meglm z x1 x2 || id:, family(binomial) eform
```

Display odds ratios on replay

```
bayes, eform
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] meglm](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Generalized linear model (GLM)

## Syntax

```
bayes [ , bayesopts ] : meglm depvar fe_equation
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>noconstant</code>	suppress constant term from the <b>fixed-effects</b> equation
<code>exposure(<i>varname<sub>e</sub></i>)</code>	include $\ln(\textit{varname}_e)$ in model with coefficient constrained to 1
<code>offset(<i>varname<sub>o</sub></i>)</code>	include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1
<code>asis</code>	retain perfect predictor variables

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <b>random effects</b> ; only structures <b>independent</b> , <b>identity</b> , and <b>unstructured</b> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
<b>Model</b>	
<code>family(<i>family</i>)</code>	distribution of <i>depvar</i> ; default is <code>family(gaussian)</code>
<code>link(<i>link</i>)</code>	link function; default varies per family
<b>Reporting</b>	
<code>eform</code>	report exponentiated coefficients
<code>irr</code>	report incidence-rate ratios
<code>or</code>	report odds ratios
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayer: meglm`, `level()` is equivalent to `bayer, clevel(): meglm`.

For a detailed description of *options*, see *Options* in [ME] `meglm`.

<i>bayeropts</i>	Description
<b>Priors</b>	
* <code>normalprior(#)</code>	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <code>igammaprior(# #)</code>	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <code>iwishartprior(# [...])</code>	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<code>prior(<i>priorspec</i>)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation

#### Simulation

<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(<i>paramref</i>)</code>	specify model parameters to be excluded from the simulation results
<code>restubs(<i>restub1 restub2</i> ...)</code>	specify stubs for random-effects parameters for all levels

#### Blocking

* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(<i>paramref</i> [, <i>blockopts</i>])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*irr</code>	report incidence-rate ratios
<code>*or</code>	report odds ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , replace ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots#[ , every(#) ]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>showreffects[ (<i>reref</i>) ]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>meglm</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayses` prefix; other options are common between `bayses` and `baysesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `baysesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, parameters as described in *Additional model parameters*, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayses` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *baysesopts*, see *Options* in [BAYES] `bayses`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] **Intro**. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `baysesmh`. For remarks and examples specific to the `bayses` prefix, see [BAYES] `bayses`. For details about the estimation command, see [ME] `meglm`.

For a simple example of the `bayses` prefix, see *Introductory example* in [BAYES] `bayses`. For multilevel examples, see *Multilevel models* in [BAYES] `bayses`. Also see *Crossed-effects model* in [BAYES] `bayses`.

## Additional model parameters

In addition to regression coefficients `{depvar:indepvars}`, `bayses: meglm` defines extra parameters that depend on the chosen family; see table 1 below.

Table 1. Additional model parameters defined by `bayses: meglm`

Family	Parameter	Model parameter	Default prior
Gaussian	Error variance	<code>{e.depvar:sigma2}</code>	$\text{InvGamma}(0.01, 0.01)$
Bernoulli/Binomial	None	None	None
Ordinal	Cutpoints	<code>{cut1}</code> , <code>{cut2}</code> , ...	Flat
Poisson	None	None	None
Negative binomial	Log-overdispersion	<code>{lnalpha}</code> (mean disp.) <code>{lndelta}</code> (constant disp.)	$N(0, 10000)$ $N(0, 10000)$
Gamma	Log-scale	<code>{lnscale}</code>	$N(0, 10000)$

Use the `dryrun` option with the `bayses` prefix to see the definitions of model parameters prior to estimation.

## Stored results

See *Stored results* in [BAYES] `bayses`.

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[ME\] meglm](#) — Multilevel mixed-effects generalized linear model

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: meintreg** — Bayesian multilevel interval regression

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: meintreg` fits a Bayesian multilevel interval regression to a continuous, interval-measured outcome; see [\[BAYES\] bayes](#) and [\[ME\] meintreg](#) for details.

## Quick start

Bayesian two-level interval regression of `y_lower` and `y_upper` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma priors for the error variance and for the variance of random intercepts

```
bayes: meintreg y_lower y_upper x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): meintreg y_lower y_upper x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y_lower: x1 x2}, uniform(-10,10)) ///  
prior({y_lower:_cons}, normal(0,10)): ///  
meintreg y_lower y_upper x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
meintreg y_lower y_upper x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
meintreg y_lower y_upper x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] meintreg](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Interval regression



## Syntax

```
bayes [ , bayesopts ] : meintreg depvarlower depvarupper fe_equation  
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code><u>noconstant</u></code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code><u>offset</u>(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

<i>re_options</i>	Description
Model	
<code><u>covariance</u>(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <a href="#">independent</a> , <a href="#">identity</a> , and <a href="#">unstructured</a> supported
<code><u>noconstant</u></code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Reporting	
<code><u>notable</u></code>	suppress coefficient table
<code><u>noheader</u></code>	suppress output header
<code><u>nogroup</u></code>	suppress table summarizing groups
<code><u>display_options</u></code>	control spacing, line width, and base and empty cells
<code><u>level</u>(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).  
*depvar*<sub>lower</sub>, *depvar*<sub>upper</sub>, *indepvars*, and *varlist* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).  
*fwrights* are allowed; see [\[U\] 11.1.6 weight](#).  
`bayes: meintreg`, `level()` is equivalent to `bayes, clevel(): meintreg`.  
For a detailed description of *options*, see [Options](#) in [\[ME\] meintreg](#).

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>meintreg</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvarlower: indepvars}`, error variance `{e.depvarlower: sigma2}`, random effects `{rename}`, and either variance components `{rename: sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub: Sigma, matrix}`; see *Likelihood model* in [BAYES] `bayes` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `meintreg`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`. For multilevel examples, see *Multilevel models* in [BAYES] `bayer`.

## Stored results

See *Stored results* in [BAYES] `bayer`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayermh`.

## Also see

- [BAYES] `bayer` — Bayesian regression models using the `bayer` prefix
- [ME] `meinteg` — Multilevel mixed-effects interval regression
- [BAYES] `Bayesian postestimation` — Postestimation tools for `bayermh` and the `bayer` prefix
- [BAYES] `Bayesian estimation` — Bayesian estimation commands
- [BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis
- [BAYES] `Intro` — Introduction to Bayesian analysis
- [BAYES] `Glossary`

# Title

bayes: melogit — Bayesian multilevel logistic regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: melogit` fits a Bayesian multilevel logistic regression to a binary outcome; see [\[BAYES\]](#) [bayes](#) and [\[ME\]](#) [melogit](#) for details.

## Quick start

Bayesian two-level logistic regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma prior for the variance of random intercepts

```
bayes: melogit y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): melogit y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): melogit y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): melogit y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): melogit y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: melogit y x1 x2 || id: , or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[ME\]](#) [melogit](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Logistic regression

# Syntax

```
bayes [ , bayesopts ] : melogit depvar fe_equation
      [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code><u>noconstant</u></code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code><u>offset</u>(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1
<code><u>asis</u></code>	retain perfect predictor variables

<i>re_options</i>	Description
Model	
<code><u>covariance</u>(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <a href="#">independent</a> , <a href="#">identity</a> , and <a href="#">unstructured</a> supported
<code><u>noconstant</u></code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Model	
<code><u>binomial</u>(<i>varname</i>   #)</code>	set binomial trials if data are in binomial form
Reporting	
<code><u>or</u></code>	report odds ratios
<code><u>notable</u></code>	suppress coefficient table
<code><u>noheader</u></code>	suppress output header
<code><u>nogroup</u></code>	suppress table summarizing groups
<code><u>display_options</u></code>	control spacing, line width, and base and empty cells
<code><u>level</u>(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 **Factor variables**.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 **Time-series varlists**.

*fweights* are allowed; see [U] 11.1.6 **weight**.

**bayes: melogit**, `level()` is equivalent to **bayes**, `clevel()`: **melogit**.

For a detailed description of *options*, see *Options* in [ME] **melogit**.

<i>bayesopts</i>	Description
<b>Priors</b>	
* <b><u>normalprior</u></b> (#)	specify standard deviation of default normal priors for regression coefficients; default is <b>normalprior</b> (100)
* <b><u>igammaprior</u></b> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <b>igammaprior</b> (0.01 0.01)
* <b><u>iwishartprior</u></b> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
<b>Simulation</b>	
<b>nchains</b> (#)	number of chains; default is to simulate one chain
<b>mcmcsize</b> (#)	MCMC sample size; default is <b>mcmcsize</b> (10000)
<b>burnin</b> (#)	burn-in period; default is <b>burnin</b> (2500)
<b>thinning</b> (#)	thinning interval; default is <b>thinning</b> (1)
<b>rseed</b> (#)	random-number seed
<b>exclude</b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>restubs</b> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <b>blocksize</b> (#)	maximum block size; default is <b>blocksize</b> (50)
<b>block</b> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b>blocksummary</b>	display block summary
* <b>noblocking</b>	do not block parameters by default
<b>Initialization</b>	
<b>initial</b> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<b>init#</b> ( <i>initspec</i> )	specify initial values for #th chain; requires <b>nchains</b> ()
<b>initall</b> ( <i>initspec</i> )	specify initial values for all chains; requires <b>nchains</b> ()
<b>nomleinitial</b>	suppress the use of maximum likelihood estimates as starting values
<b>initrandom</b>	specify random initial values
<b>initsummary</b>	display initial values used for simulation
* <b>noisily</b>	display output from the estimation command during initialization
<b>Adaptation</b>	
<b>adaptation</b> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<b>scale</b> (#)	initial multiplier for scale factor; default is <b>scale</b> (2.38)
<b>covariance</b> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*or</code>	report odds ratios
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>melogit</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `baves` prefix; other options are common between `baves` and `bavesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bavesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `baves` for how *renames* and *restub* are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bavesopts*, see *Options* in [BAYES] `baves`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bavesmh`. For remarks and examples specific to the `baves` prefix, see [BAYES] `baves`. For details about the estimation command, see [ME] `melogit`.



For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] **bayes**. For multilevel examples, see *Multilevel models* in [BAYES] **bayes**.

## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the `bayes` prefix

[ME] **melogit** — Multilevel mixed-effects logistic regression

[BAYES] **Bayesian postestimation** — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

bayes: menbreg — Bayesian multilevel negative binomial regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: menbreg` fits a Bayesian multilevel negative binomial regression to a nonnegative count outcome; see [\[BAYES\] bayes](#) and [\[ME\] menbreg](#) for details.

## Quick start

Bayesian two-level negative binomial regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and log-overdispersion parameter and default inverse-gamma prior for the variance of random intercepts

```
bayes: menbreg y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): menbreg y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): menbreg y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): menbreg y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): menbreg y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: menbreg y x1 x2 || id: , irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] menbreg](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Negative binomial regression

## Syntax

```
bayes [ , bayesopts ] : menbreg depvar fe_equation  
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>exposure(<i>varname</i><sub>e</sub>)</code>	include $\ln(\text{varname}_e)$ in model with coefficient constrained to 1
<code>offset(<i>varname</i><sub>o</sub>)</code>	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
----------------	-------------

Model	
<code>dispersion(<i>dispersion</i>)</code>	parameterization of the conditional overdispersion; <i>dispersion</i> may be <code>mean</code> (default) or <code>constant</code>

Reporting	
<code>irr</code>	report incidence-rate ratios
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

**bayer**: **menbreg**, **level()** is equivalent to **bayer**, **clevel()**: **menbreg**.

For a detailed description of *options*, see *Options* in [ME] **menbreg**.

<i>bayeropts</i>	Description
<b>Priors</b>	
* <b>normalprior</b> (#)	specify standard deviation of default normal priors for regression coefficients and log-overdispersion parameter; default is <b>normalprior</b> (100)
* <b>igammaprior</b> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <b>igammaprior</b> (0.01 0.01)
* <b>iwishartprior</b> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<b>prior</b> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<b>dryrun</b>	show model summary without estimation
<b>Simulation</b>	
<b>nchains</b> (#)	number of chains; default is to simulate one chain
<b>mcmcsize</b> (#)	MCMC sample size; default is <b>mcmcsize</b> (10000)
<b>burnin</b> (#)	burn-in period; default is <b>burnin</b> (2500)
<b>thinning</b> (#)	thinning interval; default is <b>thinning</b> (1)
<b>rseed</b> (#)	random-number seed
<b>exclude</b> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>restubs</b> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <b>blocksize</b> (#)	maximum block size; default is <b>blocksize</b> (50)
<b>block</b> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<b>blocksummary</b>	display block summary
* <b>noblocking</b>	do not block parameters by default
<b>Initialization</b>	
<b>initial</b> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<b>init#</b> ( <i>initspec</i> )	specify initial values for #th chain; requires <b>nchains</b> ()
<b>initall</b> ( <i>initspec</i> )	specify initial values for all chains; requires <b>nchains</b> ()
<b>nomleinitial</b>	suppress the use of maximum likelihood estimates as starting values
<b>initrandom</b>	specify random initial values
<b>initsummary</b>	display initial values used for simulation
* <b>noisily</b>	display output from the estimation command during initialization
<b>Adaptation</b>	
<b>adaptation</b> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<b>scale</b> (#)	initial multiplier for scale factor; default is <b>scale</b> (2.38)
<b>covariance</b> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*irr</code>	report incidence-rate ratios
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>menbreg</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, log-overdispersion parameter `{lnalpha}` with mean dispersion or `{lndelta}` with constant dispersion, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how *renames* and *restub* are defined.

Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `menbreg`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] `bayes`. For multilevel examples, see *Multilevel models* in [BAYES] `bayes`.

## Stored results

See *Stored results* in [BAYES] `bayes`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayesmh`.

## Also see

[BAYES] `bayes` — Bayesian regression models using the `bayes` prefix

[ME] `menbreg` — Multilevel mixed-effects negative binomial regression

[BAYES] `Bayesian postestimation` — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] `Bayesian estimation` — Bayesian estimation commands

[BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis

[BAYES] `Intro` — Introduction to Bayesian analysis

[BAYES] `Glossary`

# Title

**bayes: meologit** — Bayesian multilevel ordered logistic regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: meologit` fits a Bayesian multilevel ordered logistic regression to an ordinal outcome; see [\[BAYES\]](#) [bayes](#) and [\[ME\]](#) [meologit](#) for details.

## Quick start

Bayesian two-level ordered logistic regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients, flat priors for cutpoints, and default inverse-gamma prior for the variance of random intercepts

```
bayes: meologit y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): meologit y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): meologit y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): meologit y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): meologit y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: meologit y x1 x2 || id: , or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[ME\]](#) [meologit](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Ordered logistic regression

# Syntax

```
bayes [ , bayesopts ] : meologit depvar fe_equation
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
-------------------	-------------

Model	
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

<i>re_options</i>	Description
-------------------	-------------

Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <i>random effects</i> ; only structures <i>independent</i> , <i>identity</i> , and <i>unstructured</i> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
----------------	-------------

Reporting	
<code>or</code>	report odds ratios
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fwrights* are allowed; see [U] 11.1.6 weight.

`bayes: meologit`, `level()` is equivalent to `bayes, clevel(): meologit`.

For a detailed description of *options*, see *Options* in [ME] meologit.



<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*or</code>	report coefficients as odds ratios
<code>eform(<i>string</i>)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects(<i>refref</i>)</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>meolgit</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayermh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, cutpoints `{cut1}`, `{cut2}`, and so on, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayer` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [ME] `meolgit`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] **bayes**. For multilevel examples, see *Multilevel models* in [BAYES] **bayes**.

## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the `bayes` prefix

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[BAYES] **Bayesian postestimation** — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

**bayes: meoprobit** — Bayesian multilevel ordered probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: meoprobit` fits a Bayesian multilevel ordered probit regression to an ordinal outcome; see [\[BAYES\] bayes](#) and [\[ME\] meoprobit](#) for details.

## Quick start

Bayesian two-level ordered probit regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients, flat priors for cutpoints, and default inverse-gamma prior for the variance of random intercepts

```
bayes: meoprobit y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): meoprobit y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): meoprobit y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): meoprobit y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): meoprobit y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] meoprobit](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Ordered probit regression

Syntax

```
bayes [ , bayesopts ] : meoprobit depvar fe_equation
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar : R. varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <b>random effects</b> ; only structures <b>independent</b> , <b>identity</b> , and <b>unstructured</b> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Reporting	
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] **11.4.3 Factor variables**.  
*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.  
*fweights* are allowed; see [U] **11.1.6 weight**.  
`bayes: meoprobit`, `level()` is equivalent to `bayes, clevel()`: `meoprobit`.  
For a detailed description of *options*, see **Options** in [ME] **meoprobit**.

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>meoprobit</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, cutpoints `{cut1}`, `{cut2}`, and so on, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `meoprobit`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] `bayes`. For multilevel examples, see *Multilevel models* in [BAYES] `bayes`.

## Stored results

See *Stored results* in [BAYES] `bayes`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayesmh`.

## Also see

[BAYES] `bayes` — Bayesian regression models using the `bayes` prefix

[ME] `meoprobit` — Multilevel mixed-effects ordered probit regression

[BAYES] `Bayesian postestimation` — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] `Bayesian estimation` — Bayesian estimation commands

[BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis

[BAYES] `Intro` — Introduction to Bayesian analysis

[BAYES] `Glossary`



[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: mepoisson` fits a Bayesian multilevel Poisson regression to a nonnegative count outcome; see [\[BAYES\]](#) [bayes](#) and [\[ME\]](#) [mepoisson](#) for details.

## Quick start

Bayesian two-level Poisson regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma prior for the variance of random intercepts

```
bayes: mepoisson y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mepoisson y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): mepoisson y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mepoisson y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): mepoisson y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: mepoisson y x1 x2 || id: , irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[ME\]](#) [mepoisson](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Poisson regression

# Syntax

```
bayes [ , bayesopts ] : mepoisson depcvar fe_equation
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>exposure(<i>varname</i><sub>e</sub>)</code>	include $\ln(\text{varname}_e)$ in model with coefficient constrained to 1
<code>offset(<i>varname</i><sub>o</sub>)</code>	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Reporting	
<code>irr</code>	report incidence-rate ratios
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>llevel(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

*depcvar*, *indepvars*, and *varlist* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).

*fweights* are allowed; see [\[U\] 11.1.6 weight](#).

`bayes: mepoisson`, `level()` is equivalent to `bayes, clevel(): mepoisson`.

For a detailed description of *options*, see [Options](#) in [\[ME\] mepoisson](#).

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*irr</code>	report incidence-rate ratios
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>mepoisson</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how *renames* and *restub* are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `mepoisson`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] **bayes**. For multilevel examples, see *Multilevel models* in [BAYES] **bayes**.

## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the `bayes` prefix

[ME] **mepoisson** — Multilevel mixed-effects Poisson regression

[BAYES] **Bayesian postestimation** — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

bayes: meprobit — Bayesian multilevel probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: meprobit` fits a Bayesian multilevel probit regression to a binary outcome; see [\[BAYES\] bayes](#) and [\[ME\] meprobit](#) for details.

## Quick start

Bayesian two-level probit regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma prior for the variance of random intercepts

```
bayes: meprobit y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): meprobit y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): meprobit y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): meprobit y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): meprobit y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] meprobit](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Probit regression

## Syntax

```
bayes [ , bayesopts ] : meprobit depvar fe_equation
[ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1
<code>asis</code>	retain perfect predictor variables

<i>re_options</i>	Description
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
Model	
<code>binomial(<i>varname</i>   #)</code>	set binomial trials if data are in binomial form
Reporting	
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

**bayses: meprobit**, `level()` is equivalent to `bayses, clevel(): meprobit`.

For a detailed description of *options*, see *Options* in [ME] **meprobit**.

<i>baysesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix



## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>meprobit</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<i>display_options</i>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how *renames* and *restub* are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `meprobit`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] `bayes`. For multilevel examples, see *Multilevel models* in [BAYES] `bayes`.

## Stored results

See *Stored results* in [BAYES] `bayes`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayesmh`.

## Also see

- [BAYES] `bayes` — Bayesian regression models using the `bayes` prefix
- [ME] `meprobit` — Multilevel mixed-effects probit regression
- [BAYES] `Bayesian postestimation` — Postestimation tools for `bayesmh` and the `bayes` prefix
- [BAYES] `Bayesian estimation` — Bayesian estimation commands
- [BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis
- [BAYES] `Intro` — Introduction to Bayesian analysis
- [BAYES] `Glossary`

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>	<a href="#">Also see</a>

## Description

**bayes: mestreg** fits a Bayesian multilevel parametric survival model to a survival-time outcome; see [\[BAYES\] bayes](#) and [\[ME\] mestreg](#) for details.

## Quick start

Bayesian two-level Weibull survival model of [stset](#) survival-time outcome on **x1** and **x2** with random intercepts by **id**, using default normal priors for regression coefficients and log-ancillary parameters and default inverse-gamma prior for the variance of random intercepts

```
bayes: mestreg x1 x2 || id:, distribution(weibull)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mestreg x1 x2 || id:, distribution(weibull)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({_t: x1 x2}, uniform(-10,10)) ///
```

```
prior({_t:_cons}, normal(0,10)): ///
```

```
mestreg x1 x2 || id:, distribution(weibull)
```

Save simulation results to **simdata.dta**, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///
```

```
mestreg x1 x2 || id:, distribution(weibull)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): ///
```

```
mestreg x1 x2 || id:, distribution(weibull)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Use accelerated failure-time metric instead of proportional-hazards parameterization, and display time ratios instead of coefficients

```
bayes, tratio: mestreg x1 x2 || id:, distribution(weibull) time
```

Display time ratios on replay

```
bayes, tratio
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ME\] mestreg](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Parametric survival regression

## Syntax

```
bayes [ , bayesopts ] : mestreg fe_equation
[ || re_equation ] [ || re_equation ... ], distribution(distname) [ options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
-------------------	-------------

Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

<i>re_options</i>	Description
-------------------	-------------

Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
----------------	-------------

Model	
<code>*distribution(<i>distname</i>)</code>	specify survival distribution
<code>time</code>	use accelerated failure-time metric

Reporting	
<code>nohr</code>	do not report hazard ratios
<code>tratio</code>	report time ratios
<code>noshow</code>	do not show st setting information
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

\*`distribution(distname)` is required.

You must `stset` your data before using `bayes: mestreg`; see [ST] `stset`.

`indepvars` may contain factor variables; see [U] 11.4.3 **Factor variables**.

`fweights` are allowed; see [U] 11.1.6 **weight**.

`bayes: mestreg`, `level()` is equivalent to `bayes, clevel(): mestreg`.

For a detailed description of *options*, see *Options* in [ME] **mestreg**.

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u><code>normalprior(#)</code></u>	specify standard deviation of default normal priors for regression coefficients and log-ancillary parameters; default is <code>normalprior(100)</code>
* <u><code>igammaprior(# #)</code></u>	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u><code>iwishartprior(# [...])</code></u>	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<code>prior(<i>priorspec</i>)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation
<b>Simulation</b>	
<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(<i>paramref</i>)</code>	specify model parameters to be excluded from the simulation results
<code>restubs(<i>restub1 restub2 ...</i>)</code>	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(<i>paramref</i> [ , <i>blockopts</i> ])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default
<b>Initialization</b>	
<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization
<b>Adaptation</b>	
<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>*nohr</code>	do not report hazard ratios
<code>*tratio</code>	report time ratios; requires option <code>time</code> with <code>mestreg</code>
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>mestreg</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayermh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, ancillary parameters as described in [Ancillary model parameters](#), random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see [Likelihood model](#) in [BAYES] `bayer` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of `bayeropts`, see [Options](#) in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] [Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For

remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[ME\] mestreg](#).

For a simple example of the `bayes` prefix, see [Introductory example](#) in [\[BAYES\] bayes](#). For multilevel examples, see [Multilevel models](#) in [\[BAYES\] bayes](#).

Ancillary model parameters

In addition to regression coefficients `{_t:varlist}`, `bayes: mestreg` defines ancillary parameters that depend on the chosen survival model; see table 1 below. Positive ancillary parameters are transformed to be defined on the whole real line. All ancillary parameters are assigned default normal priors with zero mean and variance of 10,000.

Table 1. Ancillary model parameters defined by `bayes: mestreg`

Distribution	Ancillary parameters	Transformed model parameters
Exponential	None	None
Weibull	$p$	<code>{ln_p}</code>
Lognormal	$\sigma$	<code>{lnsigma}</code>
Loglogistic	$\gamma$	<code>{lngamma}</code>
Gamma	$s$	<code>{lnscale}</code>

Use the `dryrun` option with the `bayes` prefix to see the definitions of model parameters prior to estimation.

Stored results

See [Stored results](#) in [\[BAYES\] bayes](#).

Methods and formulas

See [Methods and formulas](#) in [\[BAYES\] bayesmh](#).

Also see

- [\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix
- [\[ME\] mestreg](#) — Multilevel mixed-effects parametric survival models
- [\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix
- [\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands
- [\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis
- [\[BAYES\] Intro](#) — Introduction to Bayesian analysis
- [\[BAYES\] Glossary](#)

# Title

bayes: metobit — Bayesian multilevel tobit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: metobit` fits a Bayesian multilevel tobit regression to a censored continuous outcome; see [\[BAYES\]](#) [bayes](#) and [\[ME\]](#) [metobit](#) for details.

## Quick start

Bayesian two-level tobit regression of `y` on `x1` and `x2` with random intercepts by `id`, using a lower censoring limit of 17, and using default normal priors for regression coefficients and default inverse-gamma priors for the error variance and for the variance of random intercepts

```
bayes: metobit y x1 x2 || id:, ll(17)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): metobit y x1 x2 || id:, ll(17)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): metobit y x1 x2 || id:, ll(17)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
metobit y x1 x2 || id:, ll(17)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
metobit y x1 x2 || id:, ll(17)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[ME\]](#) [metobit](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Tobit regression



## Syntax

```
bayes [ , bayesopts ] : metobit depvar fe_equation
      [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
<hr/>	
Model	
<code>noconstant</code>	suppress constant term from the <a href="#">fixed-effects</a> equation
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

<i>re_options</i>	Description
<hr/>	
Model	
<code>covariance(<i>vartype</i>)</code>	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<code>noconstant</code>	suppress constant term from the random-effects equation

<i>options</i>	Description
<hr/>	
Model	
<code>ll(<i>varname</i>   #)</code>	left-censoring variable or limit
<code>ul(<i>varname</i>   #)</code>	right-censoring variable or limit
Reporting	
<code>notable</code>	suppress coefficient table
<code>noheader</code>	suppress output header
<code>nogroup</code>	suppress table summarizing groups
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

*baves*: *metobit*, *level()* is equivalent to *baves*, *clevel()*: *metobit*.

For a detailed description of *options*, see *Options* in [ME] *metobit*.

<i>bavesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform(<i>string</i>)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects(<i>reref</i>)</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>metobit</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayesmh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, error variance `{e.depvar:sigma2}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayes` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ME] `metobit`.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] `bayes`. For multilevel examples, see *Multilevel models* in [BAYES] `bayes`.

## Stored results

See *Stored results* in [BAYES] `bayes`.

## Methods and formulas

See *Methods and formulas* in [BAYES] `bayesmh`.

## Also see

[BAYES] `bayes` — Bayesian regression models using the `bayes` prefix

[ME] `metobit` — Multilevel mixed-effects tobit regression

[BAYES] `Bayesian postestimation` — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] `Bayesian estimation` — Bayesian estimation commands

[BAYES] `Bayesian commands` — Introduction to commands for Bayesian analysis

[BAYES] `Intro` — Introduction to Bayesian analysis

[BAYES] `Glossary`

# Title

bayes: mixed — Bayesian multilevel linear regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: mixed` fits a Bayesian multilevel linear regression to a continuous outcome; see [\[BAYES\]](#) [bayes](#) and [\[ME\]](#) [mixed](#) for details.

## Quick start

Bayesian two-level linear regression of `y` on `x1` and `x2` with random intercepts by `id`, using default normal priors for regression coefficients and default inverse-gamma priors for the error variance and for the variance of random intercepts

```
bayes: mixed y x1 x2 || id:
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mixed y x1 x2 || id:
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): mixed y x1 x2 || id:
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mixed y x1 x2 || id:
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): mixed y x1 x2 || id:
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[ME\]](#) [mixed](#).

## Menu

Statistics > Multilevel mixed-effects models > Bayesian regression > Linear regression

# Syntax

```
bayes [ , bayesopts ] : mixed depvar fe_equation
      [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe\_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re\_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable

```
levelvar: R.varname
```

*levelvar* either is a variable identifying the group structure for the random effects at that level or is `_all`, representing one group comprising all observations.

<i>fe_options</i>	Description
-------------------	-------------

Model

<u><code>noconstant</code></u>	suppress constant term from the <a href="#">fixed-effects</a> equation
--------------------------------	--

<i>re_options</i>	Description
-------------------	-------------

Model

<u><code>covariance</code></u> ( <i>vartype</i> )	variance–covariance structure of the <a href="#">random effects</a> ; only structures <code>independent</code> , <code>identity</code> , and <code>unstructured</code> supported
<u><code>noconstant</code></u>	suppress constant term from the random-effects equation

<i>options</i>	Description
----------------	-------------

Reporting

<u><code>noheader</code></u>	suppress output header
<u><code>nogroup</code></u>	suppress table summarizing groups
<u><code>display_options</code></u>	control spacing, line width, and base and empty cells
<u><code>level</code></u> (#)	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

*depvar*, *indepvars*, and *varlist* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).

*fweights* are allowed; see [\[U\] 11.1.6 weight](#).

`bayes: mixed`, `level()` is equivalent to `bayes, clevel(): mixed`.

For a detailed description of *options*, see [Options](#) in [\[ME\] mixed](#).

<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <u>igammaprior</u> (# #)	specify shape and scale of default inverse-gamma prior for variance components; default is <code>igammaprior(0.01 0.01)</code>
* <u>iwishartprior</u> (# [...])	specify degrees of freedom and, optionally, scale matrix of default inverse-Wishart prior for unstructured random-effects covariance
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<u>restubs</u> ( <i>restub1 restub2 ...</i> )	specify stubs for random-effects parameters for all levels
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [, <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default
<b>Initialization</b>	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization
<b>Adaptation</b>	
<u>adaptation</u> ( <i>adaptopts</i> )	control the adaptive MCMC procedure
<u>scale</u> (#)	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<u>covariance</u> ( <i>cov</i> )	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[<i>(string)</i>]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>remargl</code>	compute log marginal-likelihood
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>nomesummary</code>	suppress multilevel-structure summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>dots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>showeffects[<i>(reref)</i>]</code>	specify that all or a subset of random-effects parameters be included in the output
<code>melabel</code>	display estimation table using the same row labels as <code>mixed</code>
<code>nogroup</code>	suppress table summarizing groups
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayermh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}`, error variance `{e.depvar:sigma2}`, random effects `{rename}`, and either variance components `{rename:sigma2}` or, if option `covariance(unstructured)` is specified, matrix parameter `{restub:Sigma,matrix}`; see *Likelihood model* in [BAYES] `bayer` for how `renames` and `restub` are defined. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using adaptive Metropolis–Hastings and Gibbs algorithms, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [ME] `mixed`.



For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] **bayes**. For multilevel examples, see *Multilevel models* in [BAYES] **bayes**.

By default, `bayes: mixed` uses Gibbs sampling for all model parameters except the random-effects parameters. If you specify a `prior()` distribution for which Gibbs sampling is not available, `bayes: mixed` will switch to adaptive Metropolis–Hastings sampling. In general, `bayes: mixed` will try to use a more efficient Gibbs sampling for the model parameters whenever available.

## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the `bayes` prefix

[ME] **mixed** — Multilevel mixed-effects linear regression

[BAYES] **Bayesian postestimation** — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

**bayes: mlogit** — Bayesian multinomial logistic regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: mlogit` fits a Bayesian multinomial logistic regression to a categorical outcome; see [\[BAYES\] bayes](#) and [\[R\] mlogit](#) for details.

## Quick start

Bayesian multinomial logistic regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: mlogit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mlogit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept for the category 2

```
bayes, prior({2: x1 x2}, uniform(-10,10)) ///  
prior({2:_cons}, normal(0,10)): mlogit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mlogit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): mlogit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display relative-risk ratios instead of coefficients

```
bayes: mlogit y x1 x2, rrr
```

Display relative-risk ratios on replay

```
bayes, rrr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] mlogit](#).

## Menu

Statistics > Categorical outcomes > Bayesian multinomial logistic regression

Syntax

```
bayes [ , bayesopts ] : mlogit depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model

<u>noconstant</u>	suppress constant term
<u>baseoutcome</u> (#)	value of <i>depvar</i> that will be the base outcome

Reporting

<u>rrr</u>	report relative-risk ratios
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
*fweights* are allowed; see [U] 11.1.6 weight.  
bayes: mlogit, level() is equivalent to bayes, clevel(): mlogit.  
For a detailed description of *options*, see Options in [R] mlogit.

<i>bayesopts</i>	Description
------------------	-------------

Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsz</u> ie(#)	MCMC sample size; default is mcmcsz(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

Initialization

<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init</u> #( <i>initspec</i> )	specify initial values for #th chain; requires nchains()
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires nchains()
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrando</u> m	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>rrr</code>	report relative-risk ratios
<code>eform(<i>string</i>)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots(#[, <i>every</i>(#)])</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

`priorspec` and `paramref` are defined in [BAYES] `bayermh`.

`paramref` may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{outcome1:indeps}`, `{outcome2:indeps}`, and so on, where *outcome*<sub>#</sub>'s are the values of the dependent variable or the value labels of the dependent variable if they exist. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] *Intro*. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [R] `mlogit`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`. Also see *Multinomial logistic regression* in [BAYES] `bayer`.

## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[R] **mlogit** — Multinomial (polytomous) logistic regression

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

**bayes: mprobit** — Bayesian multinomial probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: mprobit` fits a Bayesian multinomial probit regression to a categorical outcome; see [\[BAYES\] bayes](#) and [\[R\] mprobit](#) for details.

## Quick start

Bayesian multinomial probit regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: mprobit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mprobit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept for the category 2

```
bayes, prior({2: x1 x2}, uniform(-10,10)) ///  
prior({2:_cons}, normal(0,10)): mprobit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mprobit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): mprobit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] mprobit](#).

## Menu

Statistics > Categorical outcomes > Bayesian multinomial probit regression

Syntax

```
bayes [ , bayesopts ] : mprobit depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model

<u>noconstant</u>	suppress constant term
<u>baseoutcome</u> (#)	value of <i>depvar</i> that will be the base outcome
<u>probitparam</u>	use the probit variance parameterization

Reporting

<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayes: mprobit, level()` is equivalent to `bayes, clevel(): mprobit`.

For a detailed description of *options*, see *Options* in [R] **mprobit**.

<i>bayesopts</i>	Description
------------------	-------------

Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking

* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

Initialization

<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(adaptopts)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(cov)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform(string)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(search_options)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{outcome1:indepvars}`, `{outcome2:indepvars}`, and so on, where *outcome<sub>#</sub>*'s are the values of the dependent variable or the value labels of the dependent variable if they exist. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [R] `mprobit`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`. Also see *Multinomial logistic regression* in [BAYES] `bayer`.



## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[R] **mprobit** — Multinomial probit regression

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>	<a href="#">Also see</a>

Description

`bayes: mvreg` fits a Bayesian multivariate regression to multiple continuous outcomes; see [\[BAYES\] bayes](#) and [\[MV\] mvreg](#) for details.

Quick start

Bayesian multivariate regression of `y1` and `y2` on `x1` and `x2`, using default normal priors for regression coefficients and Jeffreys prior for the covariance matrix

```
bayes: mvreg y1 y2 = x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): mvreg y1 y2 = x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept of the dependent variable `y2`

```
bayes, prior({y2: x1 x2}, uniform(-10,10)) ///  
prior({y2:_cons}, normal(0,10)): mvreg y1 y2 = x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): mvreg y1 y2 = x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): mvreg y1 y2 = x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[MV\] mvreg](#).

Menu

Statistics > Linear models and related > Bayesian regression > Multivariate regression

Syntax

```
bayes [ , bayesopts ] : mvreg depvars = indepvars [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model

**noconstant** suppress constant term

Reporting

*display\_options* control spacing, line width, and base and empty cells

**level(#)** set credible level; default is **level(95)**

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

*fwrights* are allowed; see [\[U\] 11.1.6 weight](#).

**bayes: mvreg**, **level()** is equivalent to **bayes, clevel(): mvreg**.

For a detailed description of *options*, see [Options](#) in [\[MV\] mvreg](#).

<i>bayesopts</i>	Description
------------------	-------------

Priors

- \***gibbs** specify Gibbs sampling; available only with normal priors for regression coefficients and multivariate Jeffreys prior for covariance
- \***normalprior(#)** specify standard deviation of default normal priors for regression coefficients; default is **normalprior(100)**
- prior(*priorspec*)** prior for model parameters; this option may be repeated
- dryrun** show model summary without estimation

Simulation

- nchains(#)** number of chains; default is to simulate one chain
- mcmcsize(#)** MCMC sample size; default is **mcmcsize(10000)**
- burnin(#)** burn-in period; default is **burnin(2500)**
- thinning(#)** thinning interval; default is **thinning(1)**
- rseed(#)** random-number seed
- exclude(*paramref*)** specify model parameters to be excluded from the simulation results

Blocking

- \***blocksize(#)** maximum block size; default is **blocksize(50)**
- block(*paramref* [ , *blockopts* ])** specify a block of model parameters; this option may be repeated
- blocksummary** display block summary
- \***noblocking** do not block parameters by default

Initialization

- initial(*initspec*)** specify initial values for model parameters with a single chain
- init#(*initspec*)** specify initial values for #th chain; requires **nchains()**
- initall(*initspec*)** specify initial values for all chains; requires **nchains()**
- nomleinitial** suppress the use of maximum likelihood estimates as starting values
- initrandom** specify random initial values
- initsummary** display initial values used for simulation
- \***noisily** display output from the estimation command during initialization

## Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar1: indepvars}`, `{depvar2: indepvars}`, and so on, and covariance matrix `{Sigma, matrix}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Multivariate Jeffreys prior, `jeffreys(d)`, is used by default for the covariance matrix of dimension *d*.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using adaptive Metropolis–Hastings and Gibbs algorithms, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [MV] `mvreg`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`.

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[MV\] mvreg](#) — Multivariate regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: nbreg** — Bayesian negative binomial regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: nbreg` fits a Bayesian negative binomial regression to a nonnegative count outcome; see [\[BAYES\] bayes](#) and [\[R\] nbreg](#) for details.

## Quick start

Bayesian negative binomial regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients and log-overdispersion parameter

```
bayes: nbreg y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): nbreg y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): nbreg y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): nbreg y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): nbreg y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: nbreg y x1 x2, irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] nbreg](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Negative binomial regression

Syntax

```
bayes [ , bayesopts ] : nbreg depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model

<u>noconstant</u>	suppress constant term
<u>dispersion</u> ( <u>mean</u> )	parameterization of dispersion; the default
<u>dispersion</u> ( <u>constant</u> )	constant dispersion for all observations
<u>exposure</u> ( <i>varname<sub>e</sub></i> )	include ln( <i>varname<sub>e</sub></i> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname<sub>o</sub></i> )	include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1

Reporting

<u>irr</u>	report incidence-rate ratios
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varname<sub>e</sub>*, and *varname<sub>o</sub>* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fwweights* are allowed; see [U] 11.1.6 weight.

bayes: nbreg, level() is equivalent to bayes, clevel(): nbreg.

For a detailed description of options, see Options for nbreg in [R] nbreg.

<i>bayesopts</i>	Description
------------------	-------------

Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-overdispersion parameter; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

### Initialization

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

### Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

### Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

### Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` and log-overdispersion parameter `{lnalpha}` with mean dispersion or `{lndelta}` with constant dispersion. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.



## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] nbreg](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] nbreg](#) — Negative binomial regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: ologit** — Bayesian ordered logistic regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: ologit` fits a Bayesian ordered logistic regression to an ordinal outcome; see [\[BAYES\] bayes](#) and [\[R\] ologit](#) for details.

## Quick start

Bayesian ordered logistic regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients and flat priors for cutpoints

```
bayes: ologit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): ologit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): ologit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ologit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ologit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display odds ratios instead of coefficients

```
bayes: ologit y x1 x2, or
```

Display odds ratios on replay

```
bayes, or
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] ologit](#).

## Menu

Statistics > Ordinal outcomes > Bayesian regression > Ordered logistic regression

Syntax

```
bayes [ , bayesopts ] : ologit depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model	
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
Reporting	
<u>or</u>	report odds ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fwrights* are allowed; see [U] 11.1.6 weight.

`bayes: ologit`, `level()` is equivalent to `bayes, clevel(): ologit`.

For a detailed description of *options*, see *Options* in [R] `ologit`.

<i>bayesopts</i>	Description
------------------	-------------

Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

Initialization	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization

### Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

### Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>* or</code>	report odds ratios
<code>eform[(<i>string</i>)]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

### Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayses` prefix; other options are common between `bayses` and `baysesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `baysesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` and cutpoints `{cut1}`, `{cut2}`, and so on. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of *baysesopts*, see *Options* in [BAYES] `bayses`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `baysesmh`. For remarks and examples specific to the `bayses` prefix, see [BAYES] `bayses`. For details about the estimation command, see [R] `ologit`.

For a simple example of the `bayses` prefix, see *Introductory example* in [BAYES] `bayses`.

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] ologit](#) — Ordered logistic regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: oprobit** — Bayesian ordered probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: oprobit` fits a Bayesian ordered probit regression to an ordinal outcome; see [\[BAYES\] bayes](#) and [\[R\] oprobit](#) for details.

## Quick start

Bayesian ordered probit regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients and flat priors for cutpoints

```
bayes: oprobit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): oprobit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): oprobit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): oprobit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): oprobit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] oprobit](#).

## Menu

Statistics > Ordinal outcomes > Bayesian regression > Ordered probit regression

Syntax

```
bayes [ , bayesopts ] : oprobit depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model  
offset(*varname*) include *varname* in model with coefficient constrained to 1

Reporting  
*display\_options* control spacing, line width, and base and empty cells  
level(#) set credible level; default is `level(95)`

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
*fwweights* are allowed; see [U] 11.1.6 weight.  
*bayes*: `oprobit`, `level()` is equivalent to `bayes`, `clevel()`: `oprobit`.  
For a detailed description of *options*, see *Options* in [R] `oprobit`.

<i>bayesopts</i>	Description
------------------	-------------

Priors  
\*normalprior(#) specify standard deviation of default normal priors for regression coefficients; default is `normalprior(100)`  
prior(*priorspec*) prior for model parameters; this option may be repeated  
dryrun show model summary without estimation

Simulation  
nchains(#) number of chains; default is to simulate one chain  
mcmcsize(#) MCMC sample size; default is `mcmcsize(10000)`  
burnin(#) burn-in period; default is `burnin(2500)`  
thinning(#) thinning interval; default is `thinning(1)`  
rseed(#) random-number seed  
exclude(*paramref*) specify model parameters to be excluded from the simulation results

Blocking  
\*blocksize(#) maximum block size; default is `blocksize(50)`  
block(*paramref* [ , *blockopts* ]) specify a block of model parameters; this option may be repeated  
blocksummary display block summary  
\*noblocking do not block parameters by default

Initialization  
initial(*initspec*) specify initial values for model parameters with a single chain  
init#(*initspec*) specify initial values for #th chain; requires `nchains()`  
initall(*initspec*) specify initial values for all chains; requires `nchains()`  
nomleinitial suppress the use of maximum likelihood estimates as starting values  
initrandom specify random initial values  
initsummary display initial values used for simulation  
\*noisily display output from the estimation command during initialization

### Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

### Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots(#[, every(#)])</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

### Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayses` prefix; other options are common between `bayses` and `baysesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `baysesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` and cutpoints `{cut1}`, `{cut2}`, and so on. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of *baysesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `baysesmh`. For remarks and examples specific to the `bayses` prefix, see [BAYES] `bayes`. For details about the estimation command, see [R] `oprobit`.

For a simple example of the `bayses` prefix, see *Introductory example* in [BAYES] `bayes`.



## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] oprobit](#) — Ordered probit regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: poisson** — Bayesian Poisson regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: poisson` fits a Bayesian Poisson regression to a nonnegative count outcome; see [\[BAYES\] bayes](#) and [\[R\] poisson](#) for details.

## Quick start

Bayesian Poisson regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: poisson y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): poisson y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): poisson y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): poisson y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): poisson y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: poisson y x1 x2, irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] poisson](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Poisson regression

Syntax

```
bayes [ , bayesopts ] : poisson depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model	
<u>noconstant</u>	suppress constant term
<u>exposure</u> ( <i>varname<sub>e</sub></i> )	include ln( <i>varname<sub>e</sub></i> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname<sub>o</sub></i> )	include <i>varname<sub>o</sub></i> in model with coefficient constrained to 1

Reporting	
<u>irr</u>	report incidence-rate ratios
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar*, *indepvars*, *varname<sub>e</sub>*, and *varname<sub>o</sub>* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: poisson, level() is equivalent to bayes, clevel(): poisson.

For a detailed description of *options*, see Options in [R] poisson.

<i>bayesopts</i>	Description
------------------	-------------

Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking	
* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

## Initialization

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] poisson](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] poisson](#) — Poisson regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

bayes: probit — Bayesian probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: probit` fits a Bayesian probit regression to a binary outcome; see [\[BAYES\] bayes](#) and [\[R\] probit](#) for details.

## Quick start

Bayesian probit regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients

```
bayes: probit y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): probit y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): probit y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): probit y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): probit y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] probit](#).

## Menu

Statistics > Binary outcomes > Bayesian regression > Probit regression

Syntax

```
bayes [ , bayesopts ] : probit depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

Model	
<u>noconstant</u>	suppress constant term
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1
<u>asis</u>	retain perfect predictor variables

Reporting	
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayes: probit`, `level()` is equivalent to `bayes, clevel(): probit`.

For a detailed description of *options*, see *Options* in [R] `probit`.

<i>bayesopts</i>	Description
------------------	-------------

Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is <code>mcmcsize(10000)</code>
<u>burnin</u> (#)	burn-in period; default is <code>burnin(2500)</code>
<u>thinning</u> (#)	thinning interval; default is <code>thinning(1)</code>
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking	
* <u>blocksize</u> (#)	maximum block size; default is <code>blocksize(50)</code>
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

Initialization	
<u>initial</u> ( <i>initspec</i> )	specify initial values for model parameters with a single chain
<u>init#</u> ( <i>initspec</i> )	specify initial values for #th chain; requires <code>nchains()</code>
<u>initall</u> ( <i>initspec</i> )	specify initial values for all chains; requires <code>nchains()</code>
<u>nomleinitial</u>	suppress the use of maximum likelihood estimates as starting values
<u>initrandom</u>	specify random initial values
<u>initsummary</u>	display initial values used for simulation
* <u>noisily</u>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

## Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients {*depvar:indepvars*}. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] `Intro`. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayermh`. For remarks and examples specific to the `bayer` prefix, see [BAYES] `bayer`. For details about the estimation command, see [R] `probit`.

For a simple example of the `bayer` prefix, see *Introductory example* in [BAYES] `bayer`. Also see *Logistic regression with perfect predictors* in [BAYES] `bayer`.



## Stored results

See *Stored results* in [BAYES] **bayes**.

## Methods and formulas

See *Methods and formulas* in [BAYES] **bayesmh**.

## Also see

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[R] **probit** — Probit regression

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

# Title

bayes: regress — Bayesian linear regression

Description

Remarks and examples

Quick start

Stored results

Menu

Methods and formulas

Syntax

Also see

## Description

`bayes: regress` fits a Bayesian linear regression to a continuous outcome; see [\[BAYES\] bayes](#) and [\[R\] regress](#) for details.

## Quick start

Bayesian linear regression of `y` on `x1` and `x2`, using default normal priors for regression coefficients and default inverse-gamma prior for the variance

```
bayes: regress y x1 x2
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): regress y x1 x2
```

Use a shape of 1 and a scale of 2 instead of values of 0.01 for the default inverse-gamma prior

```
bayes, igammaprior(1 2): regress y x1 x2
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): regress y x1 x2
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): regress y x1 x2
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): regress y x1 x2
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] regress](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Linear regression

Syntax

```
bayes [ , bayesopts ] : regress depvar [ indepvars ] [ if ] [ in ] [ weight ] [ , options ]
```

options	Description
---------	-------------

Model	
<code>noconstant</code>	suppress constant term
Reporting	
<code>eform(string)</code> <code>display_options</code>	report exponentiated coefficients and label as <i>string</i> control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.  
*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.  
*weights* are allowed; see [U] 11.1.6 weight.  
`bayes: regress, level()` is equivalent to `bayes, clevel(): regress`.  
For a detailed description of *options*, see *Options* in [R] regress.

bayesopts	Description
-----------	-------------

Priors	
* <code>gibbs</code>	specify Gibbs sampling; available only with normal priors for regression coefficients and an inverse-gamma prior for variance
* <code>normalprior(#)</code>	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <code>igammaprior(# #)</code>	specify shape and scale of default inverse-gamma prior for variance; default is <code>igammaprior(0.01 0.01)</code>
<code>prior(priorspec)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation
Simulation	
<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(paramref)</code>	specify model parameters to be excluded from the simulation results
Blocking	
* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(paramref [ , blockopts ])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default

### Initialization

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

### Adaptation

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

### Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

### Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayer` prefix; other options are common between `bayer` and `bayermh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayermh`.

*paramref* may contain factor variables; see [U] 11.4.3 Factor variables.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}` and variance `{sigma2}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayeropts*, see *Options* in [BAYES] `bayer`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using adaptive Metropolis–Hastings and Gibbs algorithms, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] regress](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Linear regression: A case of informative default priors* in [\[BAYES\] bayes](#).

## Video examples

[Bayesian linear regression using the bayes prefix](#)

[Bayesian linear regression using the bayes prefix: How to specify custom priors](#)

[Bayesian linear regression using the bayes prefix: Checking convergence of the MCMC chain](#)

[Bayesian linear regression using the bayes prefix: How to customize the MCMC chain](#)

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] regress](#) — Linear regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: streg` fits a Bayesian parametric survival model to a survival-time outcome; see [\[BAYES\] bayes](#) and [\[ST\] streg](#) for details.

## Quick start

Bayesian Weibull survival model of `stset` survival-time outcome on `x1` and `x2`, using default normal priors for regression coefficients and log-ancillary parameters

```
bayes: streg x1 x2, distribution(weibull)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): streg x1 x2, distribution(weibull)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({_t: x1 x2}, uniform(-10,10)) ///
prior({_t:_cons}, normal(0,10)): streg x1 x2, distribution(weibull)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///
streg x1 x2, distribution(weibull)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///
streg x1 x2, distribution(weibull)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Use accelerated failure-time metric instead of proportional-hazards parameterization, and display time ratios instead of coefficients

```
bayes, tratio: streg x1 x2, distribution(weibull) time
```

Display time ratios on replay

```
bayes, tratio
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[ST\] streg](#).

## Menu

Statistics > Survival analysis > Regression models > Bayesian parametric survival models

Syntax

```
bayes [ , bayesopts ] : streg [varlist] [if] [in] [ , options ]
```

options	Description
Model	
<u>noconstant</u>	suppress constant term
<u>distribution(exponential)</u>	exponential survival distribution
<u>distribution(gompertz)</u>	Gompertz survival distribution
<u>distribution(loglogistic)</u>	loglogistic survival distribution
<u>distribution(llogistic)</u>	synonym for <code>distribution(loglogistic)</code>
<u>distribution(weibull)</u>	Weibull survival distribution
<u>distribution(lognormal)</u>	lognormal survival distribution
<u>distribution(lnormal)</u>	synonym for <code>distribution(lognormal)</code>
<u>distribution(ggamma)</u>	generalized gamma survival distribution
<u>frailty(gamma)</u>	gamma frailty distribution
<u>frailty(invgaussian)</u>	inverse-Gaussian distribution
<u>time</u>	use accelerated failure-time metric
Model 2	
<u>strata(varname)</u>	strata ID variable
<u>offset(varname)</u>	include <i>varname</i> in model with coefficient constrained to 1
<u>shared(varname)</u>	shared frailty ID variable
<u>ancillary(varlist)</u>	use <i>varlist</i> to model the first ancillary parameter
<u>anc2(varlist)</u>	use <i>varlist</i> to model the second ancillary parameter
Reporting	
<u>nohr</u>	do not report hazard ratios
<u>tratio</u>	report time ratios
<u>noshow</u>	do not show st setting information
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level(#)</u>	set credible level; default is <code>level(95)</code>

You must `stset` your data before using `bayes: streg`; see [\[ST\] stset](#).  
*varlist* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).  
`bayes: streg, level()` is equivalent to `bayes, clevel(): streg`.  
For a detailed description of *options*, see [Options](#) in [\[ST\] streg](#).

bayesopts	Description
Priors	
* <u>normalprior(#)</u>	specify standard deviation of default normal priors for regression coefficients and log-ancillary parameters; default is <code>normalprior(100)</code>
<u>prior(priorspec)</u>	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(paramref)</code>	specify model parameters to be excluded from the simulation results

## Blocking

* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(paramref[, blockopts])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default

## Initialization

<code>initial(initspec)</code>	specify initial values for model parameters with a single chain
<code>init#(initspec)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(initspec)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

## Adaptation

<code>adaptation(adaptopts)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(cov)</code>	initial proposal covariance; default is the identity matrix

## Reporting

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>nohr</code>	do not report hazard ratios
* <code>tratio</code>	report time ratios; requires option <code>time</code> with <code>streg</code>
<code>eform(string)</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(filename[, replace])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every#)]</code>	display dots as simulation is performed
<code>[no]show(paramref)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(string)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells



Advanced

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`. Options `prior()` and `block()` may be repeated. `priorspec` and `paramref` are defined in [BAYES] `bayesmh`. `paramref` may contain factor variables; see [U] 11.4.3 Factor variables. See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands. Model parameters are regression coefficients `{depvar:indepvars}` and ancillary parameters as described in Ancillary model parameters. Use the `dryrun` option to see the definitions of model parameters prior to estimation. For a detailed description of `bayesopts`, see Options in [BAYES] `bayes`.

Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] Intro. For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [BAYES] `bayesmh`. For remarks and examples specific to the `bayes` prefix, see [BAYES] `bayes`. For details about the estimation command, see [ST] `streg`. For a simple example of the `bayes` prefix, see Introductory example in [BAYES] `bayes`. Also see Parametric survival model in [BAYES] `bayes`.

Ancillary model parameters

In addition to regression coefficients `{_t:varlist}`, `bayes: streg` defines ancillary parameters that depend on the chosen survival model; see table 1 below. Positive ancillary parameters are transformed to be defined on the whole real line. All ancillary parameters are assigned default normal priors with zero mean and variance of 10,000.

Table 1. Ancillary model parameters defined by `bayes: streg`

Distribution	Ancillary parameters	Transformed model parameters
Exponential	None	None
Weibull	$p$	<code>{ln_p}</code>
Gompertz	$\gamma$	<code>{gamma}</code>
Lognormal	$\sigma$	<code>{lnsigma}</code>
Loglogistic	$\gamma$	<code>{lngamma}</code>
Generalized gamma	$\sigma, \kappa$	<code>{lnsigma}</code> , <code>{kappa}</code>

For frailty models, when option `frailty()` or option `shared()` is specified with `streg`, `bayes: streg` also defines the log-frailty parameter `{lntheta}`. If option `ancillary(varlist)` is specified, regression coefficients `{ln_p:varlist}`, `{gamma:varlist}`, and so on are defined for all ancillary parameters except  $\kappa$ . If option `anc2(varlist)` is specified, then regression coefficients `{kappa:varlist}` are defined for  $\kappa$ .

If option `strata(varname)` is specified, additional stratum-specific coefficients of the form `{eqname:#.varname}` are defined for the main regression and ancillary parameters. For example, if `drug` contains three strata, then specifying option `strata(drug)` will result in additional main regression coefficients `{_t:2.drug}` and `{_t:3.drug}` and—say, for Weibull regression—in additional parameters `{ln_p:2.drug}` and `{ln_p:3.drug}`. In the model summary with default priors, you may see these parameters labeled as `{_t:i.drug}` and `{ln_p:i.drug}`, for short.

Use the `dryrun` option with the `bayes` prefix to see the definitions of model parameters prior to estimation.

## Stored results

See *Stored results* in [BAYES] [bayes](#).

## Methods and formulas

See *Methods and formulas* in [BAYES] [bayesmh](#).

## Also see

[BAYES] [bayes](#) — Bayesian regression models using the `bayes` prefix

[ST] [streg](#) — Parametric survival models

[BAYES] [Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[BAYES] [Bayesian estimation](#) — Bayesian estimation commands

[BAYES] [Bayesian commands](#) — Introduction to commands for Bayesian analysis

[BAYES] [Intro](#) — Introduction to Bayesian analysis

[BAYES] [Glossary](#)

# Title

**bayes: tnbreg** — Bayesian truncated negative binomial regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: tnbreg** fits a Bayesian truncated negative binomial regression to a positive count outcome whose values are all above the truncation point; see [\[BAYES\] bayes](#) and [\[R\] tnbreg](#) for details.

## Quick start

Bayesian truncated negative binomial regression of *y* on *x1* and *x2*, using a lower truncation limit of 5 and using default normal priors for regression coefficients and log-overdispersion parameter

```
bayes: tnbreg y x1 x2, ll(5)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): tnbreg y x1 x2, ll(5)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): tnbreg y x1 x2, ll(5)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): tnbreg y x1 x2, ll(5)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): tnbreg y x1 x2, ll(5)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: tnbreg y x1 x2, ll(5) irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] tnbreg](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Truncated negative binomial regression

# Syntax

```
bayes [ , bayesopts ] : tnbreg depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

## Model

<u>noconstant</u>	suppress constant term
ll(#  <i>varname</i> )	truncation point; default value is ll(0), zero truncation
<u>dispersion</u> ( <u>mean</u> )	parameterization of dispersion; the default
<u>dispersion</u> ( <u>constant</u> )	constant dispersion for all observations
<u>exposure</u> ( <i>varname</i> <sub>e</sub> )	include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> <sub>o</sub> )	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1

## Reporting

<u>irr</u>	report incidence-rate ratios
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [\[U\] 11.4.3 Factor variables](#).

*depvar* and *indepvars* may contain time-series operators; see [\[U\] 11.4.4 Time-series varlists](#).

fweights are allowed; see [\[U\] 11.1.6 weight](#).

bayes: tnbreg, level() is equivalent to bayes, clevel(): tnbreg.

For a detailed description of *options*, see [Options](#) in [\[R\] tnbreg](#).

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-overdispersion parameter; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
dryrun	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` and log-overdispersion parameter `{lnalpha}` with mean dispersion or `{lndelta}` with constant dispersion. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] tnbreg](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Truncated Poisson regression* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] tnbreg](#) — Truncated negative binomial regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

bayes: tobit — Bayesian tobit regression

<a href="#">Description</a>	<a href="#">Quick start</a>	<a href="#">Menu</a>	<a href="#">Syntax</a>
<a href="#">Remarks and examples</a>	<a href="#">Stored results</a>	<a href="#">Methods and formulas</a>	<a href="#">Also see</a>

## Description

`bayes: tobit` fits a Bayesian tobit regression to a censored continuous outcome; see [\[BAYES\] bayes](#) and [\[R\] tobit](#) for details.

## Quick start

Bayesian tobit regression of `y` on `x1` and `x2`, using a lower censoring limit of 17 and using default normal priors for regression coefficients and default inverse-gamma prior for the variance

```
bayes: tobit y x1 x2, ll(17)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): tobit y x1 x2, ll(17)
```

Use a shape of 1 and a scale of 2 instead of values of 0.01 for the default inverse-gamma prior

```
bayes, igammaprior(1 2): tobit y x1 x2, ll(17)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///
prior({y:_cons}, normal(0,10)): tobit y x1 x2, ll(17)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): tobit y x1 x2, ll(17)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): tobit y x1 x2, ll(17)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] tobit](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Tobit regression

# Syntax

```
bayer [, bayesopts] : tobit depvar [indepvars] [if] [in] [weight] [, options]
```

options	Description
<b>Model</b>	
<code>noconstant</code>	suppress constant term
<code>ll[ (varname   #) ]</code>	left-censoring variable or limit
<code>ul[ (varname   #) ]</code>	right-censoring variable or limit
<code>offset(varname)</code>	include <i>varname</i> in model with coefficient constrained to 1
<b>Reporting</b>	
<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>
<p><i>indepvars</i> may contain factor variables; see <a href="#">[U] 11.4.3 Factor variables</a>.</p> <p><i>depvar</i> and <i>indepvars</i> may contain time-series operators; see <a href="#">[U] 11.4.4 Time-series varlists</a>.</p> <p><i>weights</i> are allowed; see <a href="#">[U] 11.1.6 weight</a>.</p> <p><code>bayer: tobit, level()</code> is equivalent to <code>bayer, clevel(): tobit</code>.</p> <p>For a detailed description of <i>options</i>, see <a href="#">Options</a> in <a href="#">[R] tobit</a>.</p>	
bayeropts	Description
<b>Priors</b>	
* <code>normalprior(#)</code>	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <code>igammaprior(# #)</code>	specify shape and scale of default inverse-gamma prior for variance; default is <code>igammaprior(0.01 0.01)</code>
<code>prior(priorspec)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation
<b>Simulation</b>	
<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(paramref)</code>	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(paramref[ , blockopts] )</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default



**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}` and variance `{sigma2}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] tobit](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] tobit](#) — Tobit regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: tpoisson** — Bayesian truncated Poisson regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: tpoisson** fits a Bayesian truncated Poisson regression to a positive count outcome whose values are all above the truncation point; see [\[BAYES\] bayes](#) and [\[R\] tpoisson](#) for details.

## Quick start

Bayesian truncated Poisson regression of *y* on *x1* and *x2*, using a lower truncation limit of 5 and using default normal priors for regression coefficients

```
bayes: tpoisson y x1 x2, ll(5)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): tpoisson y x1 x2, ll(5)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): tpoisson y x1 x2, ll(5)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): tpoisson y x1 x2, ll(5)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsz(20000) burnin(5000) dots(500): tpoisson y x1 x2, ll(5)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: tpoisson y x1 x2, ll(5) irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] tpoisson](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Truncated Poisson regression

# Syntax

```
bayes [ , bayesopts ] : tpoisson depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

## Model

<u>noconstant</u>	suppress constant term
ll(#   <i>varname</i> )	lower limit for truncation; default is ll(0)
ul(#   <i>varname</i> )	upper limit for truncation
<u>exposure</u> ( <i>varname</i> <sub>e</sub> )	include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> <sub>o</sub> )	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1

## Reporting

<u>irr</u>	report incidence-rate ratios
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: tpoisson, level() is equivalent to bayes, clevel(): tpoisson.

For a detailed description of *options*, see *Options* in [R] tpoisson.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] tpoisson](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Truncated Poisson regression* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] tpoisson](#) — Truncated Poisson regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

bayes: truncreg — Bayesian truncated regression

[Description](#)

[Remarks and examples](#)

[Quick start](#)

[Stored results](#)

[Menu](#)

[Methods and formulas](#)

[Syntax](#)

[Also see](#)

## Description

`bayes: truncreg` fits a Bayesian truncated linear regression to a continuous outcome; see [\[BAYES\] bayes](#) and [\[R\] truncreg](#) for details.

## Quick start

Bayesian truncated linear regression of `y` on `x1` and `x2`, using a lower truncation limit of 17 and using default normal priors for regression coefficients and default inverse-gamma prior for the variance

```
bayes: truncreg y x1 x2, ll(17)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): truncreg y x1 x2, ll(17)
```

Use a shape of 1 and a scale of 2 instead of values of 0.01 for the default inverse-gamma prior

```
bayes, igammaprior(1 2): truncreg y x1 x2, ll(17)
```

Use uniform priors for the slopes and a normal prior for the intercept

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): truncreg y x1 x2, ll(17)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
truncreg y x1 x2, ll(17)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsize(20000) burnin(5000) dots(500): ///  
truncreg y x1 x2, ll(17)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] truncreg](#).

## Menu

Statistics > Linear models and related > Bayesian regression > Truncated regression

# Syntax

```
bayer [ , bayesopts ] : truncreg depvar [indepvars] [if] [in] [weight] [ , options ]
```

<i>options</i>	Description
----------------	-------------

## Model

<code>noconstant</code>	suppress constant term
<code>ll(<i>varname</i>   #)</code>	left-truncation variable or limit
<code>ul(<i>varname</i>   #)</code>	right-truncation variable or limit
<code>offset(<i>varname</i>)</code>	include <i>varname</i> in model with coefficient constrained to 1

## Reporting

<code>display_options</code>	control spacing, line width, and base and empty cells
<code>level(#)</code>	set credible level; default is <code>level(95)</code>

*indepvars* may contain factor variables; see [U] 11.4.3 Factor variables.

*depvar* and *indepvars* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

*fweights* are allowed; see [U] 11.1.6 weight.

`bayer: truncreg, level()` is equivalent to `bayer, clevel(): truncreg`.

For a detailed description of *options*, see *Options* in [R] `truncreg`.

<i>bayesopts</i>	Description
------------------	-------------

## Priors

* <code>normalprior(#)</code>	specify standard deviation of default normal priors for regression coefficients; default is <code>normalprior(100)</code>
* <code>igammaprior(# #)</code>	specify shape and scale of default inverse-gamma prior for variance; default is <code>igammaprior(0.01 0.01)</code>
<code>prior(<i>priorspec</i>)</code>	prior for model parameters; this option may be repeated
<code>dryrun</code>	show model summary without estimation

## Simulation

<code>nchains(#)</code>	number of chains; default is to simulate one chain
<code>mcmcsize(#)</code>	MCMC sample size; default is <code>mcmcsize(10000)</code>
<code>burnin(#)</code>	burn-in period; default is <code>burnin(2500)</code>
<code>thinning(#)</code>	thinning interval; default is <code>thinning(1)</code>
<code>rseed(#)</code>	random-number seed
<code>exclude(<i>paramref</i>)</code>	specify model parameters to be excluded from the simulation results

## Blocking

* <code>blocksize(#)</code>	maximum block size; default is <code>blocksize(50)</code>
<code>block(<i>paramref</i> [ , <i>blockopts</i> ])</code>	specify a block of model parameters; this option may be repeated
<code>blocksummary</code>	display block summary
* <code>noblocking</code>	do not block parameters by default



**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar:indepvars}` and variance `{sigma2}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] truncreg](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] truncreg](#) — Truncated regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: zinb** — Bayesian zero-inflated negative binomial regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: zinb** fits a Bayesian zero-inflated negative binomial regression to a nonnegative count outcome with a high fraction of zeros; see [\[BAYES\]](#) **bayes** and [\[R\]](#) **zinb** for details.

## Quick start

Bayesian zero-inflated negative binomial regression of **y** on **x1** and **x2**, using **z** to model excess zeros and using default normal priors for regression coefficients and log-overdispersion parameter

```
bayes: zinb y x1 x2, inflate(z)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): zinb y x1 x2, inflate(z)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): zinb y x1 x2, inflate(z)
```

Save simulation results to **simdata.dta**, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): zinb y x1 x2, inflate(z)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): zinb y x1 x2, inflate(z)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: zinb y x1 x2, inflate(z) irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\]](#) **bayes** and [Quick start](#) in [\[R\]](#) **zinb**.

## Menu

Statistics > Count outcomes > Bayesian regression > Zero-inflated negative binomial regression

# Syntax

```
bayes [ , bayesopts ] : zinb depvar [ indepvars ] [ if ] [ in ] [ weight ] ,
  inflate(varlist [ , offset(varname) ] | _cons) [ options ]
```

<i>options</i>	Description
Model	
* <u>inflate</u> ( )	equation that determines whether the count is zero
<u>noconstant</u>	suppress constant term
<u>exposure</u> ( <i>varname</i> <sub>e</sub> )	include ln( <i>varname</i> <sub>e</sub> ) in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> <sub>o</sub> )	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1
<u>probit</u>	use probit model to characterize excess zeros; default is logit
Reporting	
<u>irr</u>	report incidence-rate ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

\* inflate(*varlist* [ , offset(*varname*) ] | \_cons) is required.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*fweights* are allowed; see [U] 11.1.6 weight.

bayes: zinb, level() is equivalent to bayes, clevel(): zinb.

For a detailed description of *options*, see *Options* in [R] zinb.

<i>bayesopts</i>	Description
Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients and log-overdispersion parameter; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

## Simulation

<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

## Blocking

* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[ , <i>replace</i> ])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[ , every(#) ]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{inflate: varlist}` for the inflation equation and log-overdispersion parameter `{lnalpha}`. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] zinb](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Zero-inflated negative binomial model* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] zinb](#) — Zero-inflated negative binomial regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: zioprobit** — Bayesian zero-inflated ordered probit regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

`bayes: zioprobit` fits a Bayesian zero-inflated ordered probit regression to an ordinal outcome with a high fraction of zeros; see [\[BAYES\]](#) [bayes](#) and [\[R\]](#) [zioprobit](#) for details.

## Quick start

Bayesian zero-inflated ordered probit regression of `y` on `x1` and `x2`, using `z` to model excess zeros and using default normal priors for regression coefficients and flat priors for cutpoints

```
bayes: zioprobit y x1 x2, inflate(z)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): zioprobit y x1 x2, inflate(z)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): zioprobit y x1 x2, inflate(z)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): ///  
zioprobit y x1 x2, inflate(z)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcsample(20000) burnin(5000) dots(500): ///  
zioprobit y x1 x2, inflate(z)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Also see [Quick start](#) in [\[BAYES\]](#) [bayes](#) and [Quick start](#) in [\[R\]](#) [zioprobit](#).

## Menu

Statistics > Ordinal outcomes > Bayesian regression > Zero-inflated ordered probit regression

# Syntax

```
bayan [ , bayesopts ] : zioprobit devar [ indepvars ] [ if ] [ in ] [ weight ] ,
    inflate(varlist [ , noconstant offset(varname) ] | _cons) [ options ]
```

<i>options</i>	Description
----------------	-------------

Model	
* <u>inflate</u> ( )	equation that determines excess zero values
<u>offset</u> ( <i>varname</i> )	include <i>varname</i> in model with coefficient constrained to 1

Reporting	
<i>display_options</i>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)

\* inflate(*varlist* [ , noconstant offset(*varname*) ] | \_cons) is required.

*indepvars* and *varlist* may contain factor variables; see [U] 11.4.3 Factor variables.

*fweights* are allowed; see [U] 11.1.6 weight.

bayan: zioprobit, level() is equivalent to bayan, clevel(): zioprobit.

For a detailed description of *options*, see *Options* in [R] zioprobit.

<i>bayesopts</i>	Description
------------------	-------------

Priors	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation

Simulation	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results

Blocking	
* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default



**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
<code>*noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] `bayesmh`.

*paramref* may contain factor variables; see [U] 11.4.3 **Factor variables**.

See [U] 20 **Estimation and postestimation commands** for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{inflate: varlist}` for the inflation equation and cutpoints `{cut1}`, `{cut2}`, and so on. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Flat priors, `flat`, are used by default for cutpoints.

For a detailed description of *bayesopts*, see *Options* in [BAYES] `bayes`.

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the bayes prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] zioprobit](#).

For a simple example of the bayes prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Zero-inflated negative binomial models* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the bayes prefix

[\[R\] zioprobit](#) — Zero-inflated ordered probit regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for bayesmh and the bayes prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Title

**bayes: zip** — Bayesian zero-inflated Poisson regression

[Description](#)[Remarks and examples](#)[Quick start](#)[Stored results](#)[Menu](#)[Methods and formulas](#)[Syntax](#)[Also see](#)

## Description

**bayes: zip** fits a Bayesian zero-inflated Poisson regression to a nonnegative count outcome with a high fraction of zeros; see [\[BAYES\] bayes](#) and [\[R\] zip](#) for details.

## Quick start

Bayesian zero-inflated Poisson regression of *y* on *x1* and *x2*, using *z* to model excess zeros and using default normal priors for regression coefficients

```
bayes: zip y x1 x2, inflate(z)
```

Use a standard deviation of 10 instead of 100 for the default normal priors

```
bayes, normalprior(10): zip y x1 x2, inflate(z)
```

Use uniform priors for the slopes and a normal prior for the intercept of the main regression

```
bayes, prior({y: x1 x2}, uniform(-10,10)) ///  
prior({y:_cons}, normal(0,10)): zip y x1 x2, inflate(z)
```

Save simulation results to `simdata.dta`, and use a random-number seed for reproducibility

```
bayes, saving(simdata) rseed(123): zip y x1 x2, inflate(z)
```

Specify 20,000 MCMC samples, set length of the burn-in period to 5,000, and request that a dot be displayed every 500 simulations

```
bayes, mcmcs(20000) burnin(5000) dots(500): zip y x1 x2, inflate(z)
```

In the above, request that the 90% HPD credible interval be displayed instead of the default 95% equal-tailed credible interval

```
bayes, clevel(90) hpd
```

Display incidence-rate ratios instead of coefficients

```
bayes: zip y x1 x2, inflate(z) irr
```

Display incidence-rate ratios on replay

```
bayes, irr
```

Also see [Quick start](#) in [\[BAYES\] bayes](#) and [Quick start](#) in [\[R\] zip](#).

## Menu

Statistics > Count outcomes > Bayesian regression > Zero-inflated Poisson regression

## Syntax

```
bayes [ , bayesopts ] : zip depvar [ indepvars ] [ if ] [ in ] [ weight ] ,
  inflate(varlist [ , offset(varname) ] | _cons) [ options ]
```

<i>options</i>	Description
<b>Model</b>	
* <u>inflate</u> ( )	equation that determines whether the count is zero
<u>noconstant</u>	suppress constant term
<u>exposure</u> ( <i>varname</i> <sub>e</sub> )	include $\ln(\text{varname}_e)$ in model with coefficient constrained to 1
<u>offset</u> ( <i>varname</i> <sub>o</sub> )	include <i>varname</i> <sub>o</sub> in model with coefficient constrained to 1
<u>probit</u>	use probit model to characterize excess zeros; default is logit
<b>Reporting</b>	
<u>irr</u>	report incidence-rate ratios
<u>display_options</u>	control spacing, line width, and base and empty cells
<u>level</u> (#)	set credible level; default is level(95)
* <u>inflate</u> ( <i>varlist</i> [ , <u>offset</u> ( <i>varname</i> ) ]   <u>_cons</u> ) is required. <i>indepvars</i> and <i>varlist</i> may contain factor variables; see [U] 11.4.3 Factor variables. <i>fweights</i> are allowed; see [U] 11.1.6 weight. bayes: zip, level() is equivalent to bayes, clevel(): zip. For a detailed description of <i>options</i> , see <i>Options</i> in [R] zip.	
<i>bayesopts</i>	Description
<b>Priors</b>	
* <u>normalprior</u> (#)	specify standard deviation of default normal priors for regression coefficients; default is normalprior(100)
<u>prior</u> ( <i>priorspec</i> )	prior for model parameters; this option may be repeated
<u>dryrun</u>	show model summary without estimation
<b>Simulation</b>	
<u>nchains</u> (#)	number of chains; default is to simulate one chain
<u>mcmcsize</u> (#)	MCMC sample size; default is mcmcsize(10000)
<u>burnin</u> (#)	burn-in period; default is burnin(2500)
<u>thinning</u> (#)	thinning interval; default is thinning(1)
<u>rseed</u> (#)	random-number seed
<u>exclude</u> ( <i>paramref</i> )	specify model parameters to be excluded from the simulation results
<b>Blocking</b>	
* <u>blocksize</u> (#)	maximum block size; default is blocksize(50)
<u>block</u> ( <i>paramref</i> [ , <i>blockopts</i> ])	specify a block of model parameters; this option may be repeated
<u>blocksummary</u>	display block summary
* <u>noblocking</u>	do not block parameters by default

**Initialization**

<code>initial(<i>initspec</i>)</code>	specify initial values for model parameters with a single chain
<code>init#(<i>initspec</i>)</code>	specify initial values for #th chain; requires <code>nchains()</code>
<code>initall(<i>initspec</i>)</code>	specify initial values for all chains; requires <code>nchains()</code>
<code>nomleinitial</code>	suppress the use of maximum likelihood estimates as starting values
<code>initrandom</code>	specify random initial values
<code>initsummary</code>	display initial values used for simulation
* <code>noisily</code>	display output from the estimation command during initialization

**Adaptation**

<code>adaptation(<i>adaptopts</i>)</code>	control the adaptive MCMC procedure
<code>scale(#)</code>	initial multiplier for scale factor; default is <code>scale(2.38)</code>
<code>covariance(<i>cov</i>)</code>	initial proposal covariance; default is the identity matrix

**Reporting**

<code>clevel(#)</code>	set credible interval level; default is <code>clevel(95)</code>
<code>hpd</code>	display HPD credible intervals instead of the default equal-tailed credible intervals
* <code>irr</code>	report incidence-rate ratios
<code>eform[ (<i>string</i>) ]</code>	report exponentiated coefficients and, optionally, label as <i>string</i>
<code>batch(#)</code>	specify length of block for batch-means calculations; default is <code>batch(0)</code>
<code>saving(<i>filename</i>[, <i>replace</i>])</code>	save simulation results to <i>filename.dta</i>
<code>nomodelsummary</code>	suppress model summary
<code>chainsdetail</code>	display detailed simulation summary for each chain
<code>[no]dots</code>	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is <code>nodots</code>
<code>dots#[, every(#)]</code>	display dots as simulation is performed
<code>[no]show(<i>paramref</i>)</code>	specify model parameters to be excluded from or included in the output
<code>notable</code>	suppress estimation table
<code>noheader</code>	suppress output header
<code>title(<i>string</i>)</code>	display <i>string</i> as title above the table of parameter estimates
<code>display_options</code>	control spacing, line width, and base and empty cells

**Advanced**

<code>search(<i>search_options</i>)</code>	control the search for feasible initial values
<code>corrlag(#)</code>	specify maximum autocorrelation lag; default varies
<code>corrtol(#)</code>	specify autocorrelation tolerance; default is <code>corrtol(0.01)</code>

---

\*Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.  
Options `prior()` and `block()` may be repeated.

*priorspec* and *paramref* are defined in [BAYES] [bayesmh](#).

*paramref* may contain factor variables; see [U] [11.4.3 Factor variables](#).

See [U] [20 Estimation and postestimation commands](#) for more capabilities of estimation commands.

Model parameters are regression coefficients `{depvar: indepvars}` for the main regression and `{inflate: varlist}` for the inflation equation. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

For a detailed description of *bayesopts*, see [Options](#) in [BAYES] [bayes](#).

## Remarks and examples

For a general introduction to Bayesian analysis, see [\[BAYES\] Intro](#). For a general introduction to Bayesian estimation using an adaptive Metropolis–Hastings algorithm, see [\[BAYES\] bayesmh](#). For remarks and examples specific to the `bayes` prefix, see [\[BAYES\] bayes](#). For details about the estimation command, see [\[R\] zip](#).

For a simple example of the `bayes` prefix, see *Introductory example* in [\[BAYES\] bayes](#). Also see *Zero-inflated negative binomial model* in [\[BAYES\] bayes](#).

## Stored results

See *Stored results* in [\[BAYES\] bayes](#).

## Methods and formulas

See *Methods and formulas* in [\[BAYES\] bayesmh](#).

## Also see

[\[BAYES\] bayes](#) — Bayesian regression models using the `bayes` prefix

[\[R\] zip](#) — Zero-inflated Poisson regression

[\[BAYES\] Bayesian postestimation](#) — Postestimation tools for `bayesmh` and the `bayes` prefix

[\[BAYES\] Bayesian estimation](#) — Bayesian estimation commands

[\[BAYES\] Bayesian commands](#) — Introduction to commands for Bayesian analysis

[\[BAYES\] Intro](#) — Introduction to Bayesian analysis

[\[BAYES\] Glossary](#)

# Glossary

**a posteriori.** In the context of Bayesian analysis, we use a posteriori to mean “after the sample is observed”. For example, a posteriori information is any information obtained after the data sample is observed. See [posterior distribution](#), [posterior](#).

**a priori.** In the context of Bayesian analysis, we use a priori to mean “before the sample is observed”. For example, a priori information is any information obtained before the data sample is observed. In a Bayesian model, a priori information about [model parameters](#) is specified by [prior distributions](#).

**acceptance rate.** In the context of the MH algorithm, acceptance rate is the fraction of the proposed samples that is accepted. The optimal acceptance rate depends on the properties of the [target distribution](#) and is not known in general. If the target distribution is normal, however, the optimal acceptance rate is known to be 0.44 for univariate distributions and 0.234 for multivariate distributions.

**adaptation.** In the context of the MH algorithm, adaptation refers to the process of tuning or adapting the proposal distribution to optimize the MCMC sampling. Typically, adaptation is performed periodically during the MCMC sampling. The `bayesmh` command performs adaptation every # of iterations as specified in option `adaptation(every(#))` for a maximum of `adaptation(maxiter())` iterations. In a continuous-adaptation regimes, the adaptation lasts during the entire process of the MCMC sampling. See [\[BAYES\] bayesmh](#).

**adaptation period.** Adaptation period includes all MH [adaptive iterations](#). It equals the length of the adaptation interval, as specified by `adaptation(every())`, times the maximum number of adaptations, `adaptation(maxiter())`.

**adaptive iteration.** In the adaptive MH algorithm, adaptive iterations are iterations during which [adaptation](#) is performed.

**Akaike information criterion, AIC.** Akaike information criterion (AIC) is an information-based model-selection criterion. It is given by the formula  $-2 \times \log \text{likelihood} + 2k$ , where  $k$  is the number of parameters. AIC favors simpler models by penalizing for the number of model parameters. It does not, however, account for the sample size. As a result, the AIC penalization diminishes as the sample size increases, as does its ability to guard against overparameterization.

**batch means.** Batch means are means obtained from batches of sample values of equal size. Batch means provide an alternative method for estimating MCMC standard errors ([MCSE](#)). The batch size is usually chosen to minimize the correlation between different batches of means.

**Bayes factor.** Bayes factor is given by the ratio of the [marginal likelihoods](#) of two models,  $M_1$  and  $M_2$ . It is a widely used criterion for Bayesian model comparison. Bayes factor is used in calculating the posterior odds ratio of model  $M_1$  versus  $M_2$ ,

$$\frac{P(M_1|\mathbf{y})}{P(M_2|\mathbf{y})} = \frac{P(\mathbf{y}|M_1) P(M_1)}{P(\mathbf{y}|M_2) P(M_2)}$$

where  $P(M_i|\mathbf{y})$  is a posterior probability of model  $M_i$ , and  $P(M_i)$  is a prior probability of model  $M_i$ . When the two models are equally likely, that is, when  $P(M_1) = P(M_2)$ , the Bayes factor equals the posterior odds ratio of the two models.

**Bayes’s theorem.** The Bayes’s theorem is a formal method for relating conditional probability statements. For two (random) events  $X$  and  $Y$ , the Bayes’s theorem states that

$$P(X|Y) \propto P(Y|X)P(X)$$

that is, the probability of  $X$  conditional on  $Y$  is proportional to the probability of  $X$  and the probability of  $Y$  conditional on  $X$ . In Bayesian analysis, the Bayes's theorem is used for combining prior information about model parameters and evidence from the observed data to form the [posterior distribution](#).

**Bayesian analysis.** Bayesian analysis is a statistical methodology that considers model parameters to be random quantities and estimates their [posterior distribution](#) by combining prior knowledge about parameters with the evidence from the observed data sample. Prior knowledge about parameters is described by [prior distributions](#) and evidence from the observed data is incorporated through a likelihood model. Using the [Bayes's theorem](#), the prior distribution and the likelihood model are combined to form the posterior distribution of model parameters. The posterior distribution is then used for parameter inference, hypothesis testing, and prediction.

**Bayesian estimation.** Bayesian estimation consists of fitting Bayesian models and estimating their parameters based on the resulting posterior distribution. Bayesian estimation in Stata can be done using the convenient [bayes](#) prefix or the more general [bayesmh](#) command. See [\[BAYES\] Bayesian estimation](#) for details.

**Bayesian estimation results.** Estimation results obtained after the [bayes](#) prefix or the [bayesmh](#) command.

**Bayesian hypothesis testing.** Bayesian hypothesis testing computes probabilities of hypotheses conditional on the observed data. In contrast to the frequentist hypothesis testing, the Bayesian hypothesis testing computes the actual probability of a hypothesis  $H$  by using the Bayes's theorem,

$$P(H|\mathbf{y}) \propto P(\mathbf{y}|H)P(H)$$

where  $\mathbf{y}$  is the observed data,  $P(\mathbf{y}|H)$  is the marginal likelihood of  $\mathbf{y}$  given  $H$ , and  $P(H)$  is the prior probability of  $H$ . Two different hypotheses,  $H_1$  and  $H_2$ , can be compared by simply comparing  $P(H_1|\mathbf{y})$  to  $P(H_2|\mathbf{y})$ .

**Bayesian information criterion, BIC.** The Bayesian information criterion (BIC), also known as Schwarz criterion, is an information based criterion used for model selection in classical statistics. It is given by the formula  $-0.5 \times \log \text{likelihood} + k \times \ln n$ , where  $k$  is the number of parameters and  $n$  is the sample size. BIC favors simpler, in terms of complexity, models and it is more conservative than [AIC](#).

**Bayesian model checking.** In Bayesian statistics, model checking refers to testing likelihood and prior model adequacy in the context of a research problem and observed data. A simple sanity check may include verifying that posterior inference produces results that are reasonable in the context of the problem. More substantive checks may include analysis of the sensitivity of Bayesian inference to changes in likelihood and prior distribution specifications. See [posterior predictive checking](#).

**Bayesian predictions.** Bayesian predictions are samples from the [posterior predictive distribution](#) of outcome variables and functions of these samples and, optionally, model parameters. Examples of Bayesian predictions include [replicated data](#), out-of-sample predictions, and test statistics of [simulated outcomes](#).

**blocking.** In the context of the MH algorithm, blocking refers to the process of separating model parameters into different subsets or blocks to be sampled independently of each other. MH algorithm generates proposals and applies the acceptance–rejection rule sequentially for each block. It is recommended that correlated parameters are kept in one block. Separating less-correlated or independent model parameters in different blocks may improve the [mixing](#) of the MH algorithm.

**burn-in period.** The burn-in period is the number of iterations it takes for an [MCMC](#) sequence to reach stationarity.

**central posterior interval.** See [equal-tailed credible interval](#).



**conditional conjugacy.** See *semiconjugate prior*.

**conjugate prior.** A prior distribution is conjugate for a family of likelihood distributions if the prior and posterior distributions belong to the same family of distributions. For example, the gamma distribution is a conjugate prior for the Poisson likelihood. Conjugacy may provide an efficient way of sampling from posterior distributions and is used in *Gibbs sampling*.

**continuous parameters.** Continuous parameters are parameters with continuous prior distributions.

**credible interval.** In Bayesian analysis, the credible interval of a scalar model parameter is an interval from the domain of the marginal posterior distribution of that parameter. Two types of credible intervals are typically used in practice: *equal-tailed credible intervals* and *HPD credible intervals*.

**credible level.** The credible level is a probability level between 0% and 100% used for calculating *credible intervals* in Bayesian analysis. For example, a 95% credible interval for a scalar parameter is an interval the parameter belongs to with the probability of 95%.

**cusum plot, CUSUM plot.** The cusum (CUSUM) plot of an MCMC sample is a plot of cumulative sums of the differences between sample values and their overall mean against the iteration number. Cusum plots are useful graphical summaries for detecting early drifts in MCMC samples.

**deviance information criterion, DIC.** The deviance information criterion (DIC) is an information based criterion used for Bayesian model selection. It is an analog of AIC and is given by the formula  $D(\bar{\theta}) + 2 \times p_D$ , where  $D(\bar{\theta})$  is the deviance at the sample mean and  $p_D$  is the effective complexity, a quantity equivalent to the number of parameters in the model. Models with smaller DIC are preferred.

**diminishing adaptation.** Diminishing adaptation of the adaptive algorithm is the type of adaptation in which the amount of adaptation decreases with the size of the MCMC chain.

**discrete parameters.** Discrete parameters are parameters with discrete prior distributions.

**effective sample size, ESS.** Effective sample size (ESS) is the MCMC sample size  $T$  adjusted for the autocorrelation in the sample. It represents the number of independent observations in an MCMC sample. ESS is used instead of  $T$  in calculating MCSE. Small ESS relative to  $T$  indicates high autocorrelation and consequently poor *mixing* of the chain.

**efficiency.** In the context of MCMC, efficiency is a term used for assessing the mixing quality of an MCMC procedure. Efficient MCMC algorithms are able to explore posterior domains in less time (using fewer iterations). Efficiency is typically quantified by the sample autocorrelation and effective sample size. An MCMC procedure that generates samples with low autocorrelation and consequently high ESS is more efficient.

**equal-tailed credible interval.** An equal-tailed credible interval is a credible interval defined in such a way that both tails of the marginal posterior distribution have the same probability. A  $\{100 \times (1 - \alpha)\}\%$  equal-tailed credible interval is defined by the  $\alpha/2$ th and  $\{(1 - \alpha)/2\}$ th quantiles of the marginal posterior distribution.

**feasible initial value.** An initial-value vector is feasible if it corresponds to a state with a positive posterior probability.

**fixed effects.** See *fixed-effects parameters*.

**fixed-effects parameters.** In the Bayesian context, the term “fixed effects” or “fixed-effects parameters” is a misnomer, because all model parameters are inherently random. We use this term in the context of Bayesian multilevel models to refer to regression model parameters and to distinguish them from the *random-effects parameters*. You can think of fixed-effects parameters as parameters modeling population averaged or marginal relationship of the response and the variables of interest.

**frequentist analysis.** Frequentist analysis is a form of statistical analysis where model parameters are considered to be unknown but fixed constants and the observed data are viewed as a repeatable random sample. Inference is based on the sampling distribution of the data.

**full conditionals.** A full conditional is the probability distribution of a random variate conditioned on all other random variates in a joint probability model. Full conditional distributions are used in [Gibbs sampling](#).

**full Gibbs sampling.** See [Gibbs sampling](#), [Gibbs sampler](#).

**Gelman–Rubin convergence diagnostic, Gelman–Rubin convergence statistic.** Gelman–Rubin convergence diagnostic assesses MCMC convergence by analyzing differences between multiple Markov chains. The convergence is assessed by comparing the estimated between-chains and within-chain variances for each model parameter. Large differences between these variances indicate nonconvergence. See [\[BAYES\] bayesstats grubin](#).

**Gibbs sampling, Gibbs sampler.** Gibbs sampling is an MCMC method, according to which each random variable from a joint probability model is sampled according to its [full conditional distribution](#).

**highest posterior density credible interval, HPD credible interval.** The highest posterior density (HPD) credible interval is a type of a credible interval with the highest marginal posterior density. An HPD interval has the shortest width among all other credible intervals. For some multimodal marginal distributions, HPD may not exist. See [highest posterior density region](#), [HPD region](#).

**highest posterior density region, HPD region.** The highest posterior density (HPD) region for model parameters has the highest marginal posterior probability among all domain regions. Unlike an [HPD credible interval](#), an HPD region always exist.

**hybrid MH sampling, hybrid MH sampler.** A hybrid MH sampler is an MCMC method in which some blocks of parameters are updated using the MH algorithms and other blocks are updated using Gibbs sampling.

**hyperparameter.** In Bayesian analysis, hyperparameter is a parameter of a prior distribution, in contrast to a [model parameter](#).

**hyperprior.** In Bayesian analysis, hyperprior is a prior distribution of hyperparameters. See [hyperparameter](#).

**improper prior.** A prior is said to be improper if it does not integrate to a finite number. Uniform distributions over unbounded intervals are improper. Improper priors may still yield proper posterior distributions. When using improper priors, however, one has to make sure that the resulting posterior distribution is proper for Bayesian inference to be invalid.

**independent a posteriori.** Parameters are considered independent a posteriori if their marginal posterior distributions are independent; that is, their joint posterior distribution is the product of their individual marginal posterior distributions.

**independent a priori.** Parameters are considered independent a priori if their prior distributions are independent; that is, their joint prior distribution is the product of their individual marginal prior distributions.

**informative prior.** An informative prior is a prior distribution that has substantial influence on the posterior distribution.

**in-sample predictions.** See [replicated outcome](#).

**interval hypothesis testing.** Interval hypothesis testing performs [interval hypothesis tests](#) for model parameters and functions of model parameters.

- interval test.** In Bayesian analysis, an interval test applied to a scalar model parameter calculates the marginal posterior probability for the parameter to belong to the specified interval.
- Jeffreys prior.** The Jeffreys prior of a vector of model parameters  $\theta$  is proportional to the square root of the determinant of its Fisher information matrix  $I(\theta)$ . Jeffreys priors are locally uniform and, by definition, agree with the likelihood function. Jeffreys priors are considered noninformative priors that have minimal impact on the posterior distribution.
- marginal distribution.** In Bayesian context, a distribution of the data after integrating out parameters from the joint distribution of the parameters and the data.
- marginal likelihood.** In the context of Bayesian model comparison, a marginalized over model parameters  $\theta$  likelihood of data  $y$  for a given model  $M$ ,  $P(y|M) = m(y) = \int P(y|\theta, M)P(\theta|M)d\theta$ . Also see [Bayes factor](#).
- marginal posterior distribution.** In Bayesian context, a marginal posterior distribution is a distribution resulting from integrating out all but one parameter from the joint posterior distribution.
- Markov chain.** Markov chain is a random process that generates sequences of random vectors (or states) and satisfies the Markov property: the next state depends only on the current state and not on any of the previous states. [MCMC](#) is the most common methodology for simulating Markov chains.
- matrix model parameter.** A matrix model parameter is any [model parameter](#) that is a matrix. Matrix elements, however, are viewed as [scalar model parameters](#).
- Matrix model parameters are defined and referred to within the `bayesmh` command as `{param,matrix}` or `{eqname:param,matrix}` with the equation name `eqname`. For example, `{Sigma,matrix}` and `{Scale:Omega,matrix}` are matrix model parameters. Individual matrix elements cannot be referred to within the `bayesmh` command, but they can be referred within postestimation commands accepting parameters. For example, to refer to the individual elements of the defined above, say,  $2 \times 2$  matrices, use `{Sigma_1_1}`, `{Sigma_2_1}`, `{Sigma_1_2}`, `{Sigma_2_2}` and `{Scale:Omega_1_1}`, `{Scale:Omega_2_1}`, `{Scale:Omega_1_2}`, `{Scale:Omega_2_2}`, respectively. See [\[BAYES\] bayesmh](#).
- matrix parameter.** See [matrix model parameter](#).
- MCMC, Markov chain Monte Carlo.** MCMC is a class of simulation-based methods for generating samples from probability distributions. Any MCMC algorithm simulates a [Markov chain](#) with a target distribution as its stationary or equilibrium distribution. The precision of MCMC algorithms increases with the number of iterations. The lack of a stopping rule and convergence rule, however, makes it difficult to determine for how long to run MCMC. The time needed to converge to the target distribution within a prespecified error is referred to as mixing time. Better MCMC algorithms have faster mixing times. Some of the popular MCMC algorithms are random-walk Metropolis, [Metropolis–Hastings](#), and [Gibbs sampling](#).
- MCMC replicates.** An [MCMC sample](#) of [simulated outcomes](#).
- MCMC sample.** An MCMC sample is obtained from [MCMC sampling](#). An MCMC sample approximates a target distribution and is used for summarizing this distribution.
- MCMC sample size.** MCMC sample size is the size of the [MCMC sample](#). It is specified in `bayesmh`'s option `mcmcsize()`; see [\[BAYES\] bayesmh](#).
- MCMC sampling, MCMC sampler.** MCMC sampling is an MCMC algorithm that generates samples from a target probability distribution.

**MCMC standard error, MCSE** MCSE is the standard error of the posterior mean estimate. It is defined as the standard deviation divided by the square root of [ESS](#). MCSEs are analogs of standard errors in frequentist statistics and measure the accuracy of the simulated MCMC sample.

**Metropolis–Hastings (MH) sampling, MH sampler.** A Metropolis–Hastings (MH) sampler is an MCMC method for simulating probability distributions. According to this method, at each step of the Markov chain, a new proposal state is generated from the current state according to a prespecified proposal distribution. Based on the current and new state, an acceptance probability is calculated and then used to accept or reject the proposed state. Important characteristics of MH sampling is the [acceptance rate](#) and [mixing](#) time. The MH algorithm is very general and can be applied to an arbitrary target distribution. However, its efficiency is limited, in terms of mixing time, and decreases as the dimension of the target distribution increases. [Gibbs sampling](#), when available, can provide much more efficient sampling than MH sampling.

**mixing of Markov chain.** Mixing refers to the rate at which a Markov chain traverses the parameter space. It is a property of the Markov chain that is different from convergence. Poor mixing indicates a slow rate at which the chain explores the stationary distribution and will require more iterations to provide inference at a given precision. Poor (slow) mixing is typically a result of high correlation between model parameters or of weakly-defined model specifications.

**model hypothesis testing.** Model hypothesis testing tests hypotheses about models by computing [model posterior probabilities](#).

**model parameter.** A model parameter refers to any (random) parameter in a Bayesian model. Model parameters can be [scalars](#) or [matrices](#). Examples of model parameters as defined in `bayesmh` are `{mu}`, `{scale:s}`, `{Sigma,matrix}`, and `{Scale:Omega,matrix}`. See [\[BAYES\] bayesmh](#) and, specifically, [Declaring model parameters](#) and [Referring to model parameters](#) in that entry. Also see [Different ways of specifying model parameters](#) in [\[BAYES\] Bayesian postestimation](#).

**model posterior probability.** Model posterior probability is probability of a model  $M$  computed conditional on the observed data  $\mathbf{y}$ ,

$$P(M|\mathbf{y}) = P(M)P(\mathbf{y}|M) = P(M)m(\mathbf{y})$$

where  $P(M)$  is the prior probability of a model  $M$  and  $m(\mathbf{y})$  is the [marginal likelihood](#) under model  $M$ .

**noninformative prior.** A noninformative prior is a prior with negligible influence on the posterior distribution. See, for example, [Jeffreys prior](#).

**objective prior.** See [noninformative prior](#).

**one-at-a-time MCMC sampling.** A one-at-a-time MCMC sample is an MCMC sampling procedure in which random variables are sampled individually, one at a time. For example, in [Gibbs sampling](#), individual variates are sampled one at a time, conditionally on the most recent values of the rest of the variates.

**out-of-sample predictions.** Predictions of future observations; see [simulated outcome](#).

**overdispersed initial value.** An overdispersed initial value is obtained from a distribution that is overdispersed or has larger variability relative to the true marginal posterior distribution. Overdispersed initial values are used with multiple Markov chains for diagnosing MCMC convergence. Also see [Specifying initial values](#) in [\[BAYES\] bayesmh](#).

**posterior distribution, posterior.** A posterior distribution is a probability distribution of model parameters conditional on observed data. The posterior distribution is determined by the likelihood of the parameters and their prior distribution. For a parameter vector  $\theta$  and data  $y$ , the posterior distribution is given by

$$P(\theta|y) = \frac{P(\theta)P(y|\theta)}{P(y)}$$

where  $P(\theta)$  is the prior distribution,  $P(y|\theta)$  is the model likelihood, and  $P(y)$  is the marginal distribution for  $y$ . Bayesian inference is based on a posterior distribution.

**posterior independence.** See *independent a posteriori*.

**posterior interval.** See *credible interval*.

**posterior odds.** Posterior odds for  $\theta_1$  compared with  $\theta_2$  is the ratio of posterior density evaluated at  $\theta_1$  and  $\theta_2$  under a given model,

$$\frac{p(\theta_1|y)}{p(\theta_2|y)} = \frac{p(\theta_1) p(y|\theta_1)}{p(\theta_2) p(y|\theta_2)}$$

In other words, posterior odds are prior odds times the likelihood ratio.

**posterior predictive checking.** Posterior predictive checking is a methodology for assessing goodness of fit of a Bayesian model using replicated data simulated from the posterior predictive distribution of the model. For example, graphical diagnostics of the replicated residuals may be used to check the distributional assumptions of the model error terms. A more formal and systematic approach uses test quantities and test statistics to measure discrepancies between replicated data and observed data. Test statistics such as a mean, minimum, and maximum can be used to compare different aspects of the observed data distribution with those of the replicated-data distribution. Posterior predictive  $p$ -values, also called Bayesian  $p$ -values, computed for test quantities and test statistics are used to quantify the discrepancy between the observed and replicated data. Also see *Bayesian model checking*.

**posterior predictive distribution.** Posterior predictive distribution is a distribution of unobserved (future) data conditional on observed data. Posterior predictive distribution is derived by marginalizing the likelihood function with respect to the posterior distribution of model parameters.

**posterior predictive  $p$ -value.** Posterior predictive  $p$ -value, also called a Bayesian  $p$ -value, is the probability that a test quantity (or statistic) computed for the replicated data is greater or equal to the test quantity computed for the observed data. Posterior predictive  $p$ -values are used in posterior predictive checking.  $p$ -values less than 0.05 or greater than 0.95 typically indicate model misfit (Gelman et al. 2014).

**predictive distribution.** See *prior predictive distribution* and *posterior predictive distribution*.

**predictive inference.** In Bayesian statistics, predictive inference is inference about unobserved (future) data conditionally on past data and prior knowledge of model parameters. Predictive inference is based on prior predictive or posterior predictive distribution of model parameters.

**predictive outcome.** Predictive outcome  $\tilde{y}$  is a value or a set of values simulated from a posterior predictive distribution  $p(\tilde{y}|y)$  of a Bayesian model (Gelman et al. 2014). In contrast with replicated outcome, predictive outcomes may use the values of independent variables that are different from those used to fit the model. Also see *simulated outcome*.

**prior distribution, prior.** In Bayesian statistics, prior distributions are probability distributions of model parameters formed based on some a priori knowledge about parameters. Prior distributions are independent of the observed data.

**prior independence.** See *independent a priori*.

**prior odds.** Prior odds for  $\theta_1$  compared with  $\theta_2$  is the ratio of prior density evaluated at  $\theta_1$  and  $\theta_2$  under a given model,  $p(\theta_1)/p(\theta_2)$ . Also see *posterior odds*.

**prior predictive distribution.** Prior predictive distribution is a distribution of unobserved (future) data derived by marginalizing the likelihood function with respect to the *prior distribution* of model parameters. Also see *marginal distribution*.

**proposal distribution.** In the context of the MH algorithm, a proposal distribution is used for defining the transition steps of the Markov chain. In the standard random-walk Metropolis algorithm, the proposal distribution is a multivariate normal distribution with zero mean and adaptable covariance matrix.

**pseudoconvergence.** A Markov chain may appear to converge when in fact it did not. We refer to this phenomenon as pseudoconvergence. Pseudoconvergence is typically caused by multimodality of the stationary distribution, in which case the chain may fail to traverse the weakly connected regions of the distribution space. A common way to detect pseudoconvergence is to run multiple chains using different starting values and to verify that all of the chain converge to the same target distribution.

**random effects.** See *random-effects parameters*.

**random-effects linear form.** A linear form representing a random-effects variable that can be used in substitutable expressions.

**random-effects parameters.** In the context of Bayesian multilevel models, random-effects parameters are parameters associated with a *random-effects variable*. Random-effects parameters are assumed to be conditionally independent across levels of the random-effects variable given all other model parameters. Often, random-effects parameters are assumed to be normally distributed with a zero mean and an unknown variance–covariance matrix.

**random-effects variable.** A variable identifying the group structure for the random effects at a specific level of hierarchy.

**reference prior.** See *noninformative prior*.

**replicated data.** Replicated data,  $\mathbf{y}^{\text{rep}}$ , are data that could be observed if the experiment that produced the observed data,  $\mathbf{y}^{\text{obs}}$ , were replicated using the same model and the same values of independent variables that generated  $\mathbf{y}^{\text{obs}}$ . See Gelman et al. (2014, 145), [BAYES] *bayespredict*, and [BAYES] *bayesstats pvalues*.

**replicated outcome.** Replicated outcome is a special case of a *simulated outcome* that is generated using the same values of independent variables as those used to fit the model. Also see *replicated data*.

**scalar model parameter.** A scalar model parameter is any *model parameter* that is a scalar. For example, {mean} and {hape:alpha} are scalar parameters, as declared by the *bayesmh* command. Elements of *matrix model parameters* are viewed as scalar model parameters. For example, for a  $2 \times 2$  matrix parameter {Sigma,matrix}, individual elements {Sigma\_1\_1}, {Sigma\_2\_1}, {Sigma\_1\_2}, and {Sigma\_2\_2} are scalar parameters. If a matrix parameter contains a label, the label should be included in the specification of individual elements as well. See [BAYES] *bayesmh*.

**scalar parameter.** See *scalar model parameter*.

**semiconjugate prior.** A prior distribution is semiconjugate for a family of likelihood distributions if the prior and (full) conditional posterior distributions belong to the same family of distributions. For semiconjugacy to hold, parameters must typically be independent a priori; that is, their joint prior distribution must be the product of the individual marginal prior distributions. For example,

the normal prior distribution for a mean parameter of a normal data distribution with an unknown variance (which is assumed to be independent of the mean a priori) is a semiconjugate prior. Semiconjugacy may provide an efficient way of sampling from posterior distributions and is used in [Gibbs sampling](#).

**simulated outcome.** In Bayesian predictive inference, simulated outcomes are samples from the [posterior predictive distribution](#). In the context of [bayespredict](#), we define a simulated outcome as a  $T \times n$  matrix of new outcome values simulated from the posterior predictive distribution,  $p(\tilde{\mathbf{y}}|\mathbf{y})$ , for a particular outcome variable  $\mathbf{y}$ , where  $T$  is the MCMC sample size and  $n$  is the number of observations.

**stationary distribution.** Stationary distribution of a stochastic process is a joint distribution that does not change over time. In the context of MCMC, stationary distribution is the target probability distribution to which the Markov chain converges. When MCMC is used for simulating a Bayesian model, the stationary distribution is the target joint posterior distribution of model parameters.

**subjective prior.** See [informative prior](#).

**subsampling the chain.** See [thinning](#).

**sufficient statistic.** Sufficient statistic for a parameter of a parametric likelihood model is any function of the sample that contains all the information about the model parameter.

**test quantity.** In Bayesian predictive inference, test quantity is any function of a [simulated outcome](#),  $\mathbf{y}^{\text{sim}}$ , and model parameters  $\boldsymbol{\theta}$ . It is estimated by sampling from the joint posterior distribution  $p(\mathbf{y}^{\text{sim}}, \boldsymbol{\theta})$ . A test quantity that depends only on  $\mathbf{y}^{\text{sim}}$  is called a test statistic. Test quantities are used in [posterior predictive checking](#) to assess model fit.

**test statistic.** A special case of a [test quantity](#) that depends only on the data.

**thinning.** Thinning is a way of reducing autocorrelation in the MCMC sample by subsampling the MCMC chain every prespecified number of iterations determined by the thinning interval. For example, the thinning interval of 1 corresponds to using the entire MCMC sample; the thinning interval of 2 corresponds to using every other sample value; and the thinning interval of 3 corresponds to using values from iterations 1, 4, 7, 10, and so on. Thinning should be applied with caution when used to reduce autocorrelation because it may not always be the most appropriate way of improving the precision of estimates.

**vague prior.** See [noninformative prior](#).

**valid initial state.** See [feasible initial value](#).

**vanishing adaptation.** See [diminishing adaptation](#).

**Zellner's  $g$ -prior.** Zellner's  $g$ -prior is a form of a weakly informative prior for the regression coefficients in a linear model. It accounts for the correlation between the predictor variables and controls the impact of the prior of the regression coefficients on the posterior with parameter  $g$ . For example,  $g = 1$  means that prior weight is 50% and  $g \rightarrow \infty$  means diffuse prior.

## Reference

- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. 2014. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: Chapman & Hall/CRC.

# Subject and author index

See the [combined subject index](#) and the [combined author index](#) in the *Glossary and Index*.