STATA
STRUCTURAL EQUATION MODELING
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 TEX

ISBN-10: 1-59718-295-8
ISBN-13: 978-1-59718-295-9

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

Acknowledgments 1

Intro 1 . . . . . Introduction 2
Intro 2 . . . . . Learning the language: Path diagrams and command language 7
Intro 3 . . . . . Learning the language: Factor-variable notation (gsem only) 38
Intro 4 . . . . . Substantive concepts 45
Intro 5 . . . . . Tour of models 64
Intro 6 . . . . . Comparing groups 87
Intro 7 . . . . . Postestimation tests and predictions 98
Intro 8 . . . . . Robust and clustered standard errors 107
Intro 9 . . . . . Standard errors, the full story 109
Intro 10 . . . . . Fitting models with survey data 113
Intro 11 . . . . . Fitting models with summary statistics data (sem only) 115
Intro 12 . . . . . Convergence problems and how to solve them 123

Builder . . . . . SEM Builder 133
Builder, generalized . . . . . SEM Builder for generalized models 136

estat eform . . . . . Display exponentiated coefficients 139
estat eqgof . . . . . Equation-level goodness-of-fit statistics 141
estat eqtest . . . . . Equation-level tests that all coefficients are zero 143
estat framework . . . . . Display estimation results in modeling framework 145
estat ggof . . . . . Group-level goodness-of-fit statistics 147
estat ginvariant . . . . . Tests for invariance of parameters across groups 149
estat gof . . . . . Goodness-of-fit statistics 151
estat lcgof . . . . . Latent class goodness-of-fit statistics 154
estat lcmean . . . . . Latent class marginal means 156
estat lcprob . . . . . Latent class marginal probabilities 158
estat mindices . . . . . Modification indices 160
estat residuals . . . . . Display mean and covariance residuals 162
estat scoretests . . . . . Score tests 165
estat sd . . . . . Display variance components as standard deviations and correlations 167
estat stable . . . . . Check stability of nonrecursive system 169
estat stdize . . . . . Test standardized parameters 171
estat summarize . . . . . Report summary statistics for estimation sample 173
estat teffects . . . . . Decomposition of effects into total, direct, and indirect 174

Example 1 . . . . . Single-factor measurement model 177
Example 2 . . . . . Creating a dataset from published covariances 185
Example 3 . . . . . Two-factor measurement model 190
Example 4 . . . . . Goodness-of-fit statistics 198
Example 5 . . . . . Modification indices 201
Example 6 . . . . . Linear regression 204
Example 7 . . . . . Nonrecursive structural model 208
Example 8 . . . . . Testing that coefficients are equal, and constraining them 216
Example 9 . . . . . Structural model with measurement component 220
Example 10 . . . . . MIMIC model 229
Example 11 . . . . . estat framework 236

Example 12 . . . . . Seemingly unrelated regression 239
Example 13 . . . . . Equation-level Wald test 243
Example 14 . . . . . Predicted values 244
Example 15 . . . . . Higher-order CFA 246
Example 16 . . . . . Correlation 253
Example 17 . . . . . Correlated uniqueness model 258
Example 18 . . . . . Latent growth model 265
Example 19 . . . . . Creating multiple-group summary statistics data 272
Example 20 . . . . . Two-factor measurement model by group 277
Example 21 . . . . . Group-level goodness of fit 287
Example 22 . . . . . Testing parameter equality across groups 288
Example 23 . . . . . Specifying parameter constraints across groups 291
Example 24 . . . . . Reliability 297
Example 25 . . . . . Creating summary statistics data from raw data 301
Example 26 . . . . . Fitting a model with data missing at random 309
Example 27g . . . . . Single-factor measurement model (generalized response) 313
Example 28g . . . . . One-parameter logistic IRT (Rasch) model 320
Example 29g . . . . . Two-parameter logistic IRT model 329
Example 30g . . . . . Two-level measurement model (multilevel, generalized response) 339
Example 31g . . . . . Two-factor measurement model (generalized response) 349
Example 32g . . . . . Full structural equation model (generalized response) 356
Example 33g . . . . . Logistic regression 363
Example 34g . . . . . Combined models (generalized responses) 368
Example 35g . . . . . Ordered probit and ordered logit 374
Example 36g . . . . . MIMIC model (generalized response) 382
Example 37g . . . . . Multinomial logistic regression 387
Example 38g . . . . . Random-intercept and random-slope models (multilevel) 396
Example 39g . . . . . Three-level model (multilevel, generalized response) 412
Example 40g . . . . . Crossed models (multilevel) 420
Example 41g . . . . . Two-level multinomial logistic regression (multilevel) 425
Example 42g . . . . . One- and two-level mediation models (multilevel) 435
Example 43g . . . . . Tobit regression 444
Example 44g . . . . . Interval regression 447
Example 45g . . . . . Heckman selection model 451
Example 46g . . . . . Endogenous treatment-effects model 460
Example 47g . . . . . Exponential survival model 467
Example 48g . . . . . Loglogistic survival model with censored and truncated data 471
Example 49g . . . . . Multiple-group Weibull survival model 477
Example 50g . . . . . Latent class model 483
Example 51g . . . . . Latent class goodness-of-fit statistics 489
Example 52g . . . . . Latent profile model 492
Example 53g . . . . . Finite mixture Poisson regression 501
Example 54g . . . . . Finite mixture Poisson regression, multiple responses 507

gsem . . . . . Generalized structural equation model estimation command 511
gsem estimation options . . . . . Options affecting estimation 516
gsem family-and-link options . . . . . Family-and-link options 521
gsem group options . . . . . Fitting models on different groups 527
gsem lclass options . . . . . Fitting models with latent classes 529
gsem model description options . . . . . Model description options 531
gsem path notation extensions . . . . . Command syntax for path diagrams 533

gsem postestimation . . . . . Postestimation tools for gsem 544
gsem reporting options . . . . . Options affecting reporting of results 547

lincom . . . . . Linear combinations of parameters 549
lrtest . . . . . Likelihood-ratio test of linear hypothesis 550

Methods and formulas for gsem . . . . . Methods and formulas for gsem 552
Methods and formulas for sem . . . . . Methods and formulas for sem 572

nlcom . . . . . Nonlinear combinations of parameters 584

predict after gsem . . . . . Generalized linear predictions, etc. 586
predict after sem . . . . . Factor scores, linear predictions, etc. 591

sem . . . . . Structural equation model estimation command 593
sem and gsem option constraints( ) . . . . . Specifying constraints 598
sem and gsem option covstructure( ) . . . . . Specifying covariance restrictions 600
sem and gsem option from( ) . . . . . Specifying starting values 603
sem and gsem option reliability( ) . . . . . Fraction of variance not due to measurement error 606
sem and gsem path notation . . . . . Command syntax for path diagrams 609
sem and gsem syntax options . . . . . Options affecting interpretation of syntax 615
sem estimation options . . . . . Options affecting estimation 616
sem group options . . . . . Fitting models on different groups 618
sem model description options . . . . . Model description options 620
sem option method( ) . . . . . Specifying method and calculation of VCE 622
sem option noxconditional . . . . . Computing means, etc., of observed exogenous variables 624
sem option select( ) . . . . . Using sem with summary statistics data 627
sem path notation extensions . . . . . Command syntax for path diagrams 629
sem postestimation . . . . . Postestimation tools for sem 633
sem reporting options . . . . . Options affecting reporting of results 635
sem ssd options . . . . . Options for use with summary statistics data 637
ssd . . . . . Making summary statistics data (sem only) 639

test . . . . . Wald test of linear hypotheses 643
testnl . . . . . Wald test of nonlinear hypotheses 645

Glossary 647

Subject and author index 659

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

Acknowledgments

`sem` and `gsem` were developed by StataCorp.

Neither command would exist without the help of two people outside of StataCorp. We must thank these two people profusely. They are

Jeroen Weesie, Department of Sociology at Utrecht University, The Netherlands

Sophia Rabe-Hesketh, University of California, Berkeley

Jeroen Weesie is responsible for the existence of the SEM project at StataCorp. While spending his sabbatical with us, Jeroen expressed—repeatedly—the importance of SEM, and that enthusiasm for SEM was disregarded—repeatedly. Not until after his sabbatical did StataCorp see the light. At that point, we had him back, and back, and back, so that he could inspire us, guide us, tell us what we had right, and, often, tell us what we had wrong.

Jeroen helped us with the math, the syntax, and system design, and, when we were too thick-headed, he even wrote code. By the date of first shipment, all code had been rewritten by us, but design and syntax for SEM still now and forever will show Jeroen's influence.

Thank you, Jeroen Weesie, for teaching us SEM.

Sophia Rabe-Hesketh contributed a bit later, after the second project, GSEM, was well underway. GSEM stands for generalized SEM. Sophia is the coauthor of `gllamm` and knows as much about multilevel and structural equation modeling as anybody, and probably more. She helped us a lot through her prolific published works; we did have her visit a few times, though, mainly because we knew that features in GSEM would overlap with features in GLLAMM, and we wanted to straighten out any difficulties that competing features might cause.

About the competing features, Sophia cared nothing. About the GSEM project, she was excited. About syntax and computational methods—well, she straightened us out the first day, even on things we thought we had settled. Today, enough of the underlying workings of GSEM are based on Sophia's and her coauthors' publications that anyone who uses `gsem` should cite Rabe-Hesketh, Skrondal, and Pickles (2004).

We are indebted to the works of Sophia Rabe-Hesketh, Anders Skrondal of the University of Oslo and the Norwegian Institute of Public Health, and Andrew Pickles of the University of Manchester.

Reference

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190.

Also see

[R] **gllamm** — Generalized linear and latent mixed models

Title

Intro 1 — Introduction

Description Remarks and examples Also see

Description

SEM stands for structural equation model. Structural equation modeling is

1. A notation for specifying SEMs.
2. A way of thinking about SEMs.
3. Methods for estimating the parameters of SEMs.

Stata's `sem` and `gsem` commands fit these models: `sem` fits standard linear SEMs, and `gsem` fits generalized SEMs.

In `sem`, responses are continuous and models are linear regression.

In `gsem`, responses are continuous or binary, ordinal, count, or multinomial. Models are linear regression, gamma regression, logit, probit, ordinal logit, ordinal probit, Poisson, negative binomial, multinomial logit, and more.

`sem` fits models to single-level data.

`gsem` fits models to single-level or multilevel data. Latent variables can be included at any level. `gsem` can fit models with mixed effects, including random effects such as unobserved effects within patient, nested effects such as unobserved effects within patient within doctor, and crossed effects such as unobserved effects within occupation and country.

In `sem`, all latent variables are continuous.

In `gsem`, latent variables are continuous or categorical. A model can have continuous latent variables or categorical latent variables but not both. With categorical latent variables, `gsem` can fit latent class models and finite mixture models.

Meanwhile, `sem` provides features not provided by `gsem`: estimation using observations with missing values under the assumption of joint normality; goodness-of-fit statistics, modification indices, tests of indirect effects, and more; and models fit using summary-statistic data.

There is obviously overlap between the capabilities of `sem` and `gsem`. In such cases, results will be nearly equal. Results should be exactly equal because both commands are producing estimates of the same mathematical model, but `sem` and `gsem` use different numerical machinery. `sem`'s machinery requires less calculation and fewer approximations and so is faster and slightly more accurate.

Remarks and examples

Structural equation modeling encompasses a broad array of models from linear regression to measurement models to simultaneous equations, including along the way confirmatory factor analysis (CFA), correlated uniqueness models, latent growth models, multiple indicators and multiple causes (MIMIC) models, and item response theory (IRT) models.

Structural equation modeling is not just an estimation method for a particular model in the way that Stata's `regress` and `probit` commands are, or even in the way that `stcox` and `mixed` are. Structural equation modeling is a way of thinking, a way of writing, and a way of estimating.

If you read the introductory manual pages in the front of this manual—[SEM] **Intro 2**, [SEM] **Intro 3**, and so on—we will do our best to familiarize you with SEM and our implementation of it.

Beginning with [SEM] **Intro 2**, entitled *Learning the language: Path diagrams and command language*, you will learn that

1. A particular SEM is usually described using a path diagram.
2. The `sem` and `gsem` commands allow you to use path diagrams to input models. In fact, `sem` and `gsem` share the same GUI, called the SEM Builder.
3. `sem` and `gsem` alternatively allow you to use a command language to input models. The command language is similar to the path diagrams.

[SEM] **Intro 3**, entitled *Learning the language: Factor-variable notation (gsem only)*, amounts to a continuation of [SEM] **Intro 2**.

4. We teach you Stata's factor-variable notation, a wonderfully convenient shorthand for including categorical variables in models.

In [SEM] **Intro 4**, entitled *Substantive concepts*, you will learn that

5. `sem` provides four different estimation methods; you need to specify the method appropriate for the assumptions you are willing to make. For `gsem`, there are two estimation methods.
6. There are four types of variables in SEMs: A variable is observed or latent, and simultaneously it is endogenous or exogenous. To this, `sem` and `gsem` add another type of variable, the error variable. Error variables are latent exogenous variables with a fixed-unit path coefficient, and they are associated with a single endogenous variable. Error variables are denoted with an `e.` prefix, so if `y1` is an endogenous variable, then `e.y1` is the associated error variable.
7. It is easy to specify path constraints in SEMs—you just draw them, or omit drawing them, on the diagram. It is similarly easy with the SEM Builder as well as with `sem`'s and `gsem`'s command language.
8. Determining whether an SEM is identified can be difficult. We show you how to let the software check for you.
9. Identification also includes normalization constraints. `sem` and `gsem` apply normalization constraints automatically, but you can control that if you wish. Sometimes you might even need to control it.

In [SEM] **Intro 5**, entitled *Tour of models*,

10. We take you on a whirlwind tour of some of the models that `sem` and `gsem` can fit. This is a fun and useful section because we give you an overview without getting lost in the details.

Then in [SEM] **Intro 6**, entitled *Comparing groups*,

11. We show you a highlight of `sem` and `gsem`: their ability to take an SEM and data consisting of groups—sexes, age categories, and the like—and fit the model in an interacted way that makes it easy for you to test whether and how the groups differ.

In [SEM] **Intro 7**, entitled *Postestimation tests and predictions*,

12. We show you how to redisplay results (`sem` and `gsem`), how to obtain exponentiated coefficients (`gsem` only), and how to obtain standardized results (`sem` only).
13. We show you how to obtain goodness-of-fit statistics (`sem` only).
14. We show you how to perform hypothesis tests, including tests for omitted paths, tests for relaxing constraints, and tests for model simplification.
15. We show you how to display other results, statistics, and tests.

16. We show you how to obtain predictions of observed response variables and predictions of latent variables. With `gsem`, you can obtain predicted means, probabilities, or counts that take into account the predictions of the latent variables, or you can set the latent variables to 0.

17. We show you how to access stored results.

In [SEM] **Intro 8**, entitled *Robust and clustered standard errors*,

18. We mention that `sem` and `gsem` optionally provide robust standard errors and provide clustered standard errors, which relaxes the assumption of independence of observations (or subjects) to independence within clusters of observations (subjects).

In [SEM] **Intro 9**, entitled *Standard errors, the full story*,

19. We provide lots of technical detail expanding on item 18.

In [SEM] **Intro 10**, entitled *Fitting models with survey data*,

20. We explain how `sem` and `gsem` can be used with Stata's `svy:` prefix to obtain results adjusted for complex survey designs, including clustered sampling and stratification.

In [SEM] **Intro 11**, entitled *Fitting models with summary statistics data (sem only)*,

21. We show you how to use `sem` with summary statistics data such as the correlation or covariance matrix rather than the raw data. Many sources, especially textbooks, publish data in summary statistics form.

Finally, in [SEM] **Intro 12**, entitled *Convergence problems and how to solve them*,

22. We regretfully inform you that some SEMs have difficulty converging. We figure 5% to 15% of complicated models will cause difficulty. We show you what to do and it is not difficult.

In the meantime,

23. There are many examples that we have collected for you in [SEM] **Example 1**, [SEM] **Example 2**, and so on. It is entertaining and informative simply to read the examples in order.

24. There is an alphabetical glossary in [SEM] **Glossary**, located at the end of the manual.

If you prefer, you can skip all of this introductory material and go for the details. For the full experience, go directly to [SEM] **sem** and [SEM] **gsem**. You will have no idea what we are talking about—we promise.

The technical sections, in logical order, are

Estimation
- [SEM] **sem**
- [SEM] **gsem**
- [SEM] **sem and gsem path notation**
- [SEM] **sem path notation extensions**
- [SEM] **gsem path notation extensions**
- [SEM] **Builder**
- [SEM] **Builder, generalized**
- [SEM] **sem model description options**
- [SEM] **gsem model description options**
- [SEM] **sem group options**
- [SEM] **gsem group options**
- [SEM] **gsem lclass options**
- [SEM] **sem ssd options**
- [SEM] **sem estimation options**
- [SEM] **gsem estimation options**
- [SEM] **sem reporting options**
- [SEM] **gsem reporting options**
- [SEM] **sem and gsem syntax options**
- [SEM] **sem option noxconditional**
- [SEM] **sem option select( )**
- [SEM] **sem and gsem option covstructure( )**
- [SEM] **sem option method( )**
- [SEM] **sem and gsem option reliability( )**
- [SEM] **sem and gsem option from( )**
- [SEM] **sem and gsem option constraints( )**
- [SEM] **gsem family-and-link options**
- [SEM] **ssd** (`sem` only)

Postestimation, summary of
- [SEM] **sem postestimation**
- [SEM] **gsem postestimation**

Reporting results
- [R] **estat**
- [SEM] **estat summarize**
- [SEM] **estat eform** (`gsem` only)
- [SEM] **estat teffects** (`sem` only)
- [SEM] **estat residuals** (`sem` only)
- [SEM] **estat framework** (`sem` only)
- [SEM] **estat sd** (`gsem` only)
- [SEM] **estat lcmean** (`gsem` only)
- [SEM] **estat lcprob** (`gsem` only)

Goodness-of-fit tests
- [SEM] **estat gof** (`sem` only)
- [SEM] **estat eqgof** (`sem` only)
- [SEM] **estat ggof** (`sem` only)
- [SEM] **estat lcgof** (`gsem` only)
- [R] **estat**

Hypotheses tests
- [SEM] **estat mindices** (sem only)
- [SEM] **estat eqtest** (sem only)
- [SEM] **estat scoretests** (sem only)
- [SEM] **estat ginvariant** (sem only)
- [SEM] **estat stable** (sem only)
- [SEM] **test**
- [SEM] **lrtest**
- [SEM] **testnl**
- [SEM] **estat stdize** (sem only)

Linear and nonlinear combinations of results
- [SEM] **lincom**
- [SEM] **nlcom**

Predicted values
- [SEM] **predict after sem**
- [SEM] **predict after gsem**

Methods and formulas
- [SEM] **Methods and formulas for sem**
- [SEM] **Methods and formulas for gsem**

Many of these sections are technical, but mostly in the computer sense of the word. We suggest that when you read the technical sections, you skip to *Remarks and examples*. If you read the introductory sections, you will already know how to use the commands, so there is little reason to confuse yourself with syntax diagrams that are more precise than they are enlightening. However, the syntax diagrams do serve as useful reminders.

Also see

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Acknowledgments**

Title

Intro 2 — Learning the language: Path diagrams and command language

Description Remarks and examples Reference Also see

Description

Individual structural equation models are usually described using path diagrams. Path diagrams are described here.

Path diagrams can be used in `sem`'s (`gsem`'s) GUI, known as the SEM Builder or simply the Builder, as the input to describe the model to be fit. Path diagrams differ a little from author to author, and `sem`'s and `gsem`'s path diagrams differ a little, too. For instance, we omit drawing the variances and covariances between observed exogenous variables by default.

`sem` and `gsem` also provide a command-language interface. This interface is similar to path diagrams but is typable.

Remarks and examples

Remarks are presented under the following headings:

Using path diagrams to specify standard linear SEMs

Specifying correlation

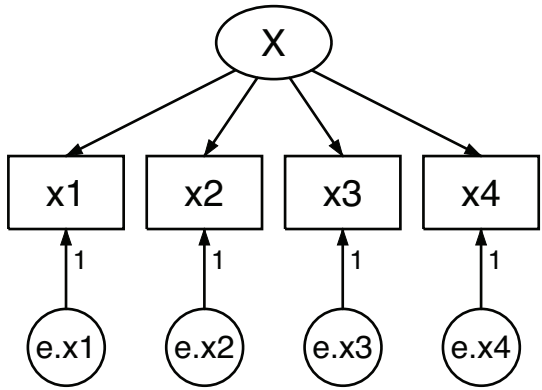
Using the command language to specify standard linear SEMs

Specifying generalized SEMs: Family and link
Specifying generalized SEMs: Family and link, multinomial logistic regression
Specifying generalized SEMs: Family and link, paths from response variables

Specifying generalized SEMs: Multilevel mixed effects (2 levels)
Specifying generalized SEMs: Multilevel mixed effects (3 levels)
Specifying generalized SEMs: Multilevel mixed effects (4+ levels)
Specifying generalized SEMs: Multilevel mixed effects with random intercepts
Specifying generalized SEMs: Multilevel mixed effects with random slopes
Specifying generalized SEMs: Latent class analysis (LCA)
Specifying generalized SEMs: Latent class analysis, class predictors
Specifying generalized SEMs: Latent class analysis, two latent variables

Using path diagrams to specify standard linear SEMs

In structural equation modeling, models are often illustrated in a path diagram such as this one:



This diagram is composed of the following:

1. Boxes and circles with variable names written inside them.
    a. Boxes contain variables that are observed in the data.
    b. Circles contain variables that are unobserved, known as latent variables.
2. Arrows, called paths, that connect some of the boxes and circles.
    a. When a path points from one variable to another, it means that the first variable affects the second.
    b. More precisely, if $s \to d$, it means to add $\beta_k s$ to the linear equation for d. β_k is called the path coefficient.
    c. Sometimes small numbers are written along the arrow connecting two variables. This means that β_k is constrained to be the value specified. (Some authors use the term "path coefficient" to mean standardized path coefficient. We do not.)
    d. When no number is written along the arrow, the corresponding coefficient is to be estimated from the data. Sometimes symbols are written along the path arrow to emphasize this and sometimes not.
    e. The same path diagram used to describe the model can be used to display the results of estimation. In that case, estimated coefficients appear along the paths.
3. There are other elements that may appear on the diagram to indicate variances and between-variable correlations. We will get to them later.

Thus the above figure corresponds to the equations

$$x_1 = \alpha_1 + \beta_1 X + e.x_1$$

$$x_2 = \alpha_2 + \beta_2 X + e.x_2$$

$$x_3 = \alpha_3 + \beta_3 X + e.x_3$$

$$x_4 = \alpha_4 + \beta_4 X + e.x_4$$

There's a third way of writing this model, namely,

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X)
```

This is the way we would write the model if we wanted to use `sem`'s or `gsem`'s command syntax rather than drawing the model in the Builder. The full command we would type is

```
. sem (x1<-X) (x2<-X) (x3<-X) (x4<-X)
```

We will get to that later.

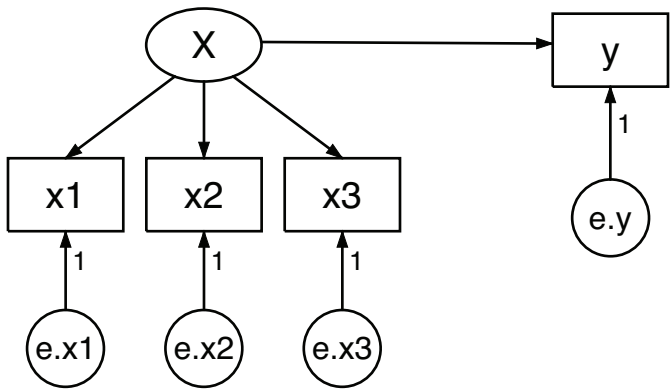
By the way, the above model is a linear single-level model. Linear single-level models can be fit by `sem` or by `gsem`. `sem` is the preferred way to fit linear single-level models because it has added features for these models that you might find useful later. Nonetheless, if you want to fit the model with `gsem`, you would type

```
. gsem (x1<-X) (x2<-X) (x3<-X) (x4<-X)
```

Whether we use `sem` or `gsem`, we obtain the same results.

However we write this model, what is it? It is a measurement model, a term loaded with meaning for some researchers. X might be mathematical ability. x_1, x_2, x_3, and x_4 might be scores from tests designed to measure mathematical ability. x_1 might be the score based on your answers to a series of questions after reading this section.

The model we have just drawn, or written in mathematical notation, or written in Stata command notation, can be interpreted in other ways, too. Look at this diagram:



Despite appearances, this diagram is identical to the previous diagram except that we have renamed x_4 to be y. The fact that we changed a name obviously does not matter substantively. The fact that we have rearranged the boxes in the diagram is irrelevant, too; paths connect the same variables in the same directions. The equations for the above diagrams are the same as the previous equations with the substitution of y for x_4:

$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$

$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

$$x_3 = \alpha_3 + X\beta_3 + e.x_3$$

$$y = \alpha_4 + X\beta_4 + e.y$$

The Stata command notation changes similarly:

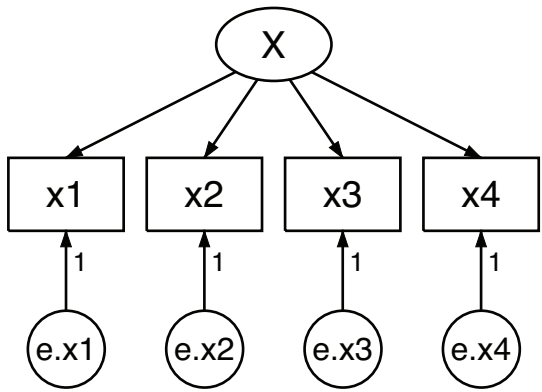
```
(x1<-X) (x2<-X) (x3<-X) (y<-X)
```

Many people looking at the model written this way might decide that it is not a measurement model but a measurement-error model. y depends on X but we do not observe X. We do observe x_1, x_2, and x_3, each a measurement of X, but with error. Our interest is in knowing β_4, the effect of true X on y.

A few others might disagree and instead see a model for interrater agreement. Obviously, we have three or four raters who each make a judgment, and we want to know how well the judgment process works and how well each of these raters performs.

Specifying correlation

One of the key features of SEM is the ease with which you can allow correlation between latent variables to adjust for the reality of the situation. In the measurement model in the previous section, we drew the following path diagram:



which corresponded to the equations

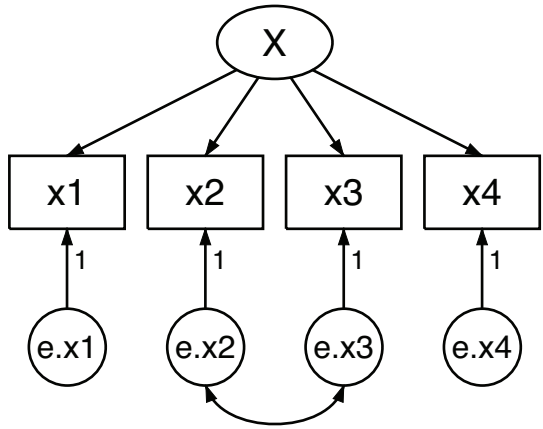
$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$
$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$
$$x_3 = \alpha_3 + X\beta_3 + e.x_3$$
$$x_4 = \alpha_4 + X\beta_4 + e.x_4$$

$(X, x_1, x_2, x_3, x_4, e.x_1, e.x_2, e.x_3, e.x_4) \sim$ i.i.d. with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

where i.i.d. means that observations are independent and identically distributed.

We must appreciate that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are estimated, just as are α_1, β_1, …, α_4, β_4. Some of the elements of $\boldsymbol{\Sigma}$, however, are constrained to be 0; which elements are constrained is determined by how we specify the model. In the above diagram, we drew the model in such a way that we assumed that error variables were uncorrelated with each other. We could have drawn the diagram differently.

If we wish to allow for a correlation between $e.x_2$ and $e.x_3$, we add a curved path between the variables:



The curved path states that there is a correlation to be estimated between the variables it connects. The absence of a curved path—say, between $e.x_1$ and $e.x_4$—means that the variables are constrained to be uncorrelated. That is not to say that x_1 and x_4 are uncorrelated; obviously, they are correlated because both are functions of the same X. Their corresponding error variables, however, are uncorrelated. The equations for this model, in their full detail, are

$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$

$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

$$x_3 = \alpha_3 + X\beta_3 + e.x_3$$

$$x_4 = \alpha_4 + X\beta_4 + e.x_4$$

$$(X, x_1, x_2, x_3, x_4, e.x_1, e.x_2, e.x_3, e.x_4) \sim \text{i.i.d. with mean } \boldsymbol{\mu} \text{ and variance } \boldsymbol{\Sigma}$$

$\boldsymbol{\Sigma}$ is constrained such that

$$\begin{aligned}
\sigma_{e.x_1,e.x_2} &= \sigma_{e.x_2,e.x_1} = 0 \\
\sigma_{e.x_1,e.x_3} &= \sigma_{e.x_3,e.x_1} = 0 \\
\sigma_{e.x_1,e.x_4} &= \sigma_{e.x_4,e.x_1} = 0 \\
\sigma_{e.x_2,e.x_4} &= \sigma_{e.x_4,e.x_2} = 0 \\
\sigma_{e.x_3,e.x_4} &= \sigma_{e.x_4,e.x_3} = 0
\end{aligned}$$

$$\begin{aligned}
\sigma_{X,e.x_1} &= \sigma_{e.x_1,X} = 0 \\
\sigma_{X,e.x_2} &= \sigma_{e.x_2,X} = 0 \\
\sigma_{X,e.x_3} &= \sigma_{e.x_3,X} = 0 \\
\sigma_{X,e.x_4} &= \sigma_{e.x_4,X} = 0
\end{aligned}$$

$\boldsymbol{\mu}$ is constrained such that

$$\begin{aligned}
\mu_X &= 0 \\
\mu_{e.x_1} &= 0 \\
\mu_{e.x_2} &= 0 \\
\mu_{e.x_3} &= 0 \\
\mu_{e.x_4} &= 0
\end{aligned}$$

The parameters to be estimated are

$$\begin{aligned}
&\alpha_1, \beta_1, \ldots, \alpha_4, \beta_4, \\
&\boldsymbol{\mu} \\
&\boldsymbol{\Sigma}
\end{aligned}$$

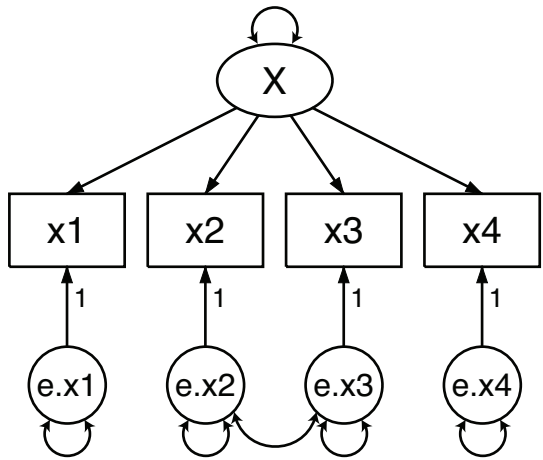
Look carefully at the above list. You will not find a line reading

$$\sigma_{e.x_2,e.x_3} = \sigma_{e.x_3,e.x_2} = 0$$

although you will find lines constraining the other covariances between error terms to be 0. The line is missing because we drew a curved path between $e.x_2$ and $e.x_3$.

There are lots of other curved arrows we could have drawn. By not drawing them, we are asserting that the corresponding covariance is 0.

Some authors would draw the above model as



A curved path from a variable to itself indicates a variance. All covariances could be shown, including those between latent variables and between observed exogenous variables.

When we draw diagrams, however, we will assume variance paths and omit drawing them, and we will similarly assume but omit drawing covariances between observed exogenous variables (there are no observed exogenous variables in this model). The Builder in `sem` mode has an option concerning the latter. Like everyone else, we will not assume correlations between latent variables unless they are shown.

In `sem`'s (`gsem`'s) command-language notation, curved paths between variables are indicated via an option:

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X), cov(e.x2*e.x3)
```

Using the command language to specify standard linear SEMs

You can describe your model to `sem` by using path diagrams with the Builder, or you can describe your model by using `sem`'s command language. Here are the trade-offs:

1. If you use path diagrams, you can see the results of your estimation as path diagrams or as standard computer output.
2. If you use the command language, only standard computer output is available.
3. Typing models in the command language is usually quicker than drawing them in the Builder.
4. You can type models in the command language and store them in do-files. By doing so, you can more easily correct the errors you make.

Translating from path diagrams to command language is easy.

1. Path diagrams have squares and circles to distinguish observed from latent variables.

   In the command language, variables are assumed to be observed if they are typed in lowercase and are assumed to be latent if the first letter is capitalized. Variable `educ` is observed, while variable `Knowledge` or `KNOWLEDGE` is latent.

   If the observed variables in your dataset have uppercase names, type `rename _all, lower` to covert them to lowercase; see [D] **rename group**.

2. When typing path diagrams in the command language, remember the `///` continuation line indicator. You may type

```
. sem (x1<-X) (x2<-X) (x3<-X) (y<-X)
```

   or you may type

```
. sem (x1<-X) (x2<-X)    ///
      (x3<-X) (y<-X)
```

   or you may type

```
. sem (x1<-X)            ///
      (x2<-X)            ///
      (x3<-X)            ///
      (y<-X)
```

3. In path diagrams, you draw arrows connecting variables to indicate paths. In the command language, you type variable names and arrows. So long as your arrow points the correct way, it does not matter which variable comes first. The following mean the same thing:

```
(x1 <- X)
(X -> x1)
```

4. In the command language, you may type multiple variables on either side of the arrow:

```
(X -> x1 x2 x3 x4)
```

   The above means the same as

```
(X -> x1) (X -> x2) (X -> x3) (X -> x4)
```

   which means the same as

```
(x1 <- X) (x2 <- X) (x3 <- X) (x4 <- X)
```

   which means the same as

```
(x1 x2 x3 x4 <- X)
```

   In a more complicated measurement model, we might have

```
(X Y -> x1 x2 x3) (X -> x4 x5) (Y -> x6 x7)
```

   The above means the same as

```
(X -> x1 x2 x3 x4 x5)  ///
(Y -> x1 x2 x3 x6 x7)
```

which means

```
(X -> x1) (X -> x2) (X -> x3) (X -> x4) (X -> x5)     ///
(Y -> x1) (Y -> x2) (Y -> x3) (Y -> x6) (Y -> x7)
```

5. In path diagrams, you are required to show the error variables. In the command language, you may omit the error variables. `sem` knows that each endogenous variable needs an error variable. You can type

```
(x1 <- X) (x2 <- X) (x3 <- X) (x4 <- X)
```

and that means the same thing as

```
(x1 <- X e.x1)   ///
(x2 <- X e.x2)   ///
(x3 <- X e.x3)   ///
(x4 <- X e.x4)
```

except that we have lost the small numeric 1s we had next to the arrows from `e.x1` to `x1`, `e.x2` to `x2`, and so on. To constrain the path coefficient, you type

```
(x1 <- X e.x1@1)    ///
(x2 <- X e.x2@1)    ///
(x3 <- X e.x3@1)    ///
(x4 <- X e.x4@1)
```

It is easier to simply type

```
(x1 <- X) (x2 <- X) (x3 <- X) (x4 <- X)
```

but now you know that if you wanted to constrain the path coefficient `x2<-X` to be 2, you could type

```
(x1 <- X) (x2 <- X@2) (x3 <- X) (x4 <- X)
```

If you wanted to constrain the path coefficients `x2<-X` and `x3<-X` to be equal, you could type

```
(x1 <- X) (x2 <- X@b) (x3 <- X@b) (x4 <- X)
```

We have not covered symbolic constraints with path diagrams yet, but typing symbolic names along the paths in either the Builder or the command language is how you constrain coefficients to be equal.

6. Curved paths are specified with the `cov()` option after you have specified your model:

```
(x1 x2 x3 x4 <- X), cov(e.x2*e.x3)
```

If you wanted to allow for correlation of `e.x2*e.x3` and `e.x3*e.x4`, you can specify that in a single `cov()` option,

```
(x1 x2 x3 x4 <- X), cov(e.x2*e.x3 e.x3*e.x4)
```

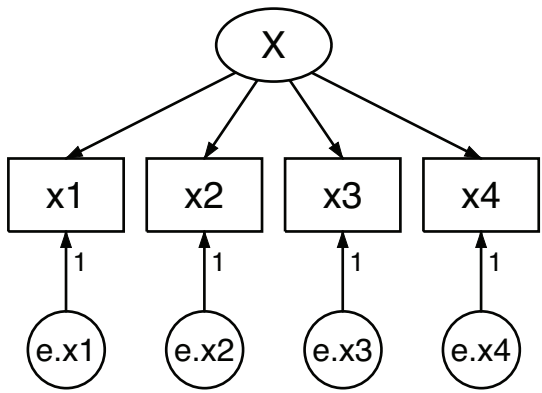
or in separate `cov()` options:

```
(x1 x2 x3 x4 <- X), cov(e.x2*e.x3) cov(e.x3*e.x4)
```

7. Nearly all the above applies equally to `gsem`. We have to say "nearly" because sometimes, in some models, some concepts simply vanish. For instance, in a logistic model, there are no error terms. For generalized responses with family Gaussian, link log, there are error terms, but they cannot be correlated. Also, for responses with family Gaussian, link identity, and censoring, there are error terms, but they cannot be correlated. `gsem` also takes observed exogenous variables as given and so cannot estimate the covariances between them.

Specifying generalized SEMs: Family and link

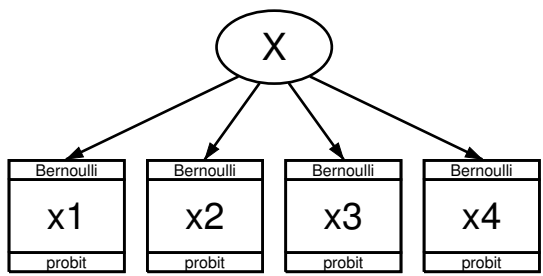
We began this discussion by showing you a linear measurement model:



In this model, we observe continuous measurements `x1`, . . . , `x4`. What if `x1`, . . . , `x4` were instead binary outcomes such as success and failure or passed and failed? That is, what if `x1`, . . . , `x4` took on values of only 1 and 0?

In that case, we would want to fit a model appropriate to binary outcomes. Perhaps we want to fit a logistic regression model or a probit model. To do either one, we will have to use `gsem` rather than `sem`. We will use a probit model.

The path diagram for the measurement model with binary outcomes is



What were plain boxes for `x1`, . . . , `x4` now say "Bernoulli" and "probit" at the top and bottom, meaning that the variable is from the Bernoulli family and is using the probit link. In addition, `e.x1`, . . . , `e.x4` (the error terms for `x1`, . . . , `x4`) have disappeared. In the generalized linear framework, there are error terms only for the Gaussian family.

Perhaps some math will clear up the issue. The generalized linear model is

$$g\{E(y\,|\,X)\} = \mathbf{x}\beta$$

and in the case of probit, $g\{E(y\,|\,X)\} = \Phi^{-1}\{E(y\,|\,X)\}$, where $\Phi(\cdot)$ is the cumulative normal distribution. Thus the equations are

$$\Phi^{-1}\{E(x_1\,|\,X)\} = \alpha_1 + X\beta_1$$

$$\Phi^{-1}\{E(x_2\,|\,X)\} = \alpha_2 + X\beta_2$$

$$\Phi^{-1}\{E(x_3\,|\,X)\} = \alpha_3 + X\beta_3$$

$$\Phi^{-1}\{E(x_4\,|\,X)\} = \alpha_4 + X\beta_4$$

Note that there are no error terms. Equivalently, the above can be written as

$$\Pr(x_1 = 1\,|\,X) = \Phi(\alpha_1 + X\beta_1)$$

$$\Pr(x_2 = 1\,|\,X) = \Phi(\alpha_2 + X\beta_2)$$

$$\Pr(x_3 = 1\,|\,X) = \Phi(\alpha_3 + X\beta_3)$$

$$\Pr(x_4 = 1\,|\,X) = \Phi(\alpha_4 + X\beta_4)$$

In `gsem`'s command language, we write this model as

```
(x1 x2 x3 x4<-X, family(bernoulli) link(probit))
```

or as

```
(x1<-X, family(bernoulli) link(probit))
(x2<-X, family(bernoulli) link(probit))
(x3<-X, family(bernoulli) link(probit))
(x4<-X, family(bernoulli) link(probit))
```

In the command language, you can simply type `probit` to mean `family(bernoulli) link(probit)`, so the model could also be typed as

```
(x1 x2 x3 x4<-X, probit)
```

or even as

```
(x1<-X, probit) (x2<-X, probit) (x3<-X, probit) (x4<-X, probit)
```

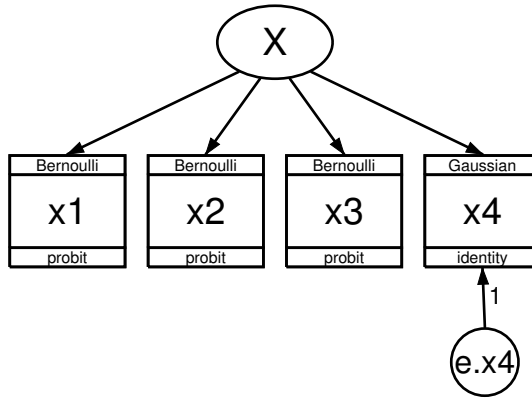
Whether you type `family(bernoulli) link(probit)` or type `probit`, when all the response variables are probit, you can type

```
(x1 x2 x3 x4<-X), probit
```

or

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X), probit
```

The response variables do not have to be all from the same family and link. Perhaps `x1`, `x2`, and `x3` are pass/fail variables but `x4` is a continuous variable. Then the model would be diagrammed as



The words "Gaussian" and "identity" now appear for variable `x4` and `e.x4` is back! Just as previously, the generalized linear model is

$$g\{E(y \mid X)\} = \mathbf{x}\boldsymbol{\beta}$$

and in the case of linear regression, $g(\mu) = \mu$, so our fourth equation becomes

$$E(x_4 \mid X) = X\beta_4$$

or

$$x_4 = \alpha + X\beta_4 + e.x_4$$

and the entire set of equations to be estimated is

$$\Pr(x_1 = 1 \mid X) = \Phi(\alpha_1 + X\beta_1)$$

$$\Pr(x_2 = 1 \mid X) = \Phi(\alpha_2 + X\beta_2)$$

$$\Pr(x_3 = 1 \mid X) = \Phi(\alpha_3 + X\beta_3)$$

$$x_4 = \alpha_4 + X\beta_4 + e.x_4$$

This can be written in the command language as

```
(x1 x2 x3<-X, family(bernoulli) link(probit))
(x4      <-X, family(gaussian)  link(identity))
```

or as

```
(x1 x2 x3<-X, probit) (x4<-X, regress)
```

`regress` is a synonym for `family(gaussian) link(identity)`. Because `family(gaussian) link(identity)` is the default, we can omit the option altogether:

```
(x1 x2 x3<-X, probit) (x4<-X)
```

We demonstrated generalized linear models above by using probit (family Bernoulli, link probit), but we could just as well have used logit (family Bernoulli, link logit). Nothing changes except that in the path diagrams, where you see probit, logit would now appear. Likewise, in the command, where you see `probit`, `logit` would appear.

The same is true with almost all the other possible generalized linear models. What they all have in common is

1. There are no `e.` error terms, except for family Gaussian.
2. Response variables appear the ordinary way except that the family is listed at the top of the box and the link is listed at the bottom of the box, except for family multinomial, link logit (also known as multinomial logistic regression or mlogit).

Concerning item 1, we just showed you a combined probit and linear regression with its $e.x4$ term. Linear regression is family Gaussian.

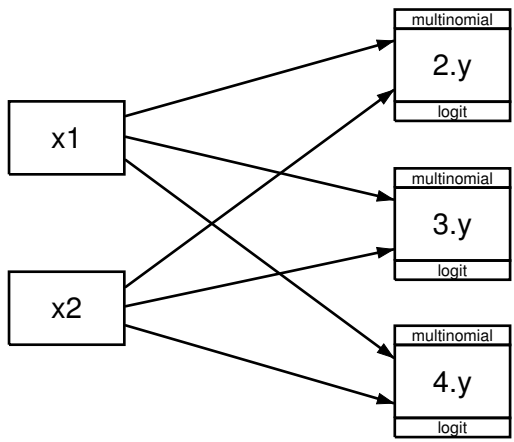
Concerning item 2, multinomial logistic regression is different enough that we need to show it to you.

Specifying generalized SEMs: Family and link, multinomial logistic regression

Let's consider a multinomial logistic model in which y takes on one of four possible outcomes and is determined by x_1 and x_2. Such a model could be fit by Stata's `mlogit` command:

```
. mlogit y x1 x2
```

The model could also be fit by `gsem`. The path diagram would be



Note that there are three boxes for y, boxes containing $2.y$, $3.y$, and $4.y$. When specifying a multinomial logistic regression model in which the dependent variables can take one of k different values, you draw $k-1$ boxes. Names like $1.y$, $2.y$, ..., mean $y=1$, $y=2$, and so on. Logically speaking, you have to omit one of them. Which you omit is irrelevant and is known as the base outcome. Estimated coefficients in the model relevant to the other outcomes will be measured as a difference from the base outcome. We said which you choose is irrelevant, but it might be easier for you to interpret your model if you chose a meaningful base outcome, such as the most frequent outcome. In fact, that is just what Stata's `mlogit` command does by default when you do not specify otherwise.

In path diagrams, you may implicitly or explicitly specify the base outcome. We implicitly specified the base by omitting $1.y$ from the diagram. We could have included it by drawing a box for $1.y$ and labeling it 1b.y. Stata understands the b to mean base category. See [SEM] **Example 37g** for an example.

Once you have handled specification of the base category, you draw path arrows from the predictors to the remaining outcome boxes. We drew paths from x_1 and x_2 to all the outcome boxes, but if we wanted to omit x_1 to $3.y$ and $4.y$, we could have omitted those paths.

Our example is simple in that y is the final outcome. If we had a more complex model where y's outcome affected another response variable, arrows would connect all or some of $2.y$, $3.y$, and $4.y$ to the other response variable.

The command syntax for our simple example is

```
(2.y 3.y 4.y<-x1 x2), mlogit
```

`2.y`, `3.y`, and `4.y` are examples of Stata's factor-variable syntax. The factor-variable syntax has some other features that can save typing in command syntax. `i.y`, for instance, means `1b.y`, `2.y`, `3.y`, and `4.y`. It is especially useful because if we had more levels, say, 10, it would mean `1b.y`, `2.y`, ..., `10.y`. To fit the model we diagrammed, we could type

```
(i.y<-x1 x2), mlogit
```

If we wanted to include some paths and not others, we need to use the more verbose syntax. For instance, to omit paths from x_1 to $3.y$ and $4.y$, we would type

```
(2.y<-x1 x2) (3.y 4.y<-x2), mlogit
```

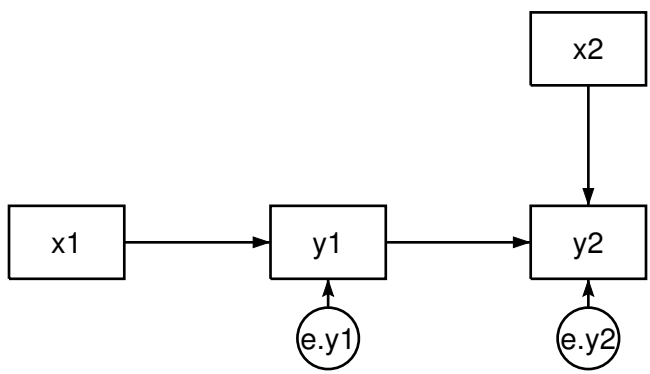
or

```
(i.y<-x2) (2.y<-x1), mlogit
```

For more information on specifying mlogit paths, see [SEM] **Intro 3**, [SEM] **Example 37g**, and [SEM] **Example 41g**.

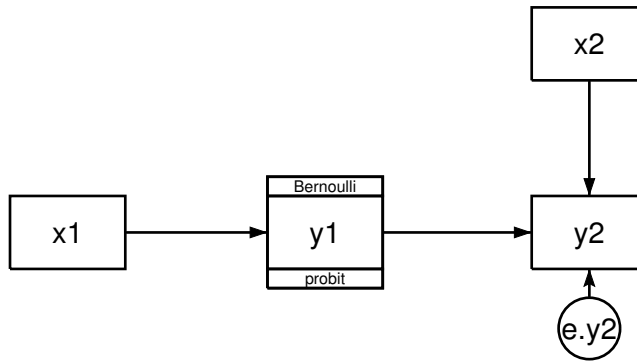
Specifying generalized SEMs: Family and link, paths from response variables

When we draw a path from one response variable to another, we are stating that the first endogenous variable is a predictor of the other endogenous variable. The diagram might look like this:



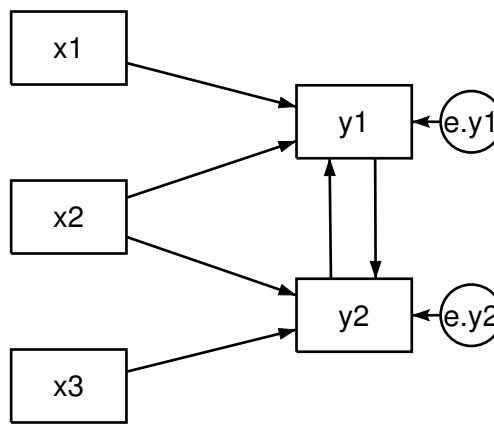
The command syntax might look like `(y1<-x1) (y2<-y1 x2)`.

The response variables in the model are linear, and note that there is a path from y_1 to y_2. Could we change y_1 to be a member of the generalized linear family, such as probit, logit, and so on? It turns out that we can:



In the command syntax, we could write this model as `(y1<-x1, probit) (y2<-y1 x2)`. (In this case, the observed values and not the expectations are used to fit the `y1->y2` coefficient. In general, this is true for all generalized responses that are not family Gaussian, link identity.)

We can make the substitution from linear to generalized linear if the path from y_1 appears in a recursive part of the model. We will define recursive shortly, but trust us that the above model is recursive all the way through. The substitution would not have been okay if the path had been in a nonrecursive portion of the model. The following is a through-and-through nonrecursive model:



It could be written in command syntax as `(y1<-y1 x1 x2) (y2<-y2 x2 x3)`.

In this model, we could not change y_1 to be family Bernoulli and link probit or any other generalized linear response variable. If we tried to fit the model with such a change, we would get an error message:

```
invalid path specification;
a loop among the paths between 'y1' and 'y2' is not allowed
r(198);
```

The software will spot the problem, but you can spot it for yourself.

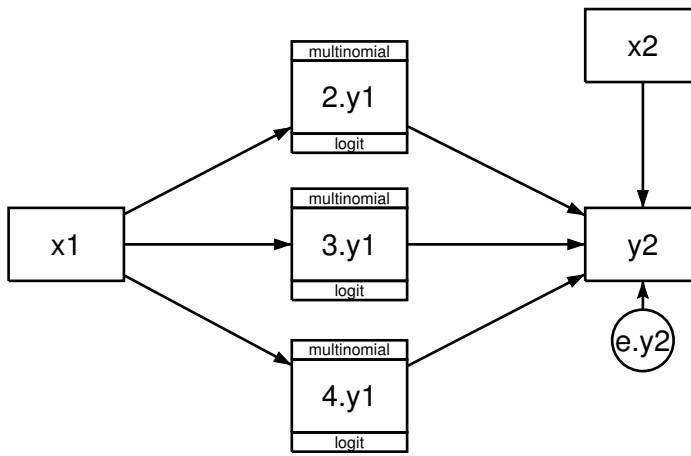
Nonrecursive models have loops. Do you see the loop in the above model? You will if you work out the total effect of a change in y_2 from a change in y_1. Assume a change in y_1. Then that change directly affects y_2, the new value of y_2 affects y_1, which in turn indirectly affects y_2 again, which affects y_1, and on and on.

Now follow the same logic with either the probit or the continuous recursive models above. The change in y_1 affects y_2 and it stops right there.

We sympathize if you think that we have interchanged the terms recursive and nonrecursive. Remember it this way: total effects in recursive models can be calculated nonrecursively because the model itself is recursive, and total effects in nonrecursive models must be calculated recursively because the model itself is nonrecursive.

Anyway, you may draw paths from generalized linear response variables to other response variables, whether linear or generalized linear, as long as no loops are formed.

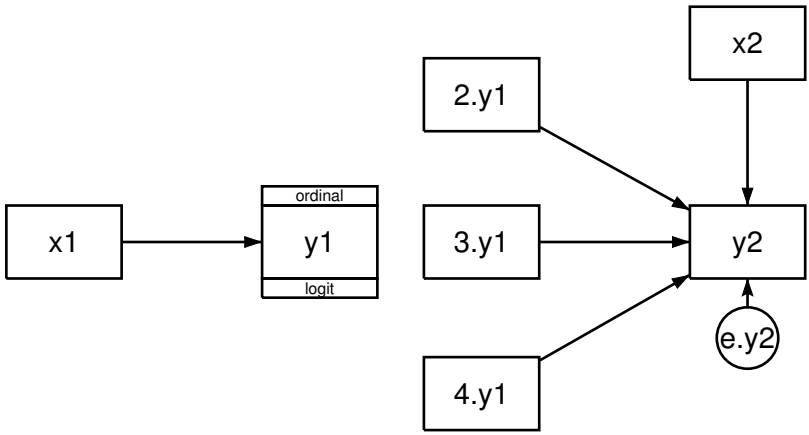
We gave special mention to multinomial logistic regression in the previous section because those models look different from the other generalized linear models. Multinomial logistic regression has a plethora of response variables. In the case of multinomial logistic regression, a recursive model with a path from the multinomial logistic outcome y_1 to (linear) y_2 would look like this:



In the command syntax, the model could be written as

```
(2.y1 3.y1 4.y1<-x1, mlogit) (y2<-2.y1 3.y1 4.y1 x2)
```

In multinomial logistic regression models, outcomes are numbered 1, 2, ..., k. The outcomes might correspond to walk, take public transportation, drive a car, and fly. In ordered probit and logistic models, outcomes are also numbered 1, 2, ..., k, and the outcomes are otherwise similar to multinomial logistic models except that the outcomes are also naturally ordered. The outcomes might be did poorly, did okay, did fairly well, and did well. After taking account of the ordering, you would probably want to model your remaining response variables in the same way you did for multinomial logistic regression. In that case, you would diagram the model like this:



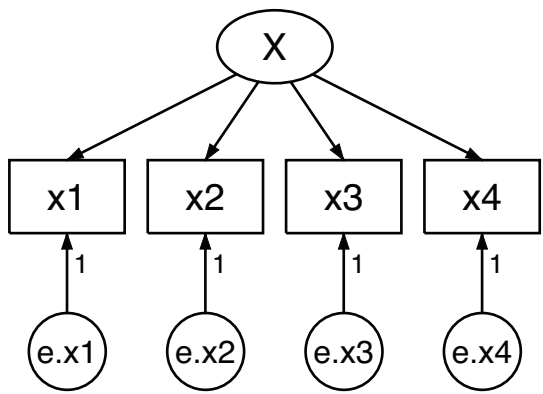
In the command syntax, the model could be written as

```
(y1<-x1, ologit) (y2<-2.y1 3.y1 4.y1 x2)
```

Unlike multinomial logistic regression, in which the k outcomes result in the estimation of $k-1$ equations, in ordered probit and logistic models, only one equation is estimated, and thus the response variable is specified simply as y_1 rather than $2.y_1$, $3.y_1$, and $4.y_1$. Even so, you probably will want the different outcomes to have separate coefficients in the y_2 equation so that the effects of being in groups 1, 2, 3, and 4 are β_0, $\beta_0+\beta_1$, $\beta_0+\beta_2$, and $\beta_0+\beta_3$, respectively. If you drew the diagram with a single path from y_1 to y_2, the effects of being in the groups would be $\beta_0+1\beta_1$, $\beta_0+2\beta_1$, $\beta_0+3\beta_1$, and $\beta_0+4\beta_1$, respectively.

Specifying generalized SEMs: Multilevel mixed effects (2 levels)

The models above are all single-level models or, equivalently, observational-level models. Let's reconsider our original measurement model:



The data for this model would look something like this:

```
. list in 1/5
```

	x1	x2	x3	x4
1.	96	82	96	43
2.	106	81	93	40
3.	127	95	96	134
4.	123	98	94	109
5.	119	99	100	108

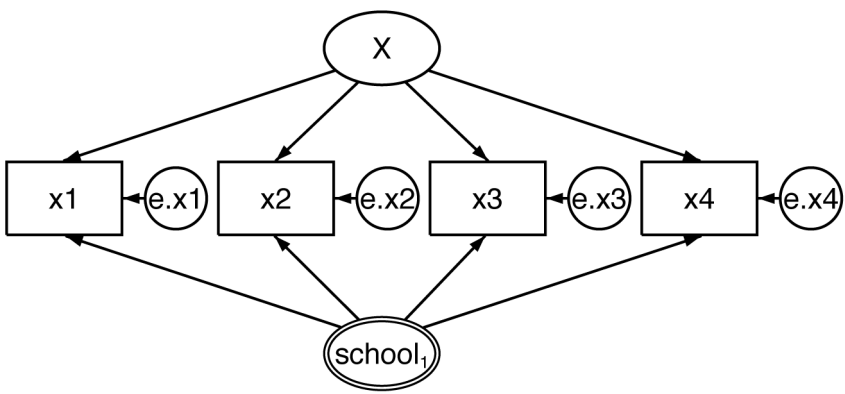
Let's pretend that the observations are students and that `x1`, . . . , `x4` are four test scores.

Now let's pretend that we have new data with students from different schools. A part of our data might look like this:

```
. list in 1/10
```

	school	x1	x2	x3	x4
1.	1	116	128	125	127
2.	1	92	104	101	103
3.	1	105	117	114	116
4.	1	117	129	126	128
5.	1	102	114	111	113
6.	2	116	122	120	121
7.	2	95	101	99	100
8.	2	80	86	84	85
9.	2	96	102	100	101
10.	2	107	113	111	112

We have four test scores from various students in various schools. Our corresponding model might be



The `school1` inside double circles in the figure tells us, "I am a latent variable at the `school` level—meaning that I am constant within school and vary across schools—and I correspond to the latent variable named `M1`." That is, double circles denote a latent variable named `M#`, where `#` is the subscript of the variable name inside the double circles. Meanwhile, the variable name inside the double circles specifies the level of the `M#` variable.

The equivalent command-language form for this model is

```
(x1 x2 x3 x4<-X M1[school])
```

The `M1` part of `M1[school]` is the latent variable's name, while the `[school]` part means "at the `school` level". Thus `M1[school]` denotes latent variable `M1`, which varies at the `school` level or, equivalently, which is constant within `school`.

By the way, we gave the latent variable the name `M1` just to match what the Builder does by default. In real life, we would probably give it a different name, such as `S`.

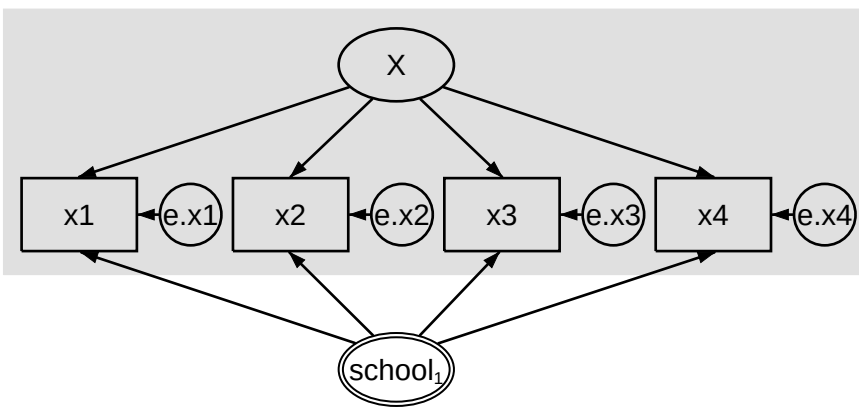
The mathematical way of writing this model is

$$x_1 = \alpha_1 + \beta_1 X + \gamma_1 M_{1,S} + e.x_1$$
$$x_2 = \alpha_2 + \beta_2 X + \gamma_2 M_{1,S} + e.x_2$$
$$x_3 = \alpha_3 + \beta_3 X + \gamma_3 M_{1,S} + e.x_3$$
$$x_4 = \alpha_4 + \beta_4 X + \gamma_4 M_{1,S} + e.x_4$$

where $S =$ school number.

Thus we have three different ways of referring to the same thing: `school1` inside double circles in the path diagram corresponds to `M1[school]` in the command language, which in turn corresponds to $M_{1,S}$ in the mathematical notation.

Rabe-Hesketh, Skrondal, and Pickles (2004) use boxes to identify different levels of the model. Our path diagrams are similar to theirs, but they do not use double circles for multilevel latent variables. We can put a box around the individual-level part of the model, producing something that looks like this:



You can do that in the Builder, but the box has no special meaning to `gsem`; however, adding the box does make the diagram easier to understand in presentations.

However you diagram the model, this model is known as a two-level model. The first level is the student or observational level, and the second level is the school level.

Specifying generalized SEMs: Multilevel mixed effects (3 levels)

A three-level model adds another latent variable at yet another level. For instance, perhaps we have data on four test scores for individual students, who are nested within school, which are nested within counties. The beginning of our data might look like this:

```
. list in 1/10
```

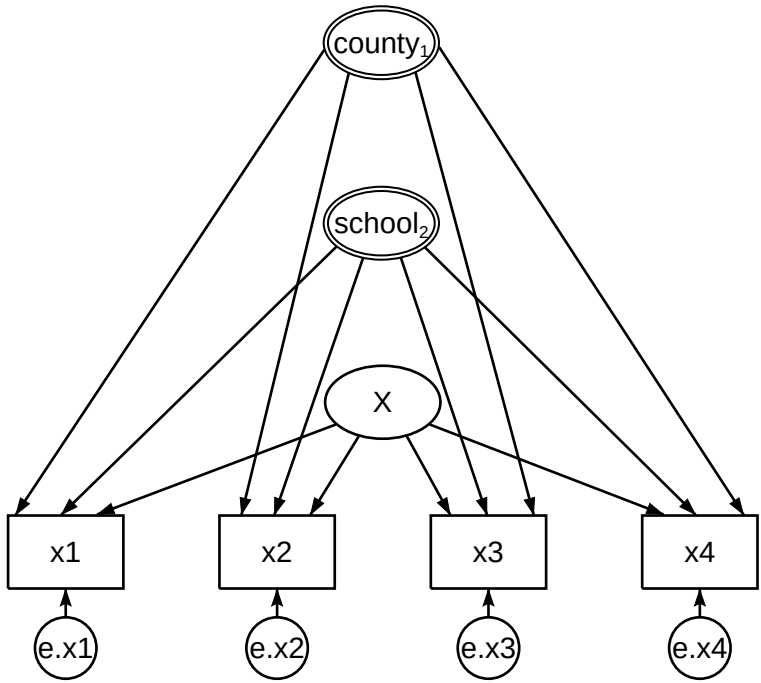
	county	school	x1	x2	x3	x4
1.	1	1	103	122	117	120
2.	1	1	88	107	102	105
3.	1	1	106	125	120	123
4.	1	1	108	127	122	125
5.	1	1	100	119	114	117
6.	1	2	90	99	97	98
7.	1	2	87	96	94	95
8.	1	2	121	130	128	129
9.	1	2	100	109	107	108
10.	1	2	88	97	95	96

Further down, the data might look like this:

```
. list in 991/1l
```

	county	school	x1	x2	x3	x4
991.	2	1	100	96	97	97
992.	2	1	99	95	96	96
993.	2	1	90	86	87	87
994.	2	1	79	75	76	76
995.	2	1	68	64	65	65
996.	2	2	105	111	109	110
997.	2	2	94	100	98	99
998.	2	2	90	96	94	95
999.	2	2	99	105	103	104
1000.	2	2	104	110	108	109

We might diagram the student-within-school-within-county model as



Now we have two variables in double circles: `county1` and `school2`.

`county1` tells us, "I am a latent variable at the `county` level—meaning that I am constant within county and vary across counties—and I correspond to the latent variable named `M1`."

`school2` tells us, "I am a latent variable at the `school` level—meaning that I am constant within school and vary across schools—and I correspond to the latent variable named `M2`."

Do not read anything into the fact that the latent variables are named `M1` and `M2` rather than `M2` and `M1`. When we diagrammed the figure in the Builder, we diagrammed `county` first and then we added `school`. Had we diagrammed them the other way around, the latent variable names would have been reversed. We can edit the names after the fact, but we seldom bother.

The equivalent command-language form for this three-level model is

```
(x1 x2 x3 x4<-X M1[county] M2[county>school])
```

or

```
(x1 x2 x3 x4<-X M1[county] M2[school<county])
```

The `[school<county]` part is usually spoken aloud as "school within county", and `[county>school]` is spoken aloud as "county containing school". When we write `[county>school]` or `[school<county]`, we are saying that the counties contain schools or, equivalently, that school is nested within county. Thus the model we are specifying is in fact a three-level nested model.

The order in which we specify the level effects is irrelevant; we could just as well type

```
(x1 x2 x3 x4<-X M2[county>school] M1[county])
```

or

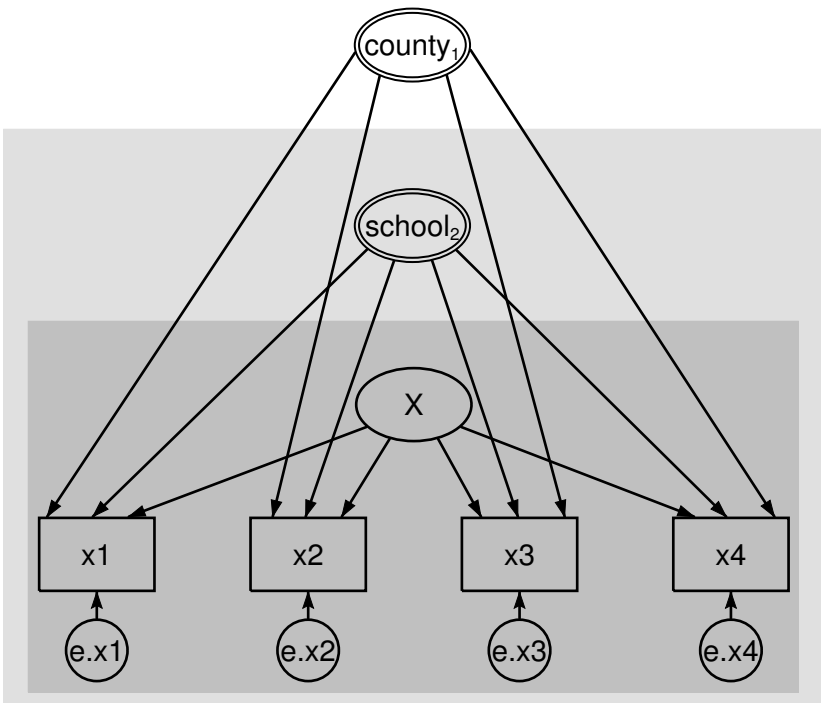
```
(x1 x2 x3 x4 <- X M2[school<county] M1[county])
```

The mathematical way of writing this model is

$$x_1 = \alpha_1 + \beta_1 X + \gamma_1 M_{1,C} + \delta_1 M_{2,S} + e.x_1$$
$$x_2 = \alpha_2 + \beta_2 X + \gamma_2 M_{1,C} + \delta_2 M_{2,S} + e.x_2$$
$$x_3 = \alpha_3 + \beta_3 X + \gamma_3 M_{1,C} + \delta_3 M_{2,S} + e.x_3$$
$$x_4 = \alpha_4 + \beta_4 X + \gamma_4 M_{1,C} + \delta_4 M_{2,S} + e.x_4$$

where $C =$ county number and $S =$ school number.

Three-level nested models are often double-boxed (or double-shaded) in presentation:



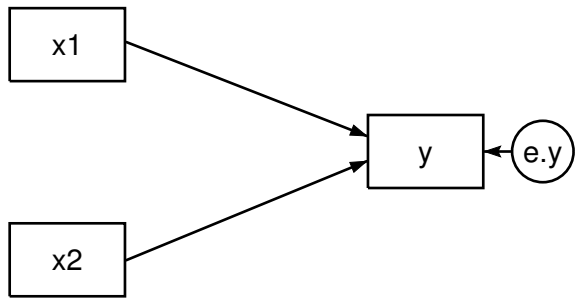
The darker box highlights the first level (student, in our case), and the lighter box highlights the second level (school, in our case). As we previously mentioned, you can add these boxes using the Builder, but they are purely aesthetic and have no meaning to `gsem`.

Specifying generalized SEMs: Multilevel mixed effects (4+ levels)

We have now diagrammed one-, two-, and three-level nested models. You can draw with the Builder and fit with `gsem` higher-level models, but you will usually need lots of data to be successful in fitting those models and, even so, you may run into other estimation problems.

Specifying generalized SEMs: Multilevel mixed effects with random intercepts

Let's change gears and consider with the following simple model a linear regression of y on x_1 and x_2:



In the command language, this model is specified as

```
(x1 x2->y)
```

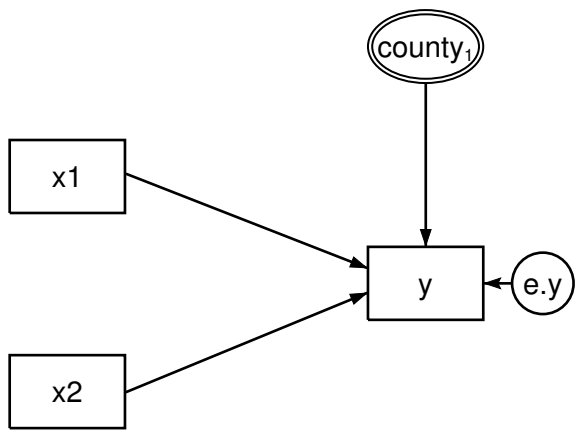
or

```
(y<-x1 x2)
```

and the mathematical representation is

$$y = \alpha + \beta x_1 + \gamma x_2 + e.y$$

Now assume that we have data not just on y, x_1, and x_2, but also on county of residence. If we wanted to add a random intercept—a random effect—for county, the diagram becomes



The command-language equivalent is

```
(x1 x2 M1[county]->y)
```

or

```
(y<-x1 x2 M1[county])
```

and the mathematical representation is

$$y = \alpha + \beta x_1 + \gamma x_2 + M_{1,C} + e.y$$

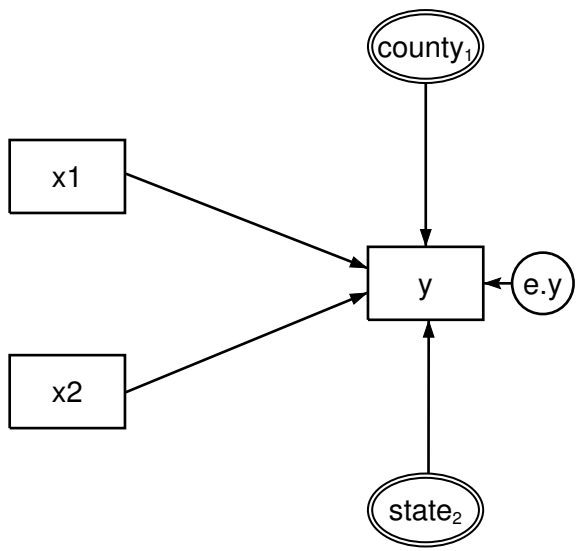
where $C =$ county number. Actually, the model is

$$y = \alpha + \beta x_1 + \gamma x_2 + \delta M_{1,C} + e.y$$

but δ is automatically constrained to be 1 by `gsem`. The software is not reading our mind; consider a solution for $M_{1,C}$ with $\delta = 0.5$. Then another equivalent solution is $M_{1,C}/2$ with $\delta = 1$, and another is $M_{1,C}/4$ with $\delta = 2$, and on and on, because in all cases, $\delta M_{1,C}$ will equal the same value. The fact is that δ is unidentified. Whenever a latent variable is unidentified because of such scaling considerations, `gsem` (and `sem`) automatically set the coefficient to 1 and thus set the scale.

This is a two-level model: the first level is the observational level and the second level is county.

Just as we demonstrated previously with the measurement model, we could have a three-level nested model. We could imagine the observational level nested within the county level nested within the state level. The path diagram would be



The command-language equivalent is

```
(y<-x1 x2 M1[county<state] M2[state])
```

and the mathematical representation is

$$y = \alpha + \beta x_1 + \gamma x_2 + M_{1,C} + M_{2,S} + e.y$$

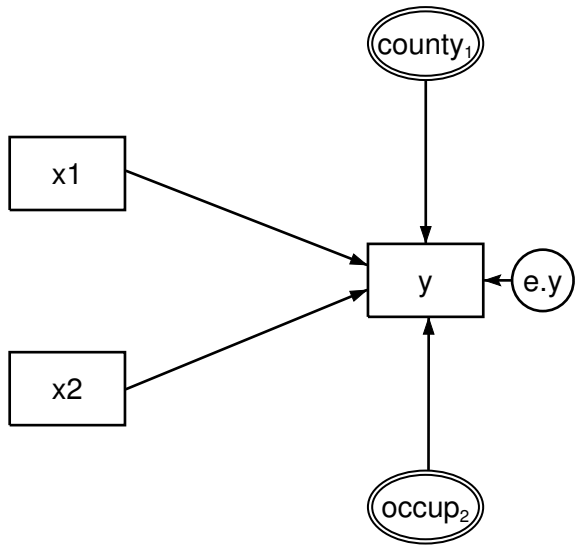
where $C =$ county number and $S =$ state number. Just as previously, the actual form of the equation is

$$y = \alpha + \beta x_1 + \gamma x_2 + \delta M_{1,C} + \zeta M_{2,S} + e.y$$

but the constraints $\delta = \zeta = 1$ are automatically applied by the software.

You can specify higher-level models, but just as we mentioned when discussing the higher-level measurement models, you will need lots of data to fit them successfully and you still may run into other estimation problems.

You can also fit crossed models, such as county and occupation. Unlike county and state, where a particular county appears only in one state, the same occupation will appear in more than one county, which is to say, occupation and county are crossed, not nested. Except for a change in the names of the variables, the path diagram for this model looks identical to the diagram for the state and county-within-state model:



When we enter the model into the Builder, however, we will specify that the two effects are crossed rather than nested.

The command-language way of expressing this crossed model is

```
(y<-x1 x2 M1[county] M2[occupation])
```

and the mathematical representation is

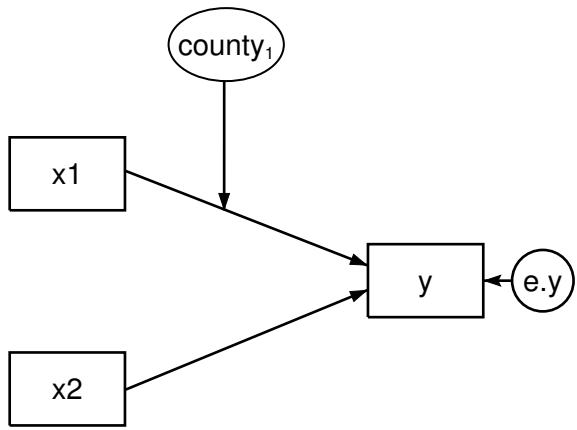
$$y = \alpha + \beta x_1 + \gamma x_2 + M_{1,C} + M_{2,O} + e.y$$

where C = county number and O = occupation number.

Higher-order crossed models are theoretically possible, and you will find `gsem` game to try to fit them. However, you are unlikely to be successful unless you have lots of data so that all the effects can be identified. In fact, you can specify crossed models anywhere you can specify nested models, but the same comment applies.

Specifying generalized SEMs: Multilevel mixed effects with random slopes

Perhaps we feel that county does not affect the intercept, but it affects the slope of `x1`. In that case, we just shift the path from `county1` to instead point to the path between y and x_1.



The command-language equivalent for this model is

```
(y<-x1 c.x1#M1[county]) (y<-x2)
```

or

```
(y<-x1 c.x1#M1[county] x2)
```

To include a random slope (coefficient) on a variable in the command language, include the variable on which you want the random slope just as you ordinarily would:

```
(y<-x1
```

Then, before closing the parenthesis, type

`c.`*variable*`#`*latent_variable*`[`*grouping_variable*`]`

In our case, the *variable* on which we want the random coefficient is `x1`, the *latent_variable* we want to create is named `M1`, and the *grouping_variable* within which the latent variable will be constant is `county`, so we type

```
(y<-x1 c.x1#M1[county]
```

Finally, finish off the command by typing the rest of the model:

```
(y<-x1 c.x1#M1[county] x2)
```

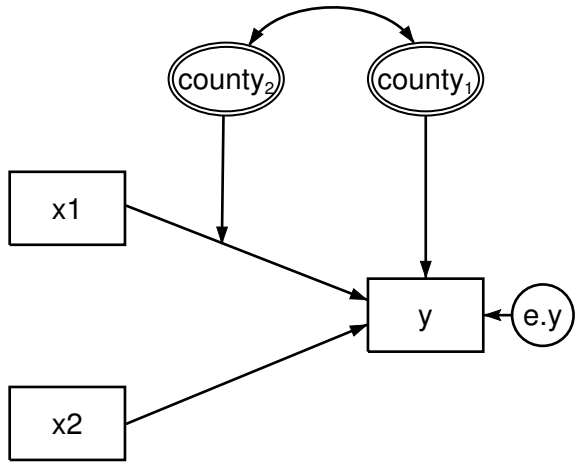
This is another example of Stata's factor-variable notation.

The mathematical representation of the random-slope model is

$$y = \alpha + \beta x_1 + \gamma x_2 + \delta M_{1,C} x_1 + e.y$$

where $C =$ county number and $\delta = 1$.

You can simultaneously specify both random slope and random intercept by putting together what we have already done. The first time you see the combined path diagram, you may think there is a mistake:



County appears twice in two different double circles, once with a subscript of 1 and the other with a subscript of 2! We would gamble at favorable odds that you would have included `county` in double circles once and then would have drawn two paths from it, one to the path between `x1` and `y`, and the other to `y` itself. That model, however, would make an extreme assumption.

Consider what you would be saying. There is one latent variable `M1`. The random intercepts would be equal to `M1`. The random slopes would be equal to γ`M1`, a scalar replica of the random intercepts! You would be constraining the random slopes and intercepts to be correlated 1.

What you usually want, however, is one latent variable for the random slopes and another for the random intercepts. They might be correlated, but they are not related by a multiplicative constant. Thus we also included a covariance between the two random effects.

The command-language equivalent of the correct path diagram (the one shown above) is

```
(y<-x1 x2 c.x1#M2[county] M1[county])
```

You will learn later that the command language assumes covariances exist between latent exogenous variables unless an option is specified. Meanwhile, the Builder assumes those same covariances are 0 unless a curved covariance path is drawn.

The mathematical representation of the model is

$$y = \alpha + \beta x_1 + \gamma x_2 + \delta M_{2,C} x_1 + \zeta M_{1,C} + e.y$$

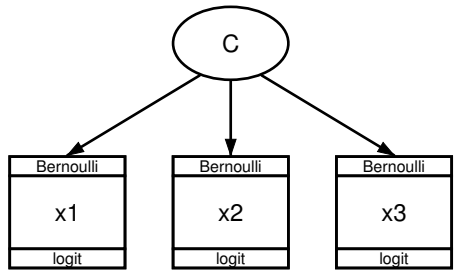
where C = county number and $\delta = \zeta = 1$.

Specifying generalized SEMs: Latent class analysis (LCA)

All the latent variables discussed in the previous sections are continuous latent variables. We can also fit models with categorical latent variables using `gsem`. The unobserved levels of a categorical latent variable are known as latent classes. The classes represent groups in a population such as groups of consumers with different buying preferences. In this manual, we refer to models with categorical latent variables as latent class models, and we refer to an analysis involving categorical latent variables as a latent class analysis (LCA).

Latent class models have distinct parameters for each class. We also estimate the probability of being in each class.

Let's consider a simple latent class model with a categorical latent variable `C` that has two classes and with three observed binary variables, `x1`, `x2`, and `x3`, that are modeled using logistic regression. Some textbooks draw a path diagram for this model as a measurement model. For instance, you may see something like



When we fit this model, we will actually estimate one intercept for `x1` in class 1 and one intercept for `x1` in class 2. Likewise, we will estimate class-specific intercepts for `x2` and for `x3`. Mathematically, the logistic equations for class 1 are

$$\Pr(x_1 = 1 \,|\, C = 1) = \frac{\exp(\alpha_{11})}{1 + \exp(\alpha_{11})}$$

$$\Pr(x_2 = 1 \,|\, C = 1) = \frac{\exp(\alpha_{21})}{1 + \exp(\alpha_{21})}$$

$$\Pr(x_3 = 1 \,|\, C = 1) = \frac{\exp(\alpha_{31})}{1 + \exp(\alpha_{31})}$$

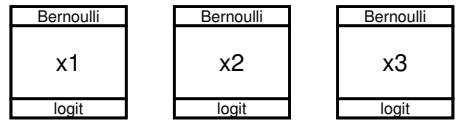
and the equations for class 2 are

$$\Pr(x_1 = 1 \,|\, C = 2) = \frac{\exp(\alpha_{12})}{1 + \exp(\alpha_{12})}$$

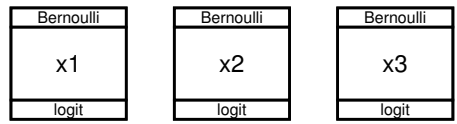
$$\Pr(x_2 = 1 \,|\, C = 2) = \frac{\exp(\alpha_{22})}{1 + \exp(\alpha_{22})}$$

$$\Pr(x_3 = 1 \,|\, C = 2) = \frac{\exp(\alpha_{32})}{1 + \exp(\alpha_{32})}$$

In a sense, `C` has an effect on `x1`, `x2`, and `x3`, but we do not estimate any coefficients on `C`. Rather, the two classes of `C` partition our model into two component models, each with the same `x` variables but with different coefficients (intercepts) on those `x` variables. In Stata's path diagrams, arrows always represent coefficients, so the path diagram above does not actually correspond with the model we fit. A more accurate description would be to draw the diagram as



for the first class and again as



for the second class. Now, we could place the six estimated intercepts in the six boxes.

However, these two diagrams do not fully describe the model. We also estimate the probability of being in each class using multinomial logistic regression,

$$\Pr(C = 1) = \frac{e^{\gamma_1}}{e^{\gamma_1} + e^{\gamma_2}}$$
$$\Pr(C = 2) = \frac{e^{\gamma_2}}{e^{\gamma_1} + e^{\gamma_2}}$$

where γ_1 and γ_2 are intercepts in the multinomial logit model. By default, the first class will be treated as the base, so $\gamma_1 = 0$. We do not have a place for the γ parameter estimates on our diagrams above. Thus, we cannot draw path diagrams in the Builder to fit latent class models. Instead, we fit latent class models using the command language.

We specify this model using the command language as

```
(x1 x2 x3 <- ), logit lclass(C 2)
```

`lclass(C 2)` says that our categorical latent variable is named `C` and has two classes. Because we are only estimating an intercept in each equation, we do not include any variables on the right side of the arrow. We could directly specify that a constant, `_cons`, is included by typing

```
(x1 x2 x3 <- _cons), logit lclass(C 2)
```

We could also be more explicit about having separate models for the two classes by including `1:` at the beginning of the path specification for class 1 and including `2:` at the beginning of the path specification for class 2.

```
(1: x1 x2 x3 <- _cons) (2: x1 x2 x3 <- _cons), logit lclass(C 2)
```

We could even write a separate path specification for each class and outcome variable combination.

```
(1: x1 <- _cons) (1: x2 <- _cons) (1: x3 <- _cons)
(2: x1 <- _cons) (2: x2 <- _cons) (2: x3 <- _cons),
logit lclass(C 2)
```

In any case, the model is the same.

We can build on this to fit different models for different classes and for different outcome variables. For instance, if x3 is ordinal, we could use an ordinal logit model for this variable by typing

```
(1: x1 <- _cons, logit) (1: x2 <- _cons, logit)
(1: x3 <- _cons, ologit) (2: x1 <- _cons, logit)
(2: x2 <- _cons, logit) (2: x3 <- _cons, ologit),
lclass(C 2)
```

Or we can add a constraint that the intercepts for x1 and x2 are equal in class 2.

```
(1: x1 <- _cons) (1: x2 <- _cons) (1: x3 <- _cons)
(2: x1 <- _cons@c) (2: x2 <- _cons@c) (2: x3 <- _cons),
logit lclass(C 2)
```

Other than including continuous latent variables, all command-language features of `gsem` discussed above can be used to modify and extend this latent class model.

Specifying generalized SEMs: Latent class analysis, class predictors

We can include variables in a latent class model that predict class membership. Extending the model above, we simply add exogenous variables to the multinomial logit model for `C`. If we believe that `z` predicts class membership, our multinomial logit model becomes

$$\Pr(C=1) = \frac{e^{\gamma_1+z\beta_1}}{e^{\gamma_1+z\beta_1}+e^{\gamma_2+z\beta_2}}$$

$$\Pr(C=2) = \frac{e^{\gamma_2+z\beta_2}}{e^{\gamma_1+z\beta_1}+e^{\gamma_2+z\beta_2}}$$

where $\gamma_1 = 0$ and $\beta_1 = 0$ when `C` $= 1$ is the base class.

To fit this model using the command language, we type

```
(x1 x2 x3 <- _cons, logit) (C <- z), lclass(C 2)
```

We can also allow different predictors for different classes. For instance, if `C` has three levels, we can type

```
(x1 x2 x3 <- _cons, logit)
(1.C <- z1 z2 z3)
(2.C <- z1 z2)
(3.C <- z1 z2 z3),
lclass(C 3)
```

where `z1`, `z2`, and `z3` are predictors of class membership, but only `z1` and `z2` are predictors of `C` $= 2$.

Specifying generalized SEMs: Latent class analysis, two latent variables

Latent class models can include more than one categorical latent variable. Let's consider an example where `C` has two classes and `D` has three classes.

We could extend our model from *Specifying generalized SEMs: Latent class analysis (LCA)* to include `D` by typing

```
(x1 x2 x3 <- _cons), logit lclass(C 2) lclass(D 3)
```

Now, if we want to refer to a specific class, we use factor-variable notation before the colon. The most verbose specification of this model is

```
(1.C#1.D: x1 <- _cons) (1.C#1.D: x2 <- _cons) (1.C#1.D: x3 <- _cons)
(2.C#1.D: x1 <- _cons) (2.C#1.D: x2 <- _cons) (2.C#1.D: x3 <- _cons)
(1.C#2.D: x1 <- _cons) (1.C#2.D: x2 <- _cons) (1.C#2.D: x3 <- _cons)
(2.C#2.D: x1 <- _cons) (2.C#2.D: x2 <- _cons) (2.C#2.D: x3 <- _cons)
(1.C#3.D: x1 <- _cons) (1.C#3.D: x2 <- _cons) (1.C#3.D: x3 <- _cons)
(2.C#3.D: x1 <- _cons) (2.C#3.D: x2 <- _cons) (2.C#3.D: x3 <- _cons),
logit lclass(C 2) lclass(D 3)
```

As before, this allows us to specify different models for different outcomes or for different classes. It also allows us to specify constraints. For instance, if we want equal intercepts for `x3` when `C` = 2 and `D` = 2 and when `C` = 2 and `D` = 3, we type

```
(1.C#1.D: x1 <- _cons) (1.C#1.D: x2 <- _cons) (1.C#1.D: x3 <- _cons)
(2.C#1.D: x1 <- _cons) (2.C#1.D: x2 <- _cons) (2.C#1.D: x3 <- _cons)
(1.C#2.D: x1 <- _cons) (1.C#2.D: x2 <- _cons) (1.C#2.D: x3 <- _cons)
(2.C#2.D: x1 <- _cons) (2.C#2.D: x2 <- _cons) (2.C#2.D: x3 <- _cons@a)
(1.C#3.D: x1 <- _cons) (1.C#3.D: x2 <- _cons) (1.C#3.D: x3 <- _cons)
(2.C#3.D: x1 <- _cons) (2.C#3.D: x2 <- _cons) (2.C#3.D: x3 <- _cons@a),
logit lclass(C 2) lclass(D 3)
```

We can allow predictors of class membership just as we did with a single categorical latent variable. For specifying `z` as a predictor of the classes of C, we type

```
(x1 x2 x3 <- _cons, logit) (C <- z), lclass(C 2) lclass(D 3)
```

We can also add predictors of specific cells in the interaction between `C` and `D`. For instance,

```
(x1 x2 x3 <- _cons, logit) (1.C#3.D <- z), lclass(C 2) lclass(D 3)
```

Adding predictors or even an intercept (by specifying `_cons`) to an individual cell induces correlation between the categorical latent variables `C` and `D`.

And now you, too, are an expert on Stata's path diagrams and command language for SEM.

Reference

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190.

Also see

[SEM] **Intro 1** — Introduction

[SEM] **Intro 3** — Learning the language: Factor-variable notation (gsem only)

[SEM] **Builder** — SEM Builder

[SEM] **Builder, generalized** — SEM Builder for generalized models

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

Title

Intro 3 — Learning the language: Factor-variable notation (gsem only)

Description Remarks and examples Also see

Description

This entry concerns `gsem` only; `sem` does not allow the use of factor variables.

`gsem` allows you to use Stata's factor-variable notation in path diagrams (in the SEM Builder) and in the command language. Use of the notation, though always optional, is sometimes useful:

1. Use of factor variables sometimes saves effort. For instance, rather than typing

   ```
   . generate femXage = female * age
   ```

   and including `femXage` in your model, you can directly include `1.female#c.age`. You can type `1.female#c.age` with the `gsem` command or type it into an exogenous-variable box in a path diagram.

2. Use of factor variables can save even more effort in command syntax. You can type things like `i.female i.skill i.female#i.skill` to include main effects and interaction effects of female interacted with indicators for all the different levels of skill.

3. Use of factor variables causes the postestimation commands `margins`, `contrast`, and `pwcompare` to produce more useful output. For instance, they will show the effect of the discrete change between `female` $= 0$ and `female` $= 1$ rather than the infinitesimal change (slope) of `female`.

In the examples in the rest of this manual, we sometimes use factor-variable notation and, at other times, ignore it. We might have variable `smokes` recording whether a person smokes. In one example, you might see `smokes` directly included in the model and yet, in another example, we might have the odd-looking `1.smokes`, which is factor-variable notation for emphasizing that `smokes` is a 0/1 variable. We probably used `1.smokes` for reason 3, although we might have done it just to emphasize that variable `smokes` is indeed 0/1.

In other examples, we will use more complicated factor-variable notation, such as `1.female#c.age`, because it is more convenient.

You should follow the same rules. Use factor-variable notation when convenient or when you plan subsequently to use postestimation commands `margins`, `contrast`, or `pwcompare`. If neither reason applies, use factor-variable notation or not. There is no benefit from consistency in this case.

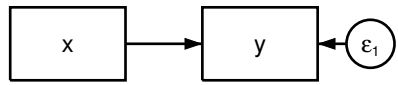
Remarks and examples

Remarks are presented under the following headings:

Specifying indicator variables
Specifying interactions with indicator variables
Specifying categorical variables
Specifying interactions with categorical variables
Specifying endogenous variables

Specifying indicator variables

We wish to fit the following model:

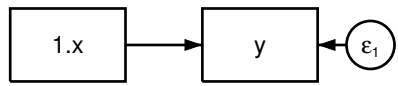


This model corresponds to the equation $y_i = \beta x_i + \epsilon_i$. It can be fit using the Builder with the path diagram above or using the command syntax by typing

```
. gsem (y<-x)
```

Say we now tell you that `x` is a 0/1 (binary) variable. Perhaps `x` indicates male/female. That changes nothing mathematically. The way we drew the path diagram and specified the command syntax are as valid when `x` is an indicator variable as they were when `x` was continuous.

Specifying `1.x` is a way you can emphasize that `x` is 0/1:



That model will produce the same results as the first model. In command syntax, we can type

```
. gsem (y<-1.x)
```

The only real advantage of `1.x` over `x` arises when we plan to use the postestimation command `margins`, `contrast`, or `pwcompare`, as we mentioned above. If we fit the model using `1.x`, those commands will know `x` is 0/1 and will exploit that knowledge to produce more useful output.

In other cases, the *#.varname* notation can sometimes save us effort. *#.varname* means a variable equal to 1 if *varname* = *#*. It is not required that *varname* itself be an indicator variable.

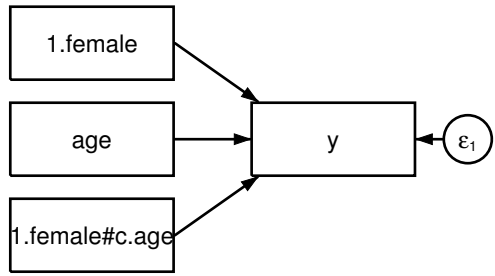
Let's say variable `skill` takes on the values 1, 2, and 3 meaning unskilled, skilled, and highly skilled. Then `3.skill` is an indicator variable for `skill` = 3. We could use `3.skill` in path diagrams or in the command language just as we previously used `1.x`.

Specifying interactions with indicator variables

We can specify interactions with `#`. Say we wish to fit the model

$$\texttt{y}_i = \beta_0 + \beta_1\ \texttt{female}_i + \beta_2\texttt{age}_i + \beta_3\ \texttt{female}_i \times \texttt{age}_i + \epsilon_i$$

We could do that by specifying `1.female`, `age`, and `1.female#c.age` in a path diagram:



We can use `1.female`, `age`, and `1.female#c.age` in command syntax, too:

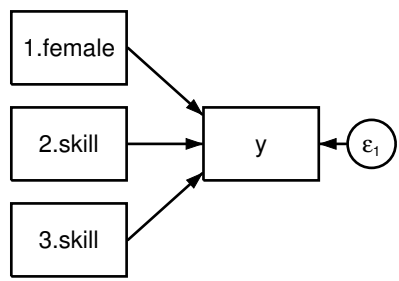
```
. gsem (y<-1.female age 1.female#c.age)
```

The `c` in `c.age` specifies that `age` is a continuous variable.

We can just as easily create interactions of indicator variables with other indicator variables. If variable `employed` is a 0/1 variable, then `1.female#1.employed` creates an indicator for employed female.

Specifying categorical variables

We use the same *#.varname* notation to handle categorical variables as we did for indicator variables. Consider the following model:



The same model would be specified in command syntax as

```
. gsem (y<-1.female 2.skill 3.skill)
```

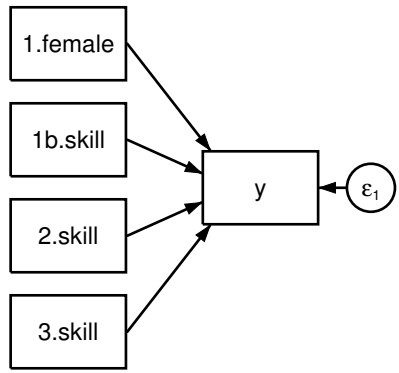
The equation for the model is

$$\texttt{y}_i = \beta_0 + \beta_1\,\texttt{1.female}_i + \beta_2\,\texttt{2.skill}_i + \beta_3\,\texttt{3.skill}_i + \epsilon_i$$

However we express the model, we are including indicator variables for skill level 2 and for skill level 3. We intentionally omit skill level 1.

Categorical variables require a base level, a level against which the other effects will be measured. We omitted `1.skill` to cause `1.skill` to become the base level. This results in β_2 measuring the difference in y between skill levels 2 and 1, and β_3 measuring the difference in y between skill levels 3 and 1.

Omission is one way to specify the base level. The other way is to include all the skill levels and indicate which one we want to be used as the base:



In command syntax, we type

```
. gsem (y<-1.female 1b.skill 2.skill 3.skill)
```

`b` specifies the base level. Specifying `1b.skill` makes skill level 1 the base. Had we wanted skill level 2 to be the base, we would have specified `2b.skill`.

Specification by omission is usually easier for simple models. For more complicated models, which might have paths from different skill levels going to different variables, it is usually better to specify all the skill levels and all the relevant paths, mark one of the skill levels as the base, and let `gsem` figure out the correct reduced-form model.

By the way, indicator variables are just a special case of categorical variables. Indicator variables are categorical variables with two levels. Everything just said about categorical variable `skill` applies equally to indicator variable `female`. We can specify `1.female` by itself, and thus implicitly specify that `female` $\neq 1$ is the base, or we can explicitly specify both `0b.female` and `1.female`.

The command language has a neat syntax for specifying all the indicator variables that can be manufactured from a categorical variable. When you type `i.skill`, it is the same as typing `1b.skill 2.skill 3.skill`. You can use the `i.` shorthand in the command language by typing

```
. gsem (y<-1.female i.skill)
```

or even by typing

```
. gsem (y<-i.female i.skill)
```

Most experienced Stata command-line users would type the model the second way, putting `i.` in front of both `skill` and `female`. They would not bother to think whether variables are indicator or categorical, nor would they concern themselves with remembering how the indicator and categorical variables are coded.

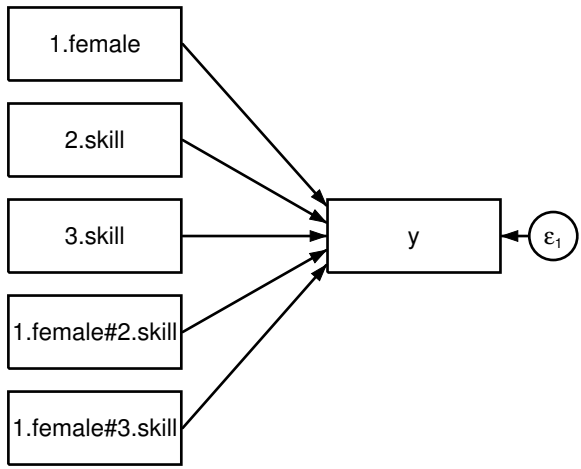
You cannot use the `i.` notation in path diagrams, however. Path diagrams allow only one variable per box. The `i.` notation produces at least two variables, and it usually produces a lot more.

Specifying interactions with categorical variables

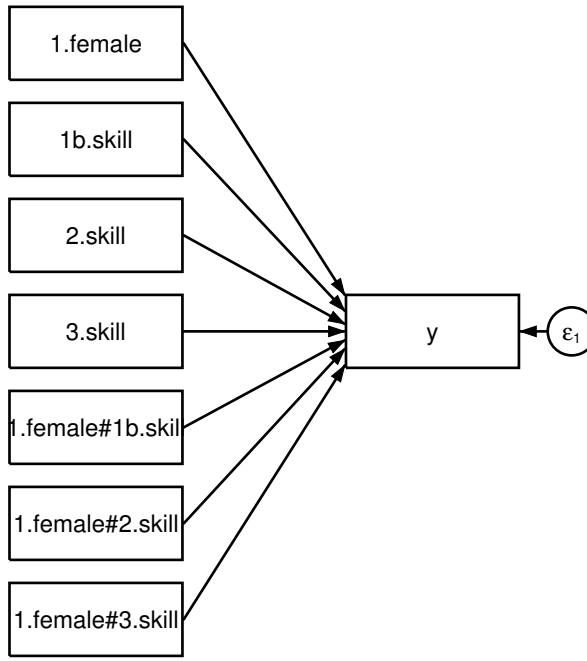
We just discussed the model `(y<-i.female i.skill)`. Obviously, the next step is to include an interaction between female and skill level. To write such a model, we use the same `#` interaction indicator that we have already discussed in the context of indicator variables: `(y<-i.female i.skill i.female#i.skill)`. We use the same notation because there is really no distinction between factor variables and indicator variables. We can specify the model exactly as shown by using the command syntax:

```
. gsem (y<-i.female i.skill i.female#i.skill)
```

In the Builder, we could diagram this model by specifying all the individual interactions but omitting the base levels,



or by including them,



In the second diagram, we did not bother to include the base levels for the indicator variable `female`, but we could have included them.

We can type the model in command syntax just as we have drawn it, either as

```
. gsem (y<-1.female
          2.skill  3.skill
          1.female#2.skill  1.female#3.skill)
```

or as

```
. gsem (y<-1.female
          1b.skill  2.skill  3.skill
          1.female#1b.skill  1.female#2.skill  1.female#3.skill)
```

But in command syntax, it is easiest to type

```
. gsem (y<-i.female i.skill i.female#i.skill)
```

Specifying endogenous variables

You may not use factor variables to specify endogenous variables except for the multinomial logistic regression model. Multinomial logistic regression is also known as mlogit and as being family multinomial, link logit.

In the mlogit case, you must specify a base level, which you can do implicitly by omission or explicitly by including a `b` on one of the levels. See [SEM] **Example 37g** and [SEM] **Example 41g** for examples.

Also see

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **Intro 4** — Substantive concepts

[SEM] **Builder** — SEM Builder

[SEM] **Builder, generalized** — SEM Builder for generalized models

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

Title

Intro 4 — Substantive concepts

Description Remarks and examples References Also see

Description

The structural equation modeling way of describing models is deceptively simple. It is deceptive because the machinery underlying structural equation modeling is sophisticated, complex, and sometimes temperamental, and it can be temperamental both in substantive statistical ways and in practical computer ways.

Professional researchers need to understand these issues.

Remarks and examples

Remarks are presented under the following headings:

Differences in assumptions between sem and gsem
- *sem: Choice of estimation method*
- *gsem: Choice of estimation method*

Treatment of missing values

Variable types: Observed, latent, endogenous, exogenous, and error

Constraining parameters
- *Constraining path coefficients to specific values*
- *Constraining intercepts to specific values (suppressing the intercept)*
- *Constraining path coefficients or intercepts to be equal*
- *Constraining covariances to be equal (or to specific values)*
- *Constraining variances to specific values (or to be equal)*

Identification 1: Substantive issues
- *Not all models are identified*
- *How to count parameters*
- *What happens when models are unidentified*
- *How to diagnose and fix the problem*

Identification 2: Normalization constraints (anchoring)
- *Why the problem arises*
- *How the problem would manifest itself*
- *How sem (gsem) solves the problem for you*
- *Overriding sem's (gsem's) solution*

Differences in assumptions between sem and gsem

sem fits standard linear SEMs.

gsem fits generalized SEMs, by which we mean

1. gsem fits not just linear but generalized linear models,
2. gsem fits multilevel mixed models,
3. gsem fits models with categorical latent variables,
4. gsem fits items 1 and 2 combined, and
5. gsem fits items 1 and 3 combined.

There is a difference in assumptions between standard linear SEMs and generalized SEMs.

Standard linear SEMs generally assume that the observed endogenous variables, the observed exogenous variables, the latent endogenous variables, and the latent exogenous variables are jointly distributed normally with mean μ and variance matrix $\mathbf{\Sigma}$. In this formulation, we include the $e.$ error variables among the latent exogenous variables.

Generalized SEMs drop the observed variables and categorical latent variables from the joint-normality assumption. Generalized SEMs treat the observed exogenous variables as given and produce estimates conditional on their values. This seems a minor difference, but for some researchers it has important implications.

Consider a researcher wishing to include a subject's age and age-squared among the observed exogenous variables of a model. Variables age and age-squared simply cannot be jointly normally distributed. Including age and age-squared violates the assumptions of standard linear SEMs, at least officially. You may now suspect that such researchers—perhaps including yourself—should use gsem and not sem. Because standard linear SEMs are a subset of generalized SEMs, that should not be much of an inconvenience.

It is interesting to note, however, that sem and gsem produce the same numeric solutions for the parameters and the standard errors when both can fit the same model.

All of which is to say that although it is typical to assume joint normality of all variables when deriving the standard linear SEM, joint normality is not strictly necessary. The lesser assumption of joint normality conditional on the observed exogenous variables is sufficient. Even the normality assumption can be relaxed and replaced with i.i.d., and even the i.i.d. assumption can be relaxed.

We will discuss these issues below.

Even so, some structural equation modeling statistics are dependent on the full joint normality assumption. Many goodness-of-fit tests fall into that category. Throughout this manual, we will be explicit about when the full joint-normality assumption is required.

gsem never requires the full joint-normality assumption. Standard linear SEMs have a history that includes the full joint-normality assumption; generalized SEMs have no such history. At the end of the day, both can be considered M estimation.

sem: Choice of estimation method

sem provides four estimation methods: maximum likelihood (ML; the default), quasimaximum likelihood (QML), asymptotic distribution free (ADF), and maximum likelihood with missing values (MLMV).

Strictly speaking, the assumptions one must make to establish the consistency of the estimates and their asymptotic normality is determined by the method used to estimate them. We want to give you advice on when to use each.

1. ML is the method that `sem` uses by default. In `sem`, the function being maximized formally assumes the full joint normality of all the variables, including the observed variables. But the full joint-normality assumption can be relaxed, and the substitute conditional-on-the-observed-exogenous-variables is sufficient to justify all reported estimates and statistics except the log-likelihood value and the model-versus-saturated χ^2 test.

   Relaxing the constraint that latent variables outside of the error variables are not normally distributed is more questionable. In the measurement model `(X->x1 x2 x3 x4)`, simulations with the violently nonnormal $X \sim \chi^2(2)$ produced good results except for the standard error of the estimated variance of X. Note that it was not the coefficient on X that was estimated poorly, it was not the coefficient's standard error, and it was not even the variance of X that was estimated poorly. It was the standard error of the variance of X. Even so, there are no guarantees.

   `sem` uses method ML when you specify `method(ml)` or when you omit the `method()` option altogether.

2. QML uses ML to fit the model parameters but relaxes the normality assumptions when estimating the standard errors. QML handles nonnormality by adjusting standard errors.

   Concerning the parameter estimates, everything just said about ML applies to QML because those estimates are produced by ML.

   Concerning standard errors, we theoretically expect consistent standard errors, and we practically observe that in our simulations. In the measurement model with $X \sim \chi^2(2)$, we even obtained good standard errors of the estimated variance of X. QML does not really fix the problem of nonnormality of latent variables, but it does tend to do a better job.

   `sem` uses method QML when you specify `method(ml) vce(robust)` or, because `method(ml)` is the default, when you specify just `vce(robust)`.

   When you specify `method(ml) vce(sbentler)` or just `vce(sbentler)`, the Satorra–Bentler scaled χ^2 test is reported. When observed data are nonnormal, the standard model-versus-saturated test statistic does not follow a χ^2 distribution. The Satorra–Bentler scaled χ^2 statistic uses a function of fourth-order moments to adjust the standard goodness-of-fit statistic so that it has a mean that more closely follows the reference χ^2 distribution. The corresponding robust standard errors, which are adjusted using a function of fourth-order moments, are reported as well. For details, see Satorra and Bentler (1994).

3. ADF makes no assumption of joint normality or even symmetry, whether for observed or latent variables. Whereas QML handles nonnormality by adjusting standard errors and not point estimates, ADF produces justifiable point estimates and standard errors under nonnormality.

   For many researchers, this is most important for relaxing the assumption of normality of the errors, and because of that, ADF is sometimes described that way. ADF in fact relaxes the normality assumption for all latent variables.

   Along the same lines, it is sometimes difficult to be certain exactly which normality assumptions are being relaxed when reading other sources. It sometimes seems that ADF uniquely relaxes the assumption of the normality of the observed variables, but that is not true. Other methods, even ML, can handle that problem.

   ADF is a form of weighted least squares (WLS). ADF is also a generalized method of moments (GMM) estimator. In simulations of the measurement model with $X \sim \chi^2(2)$, ADF produces

excellent results, even for the standard error of the variance of X. Be aware, however, that ADF is less efficient than ML when latent variables can be assumed to be normally distributed. If latent variables (including errors) are not normally distributed, on the other hand, ADF will produce more efficient estimates than ML or QML.

`sem` uses method ADF when you specify `method(adf)`.

4. MLMV aims to retrieve as much information as possible from observations containing missing values.

   In this regard, `sem` methods ML, QML, and ADF do a poor job. They are known as listwise deleters. If variable `x1` appears someplace in the model and if `x1` contains a missing value in observation 10, then observation 10 simply will not be used. This is true whether `x1` is endogenous or exogenous and even if `x1` appears in some equations but not in others.

   Method MLMV, on the other hand, is not a deleter at all. Observation 10 will be used in making all calculations.

   For method MLMV to perform what might seem like magic, joint normality of all variables is assumed and missing values are assumed to be missing at random (MAR). MAR means either that the missing values are scattered completely at random throughout the data or that values more likely to be missing than others can be predicted by the variables in the model.

   Method MLMV formally requires the assumption of joint normality of all variables, both observed and latent. If your observed variables do not follow a joint normal distribution, you may be better off using ML, QML, or ADF and simply omitting observations with missing values.

   `sem` uses method MLMV when you specify `method(mlmv)`. See [SEM] **Example 26**.

gsem: Choice of estimation method

`gsem` provides only two estimation methods: maximum likelihood (ML; the default) and quasi-maximum likelihood (QML).

1. ML is the method that `gsem` uses by default. This is the same ML used by `sem` but applied to a different likelihood function. The `sem` likelihood function assumes and includes joint normality of all variables. The `gsem` likelihood function assumes only conditional normality.

   Because likelihood functions are different, the likelihood values reported by `sem` and `gsem` are not comparable except in the case where there are no observed exogenous variables.

   ML formally assumes conditional normality, and thus continuous latent variables are still assumed to be normally distributed. What we said in the `sem` case about relaxing the assumption of normality of the latent variables applies equally in the `gsem` case.

   `gsem` uses method ML when you specify `method(ml)` or when you omit the `method()` option altogether.

2. QML uses ML to fit the model but relaxes the conditional normality assumptions when estimating the standard errors. QML handles nonnormality by adjusting standard errors.

   Everything said about `sem`'s QML applies equally to `gsem`'s QML.

   `gsem` uses method QML when you specify `vce(robust)`.

Because the choice of method often affects convergence with `sem`, in the `gsem` case there is a tendency to confuse choice of integration method with maximization method. However, there are no issues related to assumptions about integration method; choice of integration method is purely a mechanical issue. This is discussed in [SEM] **Intro 12**.

Treatment of missing values

sem, sem with method MLMV, and gsem treat missing values differently.

1. sem by default is a listwise deleter.

   If variable x1 appears in the model and if x1 contains missing in observation 10, then observation 10 will not be used. This is true whether x1 is endogenous or exogenous and even if x1 appears in some equations but not in others.

2. sem with method MLMV is not a deleter at all; it uses all observations.

   If variable x1 appears in the model and if x1 contains missing in observation 10, then observation 10 will still be used. Doing this formally requires assuming the joint normality of all observed variables and was discussed in item 4 of *sem: Choice of estimation method*.

3. gsem by default is an equationwise deleter for models without categorical latent variables.

   The abridged meaning is that gsem will often be able to use more observations from the data than sem will, assuming you do not use sem with method MLMV.

   The full meaning requires some setup. Consider a model of at least five equations. Assume that observed exogenous variable x1 appears in the first equation but not in equations 2–4; that equation 1 predicts y1; that y1 appears as a predictor in equations 2–4; and that x1 contains missing in observation 10.

   If endogenous variable y1 is latent or observed and of family Gaussian, link identity, but without censoring, then

   3.1 Observation 10 will be ignored in making calculations related to equation 1.

   3.2 Observation 10 will also be ignored in making calculations related to equations 2–4 because y1, a function of x1, appears in them.

   3.3 The calculations for the other equation(s) will include observation 10.

   Alternatively, if y1 is observed and not family Gaussian, link identity, or has censoring, then item 3.2 changes:

   3.2 Observation 10 will be used in making calculations related to equations 2–4 even though y1, a function of x1, appears in them.

   As we said at the outset, the result of all of this is that gsem often uses more observations than does sem (excluding method MLMV).

4. gsem has an option, listwise, that duplicates the sem rules. This is used in testing of Stata. There is no reason you would want to specify the option.

5. gsem uses listwise deletion for models with categorical latent variables.

Variable types: Observed, latent, endogenous, exogenous, and error

Structural equation models can contain four different types of variables:

1. observed exogenous
2. latent exogenous
3. observed endogenous
4. latent endogenous

As a software matter, it is useful to think as though there is a fifth type, too:

5. error

Errors are in fact a special case of latent exogenous variables, but there will be good reason to consider them separately.

As a language matter, it is sometimes useful to think of there being yet another type of variable, namely,

6. measure or measurement

Measurement variables are a special case of observed endogenous variables.

Let us explain:

Observed.
A variable is observed if it is a variable in your dataset. In this documentation, we often refer to observed variables with `x1`, `x2`, ..., `y1`, `y2`, and so on, but in reality observed variables have names such as `mpg`, `weight`, `testscore`, and so on.

Latent.
A variable is latent if it is not observed. A variable is latent if it is not in your dataset but you wish it were. You wish you had a variable recording the propensity to commit violent crime, or socioeconomic status, or happiness, or true ability, or even income. Sometimes, latent variables are imagined variants of real variables, variables that are somehow better, such as being measured without error. At the other end of the spectrum are latent variables that are not even conceptually measurable.

In this documentation, latent variables usually have names such as `L1`, `L2`, `F1`, ..., but in real life the names are more descriptive such as `VerbalAbility`, `SES`, and so on. The `sem` and `gsem` commands assume that variables are latent if the first letter of the name is capitalized, so we will always capitalize our latent variable names.

Endogenous.
A variable is endogenous (determined within the system) if any path points to it.

Exogenous.
A variable is exogenous (determined outside the system) if paths only originate from it or, equivalently, no path points to it.

Now that we have the above definitions, we can better understand the five types of variables:

1. *Observed exogenous.*
   A variable in your dataset that is treated as exogenous in your model.
2. *Latent exogenous.*
   An unobserved variable that is treated as exogenous in your model.
3. *Observed endogenous.*
   A variable in your dataset that is treated as endogenous in your model.
4. *Latent endogenous.*
   An unobserved variable that is treated as endogenous in your model.
5. *Error.*
   Mathematically, error variables are just latent exogenous variables. In `sem` and `gsem`, however, errors are different in that they have defaults different from the other latent exogenous variables.

   Errors are named `e`. So, for example, the error variable associated with observed endogenous variable `y1` has the full name `e.y1`; the error variable associated with latent endogenous variable `L1` has the full name `e.L1`.

In sem, each endogenous variable has a corresponding e. variable.

In gsem, observed endogenous variables associated with family Gaussian have corresponding e. variables, but other observed endogenous variables do not. All continuous latent endogenous variables have an associated e. variable, but that is not a special case because all continuous latent endogenous variables are assumed to be family Gaussian.

In the Builder, when you create an endogenous variable, the variable's corresponding error variable instantly springs into existence. The same happens in the command language, you just do not see it. In addition, error variables automatically and inalterably have their path coefficient constrained to be 1.

The covariances between the different variable types are given in the table below. The latent variables discussed here are continuous latent variables. In the table,

1. **0** means 0 and there are no options or secret tricks to change the value from 0.
2. **i** means as implied by your model and beyond your control.
3. (#) means to see the numbered note below the table.

If an entry in the matrix below is **0** or **i**, then you may not draw curved paths between variables of the specified types.

	Observed exogenous	Latent exogenous	Observed endogenous	Latent endogenous	Error
Observed exogenous	(1) & (2)				
Latent exogenous	(3)	(4) & (5)			
Observed endogenous	**i**	**i**	**i**		
Latent endogenous	**i**	**i**	**i**	**i**	
Error	**0**	**0**	**i**	**i**	(6) & (7)

1. Variances of observed exogenous variables:
   - 1.1 Builder, sem mode: Taken as given, assuming not using method MLMV. Can be estimated or constrained; if so, you are assuming normality of the observed exogenous variables unless using method ADF.
   - 1.2 sem command: Same as item 1.1. Use var() option to estimate or constrain.
   - 1.3 Builder, gsem mode: Taken as given. Cannot be estimated or constrained.
   - 1.4 gsem command: Same as item 1.3.
2. Covariances between observed exogenous variables:
   - 2.1 Builder, sem mode: Taken as given, assuming not using method MLMV. Can be estimated or constrained by drawing curved paths between variables; if so, you are assuming normality of the observed exogenous variables unless using method ADF.
   - 2.2 sem command: Same as item 2.1. Use cov() option to estimate or constrain.
   - 2.3 Builder, gsem mode: Taken as given. Cannot be estimated or constrained. Path may not be drawn.
   - 2.4 gsem command: Same as item 2.3.
3. Covariances between latent exogenous and observed exogenous variables:
   - 3.1 Builder, sem mode: Constrained to be 0 unless a curved path is drawn between variables, in which case it is estimated. Can be constrained.

3.2 `sem` command: Assumed to be nonzero and therefore estimated by default. Can be constrained (even to 0) using `cov()` option.

3.3 Builder, `gsem` mode: Assumed to be nonzero. Cannot be estimated or constrained because this covariance is not among the identified parameters of the generalized SEM.

3.4 `gsem` command: Same as item 3.3.

4. Variances of latent exogenous variables:

4.1 Builder, `sem` mode: Variances are estimated and can be constrained.

4.2 `sem` command: Variances are estimated and can be constrained using `var()` option.

4.3 Builder, `gsem` mode: Almost the same as item 4.1 except variances cannot be constrained to 0.

4.4 `gsem` command: Almost the same as item 4.2 except variances cannot be constrained to 0.

5. Covariances between latent exogenous variables:

5.1 Builder, `sem` mode: Assumed to be 0 unless a curved path is drawn between variables. Path may include constraints.

5.2 `sem` command: Assumed to be nonzero and estimated, the same as if a curved path without a constraint were drawn in the Builder. Can be constrained (even to 0) using `cov()` option.

5.3 Builder, `gsem` mode: Same as item 5.1.

5.4 `gsem` command: Same as item 5.2.

6. Variances of errors:

6.1 Builder, `sem` mode: Estimated. Can be constrained.

6.2 `sem` command: Estimated. Can be constrained using `var()` option.

6.3 Builder, `gsem` mode: Almost the same as item 6.1 except variances cannot be constrained to 0.

6.4 `gsem` command: Almost the same as item 6.2 except variances cannot be constrained to 0.

7. Covariances between errors:

7.1 Builder, `sem` mode: Assumed to be 0. Can be estimated by drawing curved paths between variables. Can be constrained.

7.2 `sem` command: Assumed to be 0. Can be estimated or constrained using `cov()` option.

7.3 Builder, `gsem` mode: Almost the same as item 7.1 except covariances between errors cannot be estimated or constrained if one or both of the error terms correspond to a generalized response with family Gaussian, link log, or link identity with censoring.

7.4 `gsem` command: Almost the same as item 7.2 except covariances between errors cannot be estimated or constrained if one or both of the error terms correspond to a generalized response with family Gaussian, link log, or link identity with censoring.

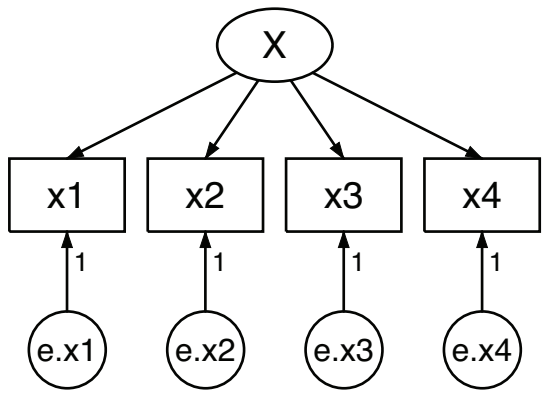
The properties of categorical latent variables differ from those of the continuous latent variables that we have discussed to this point. Categorical latent variables cannot be included in models with other types of latent variables. They do not covary by default. To specify a covariance, rather than using the `cov()` option, you add an intercept or a predictor for a cell of the interaction of the latent variables. For instance, you might type `2.C# 3.D <- _cons`. Categorical latent variables do not covary with any other types of variables in the model. Instead, parameters related to other variables are allowed to vary across the classes of the categorical latent variable.

Finally, there is a sixth variable type that we sometimes find convenient to talk about:

Measure or *measurement*.

A measure variable is an observed endogenous variable with a path from a latent variable. We introduce the word "measure" not as a computer term or even a formal modeling term but as a convenience for communication. It is a lot easier to say that `x1` is a measure of `X` than to say that `x1` is an observed endogenous variable with a path from latent variable `X` and so, in a real sense, `x1` is a measurement of `X`.

In our measurement model,



the variables are

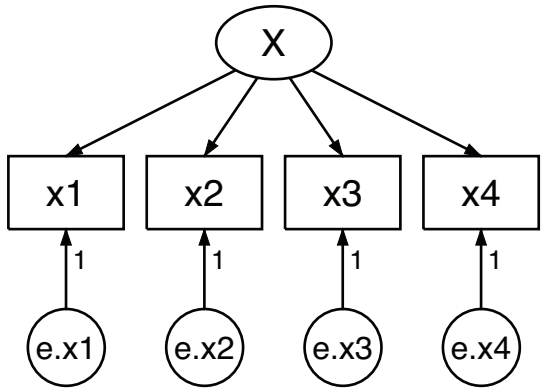
latent exogenous:	`X`
error:	`e.x1`, `e.x2`, `e.x3`, `e.x4`
observed endogenous:	`x1`, `x2`, `x3`, `x4`

All the observed endogenous variables in this model are measures of `X`.

Constraining parameters

Constraining path coefficients to specific values

If you wish to constrain a path coefficient to a specific value, you just write the value next to the path. In our measurement model without correlation of the residuals,



we indicate that the coefficients `e.x1`, ..., `e.x4` are constrained to be 1 by placing a small 1 along the path. We can similarly constrain any path in the model.

If we wanted to constrain $\beta_2 = 1$ in the equation

$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

we would write a 1 along the path between X and x_2. If we were instead using `sem`'s or `gsem`'s command language, we would write

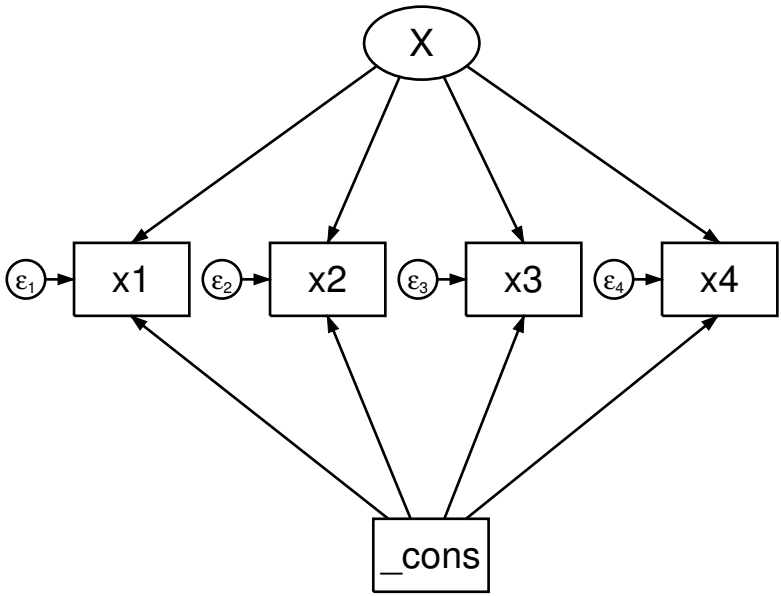
```
(x1<-X) (x2<-X@1) (x3<-X) (x4<-X)
```

That is, you type an `@` symbol immediately after the variable whose coefficient is being constrained, and then you type the value.

Constraining intercepts to specific values (suppressing the intercept)

Constraining path coefficients is common. Constraining intercepts is less so, and usually when the situation arises, you wish to constrain the intercept to 0, which is often called "suppressing the intercept".

Although it is unusual to draw the paths corresponding to intercepts in path diagrams, they are assumed, and you could draw them if you wish. A more explicit version of our path diagram for the measurement model is



The path coefficient of `_cons` to `x1` corresponds to α_1 in

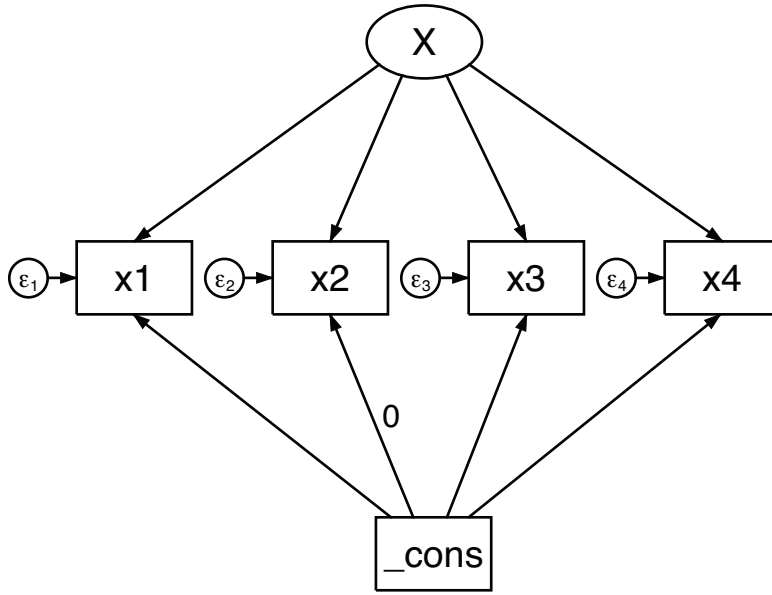
$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$

and the path coefficient of `_cons` to `x2` corresponds to α_2 in

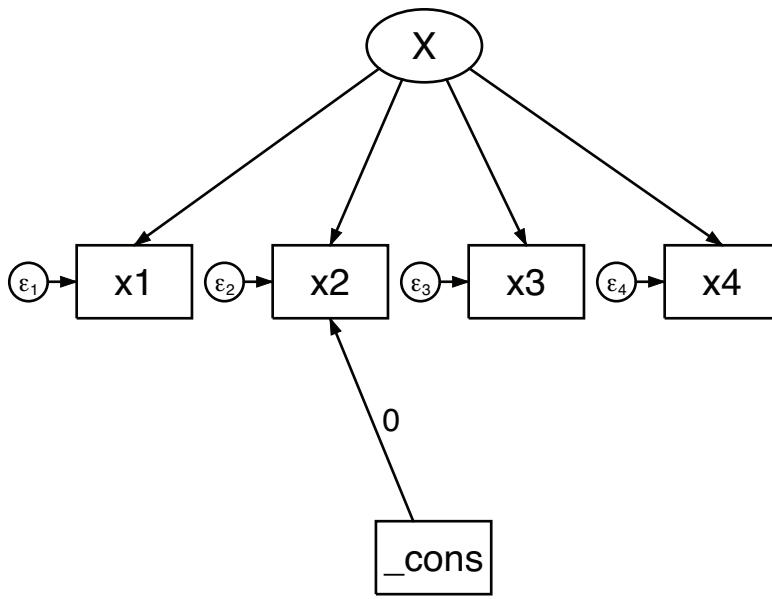
$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

and so on.

Obviously, if you wanted to constrain a particular intercept to a particular value, you would write the value along the path. To constrain $\alpha_2 = 0$, you could draw



Because intercepts are assumed, you could omit drawing the paths from `_cons` to `x1`, `_cons` to `x3`, and `_cons` to `x4`:



Just as with the Builder, the command language assumes paths from _cons to all endogenous variables, but you could type them if you wished:

```
(x1<-X _cons) (x2<-X _cons) (x3<-X _cons) (x4<-X _cons)
```

If you wanted to constrain $\alpha_2 = 0$, you could type

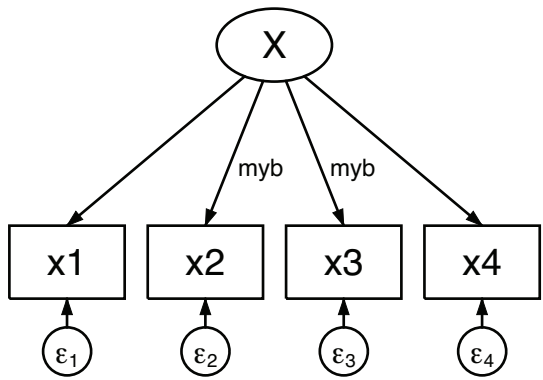
```
(x1<-X _cons) (x2<-X _cons@0) (x3<-X _cons) (x4<-X _cons)
```

or you could type

```
(x1<-X) (x2<-X _cons@0) (x3<-X) (x4<-X)
```

Constraining path coefficients or intercepts to be equal

If you wish to constrain two or more path coefficients to be equal, place a symbolic name along the relevant paths:



In the diagram above, we constrain $\beta_2 = \beta_3$ because we stated that $\beta_2 =$ myb and $\beta_3 =$ myb.

You follow the same approach in the command language:

```
(x1<-X) (x2<-X@myb) (x3<-X@myb) (x4<-X)
```

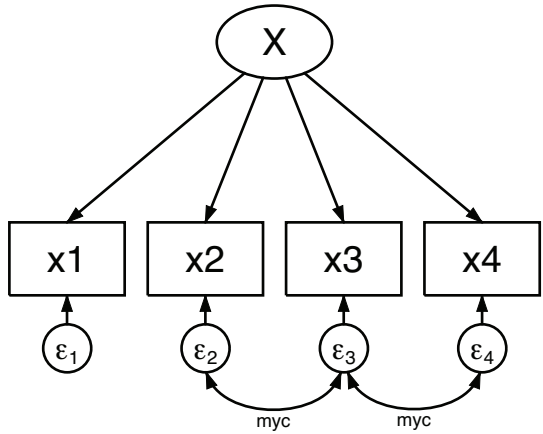
This works the same way with intercepts. Intercepts are just paths from _cons, so to constrain intercepts to be equal, you add symbolic names to their paths. In the command language, you constrain $\alpha_1 = \alpha_2$ by typing

```
(x1<-X _cons@c) (x2<-X _cons@c) (x3<-X) (x4<-X)
```

See [SEM] **Example 8**.

Constraining covariances to be equal (or to specific values)

If you wish to constrain covariances, usually you will want to constrain them to be equal instead of to a specific value. If we wanted to fit our measurement model and allow correlation between e.x2 and e.x3 and between e.x3 and e.x4, and we wanted to constrain the covariances to be equal, we could draw



If you instead wanted to constrain the covariances to specific values, you would place the value along the paths in place of the symbolic names.

In the command language, covariances (curved paths) are specified using the `cov()` option. To allow covariances between `e.x2` and `e.x3` and between `e.x3` and `e.x4`, you would type

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X), cov(e.x2*e.x3) cov(e.x3*e.x4)
```

To constrain the covariances to be equal, you would type

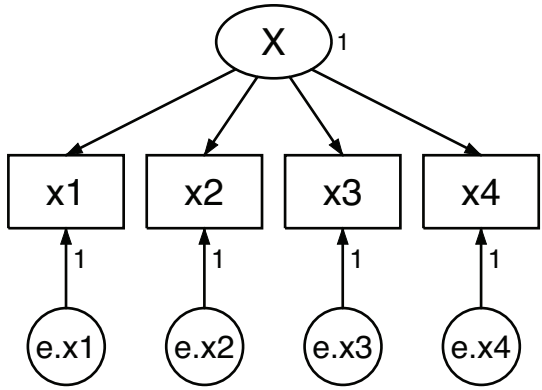
```
(x1<-X) (x2<-X) (x3<-X) (x4<-X), cov(e.x2*e.x3@myc) cov(e.x3*e.x4@myc)
```

Constraining variances to specific values (or to be equal)

Variances are like covariances except that in path diagrams drawn by some authors, variances curve back on themselves. In the Builder, variances appear inside or beside the box or circle. Regardless of how they appear, variances may be constrained to normalize latent variables, although normalization is handled by `sem` and `gsem` automatically (something we will explain in *How sem (gsem) solves the problem for you* under *Identification 2: Normalization constraints (anchoring)* below).

In the Builder, you constrain variances by clicking on the variable and using the lock box to specify the value, which can be a number or a symbol. In the command language, variances are specified using the `var()` option as we will explain below.

Let's assume that you want to normalize the latent variable `X` by constraining its variances to be 1. You do that by drawing



In the command language, we specify this model as

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X), var(X@1)
```

Constraining latent exogenous variables to have unit variance as an identifying restriction may be desirable when you wish simultaneously to constrain their correlations with other latent exogenous variables. `sem` and `gsem` allow you to constrain covariances, not correlations. Covariances are equal to correlations when variances are 1.

In more complicated models, you may wish to constrain variances of latent exogenous variables to be equal. You can do that by specifying a symbolic name.

Identification 1: Substantive issues

Not all models are identified

Just because you can draw the path diagram for a model, write its equations, or write it in Stata's command syntax, does not mean the model is identified. Identification refers to the conceptual constraints on parameters of a model that are required for the model's remaining parameters to have a unique solution. A model is said to be unidentified if these constraints are not supplied. These constraints are of two types: substantive constraints and normalization constraints. We will begin by discussing substantive constraints because that is your responsibility; the software provides normalization constraints automatically.

How to count parameters

If you are fitting a model with `sem` and your model has K observed variables, then your data contain $K(K+1)/2$ second-order moments, and thus p, the number of parameters based on second-order moments that can be estimated, cannot exceed $K(K+1)/2$.

Every path in your model contributes 1 to p unless the parameter is constrained to a specific value, and then it does not contribute at all. If two parameters are constrained to be equal, the two parameters count as one. In counting p, you must remember to count the curved paths from latent variables back to themselves, which is to say, the variances. Just counting the number of parameters can be challenging. And even if $p \leq K(K+1)/2$, your model may not be identified. Identification depends not only on the number of paths but also on their locations.

Counting parameters can be even more difficult in the case of certain generalized linear (gsem) models. For a discussion of this, see Skrondal and Rabe-Hesketh (2004, chap. 5).

Even in the non-gsem case, books have been written on this subject, and we will refer you to them. A few are Bollen (1989), Brown (2015), Kline (2016), and Kenny (1979). We will refer you to them, but do not be surprised if they refer you back to us. Brown (2015, 179) writes, "Because latent variable software programs are capable of evaluating whether a given model is identified, it is often most practical to simply try to estimate the solution and let the computer determine the model's identification status." That is not bad advice.

What happens when models are unidentified

So what happens when you attempt to fit an unidentified model? In some cases, sem (gsem) will tell you that your model is unidentified. If your model is unidentified for subtle substantive reasons, however, you will see

```
initial values not feasible
r(1400);
```

or

```
Iteration 50:   log likelihood = -337504.44  (not concave)
Iteration 51:   log likelihood = -337504.44  (not concave)
Iteration 52:   log likelihood = -337504.44  (not concave)
.
.
.
Iteration 101:   log likelihood = -337504.44  (not concave)
.
.
.
```

In the latter case, sem (gsem) will iterate forever, reporting the same criterion value (such as log likelihood) and saying "not concave" over and over again.

Observing periods of the "not concave" message is not concerning, so do not overreact at the first occurrence. Become concerned when you see "not concave" and the criterion value is not changing, and even then, stay calm for a short time because the value might be changing in digits you are not seeing. If the iteration log continues to report the same value several times, however, press *Break*. Your model is probably not identified.

How to diagnose and fix the problem

You must find and fix the problem.

If you get the "initial values not feasible" message, your goal is to find feasible initial values and thereby either solve the problem or convert it into the infinitely long iteration log problem; see [SEM] **Intro 12**.

If you get the infinitely long log, rerun the model specifying sem's (gsem's) iterate(#) option, where # is large enough to reach the "not concave" messages with constant criterion value. sem (gsem) will iterate that many times and then report the results it has at that point. Look at the output for missing standard errors. Those parameters are unidentified, and you need to think about either changing your model so that they become identified or placing constraints on them.

Identification 2: Normalization constraints (anchoring)

Normalization constraints (anchoring) are provided automatically by `sem` and `gsem`. It is rare indeed that the automatically provided constraints will be the source of convergence problems.

Models with continuous latent variables require normalization constraints because these latent variables have no natural scale. If constraints were not provided, the model would appear to the software the same as a model with a substantive lack of identification; the estimation routine would iterate forever and never arrive at a solution.

The `sem` and `gsem` commands automatically provide normalization constraints to prevent that from happening.

Below we explain why the normalization constraints are required, which normalization constraints `sem` and `gsem` automatically supply, how to override those automatic choices, and how to substitute your own constraints should you desire.

Why the problem arises

Imagine a latent variable for propensity to be violent. Your imagination might supply a scale that ranges from 0 to 1 or 1 to 100 or over other values, but regardless, the scale you imagine is arbitrary in that one scale works as well as another.

Scales have two components: mean and variance. If you imagine a latent variable with mean 0 and your colleague imagines the same variable with mean 100, the difference can be accommodated in the parameter estimates by an intercept changing by 100. If you imagine a standard deviation of 1 (variance $1^2 = 1$) and your colleague imagines a standard deviation of 10 (variance $10^2 = 100$), the difference can be accommodated by a path coefficient differing by a multiplicative factor of 10. You might measure an effect as being 1.1 and then your colleague would measure the same effect as being 0.11, but either way you both will come to the same substantive conclusions.

How the problem would manifest itself

The problem is that different scales all work equally well, and the software will iterate forever, jumping from one scale to another.

Another way of saying that the means and variances of latent variables are arbitrary is to say that they are unidentified. That's important because if you do not specify the scale you have in mind, results of estimation will look just like substantive lack of identification.

`sem` (`gsem`) will iterate forever and never arrive at a solution.

How sem (gsem) solves the problem for you

You usually do not need to worry about this problem because `sem` (`gsem`) solves it for you. `sem` (`gsem`) solves the unidentified scale problem for continuous latent variables by

1. Assuming that latent exogenous variables have mean 0.
2. Assuming that latent endogenous variables have intercept 0.
3. Setting the coefficients on paths from latent variables to the first observed endogenous variable to be 1.
4. Setting the coefficients on paths from latent variables to the first latent endogenous variable to be 1 if rule 3 does not apply—if the latent variable is measured by other latent variables only.

Rules 3 and 4 are also known as the unit-loading rules. The variable to which the path coefficient is set to 1 is said to be the anchor for the latent variable.

Applying those rules to our measurement model, when we type

```
(X->x1) (X->x2) (X->x3) (X->x4)
```

sem (gsem) acts as if we typed

```
(X@1->x1) (X->x2) (X->x3) (X->x4), means(X@0)
```

The above four rules are sufficient to provide a scale for latent variables for all models.

Overriding sem's (gsem's) solution

sem (gsem) automatically applies rules 1 through 4 to produce normalization constraints. There are, however, other normalization constraints that would work as well. In what follows, we will assume that you are well versed in deriving normalization constraints and just want to know how to bend sem (gsem) to your will.

Before you do this, however, let us warn you that substituting your normalization rules for the defaults can result in more iterations being required to fit your model. Yes, one set of normalization constraints are as good as the next, but sem's (gsem)'s starting values are based on its default normalization rules, which means that when you substitute your rules for the defaults, the required number of iterations sometimes increases.

Let's return to the measurement model:

```
(X->x1) (X->x2) (X->x3) (X->x4)
```

As we said previously, type the above and sem (gsem) acts as if you typed

```
(X@1->x1) (X->x2) (X->x3) (X->x4), means(X@0)
```

If you wanted to assume instead that the mean of X is 100, you could type

```
(X->x1) (X->x2) (X->x3) (X->x4), means(X@100)
```

The means() option allows you to specify mean constraints, and you may do so for latent or observed variables.

Let's leave the mean at 0 and specify that we instead want to constrain the second path coefficient to be 1:

```
(X->x1) (X@1->x2) (X->x3) (X->x4)
```

We did not have to tell sem (gsem) not to constrain X->x1 to have coefficient 1. We just specified that we wanted to constrain X->x2 to have coefficient 1. sem (gsem) takes all the constraints that you specify and then adds whatever normalization constraints are needed to identify the model. If what you have specified is sufficient, sem (gsem) does not add its constraints to yours.

Obviously, if we wanted to constrain the mean to be 100 and the second rather than the first path coefficient to be 1, we would type

```
(X->x1) (X@1->x2) (X->x3) (X->x4), means(X@100)
```

If we wanted to constrain the standard deviation of X to be 10 instead of constraining a path coefficient to be 1, we would type

```
(X->x1) (X->x2) (X->x3) (X->x4), means(X@100) var(X@100)
```

Standard deviations are specified in variance units when you use the var() option.

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Brown, T. A. 2015. *Confirmatory Factor Analysis for Applied Research*. 2nd ed. New York: Guilford Press.

Kenny, D. A. 1979. *Correlation and Causality*. New York: Wiley.

Kline, R. B. 2016. *Principles and Practice of Structural Equation Modeling*. 4th ed. New York: Guilford Press.

Li, C. 2013. Little's test of missing completely at random. *Stata Journal* 13: 795–809.

Satorra, A., and P. M. Bentler. 1994. Corrections to test statistics and standard errors in covariance structure analysis. In *Latent Variables Analysis: Applications for Developmental Research*, ed. A. von Eye and C. C. Clogg, 399–419. Thousand Oaks, CA: SAGE.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Also see

[SEM] **Intro 3** — Learning the language: Factor-variable notation (gsem only)

[SEM] **Intro 5** — Tour of models

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **sem and gsem option covstructure( )** — Specifying covariance restrictions

Title

Intro 5 — Tour of models

Description Remarks and examples References Also see

Description

Below is a sampling of SEMs that can be fit by `sem` or `gsem`.

Remarks and examples

If you have not read [SEM] **Intro 2**, please do so. You need to speak the language. We also recommend reading [SEM] **Intro 4**, but that is not required.

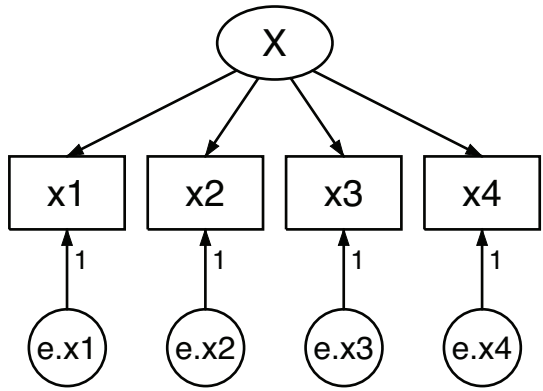
Now that you speak the language, we can start all over again and take a look at some of the classic models that `sem` and `gsem` can fit.

Remarks are presented under the following headings:

Single-factor measurement models
Item response theory (IRT) models
Multiple-factor measurement models
Confirmatory factor analysis (CFA) models
Structural models 1: Linear regression
Structural models 2: Gamma regression
Structural models 3: Binary-outcome models
Structural models 4: Count models
Structural models 5: Ordinal models
Structural models 6: Multinomial logistic regression
Structural models 7: Survival models
Structural models 8: Dependencies between response variables
Structural models 9: Unobserved inputs, outputs, or both
Structural models 10: MIMIC models
Structural models 11: Seemingly unrelated regression (SUR)
Structural models 12: Multivariate regression
Structural models 13: Mediation models
Correlations
Higher-order CFA models
Correlated uniqueness model
Latent growth models
Models with reliability
Multilevel mixed-effects models
Latent class models
Finite mixture models

Single-factor measurement models

A single-factor measurement model is



The model can be written in Stata command language as

```
(x1<-X) (x2<-X) (x3<-X) (x4<-X)
```

or as

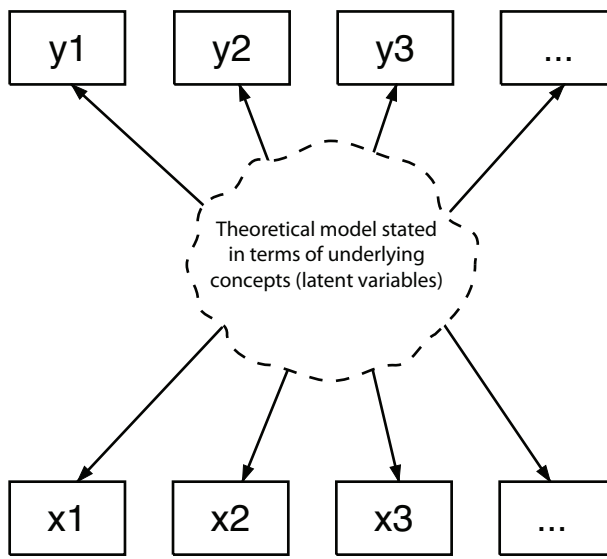
```
(x1 x2 x3 x4<-X)
```

or as

```
(X->x1 x2 x3 x4)
```

or in other ways. All the equivalent ways really are equivalent; no subtle differences will subsequently arise according to your choice.

The measurement model plays an important role in many other SEMs dealing with the observed inputs and the observed outputs:



Because the measurement model is so often joined with other models, it is common to refer to the coefficients on the paths from latent variables to observable endogenous variables as the measurement coefficients and to refer to their intercepts as the measurement intercepts. The intercepts are usually not shown in path diagrams. The other coefficients and intercepts are those not related to the measurement issue.

The measurement coefficients are often referred to as loadings.

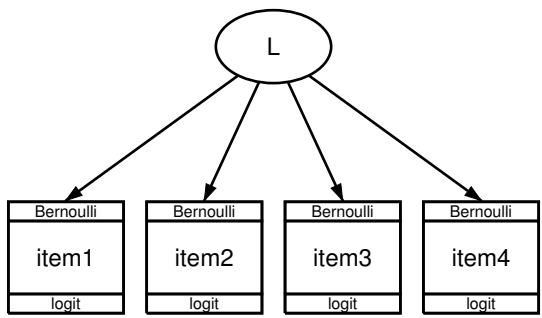
This model can be fit by `sem` or `gsem`. Use `sem` for standard linear models (standard means single level); use `gsem` when you are fitting a multilevel model or when the response variables are generalized linear such as probit, logit, multinomial logit, Poisson, and so on.

See the following examples:

1. [SEM] **Example 1**. Single-factor measurement model.
2. [SEM] **Example 27g**. Single-factor measurement model (generalized response).
3. [SEM] **Example 30g**. Two-level measurement model (multilevel, generalized response).
4. [SEM] **Example 35g**. Ordered probit and ordered logit.

Item response theory (IRT) models

Item response theory (IRT) models look like the following:



The items are the observed variables, and each has a 0/1 outcome measuring a latent variable. Often, the latent variable represents ability. These days, it is traditional to fit IRT models using logistic regression, but in the past, probit was used and they were called normal ogive models.

In one-parameter logistic models, also known as 1-PL models and Rasch models, constraints are placed on the paths and perhaps the variance of the latent variable. Either path coefficients are constrained to 1 or path coefficients are constrained to be equal and the variance of the latent variable is constrained to be 1. Either way, this results in the negative of the intercepts of the fitted model being a measure of difficulty.

1-PL and Rasch models can be fit treating the latent variable—ability—as either fixed or random. Abilities are treated as random with `gsem`.

In two-parameter logistic models (2-PL), no constraints are imposed beyond the one required to identify the latent variable, which is usually done by constraining the variance to 1. This results in path coefficients measuring discriminating ability of the items, and difficulty is measured by the negative of the intercepts divided by the corresponding (slope) coefficient.

IRT has been extended beyond 1-PL and 2-PL models, including extension to other types of generalized responses.

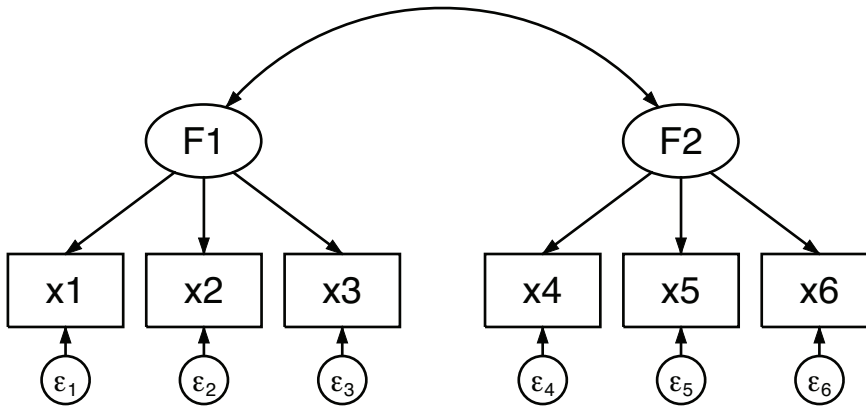
IRT models, including the extensions, can be fit by `gsem`.

See the following examples:

1. [SEM] **Example 28g**. One-parameter logistic IRT (Rasch) model.
2. [SEM] **Example 29g**. Two-parameter logistic IRT model.

Multiple-factor measurement models

A two-factor measurement model is two one-factor measurement models with possible correlation between the factors:



To obtain a correlation between `F1` and `F2`, we drew a curved path.

The model can be written in Stata command language as

```
(F1->x1) (F1->x2) (F1->x3) (F2->x4) (F2->x5) (F2->x6)
```

In the command language, we do not have to include the `cov(F1*F2)` option because, by default, `sem` assumes that exogenous latent variables are correlated with each other. This model can also be written in any of the following ways:

```
(F1->x1 x2 x3) (F2->x4 x5 x6)
```

or

```
(x1 x2 x3<-F1) (x4 x5 x6<-F2)
```

or

```
(x1<-F1) (x2<-F1) (x3<-F1) (x4<-F2) (x5<-F2) (x6<-F2)
```

Two-factor measurement models can be fit by `sem` and `gsem`.

See the following examples:

1. [SEM] **Example 3**. Two-factor measurement model.
2. [SEM] **Example 31g**. Two-factor measurement model (generalized response).

Confirmatory factor analysis (CFA) models

The measurement models just shown are also known as confirmatory factor analysis (CFA) models because they can be analyzed using CFA.

In the single-factor model, after estimation, you might want to test that all the indicators have significant loadings by using `test`; see [SEM] **test**. You might also want to test whether the correlations between the errors should have been included in the model by using `estat mindices`; see [SEM] **estat mindices**.

In the multiple-factor measurement model, you might want to test that any of the omitted paths should in fact be included in the model. The omitted paths in the two-factor measurement model above were `F1` → `x4`, `F1` → `x5`, `F1` → `x6`, and `F2` → `x1`, `F2` → `x2`, `F2` → `x3`. `estat mindices` will perform these tests.

These tests are available after `sem` only. CFA is just a measurement model and can be fit by both `sem` and `gsem`.

We show other types of CFA models below.

See the following example:

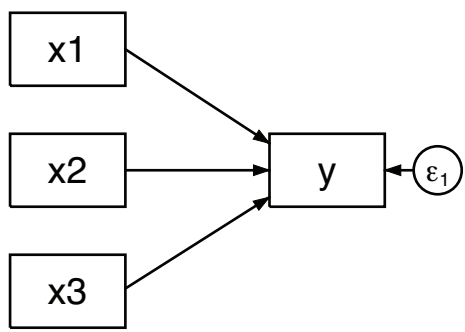
1. [SEM] **Example 5**. Modification indices.

Structural models 1: Linear regression

Different authors define the meaning of structural models in different ways. Bollen (1989, 4) defines a structural model as the parameters being not of a descriptive nature of association but instead of a causal nature. By that definition, the measurement models above could be structural models, and so could the linear regression below.

Others define structural models as models having paths reflecting causal dependencies between endogenous variables, which would thus exclude the measurement model and linear regression. We will show you a "true" structural model in the next example.

An example of a linear regression is



This model can be written in Stata command language as

```
(y<-x1 x2 x3)
```

When you estimate a linear regression by using `sem`, you obtain the same point estimates as you would with `regress` and the same standard errors up to a degree-of-freedom adjustment applied by `regress`.

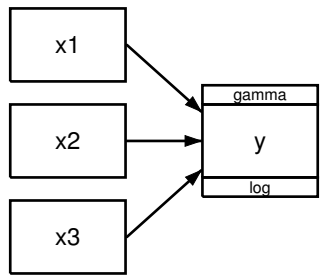
Linear regression models can be fit by `sem` and `gsem`. `gsem` also has options for censoring.

See the following examples:

1. [SEM] **Example 6**. Linear regression.
2. [SEM] **Example 38g**. Random-intercept and random-slope models (multilevel).
3. [SEM] **Example 40g**. Crossed models (multilevel).
4. [SEM] **Example 43g**. Tobit regression.
5. [SEM] **Example 44g**. Interval regression.

Structural models 2: Gamma regression

Gamma regression, also known as log-gamma regression, is used when a continuous outcome is nonnegative, when it ranges from zero to infinity, and often with positively skewed data. It is appropriate when the error variance can be assumed to increase with the mean. Gamma regression gives similar results to linear regression with a logged dependent variable.



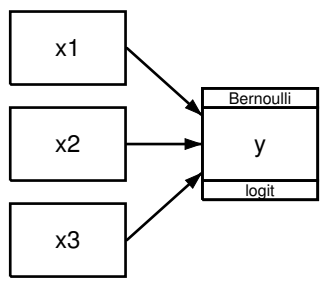
Gamma regression is fit by `gsem`; specify shorthand `gamma` or specify `family(gamma) link(log)`.

You can fit exponential regressions using gamma regression if you constrain the log of the scale parameter to be 0; see [SEM] **gsem family-and-link options**.

Structural models 3: Binary-outcome models

Binary-outcome models have 0/1 response variables. These models include logistic regression (also known as logit), probit, and complementary log-log (also known as clog-log) models.

A simple logistic regression model is



which in command syntax can be written as

```
(y<-x1 x2 x3, logit)
```

For the other binary-outcome models, all that changes in the diagram is the names of the family and link; in the command language, the option name changes; see [SEM] **gsem family-and-link options**.

Usually, the observations in binary-outcome data record whether the event occurred, but the data can instead record the number of events and the number of trials by changing the family from Bernoulli to binomial; see [SEM] **gsem family-and-link options**.

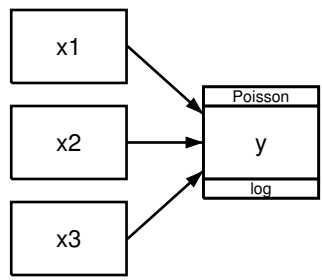
Binary-outcome models can be fit by `gsem`.

See the following examples:

1. [SEM] **Example 33g**. Logistic regression.
2. [SEM] **Example 27g**. Single-factor measurement model (generalized response).
3. [SEM] **Example 34g**. Combined models (generalized responses).

Structural models 4: Count models

Count models have response variables that are counts of things or events. These models include Poisson and negative binomial models. A simple example of a Poisson model is



which in command syntax can be written as

```
(y<-x1 x2 x3, poisson)
```

For negative binomial models, in path diagrams, the family changes to nbreg and, in the command syntax, the option changes to `nbreg`.

In Poisson models, both the mean and the variance are determined by a single parameter: the rate at which the event occurs. One use of negative binomial models is to handle overdispersion, which is when the variance is greater than the variance that would be predicted by a Poisson model.

The way we have diagrammed the model and written the command, we are assuming that each observation was at risk for the same length of time. If observations varied in this, the command would be

```
(y<-x1 x2 x3, poisson exposure(etime))
```

where variable `etime` records each observation's exposure time. The diagram would not change, but we would nonetheless enter the exposure time into the Builder. See [SEM] **gsem family-and-link options**.

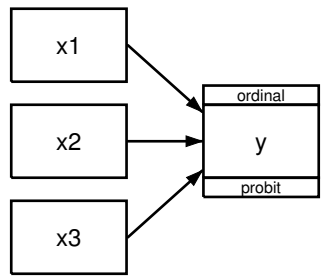
Count models are fit by `gsem`.

See the following examples:

1. [SEM] **Example 34g**. Combined models (generalized responses).
2. [SEM] **Example 39g**. Three-level model (multilevel, generalized response).

Structural models 5: Ordinal models

Ordinal models have two or more possible outcomes that are ordered, such as responses of the form "a little", "average", and "a lot"; or "strongly disagree", "disagree", . . . , "strongly agree". The outcomes are usually numbered 1, 2, . . . , k. These models include ordered probit, ordered logit, and ordered complementary log-log (also known as ordered clog-log). A simple example of an ordered probit model is



which in command syntax can be written as

```
(y<-x1 x2 x3, oprobit)
```

All that changes for the other models is the link name that appears in the path diagram or the option that appears in the command language.

Ordinal models are fit by `gsem`.

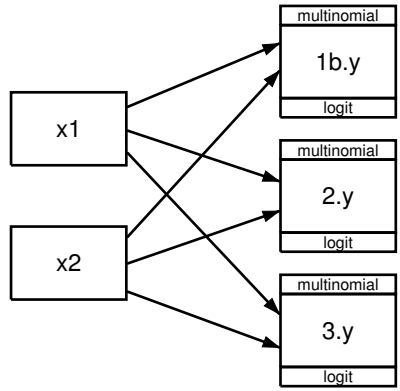
See the following examples:

1. [SEM] **Example 35g**. Ordered probit and ordered logit.
2. [SEM] **Example 31g**. Two-factor measurement model (generalized response).
3. [SEM] **Example 32g**. Full structural equation model (generalized response).
4. [SEM] **Example 36g**. MIMIC model (generalized response).

Structural models 6: Multinomial logistic regression

Multinomial logistic regression, also known as multinomial logit, is similar to ordinal models in that it, too, deals with multiple responses; however, in multinomial logistic regression, the responses cannot be ordered. Examples of multinomial logit response include method of transportation (car, public transportation, etc.) or ice-cream flavor (vanilla, chocolate, etc.).

Just as with ordinal models, the outcome is usually recorded as a single variable containing 1, 2, . . . , k, but in path diagrams, factor-variable notation is used to identify the outcomes:



In command syntax, this can be written as

```
(i.y<-x1 x2, mlogit)
```

In the example above, we have paths from all predictor variables to all outcomes. That is common but not required.

The multinomial logistic regression model is fit by `gsem`.

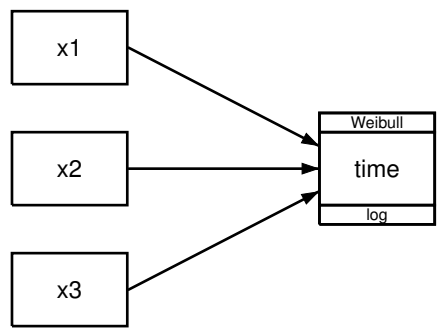
See the following examples:

1. [SEM] **Example 37g**. Multinomial logistic regression.
2. [SEM] **Example 41g**. Two-level multinomial logistic regression (multilevel).

Structural models 7: Survival models

Survival models are fit to response variables that measure survival times. The data may also contain observations for which a failure is not observed. For those observations, our response variable represents the time until the observation is censored.

Parametric survival models may be fit using a variety of distributions. A simple example of a Weibull survival model is



which in command syntax can be written as

```
(time<-x1 x2 x3, family(weibull))
```

If some of the observations were censored, the command would be

```
(time<-x1 x2 x3, family(weibull, failure(fail)))
```

where fail is an indicator variable coded as 1 for observations that failed and 0 for observations that were censored. The diagram would not change, but we would nonetheless specify fail as the failure indicator by using the Builder.

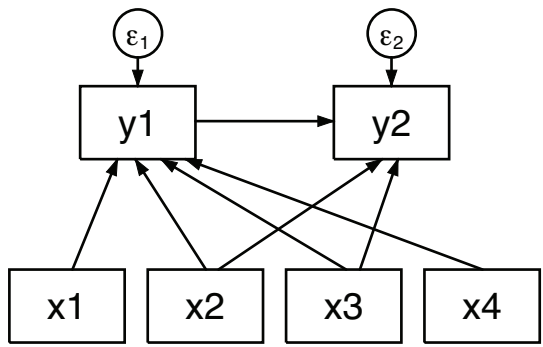
`gsem` fits parametric survival models using exponential, Weibull, gamma, loglogistic, and lognormal distributions. See [SEM] **gsem family-and-link options** for details of all options available when fitting survival models, including those for specifying the survival distribution and censoring.

See the following examples:

1. [SEM] **Example 47g**. Exponential survival model.
2. [SEM] **Example 48g**. Loglogistic survival model with censored and truncated data.
3. [SEM] **Example 49g**. Multiple-group Weibull survival model.

Structural models 8: Dependencies between response variables

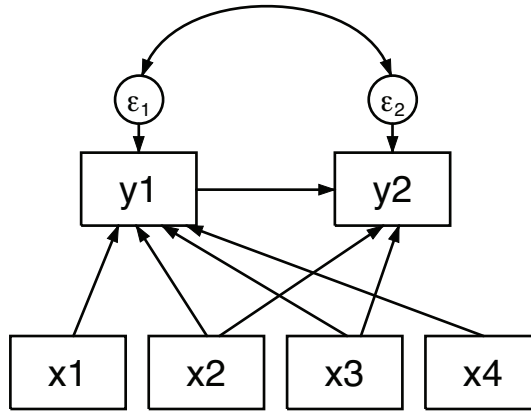
An example of a structural model having paths between response (endogenous) variables is



This model can be written in Stata command language as

```
(y1<-x1 x2 x3 x4) (y2<-y1 x2 x3)
```

In this example, all inputs and outputs are observed and the errors are assumed to be uncorrelated. In these kinds of models, it is common to allow correlation between errors:



The model above can be written in Stata command language as

```
(y1<-x1 x2 x3 x4) (y2<-y1 x2 x3), cov(e.y1*e.y2)
```

This structural model is said to be overidentified. If we omitted `y1` ← `x4`, the model would be just-identified. If we also omitted `y1` ← `x1`, the model would be unidentified.

When you fit the above model using `sem`, you obtain slightly different results from those you would obtain with `ivregress liml`. This is because `sem` with default `method(ml)` produces full-information maximum likelihood rather than limited-information maximum likelihood results.

Analysis of models for observed variables that include dependencies between endogenous variables may also be referred to as path analysis. Acock (2013, chap. 2) discusses path analysis with `sem` in more detail.

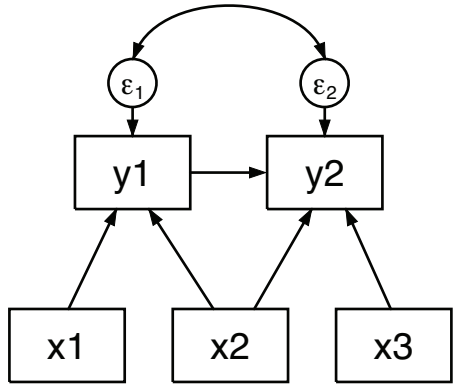
These models can be fit by `sem` or `gsem`. When using `gsem` to fit models with generalized response variables, non-Gaussian responses and Gaussian responses with the log link, or censoring, can only be included in recursive portions of models.

See the following examples:

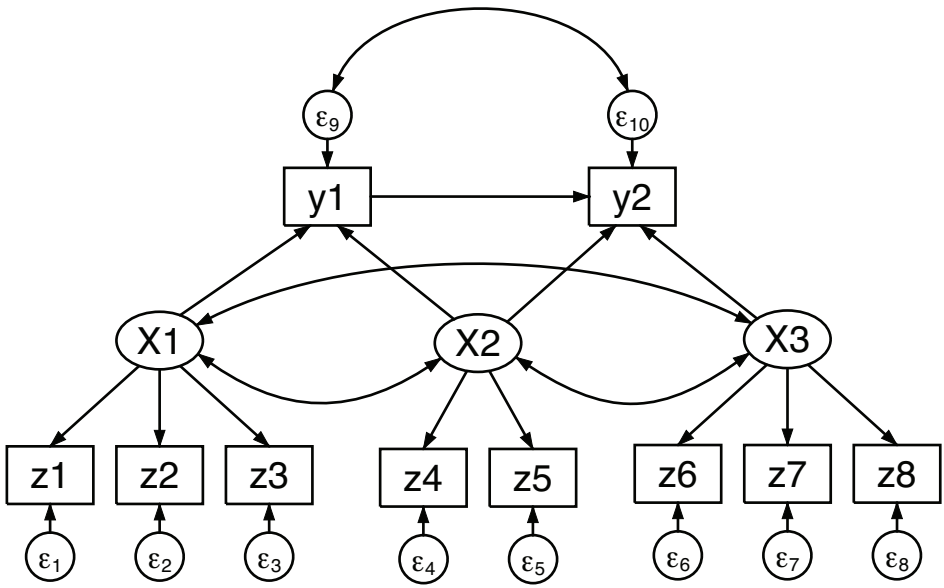
1. [SEM] **Example 7**. Nonrecursive structural model.
2. [SEM] **Example 34g**. Combined models (generalized responses).
3. [SEM] **Example 42g**. One- and two-level mediation models (multilevel).
4. [SEM] **Example 43g**. Tobit regression.
5. [SEM] **Example 44g**. Interval regression.
6. [SEM] **Example 45g**. Heckman selection model.
7. [SEM] **Example 46g**. Endogenous treatment-effects model.

Structural models 9: Unobserved inputs, outputs, or both

Perhaps in a structural model such as



the inputs `x1`, `x2`, and `x3` are concepts and thus are not observed. Assume that we have measurements for them. We can join this structural model example with a three-factor measurement model:



Note the curved arrows denoting correlation between the pairs of `X1`, `X2`, and `X3`. In the previous path diagram, we had no such arrows between the variables, yet we were still assuming that they were there. In `sem`'s path diagrams, correlation between exogenous observed variables is assumed and need not be explicitly shown. When we changed observed variables `x1`, `x2`, and `x3` to be the latent variables `X1`, `X2`, and `X3`, we needed to show explicitly the correlations we were allowing. Correlation between latent variables is not assumed and must be shown.

This model can be written in Stata command syntax as follows:

```
(y1<-X1 X2) (y2<-y1 X2 X3)    ///
(X1->z1 z2 z3)                ///
(X2->z4 z5)                   ///
(X3->z6 z7 z8),               ///
          cov(e.y1*e.y2)
```

We did not include the `cov(X1*X2 X1*X3 X2*X3)` option, although we could have. In the command language, exogenous latent variables are assumed to be correlated with each other. If we did not want `X2` and `X3` to be correlated, we would need to include the `cov(X2*X3@0)` option.

We changed `x1`, `x2`, and `x3` to be `X1`, `X2`, and `X3`. In command syntax, variables beginning with a capital letter are assumed to be latent. Alternatively, we could have left the names in lowercase and specified the identities of the latent variables:

```
(y1<-x1 x2) (y2<-y1 x2 x3)    ///
(x1->z1 z2 z3)                ///
(x2->z4 z5)                   ///
(x3->z6 z7 z8),               ///
          cov(e.y1*e.y2)      ///
          latent(x1 x2 x3)
```

Just as we have joined an observed structural model to a measurement model to handle unobserved inputs, we could join the above model to a measurement model to handle unobserved `y1` and `y2`.

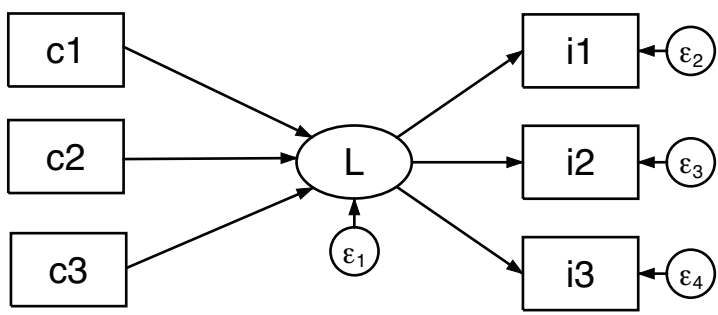
Models with unobserved inputs, outputs, or both can be fit by `sem` and `gsem`.

See the following examples:

1. [SEM] **Example 9**. Structural model with measurement component.
2. [SEM] **Example 32g**. Full structural equation model (generalized response).
4. [SEM] **Example 45g**. Heckman selection model.
5. [SEM] **Example 46g**. Endogenous treatment-effects model.

Structural models 10: MIMIC models

MIMIC stands for multiple indicators and multiple causes. An example of a MIMIC model is



In this model, the observed causes `c1`, `c2`, and `c3` determine latent variable `L`, and `L` in turn determines the observed indicators `i1`, `i2`, and `i3`.

This model can be written in Stata command syntax as

```
(i1 i2 i3<-L) (L<-c1 c2 c3)
```

MIMIC models can be fit by `sem` and `gsem`.

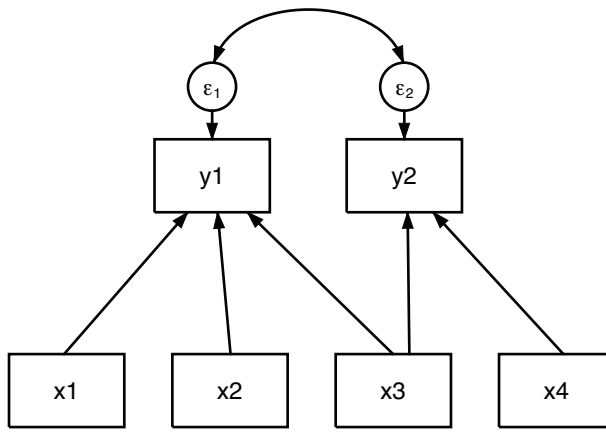
See the following examples:

1. [SEM] **Example 10**. MIMIC model.
2. [SEM] **Example 36g**. MIMIC model (generalized response).

Structural models 11: Seemingly unrelated regression (SUR)

Seemingly unrelated regression (SUR) is like having two or more separate linear regressions but allowing the errors to be correlated.

An example of an SUR model is



The model above can be written in Stata command syntax as

```
(y1<-x1 x2 x3) (y2<-x3 x4), cov(e.y1*e.y2)
```

In this example, the two regressions share a common exogenous variable, `x3`. They do not have to share a common variable, or they could share more variables. If they shared all variables, results would be the same as estimating multivariate regression, shown in the next example.

When you estimate an SUR with `sem`, you obtain the same point estimates as you would with `sureg` if you specify `sureg`'s `isure` option, which causes `sureg` to iterate until it obtains the maximum likelihood result. Standard errors will be different. If the model has exogenous variables only on the right-hand side, then standard errors will be asymptotically identical and, although the standard errors are different in finite samples, there is no reason to prefer one set over the other. If the model being fit is recursive, standard errors produced by `sem` are better than those from `sureg`, both asymptotically and in finite samples.

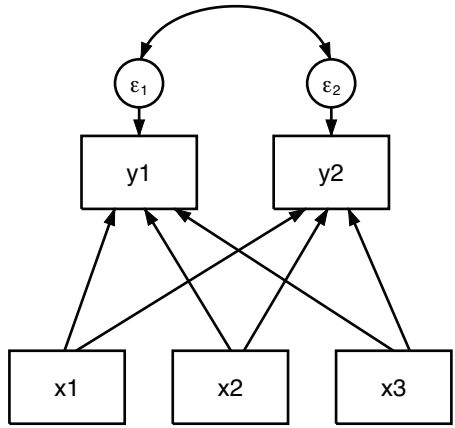
SUR models can be fit by `sem` and `gsem`. `gsem` will allow you to generalize the model to multilevel. In cases other than family Gaussian, link identity, you can sometimes introduce latent variables to create a similar effect.

See the following example:

1. [SEM] **Example 12**. Seemingly unrelated regression.

Structural models 12: Multivariate regression

A multivariate regression is just an SUR where the different dependent variables share the same exogenous variables:



The model above can be written in Stata command syntax as

```
(y1 y2<-x1 x2 x3), cov(e.y1*e.y2)
```

When you estimate a multivariate regression with `sem`, you obtain the same point estimates as you would with `mvreg` and the same standard errors up to a multiplicative $\sqrt{(N-p-1)/N}$ degree-of-freedom adjustment applied by `mvreg`.

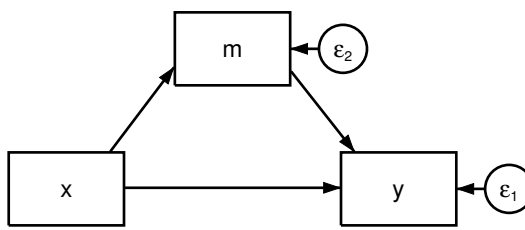
Multivariate regression can be fit by `sem` and `gsem`. `gsem` will allow you to generalize the model to multilevel. In cases other than family Gaussian, link identity, you can sometimes introduce latent variables to create a similar effect.

See the following example:

1. [SEM] **Example 12**. Seemingly unrelated regression.

Structural models 13: Mediation models

Mediation models concern effects that pass through (are mediated by) other variables. Consider response variable y affected by x mediated through m:



This can also be specified in command syntax as

```
(y<-m x) (m<-x)
```

In this simple model, x has a direct effect on y and an indirect (mediated through m) effect. The direct effect may be reasonable given the situation, or it may be included just so one can test whether the direct effect is present. If both the direct and indirect effects are significant, the effect of x is said to be partially mediated through m.

There are one-level mediation models and various two-level models, and lots of other variations, too.

`sem` and `gsem` can both fit one-level linear models, but you will be better off using `sem`. `gsem` can fit one-level generalized linear models and fit two-level (and higher) models, generalized linear or not.

See the following example:

1. [SEM] **Example 42g**. One- and two-level mediation models (multilevel).

Correlations

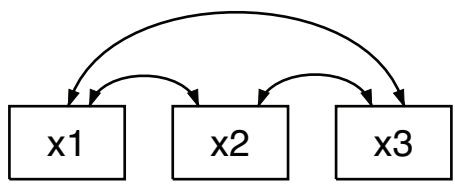
We are all familiar with correlation matrices of observed variables, such as

	x1	x2	x3
x1	1.0000		
x2	0.7700	1.0000	
x3	−0.0177	−0.2229	1.0000

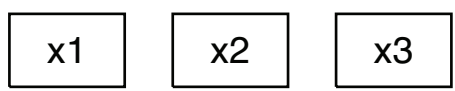
or covariances matrices, such as

	x1	x2	x3
x1	662.172		
x2	62.5157	9.95558	
x3	−0.769312	−1.19118	2.86775

These results can be obtained from `sem`. The path diagram for the model is



We could just as well leave off the curved paths because `sem` assumes them among observed exogenous variables:



Either way, this model can be written in Stata command syntax as

```
(<- x1 x2 x3)
```

That is, we simply omit specifying the target of the path, the endogenous variable.

If we fit the model, we will obtain the covariance matrix by default. `correlate` with the `covariance` option produces covariances that are divided by $N-1$ rather than by N. To match this covariance exactly, you need to specify the `nm1` option, which we can do in the command language by typing

```
(<- x1 x2 x3), nm1
```

If we want correlations rather than covariances, we ask for them by specifying the `standardized` option:

```
(<- x1 x2 x3), nm1 standardized
```

An advantage of obtaining correlation matrices from `sem` rather than from `correlate` is that you can perform statistical tests on the results, such as that the correlation of `x1` and `x3` is equal to the correlation of `x2` and `x3`.

If you are willing to assume joint normality of the variables, you can obtain more efficient estimates of the correlations in the presence of missing-at-random data by specifying the `method(mlmv)` option.

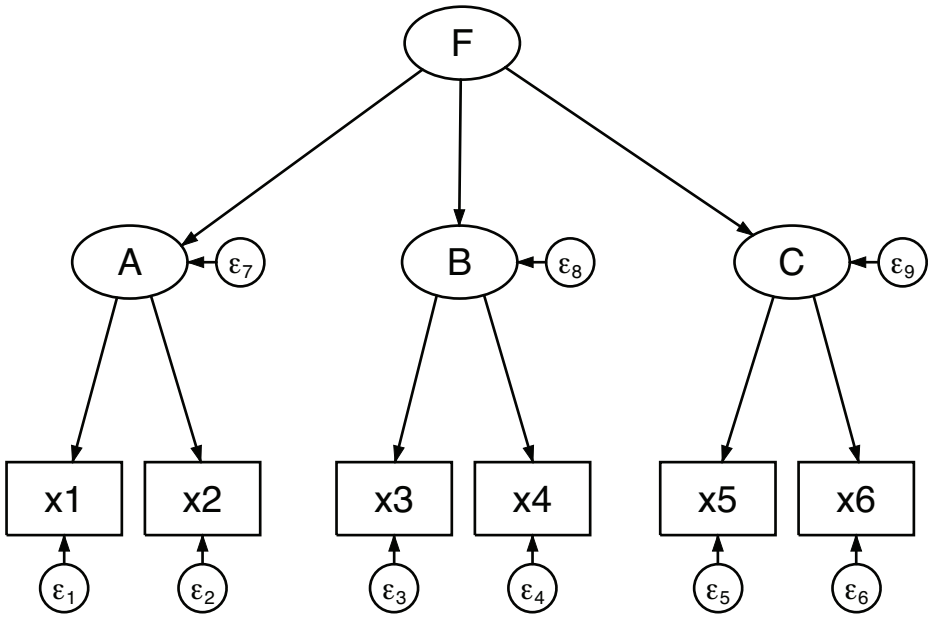
Correlations are fit using `sem`.

See the following example:

1. [SEM] **Example 16**. Correlation.

Higher-order CFA models

Observed values sometimes measure traits or other aspects of latent variables, so we insert a new layer of latent variables to reflect those traits or aspects. We have measurements—say, `x1`, . . . , `x6`—all reflecting underlying factor `F`, but `x1` and `x2` measure one trait of `F`, `x3` and `x4` measure another trait, and `x5` and `x6` measure yet another trait. This model can be drawn as



The model can be written in command syntax as

```
(A->x1 x2) (B->x3 x4) (C->x5 x6) (A B C<-F)
```

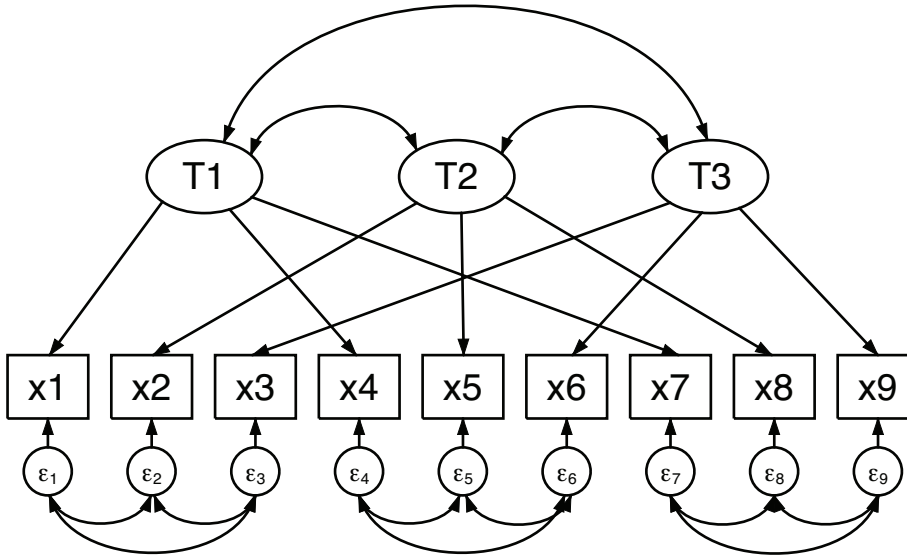
Higher-order CFA models can be fit using `sem` or `gsem`.

See the following example:

1. [SEM] **Example 15**. Higher-order CFA.

Correlated uniqueness model

Observed values sometimes are correlated just because of how the data are collected. Imagine we have factor `T1` representing a trait with measurements `x1` and `x4`. Perhaps `T1` is aggression, and then `x1` is self reported and `x4` is reported by the spouse. Imagine we also have factor `T2` with measurements `x2` and `x5`. Again, `x2` is self reported and `x5` is reported by the spouse. It would not be unlikely that `x1` and `x2` are correlated and that `x4` and `x5` are correlated. That is exactly what the correlated uniqueness model assumes:



Data that exhibit this kind of pattern are known as multitrait–multimethod (MTMM) data. Researchers historically looked at the correlations, but structural equation modeling allows us to fit a model that incorporates the correlations.

The above model can be written in Stata command syntax as

```
(T1->x1 x4 x7)                                          ///
(T2->x2 x5 x8)                                          ///
(T3->x3 x6 x9),                                         ///
            cov(e.x1*e.x2 e.x1*e.x3 e.x2*e.x3)          ///
            cov(e.x4*e.x5 e.x4*e.x6 e.x5*e.x6)          ///
            cov(e.x7*e*x8 e.x7*e.x9 e.x8*e.x9)
```

An alternative way to type the above is to use the `covstructure()` option, which we can abbreviate as `covstruct()`:

```
(T1->x1 x4 x7)                                           ///
(T2->x2 x5 x8)                                           ///
(T3->x3 x6 x9),                                          ///
            covstruct(e.x1 e.x2 e.x3, unstructured)      ///
            covstruct(e.x4 e.x5 e.x6, unstructured)      ///
            covstruct(e.x7 e.x8 e.x9, unstructured)
```

Unstructured means that the listed variables have covariances. Specifying blocks of errors as unstructured would save typing if there were more variables in each block.

The correlated uniqueness model can be fit by `sem` or `gsem`, although we recommend use of `sem` in this case. Gaussian responses with the identity link are allowed to have correlated uniqueness (error) but only in the absence of censoring. `gsem` still provides the theoretical ability to fit these models in multilevel contexts, but convergence may be difficult to achieve.

See the following example:

1. [SEM] **Example 17**. Correlated uniqueness model.

Latent growth models

A latent growth model is a variation on the measurement model. In our measurement model examples, we have assumed four observed measurements of underlying factor `X`: `x1`, `x2`, `x3`, and `x4`. In the command language, we can write this as

```
(X->x1) (X->x2) (X->x3) (X->x4)
```

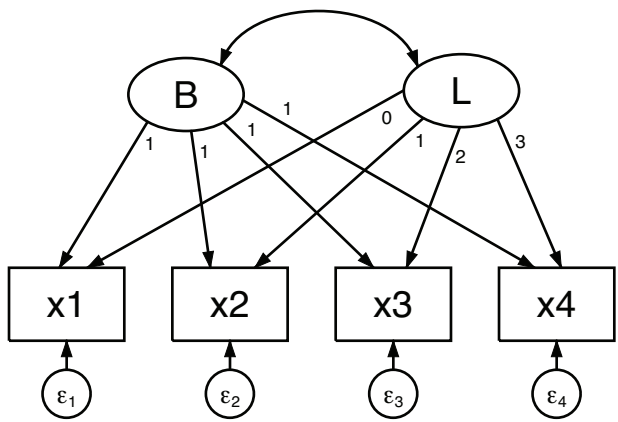
Let's assume that the observed values are collected over time. `x1` is observed at time 0, `x2` at time 1, and so on. It thus may be more reasonable to assume that the observed values represent a base value and a growth modeled with a linear trend. Thus we might write the model as

```
(B@1 L@0->x1)                ///
(B@1 L@1->x2)                ///
(B@1 L@2->x3)                ///
(B@1 L@3->x4),               ///
            noconstant
```

The equations for this model are

$$
\begin{aligned}
x_1 &= B + 0L + e.x_1 \\
x_2 &= B + 1L + e.x_2 \\
x_3 &= B + 2L + e.x_3 \\
x_4 &= B + 3L + e.x_4
\end{aligned}
$$

and the path diagram is



In evaluating this model, it is useful to review the means of the latent exogenous variables. In most models, latent exogenous variables have mean 0, and the means are thus uninteresting. `sem` usually constrains latent exogenous variables to have mean 0 and does not report that fact.

In this case, however, we ourselves have placed constraints, and thus the means are identified and in fact are an important point of the exercise. We must tell `sem` not to constrain the means of the two latent exogenous variables `B` and `L`, which we do with the `means()` option:

```
(B@1 L@0->x1)                ///
(B@1 L@1->x2)                ///
(B@1 L@2->x3)                ///
(B@1 L@3->x4),               ///
            noconstant means(B L)
```

We must similarly specify the `means()` option when using the Builder.

Latent growth models can be fit with `sem` or `gsem`.

See the following example:

1. [SEM] **Example 18**. Latent growth model.

Models with reliability

A typical solution for dealing with variables measured with error is to find multiple measurements and use those measurements to develop a latent variable. See, for example, *Single-factor measurement models* and *Multiple-factor measurement models* above.

When the reliability of the variables is known—reliability is measured as the fraction of variances that is not due to measurement error—another approach is available. This approach can be used in place of or in addition to the use of multiple measurements. See [SEM] **sem and gsem option reliability( )**.

Models with reliability can be fit with `sem` or `gsem`, although option `reliability()` is available only when responses are Gaussian with the identity link and only in the absence of censoring.

See the following example:

1. [SEM] **Example 24**. Reliability.

Multilevel mixed-effects models

Multilevel modeling concerns the inclusion of common, random effects across groups of the data, which are known as levels. You have observations on students. Students attend schools. Or you have observations on patients. Patients are served by hospitals. In either case, there may be an unmeasured effect of the institution. Some schools are better than others while some schools are worse. The same applies to hospitals. These unmeasured effects can be parameterized as a latent variable that is constant within institution and varies across institution.

What we have just described is a two-level nested model. The first level, the lowest, is the observational level. The second level is school or hospital.

In a three-level nested model, students are served by schools which are served by counties, or patients are served by hospitals which are served by states. Whatever the example, there can be unmeasured effects of each of those higher levels contributing to the effect observed at the observational level.

An alternative to nested models is crossed models. People with jobs work in an industry and in a state. The same industries are found across states and the same states are found across industries, yet both may have an effect on some aspect of the lives of their workers.

Multilevel models are fit by `gsem`.

See the following examples:

1. [SEM] **Example 30g**. Two-level measurement model (multilevel, generalized response).
2. [SEM] **Example 38g**. Random-intercept and random-slope models (multilevel).
3. [SEM] **Example 39g**. Three-level model (multilevel, generalized response).
4. [SEM] **Example 40g**. Crossed models (multilevel).
5. [SEM] **Example 41g**. Two-level multinomial logistic regression (multilevel).
6. [SEM] **Example 42g**. One- and two-level mediation models (multilevel).

Latent class models

A latent class model involves a latent variable that is categorical rather than continuous. The unobserved levels of the categorical latent variable are called latent classes. These classes correspond to unobserved groups in the population such as unobserved groups of healthy and unhealthy individuals or unobserved groups of consumers with different buying preferences. Latent class analysis can help us to identify and understand these groups.

When we fit a latent class model, we specify the number of latent classes and then estimate the probabilities of class membership. In addition, the parameters that are estimated in the rest of the model are allowed to vary across the classes. For example, we may fit intercept-only logistic regression models to a set of binary variables and allow the intercepts to be estimated separately across classes.

Latent class models are fit by `gsem`. The Stata command syntax for this type of model is

```
(y1 y2 y3 y4 <- ), logit lclass(C 2)
```

This fits a latent class model with one categorical latent variable, `C`, that has two classes. Both the name of the latent variable and the number of classes is specified in the `lclass()` option.

This basic latent class model can be extended in many ways.

1. We can specify that `C` has three, four, or even more latent classes.
2. We are not limited to having observed variables that are categorical. When variables are continuous, the analysis is sometimes called by other names such as latent profile analysis or latent cluster analysis instead of latent class analysis.
3. We can include predictors of `C`—predictors of the probabilities of being in the different classes.
4. We can include more than one categorical latent variable.

See the following examples:

1. [SEM] **Example 50g**. Latent class model.
2. [SEM] **Example 51g**. Latent class goodness-of-fit statistics.
3. [SEM] **Example 52g**. Latent profile model.

Finite mixture models

Finite mixture models also involve categorical latent variables. Here we focus on finite mixture regression models in which you can fit any regression model allowed by `gsem` and estimate the parameters of that model separately for each latent class. For a linear regression of `y` on `x1` and `x2`, the command syntax for a two-class model is

```
(y <- x1 x2), lclass(C 2)
```

The intercept and the coefficients on `x1` and `x2` will be estimated separately for the two classes. In addition, we estimate the probability of being in each class. If we have a variable `z` that predicts class membership, the command syntax becomes

```
(y <- x1 x2) (C <- z), lclass(C 2)
```

The `fmm:` prefix can also be used to fit these finite mixture regression models. For instance, `fmm: regress` can fit the model shown above. `gsem` extends the types of models that can be fit using `fmm` by allowing more than one response variable or more than one categorical latent variable.

See the following examples:

1. [SEM] **Example 53g**. Finite mixture Poisson regression.
2. [SEM] **Example 54g**. Finite mixture Poisson regression, multiple responses.

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Bartus, T. 2017. Multilevel multiprocess modeling with gsem. *Stata Journal* 17: 442–461.

Bauldry, S. 2014. miivfind: A command for identifying model-implied instrumental variables for structural equation models in Stata. *Stata Journal* 14: 60–75.

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Palmer, T. M., and J. A. C. Sterne. 2015. Fitting fixed- and random-effects meta-analysis models using structural equation modeling with the sem and gsem commands. *Stata Journal* 15: 645–671.

Also see

[SEM] **Intro 4** — Substantive concepts

[SEM] **Intro 6** — Comparing groups

[SEM] **Example 1** — Single-factor measurement model

Title

Intro 6 — Comparing groups

Description Remarks and examples Reference Also see

Description

`sem` and `gsem` have a unique feature that allows you to easily compare groups—compare males with females, compare age group 1 with age group 2 with age group 3, and so on—with respect to any SEM. Said more technically, any model fit by `sem` or `gsem` can be simultaneously estimated for different groups with some parameters constrained to be equal across groups and others allowed to vary, and those estimates can be used to perform statistical tests for comparing the groups.

`sem` and `gsem` have similar syntax for multiple group models, but there are important differences in default assumptions and in how you specify constraints. So we discuss how to compare groups using each command separately, first with `sem` and then with `gsem`.

Remarks and examples

Remarks are presented under the following headings:

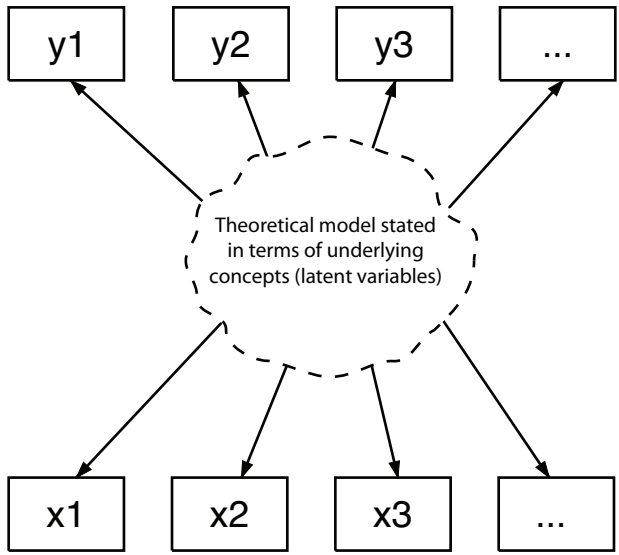
Comparing groups with sem
The generic SEM model
sem: Fitting the model for different groups of the data
sem: Which parameters vary by default, and which do not
sem: Specifying which parameters are allowed to vary in broad, sweeping terms
sem: Adding constraints for path coefficients across groups
sem: Adding constraints for means, variances, or covariances across groups
sem: Adding constraints for some groups but not others
sem: Adding paths for some groups but not others
sem: Relaxing constraints
Comparing groups with gsem
gsem: Fitting the model for different groups of the data
gsem: Which parameters vary by default, and which do not
gsem: Specifying which parameters are allowed to vary in broad, sweeping terms
gsem: Adding constraints for path coefficients across groups
gsem: Adding constraints for means, variances, or covariances across groups
gsem: Adding constraints for some groups but not others
gsem: Adding paths for some groups but not others
gsem: Relaxing constraints

Comparing groups with sem

`sem` and `gsem` have similar syntax for multiple group models. There are, however, important differences in default assumptions and in how you specify constraints, so we discuss how to compare groups using each command separately. We begin with `sem`'s syntax for group comparisons.

The generic SEM model

In [SEM] **Intro 5**, we noted that measurement models are often joined with other SEMs to produce



This can be written in the `sem` command syntax as

```
(Y1->...) (Y2->...)      ///
(...)                    /// theoretical model stated in terms of
(...)                    /// underlying concepts (latent variables)
(...)                    ///
(X1->...) (X2->...)
```

where the middle part is the theoretical model stated in terms of underlying concepts `Y1`, `Y2`, `X1`, and `X2`.

However we write the model, we are assuming the following:

1. The unobserved `X1` and `X2` are measured by the observed `x1`, `x2`, ...
2. The middle part is stated in terms of the underlying concepts `X1`, `X2`, `Y1`, and `Y2`.
3. The unobserved `Y1` and `Y2` are measured by the observed `y1`, `y2`, ...

sem: Fitting the model for different groups of the data

We can fit this model for different groups (say, age groups) by specifying the `group(`*varname*`)` option:

```
(Y1->...) (Y2->...)                    ///   part 3
(...)                                  ///
(...)                                  ///   part 2
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1
                  group(agegrp)
```

where agegrp is a variable in our dataset, perhaps taking on values 1, 2, 3, ... We can specify the model by using the command language or by drawing the model in the Builder and then choosing and filling in the group() option.

After estimation, you can use estat ginvariant (see [SEM] **estat ginvariant**) to obtain Wald tests of whether constraints should be added and score tests of whether constraints should be relaxed.

sem: Which parameters vary by default, and which do not

When we specify group(*groupvar*), the measurement parts of the model (parts 1 and 3) are constrained by default to be the same across the groups, whereas the middle part (part 2) will have separate parameters for each group. More specifically, parts 1 and 3 are constrained to be equal across groups except that the variances of the errors will be estimated separately for each group.

If there is no measurement component to the model—if there are no latent variables—then by default all parameters are estimated separately for each group.

sem: Specifying which parameters are allowed to vary in broad, sweeping terms

You can control which parameters are constrained to be equal across groups by specifying the ginvariant() option:

```
(Y1->...) (Y2->...)                    ///   part 3
(...)                                  ///
(...)                                  ///   part 2
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1
                  group(agegrp) ginvariant(pclasses)
```

The parameter classes (*pclasses*) for sem are as follows:

Parameter class description	Parameter class name	
1. structural coefficients	scoef	
2. structural intercepts	scons	
3. measurement coefficients	mcoef	
4. measurement intercepts	mcons	
5. covariances of structural errors	serrvar	
6. covariances of measurement errors	merrvar	
7. covariances between structural and measurement errors	smerrcov	
8. means of exogenous variables	meanex	(*)
9. covariances of exogenous variables	covex	(*)
10. all the above	all	(*)
11. none of the above	none	

(*) Be aware that 8, 9, and 10 (meanex, covex, and all) exclude the observed exogenous variables—that is, they include only the latent exogenous variables—unless you specify the noxconditional option or the noxconditional option is otherwise implied; see [SEM] **sem option noxconditional**. This is what you would desire in most cases.

The default when `ginvariant()` is not specified is `ginvariant(mcoef mcons)`:

```
(Y1->...) (Y2->...)                        ///   part 3, measurement
(...)                                      ///
(...)                                      ///   part 2, structural
(...)                                      ///
(X1->...) (X2->...),                       ///   part 1, measurement
                  group(agegrp) ginvariant(mcoef mcons)
```

If you also wanted covariances of errors associated with measurement to be constrained across groups, you could type

```
(Y1->...) (Y2->...)                        ///   part 3, measurement
(...)                                      ///
(...)                                      ///   part 2, structural
(...)                                      ///
(X1->...) (X2->...),                       ///   part 1, measurement
                  group(agegrp) ginvariant(mcoef mcons merrvar)
```

sem: Adding constraints for path coefficients across groups

The `ginvariant()` option allows you to state in sweeping terms which parameters vary and which are invariant across groups. You can also constrain individual parameters to be equal across groups.

Pretend that in the substantive part of the generic model, we have `Y1<-Y2`. Assume that we fit the model and allow the structural part to vary across groups:

```
(Y1->...) (Y2->...)                        ///   part 3, measurement
(...)                                      ///
(Y1<-Y2)                                   ///   part 2, structural
(...)                                      ///
(X1->...) (X2->...),                       ///   part 1, measurement
                  group(agegrp)
```

In this model, the `Y1<-Y2` path coefficient is allowed to vary across groups by default. We could constrain the coefficient to be equal across groups by typing

```
(Y1->...) (Y2->...)                        ///   part 3, measurement
(...)                                      ///
(Y1<-Y2@b)                                 ///   part 2, structural
(...)                                      ///
(X1->...) (X2->...),                       ///   part 1, measurement
                  group(agegrp)
```

Note the `@b` in `(Y1<-Y2@b)` where we previously typed `Y1<-Y2`.

Constraining a coefficient to equal a symbolic name such as `b` is how we usually constrain equality, but in the usual case, the symbolic name appears at least twice in our model. For instance, we might have `(Y1<-Y2@b)` and `(Y1<-Y3@b)` and thus constrain path coefficients to be equal.

In the case above, however, `@b` appears only once. Because we specified `group(agegrp)`, results are as if we specified this model separately for each age group, and in each group, we are specifying `@b`. Thus we are constraining the path coefficient to be equal across all groups.

sem: Adding constraints for means, variances, or covariances across groups

You use the same technique for adding constraints to means, variances, and covariances as you would for adding constraints to path coefficients. Remember that means are specified by the `means()` option, variances by the `variance()` option, and covariances by the `covariance()` option. The `variance()` and `covariance()` options are abbreviated `var()` and `cov()`, respectively.

You can specify, for instance,

```
(Y1->...) (Y2->...)                    ///   part 3, measurement
(...)                                  ///
(Y1<-Y2)                               ///   part 2, structural
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1, measurement
                group(agegrp)          ///
                means(X1@M)
```

to constrain the mean of `X1` to be the same across groups. The means would have been different across groups by default.

You can specify

```
(Y1->...) (Y2->...)                    ///   part 3, measurement
(...)                                  ///
(Y1<-Y2)                               ///   part 2, structural
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1, measurement
                group(agegrp)          ///
                var(e.Y1@V)
```

to constrain the variance of the error of `Y1` to be the same across groups.

If we wanted to allow the errors of `Y1` and `Y2` to be correlated (by default, errors are uncorrelated), we could add the `cov()` option:

```
(Y1->...) (Y2->...)                    ///   part 3, measurement
(...)                                  ///
(...)                                  ///   part 2, structural
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1, measurement
                group(agegrp)          ///
                var(e.Y1@V)            ///
                cov(e.Y1*e.Y2)
```

If we then wanted to constrain the covariance to be the same across groups, we would type

```
(Y1->...) (Y2->...)                    ///   part 3, measurement
(...)                                  ///
(...)                                  ///   part 2, structural
(...)                                  ///
(X1->...) (X2->...),                   ///   part 1, measurement
                group(agegrp)          ///
                var(e.Y1@V)            ///
                cov(e.Y1*e.Y2@C)
```

sem: Adding constraints for some groups but not others

Consider the following model:

```
... (Y1<-Y2) ..., group(agegrp)
```

Above we saw how to constrain the `Y1<-Y2` path coefficient to be the same across groups:

```
... (Y1<-Y2@b) ..., group(agegrp)
```

To constrain the path coefficients `Y1<-Y2` to be equal for groups 1 and 2 but leave the `Y1<-Y2` path coefficients unconstrained for the remaining groups, we would type

```
... (Y1<-Y2) (1: Y1<-Y2@b) (2: Y1<-Y2@b) ..., group(agegrp)
```

Think of this as follows:

1. `(Y1<-Y2)`: We set a path for all the groups.
2. `(1: Y1<-Y2@b)`: We modify the path for `agegrp` = 1.
3. `(2: Y1<-Y2@b)`: We modify the path for `agegrp` = 2.
4. We do not modify the path for any other `agegrp` value.

The result is that we constrain age groups 1 and 2 to have the same value of the path, and we do not constrain the path for the other age groups.

You can constrain variance and covariance estimates to be the same across some groups but not others in the same way. You can specify, for instance,

```
..., group(agegrp) var(1: e.Y1@V) var(2: e.Y1@V)
```

or

```
..., group(agegrp) cov(e.Y1*e.Y2) cov(1: e.Y1*e.Y2@C) ///
                                      cov(2: e.Y1*e.Y2@C)
```

Similarly, you can constrain means for some groups but not others, although this is rarely done:

```
..., group(agegrp) means(1: X1@b) means(2: X1@b)
```

sem: Adding paths for some groups but not others

In the same way that you can constrain coefficients for some groups but not others, you can add paths for some groups but not others. Consider the following model:

```
... (Y1<-Y2) ..., group(agegrp)
```

You can add the path `Y1<-Y3` for groups 1 and 2 by typing

```
... (Y1<-Y2) (1: Y1<-Y3) (2: Y1<-Y3) ..., group(agegrp)
```

You can add covariances for some groups but not others in the same way. For instance, to allow the errors of `Y1` and `Y2` to be correlated in groups 1 and 2 only, you can specify

```
..., group(agegrp) cov(1: e.Y1*e.Y2) cov(2: e.Y1*e.Y2)
```

sem: Relaxing constraints

Just as you can specify

```
..., group(agegrp) ginvariant(pclasses)
```

and then add constraints, you can also specify

```
..., group(agegrp) ginvariant(pclasses)
```

and then relax constraints that the parameter classes impose.

For instance, if we specified `ginvariant(scoef)`, then we would be constraining `(Y1<-Y2)` to be invariant across groups. We could then relax that constraint by typing

```
... (Y1<-Y2) (1: Y1<-Y2@b1) (2: Y1<-Y2@b2) ..., ///
                                  group(agegrp) ginvariant(scoef)
```

The path coefficients would be free in groups 1 and 2 and constrained in the remaining groups, if there are any. The path coefficient is free in group 1 because we specified symbolic name `b1`, and `b1` appears nowhere else in the model. The path coefficient is free in group 2 because symbolic name `b2` appears nowhere else in the model. If there are remaining groups and we want to relax the constraint on them, too, we would need to add `(3: Y1<-Y2@b3)`, and so on.

The same technique can be used to relax constraints on means, variances, and covariances:

```
..., group(agegrp) ginvariant(... meanex ...)  ///
                means(1: X1@b1) means(2: X1@b2)
..., group(agegrp) ginvariant(... serrvar ...) ///
                var(1: e.Y1@V1) var(2: e.Y1@V2)
..., group(agegrp) ginvariant(... serrvar ...) ///
                cov(1: e.Y1*e.Y2@C1) cov(2: e.Y1*e.Y2@C2)
```

Comparing groups with gsem

With `gsem`, we fit multiple-group models in a similar way by using the `group()` and `ginvariant()` options. However, the allowed parameter classes and the default constraints differ between with `sem` and `gsem`. Here, we will discuss how compare groups and how to set and release constraints across groups for models fit with `gsem`.

gsem: Fitting the model for different groups of the data

Let's begin by considering a measurement model that has binary measurements of a latent variable. If we model the measurements using probit regression, we can fit a model with the command syntax as

```
(L -> y1 y2 y3 y4), probit
```

We can fit this model for different groups such as age groups by specifying the `group(`*varname*`)` option:

```
(L -> y1 y2 y3 y4), probit group(agegrp)
```

where agegrp is a variable in our dataset, perhaps taking on values 1, 2, 3, ... We can specify the model by using the command language or by drawing the model in the Builder and then choosing and filling in the group() option.

After estimation, you can use test to obtain Wald tests of whether constraints should be added. See [SEM] **Example 49g** for an example.

gsem: Which parameters vary by default, and which do not

When we specify group(*groupvar*) with gsem, the coefficients, intercepts, and cutpoints in the model are constrained by default to be the same across the groups. All other types of parameters will be estimated separately in each group. More specifically, all variances, covariances, means, and scale parameters will be estimated separately for each group.

gsem: Specifying which parameters are allowed to vary in broad, sweeping terms

You can control which parameters are constrained to be equal across groups by specifying the ginvariant() option:

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(pclasses)
```

The parameter classes (*pclasses*) for gsem are as follows:

Parameter class description	Parameter class name
1. intercepts and cutpoints	cons
2. fixed coefficients	coef
3. latent variable coefficients	loading
4. covariances of errors	errvar
5. scaling parameters	scale
6. means of exogenous variables	means
7. covariances of exogenous latent variables	covex
8. all the above	all
9. none of the above	none

The default when ginvariant() is not specified is ginvariant(cons coef loading):

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(cons coef loading)
```

If you also wanted the variance of the latent variable to be constrained across groups, you could type

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(cons coef loading covex)
```

because L is an exogenous variable and covex refers to all covariances and variances of exogenous variables.

gsem: Adding constraints for path coefficients across groups

The ginvariant() option allows you to state in sweeping terms which parameters vary and which are invariant across groups. You can also constrain individual parameters to be equal across groups.

If we fit a model by typing

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(cons)
```

In this model, only the intercepts are constrained to be equal across groups. The L->y1, L->y2, L->y3, and L->y4 path coefficients are allowed to vary across groups because we did not include loading in the ginvariant() option. We could constrain the L->y4 coefficient to be equal across groups by typing

```
(L -> y1 y2 y3 y4@b), probit ///
        group(agegrp) ginvariant(cons)
```

Note the @b attached to y4.

Constraining a coefficient to equal a symbolic name such as b is how we usually constrain equality, but in the usual case, the symbolic name appears at least twice in our model. For instance, we might have (L->y2@b) and (L->y3@b) and thus constrain path coefficients to be equal.

In the case above, however, @b appears only once. Because we specified group(agegrp), results are as if we specified this model separately for each age group, and in each group, we are specifying @b. Thus we are constraining the path coefficient to be equal across all groups.

gsem: Adding constraints for means, variances, or covariances across groups

You use the same technique for adding constraints to means, variances, and covariances as you would for adding constraints to path coefficients. Remember that means are specified by the means() option, variances by the variance() option, and covariances by the covariance() option. The variance() and covariance() options are abbreviated var() and cov(), respectively.

You can specify, for instance,

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(cons) var(L@v)
```

to constrain the variance of L to be the same across groups.

gsem: Adding constraints for some groups but not others

Again, we consider the following model:

```
(L -> y1 y2 y3 y4), probit ///
        group(agegrp) ginvariant(cons)
```

Above we saw how to constrain the L->y4 path coefficient to be the same across groups:

```
(L -> y1 y2 y3 y4@b), probit ///
      group(agegrp) ginvariant(cons)
```

To constrain the path coefficients L->y4 to be equal for groups 1 and 2 but leave the L->y4 path coefficients unconstrained for the remaining groups, we would type

```
(L -> y1 y2 y3 y4) (1: L->y4@b) (2: L->y4@b), probit ///
      group(agegrp) ginvariant(cons)
```

Think of this as follows:

1. (L->y4): We set a path for all the groups.
2. (1: L->y4@b): We modify the path for agegrp = 1.
3. (2: L->y4@b): We modify the path for agegrp = 2.
4. We do not modify the path for any other agegrp value.

The result is that we constrain age groups 1 and 2 to have the same value of the path, and we do not constrain the path for the other age groups.

You can constrain variance, covariance, and mean estimates to be the same across some groups but not others in the same way. You can specify, for instance,

```
(L -> y1 y2 y3 y4), probit ///
      group(agegrp) var(1: L@v) var(2: L@v)
```

gsem: Adding paths for some groups but not others

In the same way that you can constrain coefficients for some groups but not others, you can add paths for some groups but not others. Consider the following model:

```
(L -> y1 y2 y3), probit group(agegrp)
```

You can add the path L->y4 for groups 1 and 2 by typing

```
(L -> y1 y2 y3) (1: L->y4) (2: L->y4), probit group(agegrp)
```

gsem: Relaxing constraints

Just as you can specify

```
..., group(agegrp) ginvariant(pclasses)
```

and then add constraints, you can also specify

```
..., group(agegrp) ginvariant(pclasses)
```

and then relax constraints that the parameter classes impose.

For instance, if we specified ginvariant(loading), then we would be constraining (L->y4) to be invariant across groups. We could then relax that constraint by typing

```
(L -> y1 y2 y3 y4) (1: L->y4@b1) (2: L->y4@b2), probit ///
      group(agegrp) ginvariant(... loading)
```

The (L->y4) path coefficients would be free in groups 1 and 2 and constrained in the remaining groups, if there are any. The path coefficient is free in group 1 because we specified symbolic name b1, and b1 appears nowhere else in the model. The path coefficient is free in group 2 because symbolic name b2 appears nowhere else in the model. If there are remaining groups and we want to relax the constraint on them, too, we would need to add (3: L->y4@b3), and so on.

The same technique can be used to relax constraints on means, variances, and covariances. For instance,

```
(L -> y1 y2 y3 y4), probit ///
      group(agegrp) ginvariant(... covex) var(1: L@V1) var(2: L@V2)
```

Reference

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Also see

[SEM] **Intro 5** — Tour of models

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **sem and gsem option covstructure( )** — Specifying covariance restrictions

[SEM] **sem group options** — Fitting models on different groups

Title

Intro 7 — Postestimation tests and predictions

Description Remarks and examples Reference Also see

Description

After fitting a model with `sem` or `gsem`, you can perform statistical tests, obtain predicted values, and more. Everything you can do is listed below.

`sem` and `gsem` vary in the tests and features available after estimation, and we mark whether each test and feature is available after `sem`, `gsem`, or both.

Remarks and examples

Remarks are presented under the following headings:

Replaying the model (sem and gsem)
Displaying odds ratios, incidence-rate ratios, etc. (gsem only)
Obtaining goodness-of-fit statistics (sem and gsem)
Performing tests for including omitted paths and relaxing constraints (sem only)
Performing tests of model simplification (sem and gsem)
Displaying other results, statistics, and tests (sem and gsem)
Obtaining predicted values (sem)
Obtaining predicted values (gsem)
Using contrast, pwcompare, and margins (sem and gsem)
Accessing stored results

Replaying the model (sem and gsem)

After estimation, you can type `sem` or `gsem` without arguments to display the estimation output:

```
. sem
(original output reappears)
```

If you wish to see results in the Bentler–Weeks formulation, after `sem` estimation type

```
. estat framework
(output omitted)
```

See [SEM] **Example 11**.

In many of the postestimation commands listed below for use after `sem` and `gsem`, you will need to refer symbolically to particular coefficients. For instance, in the model

```
. sem ... (Y<-x1) ..., ... cov(e.Y1*e.Y2)
```

the symbolic name of the coefficient corresponding to the path `Y<-x1` is `_b[Y1:x1]`, and the symbolic name of the coefficient corresponding to the covariance of `e.Y1` and `e.Y2` is `_b[/cov(e.Y1,e.Y2)]`.

Figuring out what the names are can be difficult, so instead, type

```
. sem, coeflegend
```

or

```
. gsem, coeflegend
```

With this command, `sem` (`gsem`) will produce a table looking very much like the estimation output that lists the `_b[]` notation for the estimated parameters in the model; see [SEM] **Example 8**.

Displaying odds ratios, incidence-rate ratios, etc. (gsem only)

In some generalized linear response functions, exponentiated coefficients have a special meaning. Those special meanings are

Common name	Family	Link	Meaning of exp(coef)
logit	Bernoulli	logit	odds ratio
ologit	ordinal	logit	odds ratio
mlogit	multinomial	logit	relative-risk ratio
Poisson	Poisson	log	incidence-rate ratio
nbreg	nbreg	log	incidence-rate ratio

For survival models, the special meanings of the exponentiated coefficients are

Survival distribution	Meaning of exp(coef)
exponential	hazard ratio
Weibull	hazard ratio
gamma	time ratio
loglogistic	time ratio
lognormal	time ratio

`gsem` reports coefficients, not exponentiated coefficients. You can obtain exponentiated coefficients and their standard errors by using `estat eform` after estimation. Using `estat eform` is no different from redisplaying results. The syntax is

`estat eform` *equationname*

After `gsem`, equations are named after the dependent variable, so if you want to see the equation for `cases` in exponentiated form, you can type `estat eform cases`.

See [SEM] **Example 33g**, [SEM] **Example 34g**, [SEM] **Example 47g**, [SEM] **Example 48g**, and [SEM] **estat eform**.

Obtaining goodness-of-fit statistics (sem and gsem)

One goodness-of-fit statistic and test is reported at the bottom of the `sem` output:

```
. sem                  // redisplay results
Variables in structural equation model
  (output omitted)
Structural equation model
  (coefficient table omitted)
LR test of model vs. saturated: chi2(2)   =      1.78, Prob > chi2 = 0.4111
```

Be warned that this test is based on the assumption of joint normality of the observed variables. In the case of nonnormal data, you may specify the `vce(sbentler)` option to obtain the Satorra–Bentler (1994) scaled χ^2 test. In either case, the test is a goodness-of-fit test in badness-of-fit units; a significant result implies that the model does not fit well. More mathematically, the null hypothesis of this test is that the fitted covariance matrix and mean vector of the observed variables are equal to the matrix and vector observed in the population as measured by the sample. Remember, however, the goal is not to maximize the goodness of fit. One must not add paths or covariances that are not theoretically meaningful.

In addition, other goodness-of-fit statistics are available:

1. (sem only.) Command estat gof reports a variety of goodness-of-fit statistics; see [SEM] **estat gof** and [SEM] **Example 4**.
2. (sem only.) Command estat eqgof reports R^2-like goodness-of-fit statistics for each equation separately; see [SEM] **estat eqgof** and [SEM] **Example 3**.
3. (sem only.) Command estat ggof reports goodness-of-fit statistics by group when you have estimated using sem's group() option; see [SEM] **estat ggof** and [SEM] **Example 21**.
4. (sem only.) Command estat residuals reports the element-by-element differences between the observed and fitted covariance matrix, and the observed and fitted mean vector, optionally in standardized or in normalized units; see [SEM] **estat residuals** and [SEM] **Example 10**.
5. (sem and gsem.) Commands estat ic and estimates stats report the Akaike and Bayesian information criteria; see [SEM] **Example 51g**, [SEM] **Example 52g**, [R] **estat ic**, and [R] **estimates stats**.
6. (gsem only.) Command estat lcgof reports goodness-of-fit statistics for latent class models; see [SEM] **estat lcgof** and [SEM] **Example 51g**.

Performing tests for including omitted paths and relaxing constraints (sem only)

1. (sem only.) Command estat mindices reports χ^2 modification indices and significance values for each omitted path in the model, along with the expected parameter change; see [SEM] **estat mindices**, [SEM] **Example 5**, and [SEM] **Example 9**.
2. (sem only.) Command estat scoretests performs score tests on each of the linear constraints placed on the paths and covariances; see [SEM] **estat scoretests** and [SEM] **Example 8**.
3. (sem only.) Command estat ginvariant is for use when you have estimated using sem's group() option; see [SEM] **Intro 6**. This command tests whether you can relax constraints that parameters are equal across groups; see [SEM] **estat ginvariant** and [SEM] **Example 22**.

Performing tests of model simplification (sem and gsem)

1. (sem and gsem.) Command test reports Wald tests of single or multiple linear constraints. See [SEM] **Example 8** and [SEM] **test**.
2. (sem and gsem.) Command lrtest reports likelihood-ratio tests of single or multiple linear constraints. See [SEM] **Example 10**, [SEM] **lrtest**, and [SEM] **Example 39g**.
3. (sem only.) Command estat eqtest reports an overall Wald test for each equation in the model, the test corresponding to all coefficients in the equation except the intercept being simultaneously 0; see [SEM] **estat eqtest** and [SEM] **Example 13**.
4. (sem only.) Command estat ginvariant is for use when you have estimated using sem's group() option; see [SEM] **Intro 6**. This command tests whether parameters allowed to vary across groups could be constrained; see [SEM] **estat ginvariant** and [SEM] **Example 22**.

Displaying other results, statistics, and tests (sem and gsem)

1. (sem only.) The `estat stdize:` command prefix—used in front of `test`, `testnl`, `lincom`, and `nlcom`—allows you to perform tests on standardized coefficients. See [SEM] **estat stdize** and [SEM] **Example 16**.
2. (sem only.) Command `estat teffects` reports total effects of one variable on another and decomposes the total effect into direct and indirect effects. Results can be reported in standardized or unstandardized form. See [SEM] **estat teffects** and [SEM] **Example 7**.
3. (sem only.) Command `estat stable` assesses the stability of nonrecursive structural equation systems; see [SEM] **estat stable** and [SEM] **Example 7**.
4. (sem and gsem.) Command `estat summarize` reports summary statistics for the observed variables used in the model; see [SEM] **estat summarize**.
5. (sem and gsem.) Command `lincom` reports the value, standard error, significance, and confidence interval for linear combinations of estimated parameters; see [SEM] **lincom**.
6. (sem and gsem.) Command `nlcom` reports the value, standard error, significance, and confidence interval for nonlinear (and linear) combinations of estimated parameters; see [SEM] **nlcom** and [SEM] **Example 42g**.
7. (sem and gsem.) Command `estat vce` reports the variance–covariance matrix of the estimated parameters; see [R] **estat vce**.
8. (gsem only.) Command `estat lcmean` reports marginal predicted means of each outcome within each latent classes; see [SEM] **estat lcmean**, [SEM] **Example 50g**, and [SEM] **Example 53g**.
9. (gsem only.) Command `estat lcprob` reports marginal predicted latent class probabilities; see [SEM] **estat lcprob**, [SEM] **Example 50g**, and [SEM] **Example 53g**.

Obtaining predicted values (sem)

You obtain predicted values with the `predict` command. Below we will write that predictions are the expected values, but be aware that when there are latent variables in your model, predictions are based on predicted scores; the scores can be inconsistent, and thus any prediction based on them can be inconsistent.

Available are the following:

1. `predict` *newvar*`, xb(`*odepvarname*`)` creates new variable *newvar* containing the predicted values for observed endogenous variable *odepvarname*.

   `predict` *stub*`*, xb` creates new variables *stub*`1`, *stub*`2`, ... containing the predicted values for all the observed endogenous variables in the model.

   These predicted values are the expected value of the xb given the values of the observed exogenous variables.

2. `predict` *newvar*`, latent(`*Lname*`)` creates new variable *newvar* containing the predicted values of the latent variable *Lname*, whether endogenous or exogenous.

   `predict` *stub*`*, latent` creates new variables *stub*`1`, *stub*`2`, ... containing the predicted values for all the latent variables in the model.

   Predicted values of latent variables, also known as predicted factor scores, are the expected values of the variables given the values of the observed variables.

3. `predict` *newvar*`, xblatent(`*Lname*`)` creates new variable *newvar* containing the predicted values for latent endogenous variable *Lname*.

   `predict` *stub*`*, xblatent` creates new variables *stub*1, *stub*2, ... containing the predicted values for all the latent endogenous variables in the model.

   `predict` with `xblatent(`*Lname*`)` differs from `latent(`*Lname*`)` in that the factor scores predicted by `latent()` are then used with the linear equation for *Lname* to make the prediction.

4. `predict` *stub*`*, scores` will create a slew of variables, one for each estimated parameter, containing the observation-by-observation values of the first derivative, also known as scores. This command is intended for use by programmers and may only be used after estimation using `method(ml)` or `method(mlmv)`.

See [SEM] **Example 14** and [SEM] **predict after sem**.

Obtaining predicted values (gsem)

`predict` after `gsem` is more complicated than `predict` after `sem` because generalized SEMs are more complicated. Three new issues arise: the prediction of generalized linear response variables as opposed to linear response variables, the optional presence of multilevel continuous latent variables in the model, and the optional presence of categorical latent variables in the model.

Let's start with the response functions of the observed endogenous variables. Corresponding to `predict, xb` after `sem` are the following:

1. `predict` *newvar*`, mu outcome(`*depvar*`)` predicts mean responses in natural units—linear, probability, counts, and so on. Think $g^{-1}(x_i b)$ in the generalized linear model notation.

   With continuous latent variables, calculations are made using the empirical Bayes mean predictions of latent variables, which can be computationally intensive. You can specify that empirical Bayes modes be used by specifying option `conditional(ebmodes)`. You can also specify that all continuous latent variables be set to their mean value, typically 0, by specifying the `conditional(fixedonly)` option.

   With categorical latent variables, calculations can be made for a specific latent class by specifying the `class(`*lcspec*`)` option.

2. `predict` *newvar*`, mu outcome(`*depvar*`) marginal` also predicts mean responses in natural units, and again it is appropriate to think $g^{-1}(x_i b)$.

   With continuous latent variables, calculations are made by integrating the prediction formula with respect to the continuous latent variables, which can be computationally intensive.

   With categorical latent variables, the overall mean is computed by a weighted sum of the latent class means, where the weights are latent class probabilities.

3. `predict` *newvar*`, mu outcome(`*depvar*`) pmarginal` also predicts mean responses in natural units, and again it is appropriate to think $g^{-1}(x_i b)$.

   The `pmarginal` option is allowed only for models with categorical latent variables. The overall mean is computed by a weighted sum of the latent class means, where the weights are posterior latent class probabilities.

4. Option `pr` is a synonym for `mu` when using family-and-link combinations that produce probabilities, such as logit, probit, etc.

5. `predict` *newvar*`, eta outcome(`*depvar*`)` calculates the linear prediction $x_i b$.

   With continuous latent variables, calculations are made using the empirical Bayes mean predictions of latent variables, which can be computationally intensive. You can specify that empirical Bayes modes be used by specifying option `conditional(ebmodes)`. You can also specify that all continuous latent variables be set to their mean value, typically 0, by specifying option `conditional(fixedonly)`.

   With categorical latent variables, calculations can be made for a specific latent class by specifying the `class(`*lcspec*`)` option.

6. `predict` *newvar*`, density outcome(`*depvar*`)` calculates the density function for *depvar* using the current data and model parameters.

   With continuous latent variables, calculations are made using the empirical Bayes mean predictions of latent variables, which can be computationally intensive. You can change the treatment of latent variables by specifying option `conditional(ebmodes)`, `conditional(fixedonly)`, or `marginal`.

   With categorical latent variables, calculations can be made for a specific latent class by specifying the `class(`*lcspec*`)` option.

7. `predict` *newvar*`, distribution outcome(`*depvar*`)` calculates the cumulative distribution function for *depvar* using the current data and model parameters.

   With continuous latent variables, calculations are made using the empirical Bayes mean predictions of latent variables, which can be computationally intensive. You can change the treatment of latent variables by specifying option `conditional(ebmodes)`, `conditional(fixedonly)`, or `marginal`.

   With categorical latent variables, calculations can be made for a specific latent class by specifying the `class(`*lcspec*`)` option.

   Option `distribution` is not allowed with `family(multinomial)`.

8. `predict` *newvar*`, survival outcome(`*depvar*`)` calculates the cumulative survivor function for *depvar* using the current data and model parameters.

   With continuous latent variables, calculations are made using the empirical Bayes mean predictions of latent variables, which can be computationally intensive. You can change the treatment of latent variables by specifying option `conditional(ebmodes)`, `conditional(fixedonly)`, or `marginal`.

   With categorical latent variables, calculations can be made for a specific latent class by specifying the `class(`*lcspec*`)` option.

   Option `survival` is allowed only with `family(exponential)`, `family(gamma)`, `family(loglogistic)`, `family(lognormal)`, and `family(weibull)`.

9. Option `outcome()` varies to accommodate the multiple predictions produced by multinomial logit, ordinal probit, ordinal logit, and ordinal clogclog.

   a. `outcome(`*depvar #*`)` specifies that you want the prediction for *depvar* $= \#$.

   b. `outcome(`*#.depvar*`)` is another way of specifying that you want the prediction for *depvar* $= \#$.

   c. `outcome(`*depvar* `##`*)* specifies that you want the prediction for the $\#$th category.

   For multinomial logit, you use the above form of `outcome()` when requesting `mu` or `eta` predictions.

For ordered outcomes, you use the above form of `outcome()` when requesting `mu` predictions. For `eta` predictions, there is only one linear equation, and you use the usual `outcome(`*depvar*`)` form.

10. Option `class(`*lclspec*`)` is for models with categorical latent variables and allows you to target predictions for a specific latent class. For models with one categorical latent variable, *lclspec* can be a level value, such as `class(2)` or its equivalent factor-variable notation `class(2.C)`, assuming the latent class variable is `C`. For models with two or more categorical latent variables, *lclspec* may only be in factor-variable notation, such as `class(2.C#1.D)` for categorical latent variables `C` and `D`.

11. Some GLM families, such as Poisson, allow specification of an offset or exposure. If you use one of those families and you specify an offset or exposure with `gsem` when you fit the model and now want predictions with offset = 0 (exposure = 1), you can specify option `nooffset`.

12. To obtain predicted values for all observed endogenous variables, you can omit the `outcome()` option and replace *newvar* with *stub*`*`—that is, a piece of a new variable name followed by `*`. This is not just convenient; it can also save computer time because some parts of the calculation can be shared.

    For example, you can change

    ```
    predict newvar, mu outcome(depvar)
    ```

    to be

    ```
    predict new*, mu
    ```

So much for the observed endogenous variables. Let's now move on to continuous latent variables, whether endogenous or exogenous:

13. `predict` *newvar*`, latent(`*Lname*`)` predicts the value of the specified latent variable by using empirical Bayes means. You can specify that empirical Bayes modes be used by specifying option `ebmodes`; see item 16b below.

14. `predict` *newvar*`, latent(`*Lname*`) se(`*another_newvar*`)` will predict both the latent variable and its standard error.

15. You can use the same *stub*`*` trick mentioned for predicting observed endogenous variables to obtain all the latent variables. Replace *newvar* with *stub*`*` and omit *Lname* to produce

    ```
    predict stub*, latent
    ```

    You can do this with the `se()` option, too:

    ```
    predict stub*, latent se(another_stub*)
    ```

16. Four options are related to the iterative procedure for obtaining the predicted values:

    a. `ebmodes` specifies that empirical Bayes modes rather than means be predicted. Because latent variables are assumed to be normally distributed, means and modes are usually similar. They are, however, calculated differently. The predictions are the means or modes of the empirical posterior distributions of the latent variables, which are not necessarily normal. Therefore, the means can differ from the modes. Both are computationally intensive.

       The default empirical Bayes means calculation uses adaptive quadrature, which is to say, numerical integration. All three of the options listed below apply to this calculation.

The alternative empirical Bayes mode calculation uses an optimization method that does not require adaptive quadrature. For small datasets, modes can usually be calculated more quickly than means. For large datasets, however, predicting modes can actually take longer. If you specify `ebmodes`, of the three options listed below, only `tolerance()` and `iterate()` are relevant.

b. `intpoints(#)` specifies the number of quadrature (numerical integration) points to be used to compute the empirical Bayes means. The default is the value used in estimation, which in turn defaults to 7.

   This option also controls the number of quadrature points used to compute marginal predictions for the observed endogenous variables.

c. `tolerance(#)` specifies the convergence tolerance in computing empirical Bayes means and modes and defaults to the value used in estimation, which in turn defaults to 1.0e–8.

d. `iterate(#)` specifies the maximum number of iterations in computing empirical Bayes means and modes and defaults to the value used in estimation, which in turn defaults to 1,001.

Continuous latent variables come in observation-level and multilevel flavors. If you have a multilevel latent variable such as `M1[school]`, you can predict it just as you would any other latent variable:

```
predict newvar, latent(M1[school])
```

If you predict all the latent variables with the *stub*`*` trick, any multilevel latent variables will be automatically included among the new variables.

The remaining predictions are for models with categorical latent variables.

17. `predict` *newvar*`, classpr class(`*lcspec*`)` predicts the probability of belonging to the specified latent class. This probability is the same one used to combine the class specific densities to form the marginal likelihood.

18. `predict` *newvar*`, classposteriorpr class(`*lcspec*`)` predicts the posterior probability of belonging to the specified latent class. This probability is a function of the above latent class computed by `predict, classpr` and the fitted outcome densities.

19. You can use the same *stub*`*` trick mentioned above to obtain predicted probabilities for all latent classes. Replace *newvar* with *stub*`*` and omit `class()` to produce

    ```
    predict stub*, classpr
    ```

    or

    ```
    predict stub*, classposteriorpr
    ```

See [SEM] **Example 28g**, [SEM] **Example 29g**, [SEM] **Example 50g**, [SEM] **Example 52g**, and [SEM] **predict after gsem**.

Using contrast, pwcompare, and margins (sem and gsem)

`contrast`, `pwcompare`, and `margins` are postestimation commands of Stata. All three can be used after `gsem`, and `margins` can be used after `sem` as well. Even so, these commands work best when you specify your model with Stata's factor-variable notation; see [SEM] **Intro 3**. `sem` does not allow that notation, and so obtaining desired results is a little more work.

1. Command `contrast` can be used after `gsem` to test linear hypotheses and make comparisons involving factor variables and their interactions. `contrast` allows for performing ANOVA-style tests of main effects, simple effects, or interaction effects. In addition, effects can be

decomposed into specific types of contrasts such as comparison against the grand mean, orthogonal polynomial contrasts, and more. See [R] **contrast**.

2. Command `pwcompare` can be used after `gsem` to obtain all pairwise comparisons of marginal linear predictions, including cell means, marginal means, etc. See [R] **pwcompare**.
3. Command `margins` is used after `sem` or `gsem` to obtain margins, meaning estimated marginal means, predictive margins, adjusted predictions, and marginal effects. `margins` also allows for computing contrasts and pairwise comparisons of both linear and nonlinear predictions. See [SEM] **Example 50g**, [R] **margins**, [R] **margins, contrast**, and [R] **margins, pwcompare**.

Accessing stored results

`sem` and `gsem` store all results in `e()`; see *Stored results* in [SEM] **sem** and *Stored results* in [SEM] **gsem**. To get some idea of what is stored in `e()` after `sem` or `gsem` estimation, type

```
. ereturn list
(output omitted)
```

You can save estimation results in files or temporarily in memory and do other useful things with them; see [R] **estimates**.

Not stored by `sem` in `e()` are the Bentler–Weeks matrices, but they can be obtained from the `r()` stored results of `estat framework`. (The Bentler–Weeks matrices are not relevant in the case of `gsem`.)

See [SEM] **sem** and [SEM] **estat framework**.

Reference

Satorra, A., and P. M. Bentler. 1994. Corrections to test statistics and standard errors in covariance structure analysis. In *Latent Variables Analysis: Applications for Developmental Research*, ed. A. von Eye and C. C. Clogg, 399–419. Thousand Oaks, CA: SAGE.

Also see

[SEM] **Intro 6** — Comparing groups

[SEM] **Intro 8** — Robust and clustered standard errors

Title

Intro 8 — Robust and clustered standard errors

Description Options Remarks and examples Also see

Description

`sem` and `gsem` provide two options to modify how standard-error calculations are made: `vce(robust)` and `vce(cluster` *clustvar*`)`. These standard errors are less efficient than the default standard errors, but they are valid under less restrictive assumptions.

These options are allowed only when default estimation method `method(ml)` is used or when option `method(mlmv)` is used. `ml` stands for maximum likelihood, and `mlmv` stands for maximum likelihood with missing values; see [SEM] **Intro 4**, [SEM] **sem**, and [SEM] **gsem**.

Also see [SEM] **Intro 9**, entitled *Standard errors, the full story*.

Options

`vce(`*vcetype*`)` specifies how the VCE, and thus the standard errors, is calculated. VCE stands for variance–covariance matrix of the estimators. The standard errors that `sem` and `gsem` report are the square roots of the diagonal elements of the VCE.

`vce(oim)` is the default. `oim` stands for observed information matrix (OIM). The information matrix is the matrix of second derivatives, usually of the log-likelihood function. The OIM estimator of the VCE is based on asymptotic maximum-likelihood theory. The VCE obtained in this way is valid if the errors are independent and identically distributed normal, although the estimated VCE is known to be reasonably robust to violations of the normality assumption, at least as long as the distribution is symmetric and normal-like.

`vce(robust)` specifies an alternative calculation for the VCE, called robust because the VCE calculated in this way is valid under relaxed assumptions. The method is formally known as the Huber/White/sandwich estimator. The VCE obtained in this way is valid if the errors are independently distributed. It is not required that the errors follow a normal distribution, nor is it required that they be identically distributed from one observation to the next. Thus the `vce(robust)` VCE is robust to heteroskedasticity of the errors.

`vce(cluster` *clustvar*`)` is a generalization of the `vce(robust)` calculation that relaxes the assumption of independence of the errors and replaces it with the assumption of independence between clusters. Thus the errors are allowed to be correlated within clusters.

Remarks and examples

The `vce()` option is allowed by `sem` and `gsem`. In the rest of this entry, we will use `sem` in illustrations, but everything we say applies equally to `gsem`.

The `vce(robust)` option,

```
. sem ..., ... vce(robust)
```

and the `vce(cluster` *clustvar*`)` option,

```
. sem ..., ... vce(cluster clustvar)
```

relax assumptions that are sometimes unreasonable for a given dataset and thus produce more accurate standard errors in those cases. Those assumptions are homoskedasticity of the variances of the errors—`vce(robust)`—and independence of the observations—`vce(cluster` *clustvar*`)`. `vce(cluster` *clustvar*`)` relaxes both assumptions.

Homoskedasticity means that the variances of the errors are the same from observation to observation. Homoskedasticity can be unreasonable if, for instance, the error corresponds to a dependent variable of income or socioeconomic status. It would not be unreasonable to instead assume that, in the data, the variance of income or socioeconomic status increases as the mean increases. In such cases, rather than typing

```
. sem (y<-...) (...) (...<-x1) (...<-x2)
```

you would type

```
. sem (y<-...) (...) (...<-x1) (...<-x2), vce(robust)
```

Independence implies that the observations are uncorrelated. If you have observations on people, some of whom live in the same neighborhoods, it would not be unreasonable to assume instead that the error of one person is correlated with those of others who live in the same neighborhood because neighborhoods tend to be homogeneous. In such cases, if you knew the neighborhood, rather than typing

```
. sem (y<-...) (...) (...<-x1) (...<-x2)
```

you would type

```
. sem (y<-...) (...) (...<-x1) (...<-x2), vce(cluster neighborhood)
```

Understand that if the assumptions of independent and identically distributed normal errors are met, the `vce(robust)` and `vce(cluster` *clustvar*`)` standard errors are less efficient than the standard `vce(oim)` standard errors. Less efficient means that for a given sample size, the standard errors jump around more from sample to sample than would the `vce(oim)` standard errors. `vce(oim)` standard errors are unambiguously best when the standard assumptions of homoskedasticity and independence are met.

Also see

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **Intro 9** — Standard errors, the full story

[SEM] **gsem estimation options** — Options affecting estimation

[SEM] **sem option method()** — Specifying method and calculation of VCE

Title

Intro 9 — Standard errors, the full story

Description Options Remarks and examples Reference
Also see

Description

In [SEM] **Intro 8**, we told you part of the story of the calculation of the VCE, the part we wanted to emphasize. In this section, we tell you the full story.

We at Stata try to draw a clear distinction between method and technique. The method is the process used to obtain the parameter estimates. The technique is the process used to obtain the variance–covariance matrix of the parameter estimates, which is to say, the standard errors.

The literature does not always draw such clear distinctions.

`sem` and `gsem` provide the following methods and techniques:

Methods	
ML	maximum likelihood
QML	quasimaximum likelihood
MLMV	maximum likelihood with missing values
ADF	asymptotic distribution free
Techniques	
OIM	observed information matrix
EIM	expected information matrix
OPG	outer product of the gradients
sbentler	Satorra–Bentler estimator
robust	Huber/White/sandwich estimator
clustered	generalized Huber/White/sandwich estimator
bootstrap	nonparametric bootstrap
jackknife	delete-one jackknife

They are allowed in the following combinations:

Command/ Method	Allowed techniques	Comment
`sem`		
ML	OIM	default
	EIM	
	OPG	
	sbentler	
	robust	a.k.a. QML
	clustered	
	bootstrap	
	jackknife	
MLMV	OIM	default
	EIM	
	OPG	
	robust	a.k.a. QML
	clustered	
	bootstrap	
	jackknife	
ADF	OIM	default; robust-like
	EIM	
	bootstrap	
	jackknife	
`gsem`		
ML	OIM	default
	OPG	
	robust	a.k.a. QML
	clustered	
	bootstrap	
	jackknife	

Options

The corresponding options for sem and gsem to obtain each allowed method-and-technique combination are

method()	vce()	Comment
sem		
method(ml)	vce(oim)	default
	vce(eim)	
	vce(opg)	
	vce(sbentler)	
	vce(robust)	a.k.a. QML
	vce(cluster *clustvar*)	
	vce(bootstrap)	
	vce(jackknife)	
method(mlmv)	vce(oim)	default
	vce(eim)	
	vce(opg)	
	vce(robust)	a.k.a. QML
	vce(cluster *clustvar*)	
	vce(bootstrap)	
	vce(jackknife)	
method(adf)	vce(oim)	default; vce(robust)-like
	vce(eim)	
	vce(bootstrap)	
	vce(jackknife)	
gsem		
method(ml)	vce(oim)	default
	vce(opg)	
	vce(robust)	a.k.a. QML
	vce(cluster *clustvar*)	
	bootstrap	no option; use bootstrap: prefix
	jackknife	no option; use jackknife: prefix

method(*emethod*) specifies the estimation method sem (gsem) is to use. If method() is not specified, then method(ml) is assumed.

vce(*vcetype*) specifies the technique to be used to obtain the VCE. When vce() is not specified, then vce(oim) is assumed.

In the case of gsem, vce(bootstrap) and vce(jackknife) are not allowed, although you can obtain the bootstrap or jackknife results by prefixing the gsem command with the bootstrap: or jackknife: prefix.

```
. bootstrap: gsem ...
. jackknife: gsem ...
```

See [R] **bootstrap** and [R] **jackknife**. If you are fitting a multilevel model, be sure to use bootstrap's and jackknife's cluster() and idcluster() options to obtain a correct resampling. If you have a crossed model, you cannot resample in both dimensions; there is no solution to that problem,

and therefore you cannot use the `bootstrap:` or `jackknife:` prefix. In addition, these prefixes are not allowed with models that have categorical latent variables because the categories of the latent variables (the latent classes) could change meaning from one sample to another, making it impossible to compute proper standard errors.

Remarks and examples

In [SEM] **Intro 4**, we gave reasons for why you might want to choose one method/technique over another. In that section, we did not draw the clear distinction between method and technique that we draw here.

The technique everyone is familiar with is OIM, although they might not know it by that name. OIM is the inverse of the negative matrix of second derivatives. Note that it is also the default for both `sem` and `gsem`.

The most common alternative techniques chosen are robust and cluster—options `vce(robust)` and `vce(cluster` *clustvar*`)`. That is because they relax assumptions; see [SEM] **Intro 8**.

Aside: Robust and cluster are not allowed with `method(adf)`. Nonetheless, the default OIM technique is more robust when combined with `method(adf)` than OIM usually is. OIM can be used even when errors are heteroskedastic. That is, the assumptions justifying the OIM calculation when combined with `method(adf)` are they same as those justifying `vce(robust)`—the errors are merely assumed to be independent.

Technique sbentler provides another version of standard errors that are robust to nonnormality when one fits a model with `sem`. These standard errors are a function of fourth-order moments and correspond to the Satorra–Bentler (1994) scaled chi-squared test, which is an adjustment of the model-versus-saturated goodness-of-fit statistic for nonnormal data.

Technique EIM—available with `sem` but not `gsem`—has similar properties to OIM. It is used in performing score tests. The `sem` command secretly calculates the EIM when necessary so that you can use postestimation score-test commands even if you estimate using a technique other than EIM. EIM is available to you because the `sem` command needs EIM for its own hidden purposes.

Technique OPG is not used so much anymore, although historically it was popular because it took less computer time to calculate than OIM.

For a discussion of bootstrap and jackknife variance estimation, see [R] **bootstrap** and [R] **jackknife**.

Reference

Satorra, A., and P. M. Bentler. 1994. Corrections to test statistics and standard errors in covariance structure analysis. In *Latent Variables Analysis: Applications for Developmental Research*, ed. A. von Eye and C. C. Clogg, 399–419. Thousand Oaks, CA: SAGE.

Also see

[SEM] **Intro 8** — Robust and clustered standard errors

[SEM] **Intro 10** — Fitting models with survey data

[SEM] **gsem estimation options** — Options affecting estimation

[SEM] **sem option method()** — Specifying method and calculation of VCE

Title

Intro 10 — Fitting models with survey data

Description Remarks and examples Also see

Description

Sometimes the data are not a simple random sample from the underlying population but instead are based on a complex survey design that can include stages of clustered sampling and stratification. Estimates produced by `sem` can be adjusted for these issues.

Adjustments for survey data are provided by `sem` and `gsem`.

Remarks and examples

Data obtained from surveys, properly treated, produce different point estimates because some observations represent a greater proportion of the underlying population than others. They also produce different standard errors because the observation-to-observation (sample-to-sample) variation is a function of the survey's design.

To obtain survey-corrected results, you first describe the characteristics of the survey with `svyset`:

```
. svyset county [pw=samplewgt], fpc(n_counties) strata(states) ||  ///
                                school, fpc(n_schools)          ||  ///
                                student, fpc(n_students)
```

In the above, we are telling Stata that our data are from a three-stage sampling design. The first stage samples without replacement counties within state; the second, schools within each sampled county; and the third, students within schools.

Once we have done that, we can tell Stata to make the survey adjustment by prefixing statistical commands with the `svy:` prefix:

```
. svy: regress test_result teachers_per_student sex ...
```

Point estimates and standard errors will be adjusted.

You can use the `svy:` prefix with `sem` and `gsem`:

```
. svy: sem (test_result<-...) ... (teachers_per_student->...) ...
```

See the *Stata Survey Data Reference Manual* for more information on this. From a survey perspective, `sem` is not different from any other statistical command of Stata. When `gsem` is used to fit a multilevel model, stage-level sampling weights specified in the `svyset` command are applied to the corresponding hierarchical group level in the model.

Once results are estimated, you do not include the `svy:` prefix in front of the postestimation commands. You type, for instance,

```
. estat eqtest ...
```

You do not type `svy: estat eqtest ...`.

Some postestimation procedures you might ordinarily perform can be inappropriate with survey estimation results. This is because you no longer have a sample likelihood value. The postestimation command `lrtest` is an example. If you attempt to use an inappropriate postestimation command, you will be warned.

```
. lrtest ...
lrtest is not appropriate with survey estimation results
r(322);
```

Also see

[SEM] **Intro 9** — Standard errors, the full story

[SEM] **Intro 11** — Fitting models with summary statistics data (sem only)

[SVY] *Stata Survey Data Reference Manual*

Title

Intro 11 — Fitting models with summary statistics data (sem only)

Description Remarks and examples Reference Also see

Description

In textbooks and research papers, the data used are often printed in summary statistic form. These summary statistics include means, standard deviations or variances, and correlations or covariances. These summary statistics can be used in place of the underlying raw data to fit models with `sem`.

Summary statistics data (SSD) are convenient for publication because of their terseness. By not revealing individual responses, they do not violate participant confidentiality, which is sometimes important.

Support for SSD is provided by `sem` but not by `gsem`.

Remarks and examples

Remarks are presented under the following headings:

Background
How to use sem with SSD
What you cannot do with SSD
Entering SSD
Entering SSD for multiple groups
What happens when you do not set all the summary statistics
Labeling SSD
Making summary statistics from data for use by others

Background

The structural equation modeling estimator is a function of the first and second moments—the means and covariances—of the data. Thus it is possible to obtain estimates of the parameters of an SEM by using means and covariances. One does not need the original dataset.

In terms of `sem`, one can create a dataset containing these summary statistics and then use that dataset to obtain fitted models. The `sem` command is used just as one would use it with the original, raw data.

How to use sem with SSD

To use `sem` with SSD,

1. Enter the summary statistics by using the `ssd` command. How you do that is the topic of an upcoming section.
2. Save the data just as you would any dataset, namely, with the `save` command.
3. Use the `sem` command just as you would ordinarily. You `use` the SSD if they are not already in memory, and no special syntax or options are required by `sem`, except

a. Do not use `sem`'s `if` *exp* or `in` *range* qualifiers. You do not have the raw data in memory and so you cannot select subsets of the data.

b. If you have entered summary statistics for groups of observations (for example, males and, separately, females), use `sem`'s `select()` option if you want to fit the model with a subset of the groups. That is, where you would ordinarily type

```
. sem ... if sex==1, ...
```

you instead type

```
. sem ..., ... select(1)
```

Where you would ordinarily type

```
. sem ... if region==1 | region==3, ...
```

you instead type

```
. sem ..., ... select(1 3)
```

See [SEM] **Example 3**.

What you cannot do with SSD

With SSD in memory,

1. You cannot obtain Satorra–Bentler standard errors and the Satorra–Bentler scaled χ^2 test, which you would normally do by specifying `sem` option `vce(sbentler)`.
2. You cannot obtain robust standard errors, which you would normally do by specifying `sem` option `vce(robust)`.
3. You cannot obtain clustered standard errors, which you would normally do by specifying `sem` option `vce(cluster` *clustvar*`)`.
4. You cannot obtain survey-adjusted results, which you would normally do by specifying the `svy:` prefix in front of the `sem` command.
5. You cannot obtain bootstrap or jackknife standard errors, which you would normally do by specifying `sem` option `vce(bootstrap)` or `vce(jackknife)`.
6. You cannot obtain VCE estimates from the observation-level outer product of the gradients, which you would normally do by specifying `vce(opg)`.
7. You cannot use weights, which you would normally do by specifying, for instance, `[fw=`*varname*`]`.
8. You cannot restrict the estimation sample with `if` *exp* or `in` *range*.
9. You cannot fit the model by using maximum likelihood with missing values or by using the asymptotic distribution free method, which you would normally do by specifying `method(mlmv)` or `method(adf)`.

Entering SSD

Entering SSD is easy. You need to see an example of how easy it is before continuing: see [SEM] **Example 2**.

What follows is an outline of the procedure. Let us begin with the data you need to have. You have

1. The names of the variables. We will just call them `x1`, `x2`, and `x3`.
2. The number of observations, say, 74.
3. The correlations, say,

```
 1
-0.8072    1
 0.3934   -0.5928    1
```

   or you may have the covariances,

```
33.4722
-3.6294    0.6043
 1.0374   -0.2120    0.2118
```

4. The variances: 33.4722, 0.6043, and 0.2118.

   Or the standard deviations: 5.6855, 0.7774, and 0.4602.

   Or neither.

   If you have the covariances in step 3, you in fact have the variances—they are just the diagonal elements of the covariance matrix—but the software will not make you enter the values twice.

5. The means: 21.2973, 3.0195, and 0.2973.

   Or not.

With that information at hand, do the following:

1. Start with no data in memory:

```
. clear all
```

2. Initialize the SSD by stating the names of the variables:

```
. ssd init x1 x2 x3
```

The remaining steps can be done in any order.

3. Set the number of observations:

```
. ssd set obs 74
```

4. Set the covariances:

```
. ssd set cov 33.4722 \ -3.6294 .6043 \ 1.0374 -.2120 .2118
```

   Or the correlations:

```
. ssd set cor  1 \ -.8072 1 \ .3934 -.5928 1
```

5. If you set covariances in step 4, skip to step 6. Otherwise, if you have them, set the variances:

```
. ssd set var 33.4722 .6043 .2118
```

   Or set the standard deviations:

```
. ssd set sd  5.6855 .7774 .4602
```

6. Set the means if you have them:

```
. ssd set means 21.2973 3.0195 .2973
```

7. If at any point you become confused as to what you have set and what remains to be set, type

```
. ssd status
```

8. If you want to review what you have set, type

```
. ssd list
```

9. If you make a mistake, you can repeat any `ssd set` command by adding the `replace` option to the end. For instance, you could reenter the means by typing

```
. ssd set means 21.2973 3.0195 .2973, replace
```

10. Save the dataset just as you would with any dataset:

```
. save mydata
```

You are now ready to use `sem` with the SSD. With the SSD in memory, you issue the `sem` command just as you would if you had the raw data:

```
. sem ...
```

Entering SSD for multiple groups

You can enter summary statistics for groups of the data. Perhaps you have summary statistics not for the data as a whole, but for males and for females, or for the young, for the middle-aged, and for the old.

Let's pretend you have the following data:

The young:			
observations:	74		
correlations:	1		
	−0.8072	1	
	0.3934	−0.5928	1
standard deviations:	5.6855	0.7774	0.4602
means:	21.2973	3.0195	0.2973
The middle-aged:			
observations:	141		
correlations:	1		
	−0.5721	1	
	0.3843	−0.4848	1
standard deviations:	4.9112	0.7010	0.5420
means:	38.1512	5.2210	0.2282
The old:			
observations:	36		
correlations:	1		
	−0.8222	1	
	0.3712	−0.3113	1
standard deviations:	6.7827	0.7221	0.4305
means:	58.7171	2.1511	0.1623

The commands for entering these summary statistics are

```
. ssd init x1 x2 x3
. ssd set obs   74
. ssd set cor    1    \  -.8072  1    \ .3934 -.5928 1
. ssd set sd     5.6855   .7774  .4602
. ssd set means 21.2973  3.0195  .2973
. ssd addgroup agecategory
. ssd set obs  141
. ssd set cor    1    \  -.5721  1    \ .3843 -.4848 1
. ssd set sd     4.9112   .7010  .5420
. ssd set means 38.1512  5.2210  .2282
. ssd addgroup
. ssd set obs   36
. ssd set cor    1    \  -.8222  1    \ .3712 -.3113 1
. ssd set sd     6.7827   .7221   .4305
. ssd set means 58.7171  2.1511   .1623
. save mygroupdata
```

The general procedure is as follows:

1. Enter the summary statistics for the first group just as outlined in the previous section.
2. Next add a group by typing

   . ssd addgroup *newgroupvar*

   In that one command, you are telling `ssd` that the summary statistics you entered in step 1 were for a group you are now calling *newgroupvar*, and in particular they were for *newgroupvar* = 1. You are also telling `ssd` that you now want to enter the summary statistics for the next group, namely, *newgroupvar* = 2.
3. Enter the summary statistics for the second group in the same way you entered them for the first group, just as outlined in the previous section.
4. If you have a third group, add it by typing

   . ssd addgroup

   In this case, you are telling `ssd` only one thing: that you now want to enter data for the next group, namely, *newgroupvar* = 3.
5. Enter the summary statistics for the third group in the same way you entered them for the second group, and just as outlined in the previous section.
6. If you want to add more groups, continue in the same way. Declare the next group of data by typing

   . ssd addgroup

   and then enter the data by using the `ssd set` command.
7. If you mistakenly add a group and wish to rescind that, type

   . ssd unaddgroup

8. If you wish to go back and modify the values entered for a previous group, put the group number between `ssd set` and what is being set—for instance, type `ssd set 2 observations` . . . —and specify the `replace` option. For instance, to reenter the correlations for group 1, type

   . ssd set 1 correlations *values*, replace

What happens when you do not set all the summary statistics

You are required to set the number of observations and to set the covariances or the correlations. Setting the variances (standard deviations) and setting the means are optional.

1. If you set correlations only, then

   a. Means are assumed to be 0.

   b. Standard deviations are assumed to be 1.

   c. You will not be able to pool across groups if you have group data.

   As a result of (a) and (b), the parameters `sem` estimates will be standardized even when you do not specify `sem`'s `standardized` reporting option. Estimated means and intercepts will be 0.

   Concerning (c), we need to explain. This concerns group data. If you type

   ```
   . sem ...
   ```

   then `sem` fits a model with all the data. `sem` does that whether you have raw data or SSD in memory. If you have SSD with groups—say, males and females or age groups 1, 2, and 3—`sem` combines the summary statistics to obtain the summary statistics for the overall data. It is only possible to do this when covariances and means are known for each group. If you set correlations without variances or standard deviations and without means, the necessary statistics are not known and the groups cannot be combined. Thus if you type

   ```
   . sem ...
   ```

   you will get an error message. You can still estimate using `sem`; you just have to specify on which group you wish to run `sem`, and you do that with the `select()` option:

   ```
   . sem ..., select(#)
   ```

2. If you set correlations and means,

   a. Standard deviations are assumed to be 1.

   b. You will not be able to pool across groups if you have group data.

   This situation is nearly identical to situation 1. The only difference is that estimated means and intercepts will be nonzero.

3. If you set correlations and standard deviations or variances, or if you set covariances only,

   a. Means are assumed to be 0.

   b. You will not be able to pool across groups if you have group data.

   This situation is a little better than situation 1. Estimated intercepts will be 0, but the remaining estimated coefficients will not be standardized unless you specify `sem`'s `standardized` reporting option.

Labeling SSD

You may use the following commands on SSD, and you use them in the same way you would with an ordinary dataset:

1. `rename` *oldvarname newvarname*
   You may rename the variables; see [D] **rename**.
2. `label data "`*dataset label*`"`
   You may label the dataset; see [D] **label**.

3. `label variable` *varname* "*variable label*"
   You may label variables; see [D] **label**.
4. `label values` *groupvarname valuelabelname*
   You may place a value label on the group variable; see [D] **label**. The group variable always takes on the values 1, 2, ...
5. `note:` *my note*
   `note` *varname*`:` *my note*
   You may places notes on the dataset or on its variables; see [D] **notes**.

Do not modify the SSD except by using the `ssd` command. Most importantly, do not drop variables or observations.

Making summary statistics from data for use by others

If you have raw data and wish to make the summary statistics available for subsequent publication, type

```
. ssd build varlist
```

where *varlist* lists the variables you wish to include in the dataset. The SSD will replace the raw data you had in memory. The full syntax is

```
. ssd build varlist if exp in range
```

so you may specify `if` and `in` to restrict the observations that are included.

For instance, to build an SSD for variables `occ_prestige`, `income`, and `social_status`, type

```
. ssd build occ_prestige income social_status
```

If you wish to build the dataset to include separate groups for males and females, type

```
. ssd build occ_prestige income social_status, group(sex)
```

However the `sex` variable was coded in your original data, the two sexes will be now be coded 1 and 2 in the resulting SSD. Which sex is 1 and which is 2 will correspond to however `sort` would have ordered `sex` in its original coding. For instance, if variable `sex` took on values "`male`" and "`female`", the resulting variable `sex` would take on values 1 corresponding to female and 2 corresponding to male.

Once you have built the SSD, you can describe it and list it:

```
. ssd describe
. ssd list
```

See [SEM] **Example 25**.

Reference

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Also see

[SEM] **Intro 10** — Fitting models with survey data

[SEM] **Intro 12** — Convergence problems and how to solve them

[SEM] **Example 2** — Creating a dataset from published covariances

[SEM] **Example 3** — Two-factor measurement model

[SEM] **Example 19** — Creating multiple-group summary statistics data

[SEM] **Example 25** — Creating summary statistics data from raw data

[SEM] **sem option select( )** — Using sem with summary statistics data

[SEM] **ssd** — Making summary statistics data (sem only)

Title

Intro 12 — Convergence problems and how to solve them

Description Remarks and examples Also see

Description

It can be devilishly difficult for software to obtain results for SEMs. Here is what can happen:

```
. sem ...
Variables in structural equation model
  (output omitted)
Fitting target model:
initial values not feasible
r(1400);
```

or,

```
. gsem ...
Fitting fixed-effects model:
Iteration 0:  log likelihood = -914.65237
Iteration 1:  log likelihood = -661.32533
Iteration 2:  log likelihood = -657.18568
  (output omitted)
Refining starting values:
Grid node 0:  log likelihood = .
Grid node 1:  log likelihood = .
Grid node 2:  log likelihood = .
Grid node 3:  log likelihood = .
Fitting full model:
initial values not feasible
r(1400);
```

or,

```
. sem ...
Endogenous variables
  (output omitted)
Fitting target model:
Iteration  1:  log likelihood = ...
.
.
.
Iteration 50:  log likelihood = -337504.44  (not concave)
Iteration 51:  log likelihood = -337503.52  (not concave)
Iteration 52:  log likelihood = -337502.13  (not concave)
.
.
.
Iteration 101: log likelihood = -337400.69  (not concave)
--Break--
r(1);
```

In the first two cases, `sem` and `gsem` gave up. The error message is perhaps informative if not helpful. In the last case, `sem` (it could just as well have been `gsem`) iterated and iterated while producing little improvement in the log-likelihood value. We eventually tired of watching a process that was going nowhere slowly and pressed *Break*.

Now what?

Remarks and examples

Remarks are presented under the following headings:

Is your model identified?
Convergence solutions generically described
Temporarily eliminate option reliability()
Use default normalization constraints
Temporarily eliminate feedback loops
Temporarily simplify the model
Try other numerical integration methods (gsem only)
Get better starting values (sem and gsem)
Get better starting values (gsem)

Is your model identified?

You may not know whether your model is identified. Infinite iteration logs are an indication of lack of identification or of poor starting values:

```
. sem ...
Endogenous variables
  (output omitted)
Fitting target model:
Iteration  1:  log likelihood = ...
.
.
.
Iteration 50:  log likelihood = -337504.44  (not concave)
Iteration 51:  log likelihood = -337503.52  (not concave)
Iteration 52:  log likelihood = -337502.13  (not concave)
.
.
.
Iteration 101: log likelihood = -337400.69  (not concave)
--Break--
r(1);
```

If the problem is lack of identification, the criterion function being optimized (the log likelihood in this case) will eventually stop improving at all and yet `sem` or `gsem` will continue iterating.

If the problem is poor starting values, the criterion function will continue to increase slowly.

So if your model might not be identified, do not press *Break* too soon.

There is another way to distinguish between identification problems and poor starting values. If starting values are to blame, it is likely that a variance estimate will be heading toward 0. If the problem is lack of identification, you are more likely to see an absurd path coefficient.

To distinguish between those two alternatives, you will need to rerun the model and specify the `iterate()` option:

```
. sem ..., ... iterate(100)
  (output omitted)
```

We omitted the output, but specifying `iterate(100)` allowed us to see the current parameter values at the point. We chose to specify `iterate(100)` because we knew that the likelihood function was changing slowly by that point.

If you are worried about model identification, you have a choice: Sit it out and do not press *Break* too soon, or press *Break* and rerun.

If you discover that your model is not identified, see *Identification 1: Substantive issues* in [SEM] **Intro 4**.

Convergence solutions generically described

There are three generic solutions to convergence problems that we will use:

G1. The improved-starting-values procedure:
Obtain the current parameter values from a failed attempt, modify those lousy values to make them better, use the improved values as starting values to try to fit the model again, and repeat as necessary.

G2. The alternative-starting-values procedure:
Simplify the model to produce an easier-to-fit model, fit the simplified model, use the simplified model's solution as starting values to fit the original, more complicated model, and repeat as necessary.

G3. The alternative-software-logic procedure:
Specify strange options that make the software behave differently in hopes that a different approach will produce a solution. `sem` does not have any such strange options, but `gsem` does. In following this approach, it does not matter whether we in fact understand what we are doing because, once we find a solution, we can obtain the parameter values from the successful model and use those values as starting values to fit the model without the strange and confusing options.

Sometimes we will use all three of these procedures. We will talk about convergence and these procedures substantively below, but before we do that, we want to show you how to use a tool that you will need.

The generic solutions share a mechanical step in common, namely, obtaining parameter values from one attempt at fitting the model to use as starting values in a subsequent attempt. Here is how you do that:

```
. sem ..., ...                      // fit one model
. matrix b = e(b)                   // save parameter estimates in b
. ...                               // optionally modify b
. sem ..., ... from(b)              // fit same model or different model
```

If the first `sem` or `gsem` command fails because you pressed *Break* or if the command issued an error message, you must reissue the command adding option `noestimate` or `iterate(#)`. Specify `noestimate` if the failure came early before the iteration log started, and otherwise specify `iterate(#)`, making # the iteration number close to but before the failure occurred:

```
. sem ..., ... noestimate
```

or

```
. sem ..., ... iterate(50)
```

Once you have obtained the parameter values in `b`, you can list `b`,

```
. matrix b = e(b)
. matrix list b
b[1,10]
           x1:        x1:        x2:        x2:        x3:        x3:
            L      _cons          L      _cons          L      _cons
y1          1     99.518  1.1207418     99.954  1.1149982     99.052
            /:         /:         /:         /:
    var(e.x1)  var(e.x2)  var(e.x3)     var(L)
y1  123.31427  97.675646  100.72204  82.311401
```

and you can modify it:

```
. matrix b[1, 10] = 500
. matrix list b
b[1,10]
           x1:        x1:        x2:        x2:        x3:        x3:
            L      _cons          L      _cons          L      _cons
y1          1     99.518  1.1207418     99.954  1.1149982     99.052
            /:         /:         /:         /:
    var(e.x1)  var(e.x2)  var(e.x3)     var(L)
y1  123.31427  97.675646  100.72204        500
```

And, whether you modify it or not, you can use `b` as the starting values for another attempt of the same model or for an attempt of a different model:

```
. sem ..., ... from(b)
```

Temporarily eliminate option reliability()

If you specified `sem`'s or `gsem`'s `reliability()` option, remove it and try fitting the model again. If the model converges, then your estimate of the reliability is too low; see *What can go wrong* in [SEM] **sem and gsem option reliability( )**.

Use default normalization constraints

Let `sem` and `gsem` provide their default normalization constraints. By default, `sem` and `gsem` constrain all latent exogenous variables to have mean 0; constrain all latent endogenous variables to have intercept 0; and constrain the paths from latent variables to the first observed endogenous variable to have coefficient 1.

Replacing any of the above defaults can cause problems, but problems are most likely to arise if you replace the last listed default. Do not constrain path coefficients merely to obtain model identification. Let `sem` choose those constraints. It is possible that the default-coefficient-1 constraint is inappropriate for how you want to interpret your model. Relax it anyway and reimpose it later.

If default constraints solve the problem, you are done unless you want to reimpose your original, alternative constraints. That you do by typing

```
. sem ..., ...           // model with default constraints
. matrix b = e(b)
. sem ..., ... from(b)   // model with desired constraints
```

If you have multiple constraints that you want to reimpose, you may need to do them in sets.

Temporarily eliminate feedback loops

Check whether your model has any feedback loops, such as

```
. sem ... (y1<-y2 x2) (y2<-y1 x3) ...
```

In this example, variable `y1` affects `y2` affects `y1`. Models with such feedback loops are said to be nonrecursive. Assume you had a solution to the above model. The results might be unstable in a substantive sense; see *nonrecursive (structural) model (system)* in [SEM] **Glossary**. The problem is that finding such truly unstable solutions is often difficult and the stability problem manifests itself as a convergence problem.

If you have convergence problems and you have feedback loops, that is not proof that the underlying values are unstable.

Regardless, temporarily remove the feedback loop,

```
. sem ... (y1<-y2 x2) (y2<-   x3) ...
```

and see whether the model converges. If it does, save the parameter estimates and refit the original model with the feedback, but using the saved parameter estimates as starting values.

```
. matrix b = e(b)
. sem ... (y1<-y2 x2) (y2<-y1 x3) ..., ... from(b)
```

If the model converges, the feedback loop is probably stable. If you are using `sem`, you can check for stability with `estat stable`. If the model does not converge, you now must find which other variables need to have starting values modified.

Temporarily simplify the model

At this point, it is difficult to know whether you should temporarily simplify your model or simply proceed with the subsequent steps and come back to simplification later should it be necessary. So proceed in whatever order seems best to you.

The idea of temporarily simplifying the model is to simplify the model, get that model to converge, and use the simplified model's solution as starting values for the more complicated model.

The more orthogonal and independent you can make the pieces of the model, the better. Remove covariances. If you have measurement, fit it separately. Basically, remove whatever you hold most dear, because you are probably looking for subtle and correlated effects.

Once you find a simplified model that converges, do the following:

```
. sem ..., ...           // fit simplified model
. matrix b = e(b)
. sem ..., ... from(b)   // fit original model
```

Try other numerical integration methods (gsem only)

For models with continuous latent variables, `gsem` provides four numerical integration methods just so you can try them. The `intmethod()` option specifies the integration method.

1. `intmethod(mvaghermite)` is the default and performs mean-and-variance adaptive Gauss–Hermite quadrature. It is fast and accurate.
2. `intmethod(mcaghermite)` performs mode-and-curvature adaptive Gauss–Hermite quadrature. It is accurate but not as fast. If you are fitting a multilevel model, there are cases where method 1 will not work but this method will work.

3. `intmethod(ghermite)` performs nonadaptive Gauss–Hermite quadrature. It is less accurate but quicker, and the calculation it makes converges more readily than either of the above methods.
4. `intmethod(laplace)` performs the Laplacian approximation instead of quadrature. It is the least accurate, sometimes the fastest, and the calculation it makes has no convergence issues whatsoever.

Try all of them. We view methods 1 and 2 as completely trustworthy. If your model will only converge with method 3 or 4, we recommend using the result as starting values for method 1 or 2. Thus you might type

```
. gsem ..., ... intmethod(ghermite)
. matrix b = e(b)
. gsem ..., ... from(b)
```

There is another option we should mention, namely, `intpoints(#)`. Methods 1, 2, and 3 default to using seven integration points. You can change that. A larger number of integration points produces more accurate results but does not improve model convergence. You might reduce the number of integration points to, say, 3. A lower number of integration points slightly improves convergence of the model, and it certainly makes model fitting much quicker. Obviously, model results are less accurate. We have been known to use fewer integration points, but mainly because of the speed issue. We can experiment more quickly. At the end, we always return to the default number of integration points.

Get better starting values (sem and gsem)

If you observe,

```
. sem ..., ...
Variables in structural equation model
  (output omitted)
Fitting target model:
initial values not feasible
r(1400);
```

and you are using `sem`, we direct you back to *Temporarily simplify the model*. The problem that we discuss here seldom reveals itself as "initial values not feasible" with `sem`. If you are using `gsem`, we direct you to the next section, *Get better starting values (gsem)*. It is not impossible that what we discuss here is the solution to your problem, but it is unlikely.

We discuss here problems that usually reveal themselves by producing an infinite iteration log:

```
. sem ..., ...
Endogenous variables
  (output omitted)
Fitting target model:
Iteration  1:  log likelihood = ...
.
.
.
Iteration 50:  log likelihood = -337504.44  (not concave)
Iteration 51:  log likelihood = -337503.52  (not concave)
Iteration 52:  log likelihood = -337502.13  (not concave)
.
.
.
Iteration 101: log likelihood = -337400.69  (not concave)
--Break--
r(1);
```

The first thing to do is look at the parameter values. To do that, type

```
. sem ..., ... iterate(100)
```

We specified 100; you should specify an iteration value based on your log.

In most cases, you will discover that you have a variance of a latent exogenous variable going to 0, or you have a variance of an error (`e.`) variable going to 0.

Based on what you see, say that you suspect the problem is with the variance of the error of a latent endogenous variable `F` going to 0, namely, `e.F`. You need to give that variance a larger starting value, which you can do by typing

```
. sem ..., ... var(e.F, init(1))
```

or

```
. sem ..., ... var(e.F, init(2))
```

or

```
. sem ..., ... var(e.F, init(10))
```

We recommend choosing a value for the variance that is larger than you believe is necessary.

To obtain that value,

1. If the variable is observed, use `summarize` to obtain the summary statistics for the variable, square the reported standard deviation, and then increase that by, say, 20%.
2. If the variable is latent, use `summarize` to obtain a summary of the latent variable's anchor variable and then follow the same rule: use `summarize`, square the reported standard deviation, and then increase that by 20%. (The anchor variable is the variable whose path is constrained to have coefficient 1.)
3. If the variable is latent and has paths only to other latent variables so that its anchor variable is itself latent, follow the anchor's paths to an observed variable and follow the same rule: use `summarize`, square the reported standard deviation, and then increase that by 20%.
4. If you are using `gsem` to fit a multilevel model and the latent variable is at the observational level, follow advice 2 above. If the latent variable is at a higher level—say `school`—and its anchor is `x`, a Gaussian response with the identity link, type

```
. sort school
. by school: egen avg = mean(x)
. by school: generate touse = _n==1 if school<.
. summarize avg if touse==1
```

Square the reported standard deviation and add 20%.

5. If you are using `gsem` and the anchor of the latent variable is not Gaussian with the identity link, see the next section.

Do not dismiss the possibility that the bad starting values concern estimated parameters other than variances of latent exogenous or error variables, although variances of those kinds of variables is the usual case. Covariances are rarely the problem because covariances can take on any value and, whether too small or too large, usually get themselves back on track as the iterative process proceeds. If you need to specify an initial value for a covariance, the syntax is

```
. sem ..., ... cov(e.F*e.G, init(-25))
```

Substitute for −25 the value you consider reasonable.

The other possibility is that a path coefficient needs a better starting value, which is as unlikely as a covariance being the problem, and for the same reasons. To set the initial value of a path coefficient, add the `init()` option where the path is specified. Say the original `sem` command included `y<-x1`:

```
. sem ... (y<-x1 x2) ...
```

If you wanted to set the initial value of the path from `x1` to 3, modify the command to read

```
. sem ... (y<-(x1, init(3)) x2) ...
```

Get better starting values (gsem)

What we say below applies regardless of how the convergence problem revealed itself. You might have seen the error message "initial values not feasible", or some other error message, or you might have an infinite iteration log.

`gsem` provides two options to help you obtain better starting values: `startvalues()` and `startgrid()`.

For models with continuous latent variables, `startvalues(`*svmethod*`)` allows you to specify one of five starting-value calculation methods: `zero`, `constantonly`, `fixedonly`, `ivloadings`, and `iv`. By default, `gsem` uses `ivloadings`. Evidently, that did not work for you. Try the others, starting with `iv`:

```
. gsem ..., ... startvalues(iv)
```

If that does not solve the problem, proceed through the others in the following order: `fixedonly`, `constantonly`, and `zero`.

For models with categorical latent variables, `startvalues(`*svmethod*`)` allows you to specify one of seven starting-value calculation methods: `factor`, `classid`, `randomid`, `classpr`, `randompr`, `jitter`, and `zero`. By default, `gsem` uses `factor`.

If you have a variable, say, *guessid*, that provides a guess for the latent class membership of each observation, try

```
. gsem ..., ... startvalues(classid guessid)
```

You might even use another command, such as `cluster kmeans` with the `generate()` option to create a new variable to specify in place of *guessid*.

The option `startvalues(randomid)` provides a function similar to `startvalues(classid` *guessid*`)` but uses random guesses for the latent class membership. You may find it helpful to try multiple random guesses and use the one with the best log likelihood after a given number of EM iterations for starting values. To do this, specify the `draws(#)` suboption. For instance, to try 10 random guesses, you can specify `startvalues(randomid, draws(10))`. You can also specify the number of EM iterations that should be taken before deciding which model is best. To do this, specify the `emopts(iterate(#))` option.

If you have a set of variables, say, *guessprlist*, that provide guesses for the probabilities for latent class membership in each observation, try

```
. gsem ..., ... startvalues(classpr guessprlist)
```

The option `startvalues(randompr)` provides a similar function but uses random guesses for the latent class probabilities. You can also specify the `draws(#)` suboption here to specify the number of random guesses that should be tried. For instance, `startvalues(randompr, draws(10))` specifies that 10 random guesses for class probabilities should be tried and that the one with the best log likelihood after the EM iterations should be used as starting values.

The option `startvalues(jitter)` applies random adjustments to starting values computed from a normal approximation to each outcome.

The option `startvalues(zero)` is really only useful if you have your own set of starting values and you want `gsem` to use zero for any model parameter without a specified value.

Whether you have continuous or categorical latent variables, if you have starting values for some parameters but not others—perhaps you fit a simplified model to get them—you can combine the options `startvalues()` and `from()`:

```
. gsem ..., ...                              // simplified model
. matrix b = e(b)
. gsem ..., ... from(b) startvalues(iv)      // full model
```

You can combine `startvalues()` with the `init()` option, too. We described `init()` in the previous section.

The other special option `gsem` provides is `startgrid()`. This option is mostly helpful for models with continuous latent variables, but not so much for models with categorical latent variables. `startgrid()` can be used with or without `startvalues()`. `startgrid()` is a brute-force approach that tries various values for variances and covariances and chooses the ones that work best.

1. You may already be using a default form of `startgrid()` without knowing it. If you see `gsem` displaying Grid node 1, Grid node 2, ... following Grid node 0 in the iteration log, that is `gsem` doing a default search because the original starting values were not feasible.

   The default form tries 0.1, 1, and 10 for all variances of all latent variables, by which we mean the variances of latent exogenous variables and the variances of errors of latent endogenous variables.

2. `startgrid(`*numlist*`)` specifies values to try for variances of latent variables.

3. `startgrid(`*covspec*`)` specifies the particular variances and covariances in which grid searches are to be performed. Variances and covariances are specified in the usual way. `startgrid(e.F e.F*e.L M1[school] G*H e.y e.y1*e.y2)` specifies that 0.1, 1, and 10 be tried for each member of the list.

4. `startgrid(`*numlist covspec*`)` allows you to combine the two syntaxes, and you can specify multiple `startgrid()` options so that you can search the different ranges for different variances and covariances.

Our advice to you is this:

1. If you got an iteration log and it did not contain Grid node 1, Grid node 2, . . . , then specify `startgrid(.1 1 10)`. Do that whether the iteration log was infinite or ended with some other error. In this case, we know that `gsem` did not run `startgrid()` on its own because it did not report Grid node 1, Grid node 2, etc. Your problem is poor starting values, not infeasible ones.

   A synonym for `startgrid(.1 1 10)` is just `startgrid` without parentheses.

   Be careful, however, if you have a large number of continuous latent variables. `startgrid` could run a long time because it runs all possible combinations. If you have 10 latent variables, that means $3^10 = 59{,}049$ likelihood evaluations.

   If you have a large number of continuous latent variables, rerun your difficult `gsem` command including option `iterate(#)` and look at the results. Identify the problematic variances and search across them only. Do not just look for variances going to 0. Variances getting really big can be a problem, too, and even reasonable values can be a problem. Use your knowledge and intuition about the model.

   Perhaps you will try to fit your model by specifying `startgrid(.1 1 10 e.F L e.X)`. Because values 0.1, 1, and 10 are the default, you could equivalently specify `startgrid(e.F L e.X)`.

   Look at covariances as well as variances. If you expect a covariance to be negative and it is positive, try negative starting values for the covariance by specifying `startgrid(-.1 -1 -10 G*H)`.

   Remember that you can have multiple `startgrid()` options, and thus you could specify `startgrid(e.F L e.X) startgrid(-.1 -1 -10 G*H)`.

2. If you got "initial values not feasible", you know that `gsem` already tried the default `startgrid`.

   The default `startgrid` only tried the values 0.1, 1, and 10, and only tried them on the variances of latent variables. You may need to try different values or try the same values on covariances or variances of errors of observed endogenous variables.

   We suggest you first rerun the model causing difficulty including the `noestimate` option. We also direct you back to the idea of first simplifying the model; see *Temporarily simplify the model*.

   If, looking at the results, you have an idea of which variance or covariance is a problem, or if you have few variances and covariances, we would recommend running `startgrid()` first. On the other hand, if you have no idea as to which variance or covariance is the problem and you have a large number of them, you will be better off if you first simplify the model. If, after doing that, your simplified model does not include all the variances and covariances, you can specify a combination of `from()` and `startgrid()`.

Also see

[SEM] **Intro 11** — Fitting models with summary statistics data (sem only)

Title

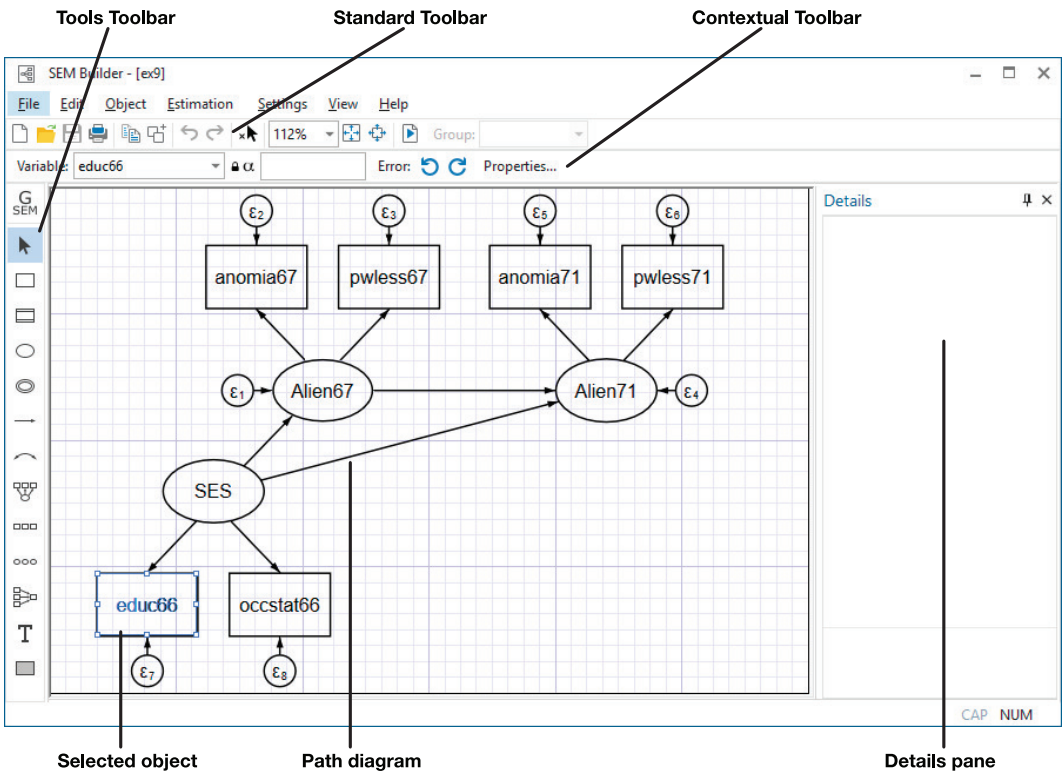
Builder — SEM Builder

Description Remarks and examples Reference

Description

The SEM Builder lets you create path diagrams for SEMs, fit those models, and show results on the path diagram. Here we discuss standard linear SEMs; see [SEM] **Builder, generalized** for information on using the Builder to create models with generalized responses and multilevel structure.

Remarks and examples



Launch the SEM Builder by selecting the menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**. You can also type `sembuilder` in the Command window.

Select the Add observed variable tool, ▭, and click within the diagram to add an observed variable. Type a name in the *Name* control of the Contextual Toolbar to name the variable. If the variable is not placed exactly where you want it, simply select the Select tool and drag the variable to your preferred location.

After adding the variable, use the *Variable* control in the Contextual Toolbar to select a variable from your dataset or type a variable name into the edit field.

Add latent variables to the model by using the tool.

Clicking within the diagram with either the or tool places a variable of the default size and shape. If you instead wish to create a variable of custom size and shape, click and hold the mouse button and drag until the variable is your preferred size. Select the **Properties...** button in the Contextual Toolbar or double-click on a variable to launch a dialog box where you can change even more properties of the variable. You can customize the size, shape, label font, and other appearance characteristics of the variable from the **Appearance** tab of the dialog, but you will rarely do that.

More often, you will want to change the appearance of all variables or a class of variables from the **Settings** menu. From the **Settings** menu, you can change any aspect of the appearance of variables. The **Settings > Variables** menu lets you change the appearance of **All** variables, **Latent** variables, **Observed** variables, **Latent Exogenous** variables, **Latent Endogenous** variables, **Observed Exogenous** variables, **Observed Endogenous** variables, **Error** variables, **Latent Error** variables, and **Observed Error** variables.

Draw paths between variables with the tool. Simply click in the source variable's box or oval and drag to the target variable's box or oval. The new path can connect variables either along the line between their center points or at the edge nearest where you click and release. This connection behavior is set in the **Settings > Automation...** dialog.

Whenever a path is connected to an exogenous variable, that variable becomes endogenous and an error variable is created. If you do not like the direction of the error variable relative to the new endogenous variable, first use the tool to select the endogenous variable and then use the buttons in the Contextual Toolbar to rotate the error variable.

Draw covariances between variables with the tool. If you drag to the right, the curve of the covariance will bend up. If you drag to the left, the curve will bend down. If you drag down, the curve will bend right, and if you drag up, it will bend left. That does not matter greatly; if the curve bends the opposite of what you want, simply click the **Mirror** button in the Contextual Toolbar.

As with variables, you can change more properties of a path or covariance (connection) by double-clicking on it or by selecting Properties from the Contextual Toolbar. You can also change the appearance of all connections from the **Settings > Connections** menu. From **Settings > Connections**, you can change the appearance of **All** connections, all **Paths**, all **Covariances**, or all **Error Paths**.

Manage variables and connections on the diagram with the Select tool, . As you select an object, its Contextual Toolbar will appear. You can change the name and other properties of selected variables or the properties of selected connections. You can also drag-select or hold the *Shift* key and click to select multiple objects. Changing a property from the Contextual Toolbar or **Properties** dialog will affect all the selected objects. You can also drag selected variables as a group to other locations on the diagram.

You can place a measurement model (or measurement component of a larger model) on the diagram with the tool. Clicking on the diagram with this tool launches a dialog box where you can name the latent variable, select or type the names of the measurement variables, and specify the direction of the measurements relative to the latent variable. You can even set the spacing between the measurement variables and the distance from the latent variable to the measurements. The tool creates measurement components quickly and with even spacing. Similarly, neatly organized sets of observed variables can be placed with the tool; sets of latent variables, with the tool; and regression components, with the tool.

Place annotations and other text by using the T tool.

We have ignored the locks, 🔒, in the Contextual Toolbars. These locks apply constraints to the parameters of the SEM. You can constrain variances, 🔒σ^2; means, 🔒μ; intercepts, 🔒α; and path coefficients and covariances, 🔒β. For example, select an exogenous variable or an error variable and type a number in 🔒σ^2 to constrain that variance to a fixed value. Or, select three path variables and type a name (a placeholder) in 🔒β to constrain all the path coefficients to be equal. You can type numbers, names, or linear expressions in the 🔒 controls. The linear expressions can involve only numbers and names that are used in other 🔒 controls.

Do not be afraid to try things. If you do not know what a tool, control, or dialog item does, try it. If you do not like the result, click on the **Undo** button, ↶, in the Standard Toolbar.

Click on ▶ in the Standard Toolbar to fit the model. A dialog is launched that allows you to set all the estimation options that are not defined by the path diagram. After estimation, some of the estimation results are displayed on the path diagram. Use the **Results** tab of the **Settings > Variables** and **Settings > Connections** dialogs to change what results are shown and how they appear (font sizes, locations, etc.). Also, as you click on connections or variables, the Details pane displays all the estimation results for the selected object.

If you wish to create another model derived from the current model, click ⧉+ in the Standard Toolbar.

Video example

SEM Builder in Stata

Visit the Stata YouTube channel for more videos.

Reference

Huber, C. 2012. Using Stata's SEM features to model the Beck Depression Inventory. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2012/10/17/using-statas-sem-features-to-model-the-beck-depression-inventory/.

Title

Builder, generalized — SEM Builder for generalized models

Description Remarks and examples Reference

Description

The SEM Builder allows you to create path diagrams of generalized SEMs, fit those models, and show the results on the path diagram. Here we extend the discussion from [SEM] **Builder**. Read that manual entry first.

Remarks and examples

Launch the SEM Builder by selecting the menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**. You can also type `sembuilder` in the Command window. Switch the Builder into generalized SEM mode by clicking on the **Change to generalized SEM** button, GSEM. The generalized SEM mode allows you to add generalized responses and multilevel latent variables to your path diagram. This mode also disallows constraints on the means or variances of observed exogenous variables and on the variances of errors. In technical terms, the generalized mode draws and fits `gsem` models rather than `sem` models; see [SEM] **Intro 1** for a discussion.

Aside from activating tools to add generalized responses and multilevel latent variables, everything about the Builder works the same in generalized SEM mode as it did in standard SEM mode. You can still add observed variables (▭) and latent variables (⬭) and create connections using paths (→) and covariances (⌒).

You can now add to your diagram a special kind of observed variable called a generalized response variable by using the ▭ tool. A generalized response is a dependent variable that may be continuous, count, binary, ordinal, or multinomial, and whose statistical properties are described by its family and link; see [SEM] **Intro 2** for a discussion. Simply click on the ▭ tool and then click in the diagram to add a generalized response variable. After adding the generalized response, use the Contextual Toolbar to do the following: 1) select its *Family/Link* and 2) select its variable from the *Variable* control. You may also type a variable name into the *Variable* control.

You can also add multilevel latent variables—random effects assumed to have a Gaussian distribution whose realizations are at the level of a grouping variable or variables. See [SEM] **Intro 2** for a discussion. Click on the ◎ tool and then click on the diagram to add a multilevel latent variable. After adding the variable, click on the ··· button of the *Name/Level* control in the Contextual Toolbar. In the resulting dialog box, choose the nesting depth of your latent variable, and then choose the variables that define the nesting. For example, for a three-level model of doctors nesting patients nesting observations, you would select `3` in the *Nesting depth* control, select `doctor` in the first variable control, and select `patient` in the second variable control, leaving `> Observations`, which is fixed. You do not need to change the default *Base name*, but you may type a different name of your choice. Do read the note on the dialog carefully because it provides guidance on successfully creating multilevel models. Also see [SEM] **Example 39g** for an example that demonstrates creating a nested model in the Builder.

As with standard observed and latent variables, you can also click and drag as you add generalized responses and multilevel latent variables to control the size of the resulting variables. You can also select the **Properties...** button from the Contextual Toolbar or double-click on a variable to change even more properties of the variable. As with standard variables, you can control their appearance from the **Appearance** tab of the resulting dialog box or, more likely, control the appearance of all variables of a given type from the **Settings > Variables** menu.

As with standard models, you create the structure of your model by connecting the variables with the tool. You may also specify covariances among variables with the tool.

There is one other construct available in generalized mode that is not available in standard mode—paths to other paths. In addition to creating paths from a latent variable to an observed variable or to another latent variable at the same nesting level, you can connect a standard or multilevel latent variable to a path from one observed variable to another observed variable. This makes that path a random path, or a random slope. As with other path connections, think of the path as adding the source variable to the target. So we are adding a random variable to a path (or slope), making it a random path. See [SEM] **Example 38g** for an example of creating a random path in the Builder.

In generalized mode, the Add Measurement Component tool, , allows the measurements to be created as generalized responses. Simply click on the *Make measurements generalized* box on the dialog box and select a *Family/Link*. It also allows the latent variable to be either standard observation-level or multilevel with the radio control at the top of the dialog box.

The Add observed variables set tool, , also allows you to create multiple generalized responses. Click on the *Make variables generalized responses* box on the dialog box and select a *Family/Link*.

Creating multinomial logit responses can be tricky. You must create a generalized response for each level of your outcome and use factor-variable notation to designate the levels of the outcome. The tool is the safest way to create a multinomial logit response. With your dataset in memory, select the tool and then click on the diagram. Click on the *Make variables generalized responses* box on the dialog box, and then select `Multinomial, Logit` in the *Family/Link* control. Select your multinomial response variable from the *Variable* control, and the levels for your multinomial response will be automatically populated. The level designated with a `b` will be the base level. You can type the `b` on another level to make it the base. When you click **OK**, multinomial generalized responses will be created for each level of your variable. See [SEM] **Example 37g** for an example of creating multinomial logit responses in the Builder.

The Add Regression Component tool, , also allows the dependent variable to be specified as a generalized response.

Place shaded boxes on the diagram with the tool. This tool can be used in both standard and generalized mode. It is most useful in the generalized mode, where we can shade different nesting levels of multilevel models.

As in standard mode, constraints are specified with lock controls , and you fit the model by using the button on the Standard Toolbar.

Video example

SEM Builder in Stata

Visit the Stata YouTube channel for more videos.

Reference

Huber, C. 2012. Using Stata's SEM features to model the Beck Depression Inventory. *The Stata Blog: Not Elsewhere Classified*. http://blog.stata.com/2012/10/17/using-statas-sem-features-to-model-the-beck-depression-inventory/.

Title

estat eform — Display exponentiated coefficients

Description Menu Syntax Options
Remarks and examples Also see

Description

`estat eform` is for use after `gsem` but not `sem`.

`gsem` reports coefficients. You can obtain exponentiated coefficients and their standard errors by using `estat eform` after estimation to redisplay results.

Menu

Statistics > SEM (structural equation modeling) > Other > Display exponentiated coefficients

Syntax

`estat eform` [*eqnamelist*] [`,` `level(#)` *display_options*]

where *eqnamelist* is a list of equation names. In `gsem`, equation names correspond to the names of the response variables. If no *eqnamelist* is specified, exponentiated results for the first equation are shown.

Options

`level(#)`; see [R] **Estimation options**; default is `level(95)`.

display_options control the display of factor variables and more. Allowed *display_options* are `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(style)`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`. See [R] **Estimation options**.

Remarks and examples

In some generalized linear response functions, exponentiated coefficients have a special meaning. Those special meanings are as follows:

Common name	Family	Link	Meaning of exp(coef)
logit	Bernoulli	logit	odds ratio
ologit	ordinal	logit	odds ratio
mlogit	multinomial	logit	relative-risk ratio
Poisson	Poisson	log	incidence-rate ratio
nbreg	nbreg	log	incidence-rate ratio

Survival distribution	Meaning of exp(coef)
exponential	hazard ratio
Weibull	hazard ratio
gamma	time ratio
loglogistic	time ratio
lognormal	time ratio

See [SEM] **Example 33g** and [SEM] **Example 34g**.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **Example 33g** — Logistic regression

[SEM] **Example 34g** — Combined models (generalized responses)

[SEM] **Example 47g** — Exponential survival model

[SEM] **Example 48g** — Loglogistic survival model with censored and truncated data

Title

estat eqgof — Equation-level goodness-of-fit statistics

Description | Menu | Syntax | Option
Remarks and examples | Stored results | Reference | Also see

Description

`estat eqgof` is for use after `sem` but not `gsem`.

`estat eqgof` displays equation-by-equation goodness-of-fit statistics. Displayed are R^2 and the Bentler–Raykov squared multiple-correlation coefficient (Bentler and Raykov 2000).

These two concepts of fit are equivalent for recursive SEMs and univariate linear regression. For nonrecursive SEMs, these measures are distinct.

Equation-level variance decomposition is also reported, along with the overall model coefficient of determination.

Menu

Statistics > SEM (structural equation modeling) > Goodness of fit > Equation-level goodness of fit

Syntax

`estat eqgof` [`,` `format(`*%fmt*`)`]

Option

`format(`*%fmt*`)` specifies the display format. The default is `format(%9.0f)`.

Remarks and examples

See [SEM] **Example 3**.

In rare circumstances, these equation-level goodness-of-fit measures in nonrecursive structural equations have unexpected values. It is possible to obtain negative R^2 and multiple-correlation values.

It is recommended to use the Bentler–Raykov squared multiple correlations as a measure of explained variance for nonrecursive systems that involve endogenous variables with reciprocal causations.

Stored results

`estat eqgof` stores the following in `r()`:

Scalars

`r(N_groups)`	number of groups
`r(CD[_#])`	overall coefficient of determination (for group #)

Matrices

`r(nobs)`	sample size for each group
`r(eqfit[_#])`	fit statistics (for group #)

Reference

Bentler, P. M., and T. Raykov. 2000. On measures of explained variance in nonrecursive structural equation models. *Journal of Applied Psychology* 85: 125–131.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat gof** — Goodness-of-fit statistics

[SEM] **estat ggof** — Group-level goodness-of-fit statistics

[SEM] **Example 3** — Two-factor measurement model

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat eqtest — Equation-level tests that all coefficients are zero

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`estat eqtest` is for use after `sem` but not `gsem`.

`estat eqtest` displays Wald tests that all coefficients excluding the intercept are 0 for each equation in the model.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Equation-level Wald tests

Syntax

`estat eqtest [, total nosvyadjust]`

Options

`total` is for use when estimation was with `sem, group()`. It specifies that the tests be aggregated across the groups.

`nosvyadjust` is for use with `svy` estimation commands. It specifies that the Wald tests be carried out without the default adjustment for the design degrees of freedom. That is to say the tests are carried out as $W/k \sim F(k,d)$ rather than as $(d-k+1)W/(kd) \sim F(k,d-k+1)$, where k is the dimension of the tests and d is the total number of sampled PSUs minus the total number of strata.

Remarks and examples

See [SEM] **Example 13**.

Stored results

`estat eqtest` stores the following in `r()`:

Scalars
- `r(N_groups)` number of groups

Matrices
- `r(nobs)` sample size for each group
- `r(test[_#])` test statistics (for group #)
- `r(test_total)` aggregated test statistics (`total` only)

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **lrtest** — Likelihood-ratio test of linear hypothesis

[SEM] **test** — Wald test of linear hypotheses

[SEM] **Example 13** — Equation-level Wald test

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat framework — Display estimation results in modeling framework

Description Menu Syntax Options
Remarks and examples Stored results Reference Also see

Description

`estat framework` is a postestimation command for use after `sem` but not `gsem`.

`estat framework` displays the estimation results as a series of matrices derived from the Bentler–Weeks form; see Bentler and Weeks (1980).

Menu

Statistics > SEM (structural equation modeling) > Other > Report model framework

Syntax

`estat framework` [, *options*]

options	Description
`standardized`	report standardized results
`compact`	display matrices in compact form
`fitted`	include fitted means, variances, and covariances
`format(%`*fmt*`)`	display format to use

Options

`standardized` reports results in standardized form.

`compact` displays matrices in compact form. Zero matrices are displayed as a description. Diagonal matrices are shown as a row vector.

`fitted` displays the fitted mean and covariance values.

`format(%`*fmt*`)` specifies the display format to be used. The default is `format(%9.0g)`.

Remarks and examples

See [SEM] **Example 11**.

❑ Technical note

If sem's nm1 option was specified when the model was fit, all covariance matrices are calculated using $N - 1$ in the denominator instead of N.

❑

Stored results

estat framework stores the following in r():

Scalars

r(N_groups)	number of groups
r(standardized)	indicator of standardized results (+)

Matrices

r(nobs)	sample size for each group
r(Beta[_#])	coefficients of endogenous variables on endogenous variables (for group #)
r(Gamma[_#])	coefficients of endogenous variables on exogenous variables (for group #)
r(alpha[_#])	intercepts (for group #) (*)
r(Psi[_#])	covariances of errors (for group #)
r(Phi[_#])	covariances of exogenous variables (for group #)
r(kappa[_#])	means of exogenous variables (for group #) (*)
r(Sigma[_#])	fitted covariances (for group #)
r(mu[_#])	fitted means (for group #) (*)

(+) If r(standardized)=1, the returned matrices contain standardized values.

(*) If there are no estimated means or intercepts in the sem model, these matrices are not returned.

Reference

Bentler, P. M., and D. G. Weeks. 1980. Linear structural equations with latent variables. *Psychometrika* 45: 289–308.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **Intro 7** — Postestimation tests and predictions (*Replaying the model (sem and gsem)*)

[SEM] **Intro 7** — Postestimation tests and predictions (*Accessing stored results*)

[SEM] **Example 11** — estat framework

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat ggof — Group-level goodness-of-fit statistics

Description Menu Syntax Option
Remarks and examples Stored results Also see

Description

`estat ggof` is for use after estimation with `sem, group()`.

`estat ggof` displays, by group, the standardized root mean squared residual (SRMR), the coefficient of determination (CD), and the model versus saturated χ^2 along with its associated degrees of freedom and p-value.

Menu

Statistics > SEM (structural equation modeling) > Group statistics > Group-level goodness of fit

Syntax

`estat ggof` [`, format(%`*fmt*`)`]

Option

`format(%`*fmt*`)` specifies the display format. The default is `format(%9.3f)`.

Remarks and examples

See [SEM] **Example 21**.

`estat ggof` provides group-level goodness-of-fit statistics after estimation by `sem, group()`; see [SEM] **sem group options**.

The SRMR, CD, and χ^2 statistics are not computed for models fit by `gsem`; therefore, `estat ggof` is not for use after estimation with `gsem, group()`.

Stored results

`estat ggof` stores the following in `r()`:

Scalars
- `r(N_groups)` number of groups

Matrices
- `r(gfit)` fit statistics
- `r(gfit_sb)` Satorra–Bentler scaled fit statistics

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat eqgof** — Equation-level goodness-of-fit statistics

[SEM] **estat gof** — Goodness-of-fit statistics

[SEM] **Example 21** — Group-level goodness of fit

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

[SEM] **sem group options** — Fitting models on different groups

Title

estat ginvariant — Tests for invariance of parameters across groups

Description Menu Syntax Options
Remarks and examples Stored results References Also see

Description

`estat ginvariant` is for use after estimation with `sem, group()`; see [SEM] **sem group options**.

`estat ginvariant` performs score tests (Lagrange multiplier tests) and Wald tests of whether parameters constrained to be equal across groups should be relaxed and whether parameters allowed to vary across groups could be constrained.

See Sörbom (1989) and Wooldridge (2010, 421–428).

Menu

Statistics > SEM (structural equation modeling) > Group statistics > Test invariance of parameters across groups

Syntax

`estat ginvariant` [, *options*]

options	Description
`showpclass(`*pclassname*`)`	restrict output to parameters in the specified parameter class
`class`	include joint tests for parameter classes
`legend`	include legend describing parameter classes

pclassname	Description
`scoef`	structural coefficients
`scons`	structural intercepts
`mcoef`	measurement coefficients
`mcons`	measurement intercepts
`serrvar`	covariances of structural errors
`merrvar`	covariances of measurement errors
`smerrcov`	covariances between structural and measurement errors
`meanex`	means of exogenous variables
`covex`	covariances of exogenous variables
`all`	all the above
`none`	none of the above

Options

`showpclass(`*pclassname*`)` displays tests for the classes specified. `showpclass(all)` is the default.

`class` displays a table with joint tests for group invariance for each of the nine parameter classes.

`legend` displays a legend describing the parameter classes. This option may only be used with the `class` option.

Remarks and examples

See [SEM] **Example 22**.

Score tests are not available after `gsem`; therefore, `estat ginvariant` is not for use after estimation with `gsem, group()`.

Stored results

`estat ginvariant` stores the following in `r()`:

Scalars

`r(N_groups)`	number of groups

Matrices

`r(nobs)`	sample size for each group
`r(test)`	Wald and score tests
`r(test_pclass)`	parameter classes corresponding to `r(test)`
`r(test_class)`	joint Wald and score tests for each class

References

MacDonald, K. 2016. Group comparisons in structural equation models: Testing measurement invariance. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/08/23/group-comparisons-in-structural-equation-models-testing-measurement-invariance/.

Sörbom, D. 1989. Model modification. *Psychometrika* 54: 371–384.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat mindices** — Modification indices

[SEM] **estat scoretests** — Score tests

[SEM] **Example 22** — Testing parameter equality across groups

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat gof — Goodness-of-fit statistics

Description | Menu | Syntax | Options
Remarks and examples | Stored results | References | Also see

Description

`estat gof` is for use after `sem` but not `gsem`.

`estat gof` displays a variety of overall goodness-of-fit statistics.

Menu

Statistics > SEM (structural equation modeling) > Goodness of fit > Overall goodness of fit

Syntax

`estat gof` [, *options*]

options	Description
`stats(`*statlist*`)`	statistics to be displayed
`nodescribe`	suppress descriptions of statistics

statlist	Description
`chi2`	χ^2 tests; the default
`rmsea`	root mean squared error of approximation
`ic`	information indices
`indices`	indices for comparison against baseline
`residuals`	measures based on residuals
`all`	all the above

Note: The statistics reported by `chi2`, `rmsea`, and `indices` are dependent on the assumption of joint normality of the observed variables. If `vce(sbentler)` is specified with `sem`, modified versions of these statistics that are computed using the Satorra–Bentler scaled χ^2 statistics will also be reported.

Options

`stats(`*statlist*`)` specifies the statistics to be displayed. The default is `stats(chi2)`.

`stats(chi2)` reports the model versus saturated test and the baseline versus saturated test. The saturated model is the model that fits the covariances perfectly.

The model versus saturated test is a repeat of the test reported at the bottom of the `sem` output.

In the baseline versus saturated test, the baseline model includes the means and variances of all observed variables plus the covariances of all observed exogenous variables. For a covariance model (a model with no endogenous variables), the baseline includes only the means and variances of observed variables. Be aware that different authors define the baseline model differently.

`stats(rmsea)` reports the root mean squared error of approximation (RMSEA) and its 90% confidence interval, and pclose, the p-value for a test of close fit, namely, RMSEA < 0.05. Most interpreters of this test label the fit close if the lower bound of the 90% CI is below 0.05 and label the fit poor if the upper bound is above 0.10. See Browne and Cudeck (1993).

`stats(ic)` reports the Akaike information criterion (AIC) and Bayesian (or Schwarz) information criterion (BIC). These statistics are available only after estimation with `sem method(ml)` or `method(mlmv)`. These statistics are used not to judge fit in absolute terms but instead to compare the fit of different models. Smaller values indicate a better fit. Be aware that there are many variations (minor adjustments) to statistics labeled AIC and BIC. Reported here are statistics that match `estat ic`; see [R] **estat ic**.

To compare models that use statistics based on likelihoods, such as AIC and BIC, models should include the same variables; see [SEM] **lrtest**. See Akaike (1987), Schwarz (1978), and Raftery (1993).

`stats(indices)` reports CFI and TLI, two indices such that a value close to 1 indicates a good fit. CFI stands for comparative fit index. TLI stands for Tucker–Lewis index and is also known as the nonnormed fit index. See Bentler (1990).

`stats(residuals)` reports the standardized root mean squared residual (SRMR) and the coefficient of determination (CD).

A perfect fit corresponds to an SRMR of 0. A good fit is a small value, considered by some to be limited to 0.08.

Concerning CD, a perfect fit corresponds to a CD of 1. CD is like R^2 for the whole model.

`stats(all)` reports all the statistics. You can also specify just the statistics you wish reported, such as

```
. estat gof, stats(indices residuals)
```

`nodescribe` suppresses the descriptions of the goodness-of-fit measures.

Remarks and examples

See [SEM] **Example 4**.

Stored results

`estat gof` stores the following in `r()`:

Scalars

`r(chi2_ms)`	test of target model against saturated model
`r(df_ms)`	degrees of freedom for `r(chi2_ms)`
`r(p_ms)`	p-value for `r(chi2_ms)`
`r(chi2sb_ms)`	Satorra–Bentler scaled test of target model against saturated model
`r(psb_ms)`	p-value for `r(chi2sb_ms)`
`r(chi2_bs)`	test of baseline model against saturated model
`r(df_bs)`	degrees of freedom for `r(chi2_bs)`
`r(p_bs)`	p-value for `r(chi2_bs)`
`r(chi2sb_bs)`	Satorra–Bentler scaled test of baseline model against saturated model
`r(psb_bs)`	p-value for `r(chi2sb_bs)`
`r(rmsea)`	root mean squared error of approximation
`r(lb90_rmsea)`	lower bound of 90% CI for RMSEA
`r(ub90_rmsea)`	upper bound of 90% CI for RMSEA
`r(pclose)`	p-value for test of close fit: RMSEA $<$ 0.05
`r(rmsea_sb)`	RMSEA using Satorra–Bentler χ^2
`r(aic)`	Akaike information criterion
`r(bic)`	Bayesian information criterion
`r(cfi)`	comparative fit index
`r(cfi_sb)`	CFI using Satorra–Bentler χ^2
`r(tli)`	Tucker–Lewis fit index
`r(tli_sb)`	TLI using Satorra–Bentler χ^2
`r(cd)`	coefficient of determination
`r(srmr)`	standardized root mean squared residual
`r(N_groups)`	number of groups

Matrices

`r(nobs)`	sample size for each group

References

Akaike, H. 1987. Factor analysis and AIC. *Psychometrika* 52: 317–332.

Bentler, P. M. 1990. Comparative fit indexes in structural models. *Psychological Bulletin* 107: 238–246.

Browne, M. W., and R. Cudeck. 1993. Alternative ways of assessing model fit. Reprinted in *Testing Structural Equation Models*, ed. K. A. Bollen and J. S. Long, pp. 136–162. Newbury Park, CA: SAGE.

Raftery, A. E. 1993. Bayesian model selection in structural equation models. Reprinted in *Testing Structural Equation Models*, ed. K. A. Bollen and J. S. Long, pp. 163–180. Newbury Park, CA: SAGE.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat eqgof** — Equation-level goodness-of-fit statistics

[SEM] **estat ggof** — Group-level goodness-of-fit statistics

[SEM] **estat residuals** — Display mean and covariance residuals

[SEM] **Example 4** — Goodness-of-fit statistics

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

[R] **estat ic** — Display information criteria

Title

estat lcgof — Latent class goodness-of-fit statistics

Description Menu Syntax Option
Remarks and examples Stored results References Also see

Description

`estat lcgof` is for use after `gsem` but not `sem`.

`estat lcgof` displays a variety of overall goodness-of-fit statistics.

Menu

Statistics > LCA (latent class analysis) > Goodness of fit

Syntax

```
estat lcgof [, nodescribe]
```

Option

`nodescribe` suppresses the descriptions of the goodness-of-fit measures.

Remarks and examples

`estat lcgof` reports AIC and BIC for the fitted model.

For standard latent class models, `estat lcgof` also reports a likelihood-ratio test of the fitted model versus the saturated model. The likelihood-ratio statistic is also known as the G^2 statistic.

See [SEM] **Example 51g**.

Stored results

`estat lcgof` stores the following in `r()`:

Scalars

`r(chi2_ms)`	test of target model against saturated model
`r(df_ms)`	degrees of freedom for `r(chi2_ms)`
`r(p_ms)`	p-value for `r(chi2_ms)`
`r(aic)`	Akaike information criterion
`r(bic)`	Bayesian information criterion

References

Akaike, H. 1987. Factor analysis and AIC. *Psychometrika* 52: 317–332.

Goodman, L. A. 2002. Latent class analysis: The empirical study of latent types, latent variables, and latent structures. In *Applied Latent Class Analysis*, ed. J. A. Hagenaars and A. L. McCutcheon, 3–55. Cambridge: Cambridge University Press.

Raftery, A. E. 1993. Bayesian model selection in structural equation models. Reprinted in *Testing Structural Equation Models*, ed. K. A. Bollen and J. S. Long, pp. 163–180. Newbury Park, CA: SAGE.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Example 51g** — Latent class goodness-of-fit statistics

[R] **estat ic** — Display information criteria

Title

estat lcmean — Latent class marginal means

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`estat lcmean` is for use after `gsem` but not `sem`.

`estat lcmean` reports a table of the marginal predicted means of each outcome within each latent class.

Menu

Statistics > LCA (latent class analysis) > Class marginal means

Syntax

`estat lcmean` [, *options*]

options	Description
`nose`	do not estimate SEs
`post`	post margins and their VCE as estimation results
display_options	control column formats, row spacing, and line width

Options

`nose` suppresses calculation of the VCE and standard errors.

`post` causes `estat lcmean` to behave like a Stata estimation (e-class) command. `estat lcmean` posts the vector of estimated margins along with the estimated variance–covariance matrix to `e()`, so you can treat the estimated margins just as you would results from any other estimation command.

display_options: `vsquish`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`.

Remarks and examples

See [SEM] **Example 50g**, [SEM] **Example 53g**, and [SEM] **Example 54g**.

Stored results

`estat lcmean` stores the following in `r()`:

Scalars
- `r(N)` number of observations

Macros
- `r(title)` title in output

Matrices
- `r(b)` estimates
- `r(V)` variance–covariance matrix of the estimates
- `r(table)` matrix containing the margins with their standard errors, test statistics, p-values, and confidence intervals

`estat lcmean` with the `post` option also stores the following in `e()`:

Scalars
- `e(N)` number of observations

Macros
- `e(title)` title in output
- `e(properties)` `b V`

Matrices
- `e(b)` estimates
- `e(V)` variance–covariance matrix of the estimates

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Example 50g** — Latent class model

[SEM] **Example 53g** — Finite mixture Poisson regression

[SEM] **Example 54g** — Finite mixture Poisson regression, multiple responses

Title

estat lcprob — Latent class marginal probabilities

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`estat lcprob` is for use after `gsem` but not `sem`.

`estat lcprob` reports a table of the marginal predicted latent class probabilities.

Menu

Statistics > LCA (latent class analysis) > Class marginal probabilities

Syntax

`estat lcprob` [, *options*]

options	Description
`classpr`	latent class probability; the default
`classposteriorpr`	posterior latent class probability
`nose`	do not estimate SEs
`post`	post margins and their VCE as estimation results
display_options	control column formats, row spacing, and line width

Options

`classpr`, the default, calculates marginal predicted probabilities for each latent class.

`classposteriorpr` calculates marginal predicted posterior probabilities for each latent class. The posterior probabilities are a function of the latent class predictors and the fitted outcome densities.

`nose` suppresses calculation of the VCE and standard errors.

`post` causes `estat lcprob` to behave like a Stata estimation (e-class) command. `estat lcprob` posts the vector of estimated margins along with the estimated variance–covariance matrix to `e()`, so you can treat the estimated margins just as you would results from any other estimation command.

display_options: `vsquish`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`.

Remarks and examples

See [SEM] **Example 50g**, [SEM] **Example 53g**, and [SEM] **Example 54g**.

Stored results

`estat lcprob` stores the following in `r()`:

Scalars

`r(N)`	number of observations

Macros

`r(title)`	title in output
`r(classposteriorpr)`	`classposteriorpr`

Matrices

`r(b)`	estimates
`r(V)`	variance–covariance matrix of the estimates
`r(table)`	matrix containing the margins with their standard errors, test statistics, p-values, and confidence intervals

`estat lcprob` with the `post` option also stores the following in `e()`:

Scalars

`e(N)`	number of observations

Macros

`e(title)`	title in output
`e(classposteriorpr)`	`classposteriorpr`
`e(properties)`	`b V`

Matrices

`e(b)`	estimates
`e(V)`	variance–covariance matrix of the estimates

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Example 50g** — Latent class model

[SEM] **Example 53g** — Finite mixture Poisson regression

[SEM] **Example 54g** — Finite mixture Poisson regression, multiple responses

Title

estat mindices — Modification indices

Description | Menu | Syntax | Options
Remarks and examples | Stored results | References | Also see

Description

`estat mindices` is for use after `sem` but not `gsem`.

`estat mindices` reports modification indices for path coefficients and covariances that were constrained or omitted in the fitted model. Modification indices are score tests (Lagrange multiplier tests) for the statistical significance of the constrained parameters. See Sörbom (1989) and Wooldridge (2010, 421–428).

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Modification indices

Syntax

`estat mindices` [, *options*]

options	Description
`showpclass(`*pclassname*`)`	restrict output to parameters in the specified parameter classes
`minchi2(#)`	display only tests with modification index (MI) $\geq$ #

pclassname	Description
`scoef`	structural coefficients
`scons`	structural intercepts
`mcoef`	measurement coefficients
`mcons`	measurement intercepts
`serrvar`	covariances of structural errors
`merrvar`	covariances of measurement errors
`smerrcov`	covariances between structural and measurement errors
`meanex`	means of exogenous variables
`covex`	covariances of exogenous variables
`all`	all the above
`none`	none of the above

Options

`showpclass(`*pclassname*`)` specifies that results be limited to parameters that belong to the specified parameter classes. The default is `showpclass(all)`.

`minchi2(#)` suppresses listing paths with modification indices (MIs) less than #. By default, `estat mindices` lists values significant at the 0.05 level, corresponding to $\chi^2(1)$ value `minchi2(3.8414588)`. Specify `minchi2(0)` if you wish to see all tests.

Remarks and examples

See [SEM] **Example 5**.

Stored results

`estat mindices` stores the following in `r()`:

Scalars	
`r(N_groups)`	number of groups
Matrices	
`r(nobs)`	sample size for each group
`r(mindices_pclass)`	parameter class of modification indices
`r(mindices)`	matrix containing the displayed table values

References

Sörbom, D. 1989. Model modification. *Psychometrika* 54: 371–384.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat ginvariant** — Tests for invariance of parameters across groups

[SEM] **estat scoretests** — Score tests

[SEM] **Example 5** — Modification indices

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat residuals — Display mean and covariance residuals

Description Menu Syntax Options
Remarks and examples Stored results References Also see

Description

`estat residuals` is for use after `sem` but not `gsem`.

`estat residuals` displays the mean and covariance residuals. Normalized and standardized residuals are available.

Both mean and covariance residuals are reported unless `sem`'s option `nomeans` was specified or implied at the time the model was fit, in which case mean residuals are not reported.

`estat residuals` usually does not work following `sem` models fit with `method(mlmv)`. It also does not work if there are any missing values, which after all is the whole point of using `method(mlmv)`.

Menu

Statistics > SEM (structural equation modeling) > Goodness of fit > Matrices of residuals

Syntax

`estat residuals` [, *options*]

options	Description
`normalized`	report normalized residuals
`standardized`	report standardized residuals
`sample`	use sample covariances in residual variance calculations
`nm1`	use adjustment $N - 1$ in residual variance calculations
`zerotolerance(`*tol*`)`	apply tolerance to treat residuals as 0
`format(%`*fmt*`)`	display format

Options

`normalized` and `standardized` are alternatives. If neither is specified, raw residuals are reported.

Normalized residuals and standardized residuals attempt to adjust the residuals in the same way, but they go about it differently. The normalized residuals are always valid, but they do not follow a standard normal distribution. The standardized residuals do follow a standard normal distribution but only if they can be calculated; otherwise, they will equal missing values. When both can be calculated (equivalent to both being appropriate), the normalized residuals will be a little smaller than the standardized residuals. See Jöreskog and Sörbom (1986).

`sample` specifies that the sample variance and covariances be used in variance formulas to compute normalized and standardized residuals. The default uses fitted variance and covariance values as described by Bollen (1989).

`nm1` specifies that the variances be computed using $N-1$ in the denominator rather than using sample size N.

`zerotolerance(`*tol*`)` treats residuals within *tol* of 0 as if they were 0. *tol* must be a numeric value less than 1. The default is `zerotolerance(0)`, meaning that no tolerance is applied. When standardized residuals cannot be calculated, it is because a variance calculated by the Hausman (1978) theorem turns negative. Applying a tolerance to the residuals turns some residuals into 0 and then division by the negative variance becomes irrelevant, and that may be enough to solve the calculation problem.

`format(%`*fmt*`)` specifies the display format. The default is `format(%9.3f)`.

Remarks and examples

See [SEM] **Example 10**.

Stored results

`estat residuals` stores the following in `r()`:

Scalars

`r(N_groups)`	number of groups

Macros

`r(sample)`	empty or `sample`, if `sample` was specified
`r(nm1)`	empty or `nm1`, if `nm1` was specified

Matrices

`r(nobs)`	sample size for each group
`r(res_mean[_#])`	raw mean residuals (for group #) (*)
`r(res_cov[_#])`	raw covariance residuals (for group #)
`r(nres_mean[_#])`	normalized mean residuals (for group #) (*)
`r(nres_cov[_#])`	normalized covariance residuals (for group #)
`r(sres_mean[_#])`	standardized mean residuals (for group #) (*)
`r(sres_cov[_#])`	standardized covariance residuals (for group #)

(*) If there are no estimated means or intercepts in the `sem` model, these matrices are not returned.

References

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.

Jöreskog, K. G., and D. Sörbom. 1986. *Lisrel VI: Analysis of linear structural relationships by the method of maximum likelihood*. Mooresville, IN: Scientific Software.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat eqgof** — Equation-level goodness-of-fit statistics

[SEM] **estat ggof** — Group-level goodness-of-fit statistics

[SEM] **estat gof** — Goodness-of-fit statistics

[SEM] **Example 10** — MIMIC model

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat scoretests — Score tests

Description | Menu | Syntax | Option
Remarks and examples | Stored results | References | Also see

Description

`estat scoretests` is for use after `sem` but not `gsem`.

`estat scoretests` displays score tests (Lagrange multiplier tests) for each of the user-specified linear constraints imposed on the model when it was fit. See Sörbom (1989) and Wooldridge (2010, 421–428).

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Score tests of linear constraints

Syntax

`estat scoretests [, minchi2(#)]`

Option

`minchi2(#)` suppresses output of tests with $\chi^2(1) < \#$. By default, `estat mindices` lists values significant at the 0.05 level, corresponding to $\chi^2(1)$ value `minchi2(3.8414588)`. Specify `minchi2(0)` if you wish to see all tests.

Remarks and examples

See [SEM] **Example 8**.

Stored results

`estat scoretests` stores the following in `r()`:

Scalars
- `r(N_groups)` number of groups

Matrices
- `r(nobs)` sample size for each group
- `r(Cns_sctest)` matrix containing the displayed table values

References

Sörbom, D. 1989. Model modification. *Psychometrika* 54: 371–384.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat ginvariant** — Tests for invariance of parameters across groups

[SEM] **estat mindices** — Modification indices

[SEM] **Example 8** — Testing that coefficients are equal, and constraining them

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat sd — Display variance components as standard deviations and correlations

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`estat sd` is for use after `gsem` but not `sem`.

`estat sd` displays the fitted variance components as standard deviations and correlations.

Menu

Statistics > SEM (structural equation modeling) > Other > Display standard deviations and correlations

Syntax

```
estat sd [, verbose post coeflegend]
```

Options

`verbose` specifies that the full estimation table be displayed. By default, only the variance components are displayed. This option is implied when `post` is specified.

`post` causes `estat sd` to behave like a Stata estimation (e-class) command. `estat sd` posts the vector of calculated standard deviation and correlation parameters along with the corresponding variance–covariance matrix to `e()`, so that you can treat the estimated parameters just as you would results from any other estimation command. For example, you could use `test` to perform simultaneous tests of hypotheses on the parameters, or you could use `lincom` to create linear combinations.

The following option is not shown in the dialog box:

`coeflegend` specifies that the legend of the coefficients and how to specify them in an expression be displayed rather than displaying the statistics for the coefficients.

Remarks and examples

See [SEM] **Example 31g**.

Stored results

`estat sd` stores the following in `r()`:

Scalars

`r(level)`	confidence level

Matrices

`r(b)`	coefficient vector
`r(V)`	variance–covariance matrix of the estimators
`r(table)`	table of results

If `post` is specified, `estat sd` stores the following in `e()`:

Matrices

`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix of the estimators

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Example 31g** — Two-factor measurement model (generalized response)

Title

estat stable — Check stability of nonrecursive system

Description | Menu | Syntax | Option
Remarks and examples | Stored results | Reference | Also see

Description

`estat stable` is for use after `sem` but not `gsem`.

`estat stable` reports the eigenvalue stability index for nonrecursive models. The stability index is computed as the maximum modulus of the eigenvalues for the matrix of coefficients on endogenous variables predicting other endogenous variables. If the model was fit by `sem` with the `group()` option, `estat stable` reports the index for each group separately.

There are two formulas commonly used to calculate the index. `estat stable` uses the formulation of Bentler and Freeman (1983).

Menu

Statistics > SEM (structural equation modeling) > Other > Assess stability of nonrecursive systems

Syntax

`estat stable [, detail]`

Option

`detail` displays the matrix of coefficients on endogenous variables predicting other endogenous variables, also known as the β matrix.

Remarks and examples

See *nonrecursive (structural) model (system)* in [SEM] **Glossary**. The issue of stability is described there. Also see *Remarks and examples* of [SEM] **estat teffects**.

Stored results

`estat stable` stores the following in `r()`:

Scalars

`r(N_groups)`	number of groups
`r(stindex[_#])`	stability index (for group #)

Matrices

`r(nobs)`	sample size for each group
`r(Beta[_#])`	coefficients of endogenous variables on endogenous variables (for group #)
`r(Re[_#])`	real parts of the eigenvalues of A (for group #)
`r(Im[_#])`	imaginary parts of the eigenvalues of A (for group #)
`r(Modulus[_#])`	modulus of the eigenvalues of A (for group #)

Reference

Bentler, P. M., and E. H. Freeman. 1983. Tests for stability in linear structural equation systems. *Psychometrika* 48: 143–145.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat teffects** — Decomposition of effects into total, direct, and indirect

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat stdize — Test standardized parameters

Description Menu Syntax Remarks and examples Stored results Also see

Description

`estat stdize:` is for use after `sem` but not `gsem`.

`estat stdize:` can be used to prefix `test`, `lincom`, `testnl`, and `nlcom`; see [SEM] **test**, [SEM] **lincom**, [SEM] **testnl**, and [SEM] **nlcom**.

These commands without a prefix work in the underlying metric of SEM, which is to say path coefficients, variances, and covariances. If the commands are prefixed with `estat stdize:`, they will work in the metric of standardized coefficients and correlation coefficients. There is no counterpart to variances in the standardized metric because variances are standardized to be 1.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Testing standardized parameters

Syntax

```
estat stdize: test ...

estat stdize: lincom ...

estat stdize: testnl ...

estat stdize: nlcom ...
```

Remarks and examples

See [SEM] **Example 16**.

Exercise caution when using the `estat stdize:` prefix to perform tests on estimated second moments (correlations). Do not test that correlations are 0. Instead, omit the `estat stdize:` prefix and test that covariances are 0. Covariances are more likely to be normally distributed than are correlations.

Stored results

Stored results are the results stored by the command being used with the `estat stdize:` prefix.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **lincom** — Linear combinations of parameters

[SEM] **nlcom** — Nonlinear combinations of parameters

[SEM] **test** — Wald test of linear hypotheses

[SEM] **testnl** — Wald test of nonlinear hypotheses

[SEM] **Example 16** — Correlation

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

estat summarize — Report summary statistics for estimation sample

Description Menu Syntax Options Stored results Also see

Description

`estat summarize` is a standard postestimation command of Stata. This entry concerns use of `estat summarize` after `sem` or `gsem`.

`estat summarize` reports the summary statistics in the estimation sample for the observed variables in the model. `estat summarize` is mentioned here because

1. `estat summarize` cannot be used if `sem` was run on summary statistics data; see [SEM] **Intro 11**.
2. `estat summarize` allows the additional option `group` after estimation by `sem` and `gsem`.

Menu

Statistics > SEM (structural equation modeling) > Other > Estimation-sample summary statistics

Syntax

`estat summarize` [*eqlist*] [`, group` *estat_summ_options*]

Options

`group` may be specified if `group(`*varname*`)` was specified with `sem` or `gsem` at the time the model was fit. It requests that summary statistics be reported by group.

estat_summ_options are the standard options allowed by `estat summarize` and are outlined in *Options* of [R] **estat summarize**.

Stored results

See *Stored results* of [R] **estat summarize**.

Also see

[R] **estat summarize** — Summarize estimation sample

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **sem postestimation** — Postestimation tools for sem

Title

estat teffects — Decomposition of effects into total, direct, and indirect

Description Menu Syntax Options
Remarks and examples Stored results References Also see

Description

`estat teffects` is for use after `sem` but not `gsem`.

`estat teffects` reports direct, indirect, and total effects for each path (Sobel 1987), along with standard errors obtained by the delta method.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Direct and indirect effects

Syntax

`estat teffects` [, *options*]

options	Description
`compact`	do not display effects with no path
`standardized`	report standardized effects
`nolabel`	display group values, not labels
`nodirect`	do not display direct effects
`noindirect`	do not display indirect effects
`nototal`	do not display total effects
display_options	control columns and column formats, row spacing, and display of omitted paths

Options

`compact` is a popular option. Consider the following model:

```
. sem (y1<-y2 x1) (y2<-x2)
```

`x2` has no direct effect on `y1` but does have an indirect effect. `estat teffects` formats all its effects tables the same way by default, so there will be a row for the direct effect of `x2` on `y1` just because there is a row for the indirect effect of `x2` on `y1`. The value reported for the direct effect, of course, will be 0. `compact` says to omit these unnecessary rows.

`standardized` reports effects in standardized form, but standard errors of the standardized effects are not reported.

`nolabel` is relevant only if estimation was with `sem`'s `group()` option and the group variable has a value label. Groups are identified by group value rather than label.

`nodirect`, `noindirect`, and `nototal` suppress the display of the indicated effect. The default is to display all effects.

display_options: noci, nopvalues, noomitted, vsquish, cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**. Although estat teffects is not an estimation command, it allows these options.

Remarks and examples

See [SEM] **Example 7**.

Direct effects are the path coefficients in the model.

Indirect effects are all mediating effects. For instance, consider

```
. sem ... (y1<-y2) (y1<-x2) (y2<-x3) ..., ...
```

The direct effect of y2 on y1 is the path coefficient (y1<-y2).

In this example, changes in x3 affect y1, too. That is called the indirect effect and is the product of the path coefficients (y2<-x3) and (y1<-y2). If there were other paths in the model such that y1 changed when x3 changed, those effects would be added to the indirect effect as well. estat teffects reports total indirect effects.

The total effect is the sum of the direct and indirect effects.

When feedback loops are present in the model, such as

```
. sem ... (y1<-y2) (y1<-x2) (y2<-x3 y1) ..., ...
```

care must be taken when interpreting indirect effects. The feedback loop is when a variable indirectly affects itself, as y1 does in the example; y1 affects y2 and y2 affects y1. Thus in calculating the indirect effect, the sum has an infinite number of terms although the term values get smaller and smaller and thus usually converge to a finite result. It is important that you check nonrecursive models for stability; see Bollen (1989, 397) and see [SEM] **estat stable**. Caution: if the model is unstable, the calculation of the indirect effect can sometimes still converge to a finite result.

Stored results

estat teffects stores the following in r():

Scalars

r(N_groups)	number of groups

Matrices

r(nobs)	sample size for each group
r(direct)	direct effects
r(indirect)	indirect effects
r(total)	total effects
r(V_direct)	covariance matrix of the direct effects
r(V_indirect)	covariance matrix of the indirect effects
r(V_total)	covariance matrix of the total effects

estat teffects with the standardized option additionally stores the following in r():

Matrices

r(direct_std)	standardized direct effects
r(indirect_std)	standardized indirect effects
r(total_std)	standardized total effects

References

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Sobel, M. E. 1987. Direct and indirect effects in linear structural equation models. *Sociological Methods and Research* 16: 155–176.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **estat stable** — Check stability of nonrecursive system

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

Title

Example 1 — Single-factor measurement model

Description Remarks and examples References Also see

Description

The single-factor measurement model is demonstrated using the following data:

```
. use https://www.stata-press.com/data/r16/sem_1fmm
(single-factor measurement model)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	500	99.518	14.35402	60	137
x2	500	99.954	14.1939	52	140
x3	500	99.052	14.26395	59	150
x4	500	94.474	70.11603	-113	295

```
. notes
_dta:
  1.  fictional data
  2.  Variables x1, x2, x3, and x4 each contain a test score designed to
      measure X. The test is scored to have mean 100.
```

See *Single-factor measurement models* in [SEM] **Intro 5** for background.

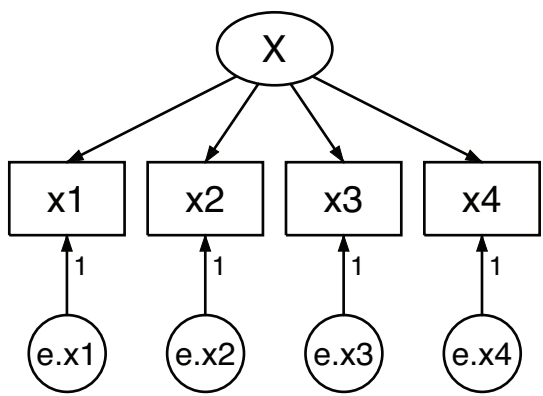
Remarks and examples

Remarks are presented under the following headings:

Single-factor measurement model
Satorra–Bentler scaled χ^2 test
Fitting the same model with gsem
Fitting the same model with the Builder
The measurement-error model interpretation

Single-factor measurement model

Below we fit the following model:



```
. sem (x1 x2 x3 x4 <- X)
Endogenous variables
Measurement:  x1 x2 x3 x4
Exogenous variables
Latent:       X
Fitting target model:
Iteration 0:   log likelihood = -8487.5905
Iteration 1:   log likelihood = -8487.2358
Iteration 2:   log likelihood = -8487.2337
Iteration 3:   log likelihood = -8487.2337
Structural equation model                       Number of obs     =        500
Estimation method  = ml
Log likelihood     = -8487.2337
 ( 1)  [x1]X = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
x1						
X	1	(constrained)				
_cons	99.518	.6412888	155.18	0.000	98.2611	100.7749
x2						
X	1.033249	.0723898	14.27	0.000	.8913676	1.17513
_cons	99.954	.6341354	157.62	0.000	98.71112	101.1969
x3						
X	1.063876	.0729725	14.58	0.000	.9208526	1.2069
_cons	99.052	.6372649	155.43	0.000	97.80298	100.301
x4						
X	7.276754	.4277638	17.01	0.000	6.438353	8.115156
_cons	94.474	3.132547	30.16	0.000	88.33432	100.6137
var(e.x1)	115.6865	7.790423			101.3823	132.0089
var(e.x2)	105.0445	7.38755			91.51873	120.5692
var(e.x3)	101.2572	7.17635			88.12499	116.3463
var(e.x4)	144.0406	145.2887			19.94838	1040.069
var(X)	89.93921	11.07933			70.64676	114.5001

```
LR test of model vs. saturated: chi2(2)   =      1.46, Prob > chi2 = 0.4827
```

The equations for this model are

$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$

$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

$$x_3 = \alpha_3 + X\beta_3 + e.x_3$$

$$x_4 = \alpha_4 + X\beta_4 + e.x_4$$

Notes:

1. Variable `X` is latent exogenous and thus needs a normalizing constraint. The variable is anchored to the first observed variable, `x1`, and thus the path coefficient is constrained to be 1. See *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.

2. The path coefficients for `X->x1`, `X->x2`, and `X->x3` are 1 (constrained), 1.03, and 1.06. Meanwhile, the path coefficient for `X->x4` is 7.28. This is not unexpected; we at StataCorp generated this data, and the true coefficients are 1, 1, 1, and 7.
3. A test for "model versus saturated" is reported at the bottom of the output; the $\chi^2(2)$ statistic is 1.46 and its significance level is 0.4827. We cannot reject the null hypothesis of this test.

   This test is a goodness-of-fit test in badness-of-fit units; a significant result implies that the model does not fit well.

   More mathematically, the null hypothesis of the test is that the fitted covariance matrix and mean vector of the observed variables are equal to the matrix and vector observed in the population.

Satorra–Bentler scaled χ^2 test

The model-versus-saturated goodness-of-fit statistic shown above does not follow the χ^2 distribution that it is referred to when the data are nonnormal. Satorra and Bentler (1994) provide a scaled version of this statistic that more closely follows the mean of the reference distribution in the presence of nonnormal data. We can request this statistic and the corresponding robust standard errors by specifying the `vce(sbentler)` option.

```
. sem (x1 x2 x3 x4 <- X), vce(sbentler)
Endogenous variables
Measurement:  x1 x2 x3 x4
Exogenous variables
Latent:       X
Fitting target model:
Iteration 0:   log pseudolikelihood = -8487.5905
Iteration 1:   log pseudolikelihood = -8487.2358
Iteration 2:   log pseudolikelihood = -8487.2337
Iteration 3:   log pseudolikelihood = -8487.2337
Structural equation model                       Number of obs     =        500
Estimation method    = ml
Log pseudolikelihood = -8487.2337
 ( 1)  [x1]X = 1
```

	Satorra-Bentler Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
x1						
X	1	(constrained)				
_cons	99.518	.6419311	155.03	0.000	98.25984	100.7762
x2						
X	1.033249	.0767608	13.46	0.000	.8828006	1.183698
_cons	99.954	.6347705	157.46	0.000	98.70987	101.1981
x3						
X	1.063876	.0751028	14.17	0.000	.9166773	1.211075
_cons	99.052	.6379032	155.28	0.000	97.80173	100.3023
x4						
X	7.276754	.4386592	16.59	0.000	6.416998	8.13651
_cons	94.474	3.135684	30.13	0.000	88.32817	100.6198
var(e.x1)	115.6865	7.744173			101.4617	131.9055
var(e.x2)	105.0445	6.499187			93.04833	118.5872
var(e.x3)	101.2572	7.00047			88.42551	115.9509
var(e.x4)	144.0406	145.6607			19.84766	1045.347
var(X)	89.93921	11.2763			70.34416	114.9927

```
LR test of model vs. saturated: chi2(2)   =      1.46, Prob > chi2 = 0.4827
Satorra-Bentler scaled test:    chi2(2)   =      1.59, Prob > chi2 = 0.4526
```

The rescaled statistic is labeled “Satorra–Bentler scaled test” and has a value of 1.59 with a significance level of 0.4526. As with the unadjusted test, we cannot reject the null hypothesis.

Fitting the same model with gsem

sem and gsem produce the same results for standard linear SEMs. We are going to demonstrate that just this once.

```
. gsem (x1 x2 x3 x4 <- X)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -8948.2394
Iteration 1:   log likelihood = -8948.2394
Refining starting values:
Grid node 0:   log likelihood = -8487.5916
Fitting full model:
Iteration 0:   log likelihood = -8487.5916
Iteration 1:   log likelihood = -8487.5051
Iteration 2:   log likelihood = -8487.3836
Iteration 3:   log likelihood = -8487.2697
Iteration 4:   log likelihood = -8487.2337
Iteration 5:   log likelihood = -8487.2337
Generalized structural equation model           Number of obs     =        500
Response       : x1
Family         : Gaussian
Link           : identity
Response       : x2
Family         : Gaussian
Link           : identity
Response       : x3
Family         : Gaussian
Link           : identity
Response       : x4
Family         : Gaussian
Link           : identity
Log likelihood = -8487.2337
 ( 1)  [x1]X = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1						
X	1	(constrained)				
_cons	99.518	.6412888	155.18	0.000	98.2611	100.7749
x2						
X	1.033249	.0723898	14.27	0.000	.8913676	1.17513
_cons	99.954	.6341354	157.62	0.000	98.71112	101.1969
x3						
X	1.063876	.0729725	14.58	0.000	.9208526	1.206899
_cons	99.052	.6372649	155.43	0.000	97.80298	100.301
x4						
X	7.276753	.4277636	17.01	0.000	6.438352	8.115154
_cons	94.474	3.132547	30.16	0.000	88.33432	100.6137
var(X)	89.93923	11.07933			70.64678	114.5001
var(e.x1)	115.6865	7.790422			101.3822	132.0089
var(e.x2)	105.0444	7.387549			91.51872	120.5692
var(e.x3)	101.2572	7.176349			88.12498	116.3463
var(e.x4)	144.0408	145.2886			19.94848	1040.067

Notes:

1. Results are virtually the same. Coefficients, variance estimates, and standard errors may differ in the last digit; for instance, `x4<-X` was 7.276754 and now it is 7.276753.

   These are the kind of differences we would expect to see. `gsem` follows a different approach for obtaining results that involves far more numeric machinery, which correspondingly results in slightly less accuracy.

2. The log-likelihood values reported are the same. This model is one of the few models we could have chosen where `sem` and `gsem` would produce the same log-likelihood values. In general, `gsem` log likelihoods are on different metrics from those of `sem`. In the case where the model does not include observed exogenous variables, however, they share the same metric.

3. There is no reason to use `gsem` over `sem` when both can fit the same model. `sem` is slightly more accurate, is quicker, and has more postestimation features.

Fitting the same model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_1fmm
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for `X`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `X`;

   b. select `x1`, `x2`, `x3`, and `x4` by using the *Measurement variables* control;

   c. select `Down` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

   Notice that the constraints of 1 on the paths from the error terms to the observed measures are implied, so we do not need to add these to our diagram.

4. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_1fmm
```

The measurement-error model interpretation

As we pointed out in *Using path diagrams to specify standard linear SEMs* in [SEM] **Intro 2**, if we rename variable `x4` to be `y`, we can reinterpret this measurement model as a measurement-error model. In this interpretation, `X` is the unobserved true value. `x1`, `x2`, and `x3` are each measurements of `X`, but with error. Meanwhile, `y` (`x4`) is really something else entirely. Perhaps `y` is earnings, and we believe

$$\texttt{y} = \alpha_4 + \beta_4\texttt{X} + \texttt{e.y}$$

We are interested in β_4, the effect of true `X` on `y`.

If we were to go back to the data and type `regress y x1`, we would obtain an estimate of β_4, but we would expect that estimate to be biased toward 0 because of the errors-in-variable problem. The same applies for `y` on `x2` and `y` on `x3`. If we do that, we obtain

β_4 based on `regress y x1`	3.19
β_4 based on `regress y x2`	3.36
β_4 based on `regress y x3`	3.43

In the `sem` output above, we have an estimate of β_4 with the bias washed away:

β_4 based on `sem (y<-X)`	7.28

The number 7.28 is the value reported for `(x4<-X)` in the `sem` output.

That β_4 might be 7.28 seems plausible because we expect that the estimate should be larger than the estimates we obtain using the variables measured with error. In fact, we can tell you that the 7.28 estimate is quite good because we at StataCorp know that the true value of β_4 is 7. Here is how we manufactured this fictional dataset:

```
set seed 83216
set obs 500
gen X  = round(rnormal(0,10))
gen x1 = round(100 + X + rnormal(0, 10))
gen x2 = round(100 + X + rnormal(0, 10))
gen x3 = round(100 + X + rnormal(0, 10))
gen x4 = round(100 + 7*X + rnormal(0, 10))
drop X
```

The data recorded in `sem_1fmm.dta` were obviously generated using normality, the same assumption that is most often used to justify the SEM maximum likelihood estimator. In [SEM] **Intro 4**, we explained that the normality assumption can be relaxed and conditional normality can usually be substituted in its place.

So let's consider nonnormal data. Let's make `X` be $\chi^2(2)$, a violently nonnormal distribution, resulting in the data-manufacturing code

```
set seed 83216
set obs 500
gen X  = (rchi2(2)-2)*(10/2)
gen x1 = round(100 + X + rnormal(0, 10))
gen x2 = round(100 + X + rnormal(0, 10))
gen x3 = round(100 + X + rnormal(0, 10))
gen x4 = round(100 + 7*X + rnormal(0, 10))
drop X
```

All the `rnormal()` functions remaining in our code have to do with the assumed normality of the errors. The multiplicative and additive constants in the generation of `X` simply rescale the $\chi^2(2)$ variable to have mean 100 and standard deviation 10, which would not be important except for the subsequent `round()` functions, which themselves were unnecessary except that we wanted to produce a pretty dataset when we created the original `sem_1fmm.dta`.

In any case, if we rerun the commands with these data, we obtain

β_4 based on `regress y x1`	3.24
β_4 based on `regress y x2`	3.14
β_4 based on `regress y x3`	3.36
β_4 based on `sem (y<-X)`	7.25

We will not burden you with the details of running simulations to assess coverage; we will just tell you that coverage is excellent: reported test statistics and significance levels can be trusted.

By the way, errors in the variables is something that does not go away with progressively larger sample sizes. Change the code above to produce a 100,000-observation dataset instead of a 500-observation one, and you will obtain

β_4 based on `regress y x1`	3.47
β_4 based on `regress y x2`	3.48
β_4 based on `regress y x3`	3.50
β_4 based on `sem (y<-X)`	6.97

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Satorra, A., and P. M. Bentler. 1994. Corrections to test statistics and standard errors in covariance structure analysis. In *Latent Variables Analysis: Applications for Developmental Research*, ed. A. von Eye and C. C. Clogg, 399–419. Thousand Oaks, CA: SAGE.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 5** — Tour of models

[SEM] **Example 3** — Two-factor measurement model

[SEM] **Example 24** — Reliability

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

Title

Example 2 — Creating a dataset from published covariances

Description Remarks and examples Reference Also see

Description

Williams, Eaves, and Cox (2002) publish covariances from their data. We will use those published covariances in [SEM] **Example 3** to fit an SEM.

In this example, we show how to create the summary statistics dataset (SSD) that we will analyze in that example.

Remarks and examples

Remarks are presented under the following headings:

Background
Creating the SSD
At this point, we could save the dataset and stop
Labeling the SSD
Listing the SSD

For more explanation, also see [SEM] **Intro 11**.

Background

In Williams, Eaves, and Cox (2002), the authors report a covariance matrix in a table that looks something like this:

	Affective		Miniscale		Cognitive		Miniscale
	1	2	. . .	5	1	2	. . .
Affective							
1	2038.035	1631.766	. . .	1721.830	659.795	779.519	. . .
2		1932.163	. . .	1688.292	702.969	790.448	. . .
.							
.							
5				2061.875	775.118	871.211	. . .
Cognitive							
1					630.518	500.128	. . .
.							
.							

Creating the SSD

```
. clear all
. ssd init  a1 a2 a3 a4 a5  c1 c2 c3 c4 c5
Summary statistics data initialized.  Next use, in any order,
    ssd set observations (required)
        It is best to do this first.
    ssd set means (optional)
        Default setting is 0.
    ssd set variances or ssd set sd (optional)
        Use this only if you have set or will set correlations and, even
        then, this is optional but highly recommended.  Default setting is 1.
    ssd set covariances or ssd set correlations (required)
. ssd set obs 216
  (value set)
    Status:
                        observations:    set
                               means:  unset
                    variances or sd:  unset
         covariances or correlations:  unset (required to be set)
. #delimit ;
delimiter now ;
. ssd set cov 2038.035 \
>             1631.766 1932.163 \
>             1393.567 1336.871 1313.809 \
>             1657.299 1647.164 1273.261 2034.216 \
>             1721.830 1688.292 1498.401 1677.767 2061.875 \
>              659.795  702.969  585.019  656.527  775.118 630.518 \
>              779.519  790.448  653.734  764.755  871.211 500.128 741.767 \
>
>              899.912  879.179  750.037  897.441 1008.287 648.935 679.970
>             1087.409 \
>
>              775.235  739.157  659.867  751.860  895.616 550.476 603.950
>              677.865  855.272 \
>
>              821.170  785.419  669.951  802.825  877.681 491.042 574.775
>              686.391  622.830  728.674 ;
  (values set)
    Status:
                        observations:    set
                               means:  unset
                    variances or sd:    set
         covariances or correlations:    set
. #delimit cr
delimiter now cr
```

Notes:

1. We used `#delimit` to temporarily set the end-of-line character to semicolon. That was not necessary, but it made it easier to enter the data in a way that would be subsequently more readable. You can use `#delimit` only in do-files; see [P] **#delimit**.
2. We recommend entering SSD by using do-files. That way, you can edit the file and get it right.
3. We did not have to reset the delimiter. We could have entered the numbers on one (long) line. That works well when there are only a few summary statistics.

At this point, we could save the dataset and stop

We could save the dataset and stop right now if we wished:

```
. save sem_2fmm
file sem_2fmm.dta saved
```

Obviously, we can save the dataset anytime we wish. We know we could stop because `ssd status` tells us whether there is anything more that we need to define:

```
. ssd status
    Status:
                        observations:    set
                               means:  unset
                    variances or sd:    set
        covariances or correlations:    set
```

Notes:

1. The means have not been set. The authors did not provide the means.
2. `ssd status` would mention if anything that was not set was required to be set.

Labeling the SSD

If we were to use `ssd describe` to describe these data, the output would look like this:

```
. ssd describe
Summary statistics data
  obs:             216
 vars:              10
-------------------------------------------------------------------------------
variable name                  variable label
-------------------------------------------------------------------------------
a1
a2
a3
a4
a5
c1
c2
c3
c4
c5
-------------------------------------------------------------------------------
```

We can add labels and notes to our dataset:

```
. label data "Affective and cognitive arousal"
. label var a1 "affective arousal 1"
. label var a2 "affective arousal 2"
. label var a3 "affective arousal 3"
. label var a4 "affective arousal 4"
. label var a5 "affective arousal 5"
. label var c1 "cognitive arousal 1"
. label var c2 "cognitive arousal 2"
. label var c3 "cognitive arousal 3"
. label var c4 "cognitive arousal 4"
```

```
. label var c5 "cognitive arousal 5"
. #delimit ;
delimiter now ;
. notes: Summary statistics data containing published covariances
>        from Thomas O. Williams, Ronald C. Eaves, and Cynthia Cox,
>        2 Apr 2002, "Confirmatory factor analysis of an instrument
>        designed to measure affective and cognitive arousal",
>        _Educational and Psychological Measurement_,
>        vol. 62 no. 2, 264-283. ;
. notes: a1-a5 report scores from 5 miniscales designed to measure
>        affective arousal. ;
. notes: c1-c5 report scores from 5 miniscales designed to measure
>        cognitive arousal. ;
. notes: The series of tests, known as the VST II
>        (Visual Similes Test II) were administered to 216 children
>        ages 10 to 12.  The miniscales are sums of scores of
>        5 to 6 items in VST II. ;
. #delimit cr
delimiter now cr
. ssd describe
  Summary statistics data
    obs:             216                  Affective and cognitive arousal
   vars:              10
                                          (_dta has notes)
-------------------------------------------------------------------------------
  variable name                  variable label
-------------------------------------------------------------------------------
  a1                             affective arousal 1
  a2                             affective arousal 2
  a3                             affective arousal 3
  a4                             affective arousal 4
  a5                             affective arousal 5
  c1                             cognitive arousal 1
  c2                             cognitive arousal 2
  c3                             cognitive arousal 3
  c4                             cognitive arousal 4
  c5                             cognitive arousal 5
-------------------------------------------------------------------------------
. save sem_2fmm, replace
file sem_2fmm.dta saved
```

Notes:

1. You can label the variables and the data, and you can add notes just as you would to any dataset.
2. You `save` and `use` SSD just as you `save` and `use` any dataset.

Listing the SSD

```
. ssd list
Observations = 216
Means undefined; assumed to be 0
Variances implicitly defined; they are the diagonal of the covariance
matrix.
Covariances:
      a1        a2        a3        a4        a5        c1        c2
2038.035
1631.766  1932.163
1393.567  1336.871  1313.809
1657.299  1647.164  1273.261  2034.216
 1721.83  1688.292  1498.401  1677.767  2061.875
 659.795   702.969   585.019   656.527   775.118   630.518
 779.519   790.448   653.734   764.755   871.211   500.128   741.767
 899.912   879.179   750.037   897.441  1008.287   648.935    679.97
 775.235   739.157   659.867    751.86   895.616   550.476    603.95
  821.17   785.419   669.951   802.825   877.681   491.042   574.775

      c3        c4        c5
1087.409
 677.865   855.272
 686.391    622.83   728.674
```

Reference

Williams, T. O., Jr., R. C. Eaves, and C. Cox. 2002. Confirmatory factor analysis of an instrument designed to measure affective and cognitive arousal. *Educational and Psychological Measurement* 62: 264–283.

Also see

[SEM] **Example 3** — Two-factor measurement model

[SEM] **ssd** — Making summary statistics data (sem only)

Title

Example 3 — Two-factor measurement model

Description Remarks and examples References Also see

Description

The multiple-factor measurement model is demonstrated using summary statistics dataset (SSD) `sem_2fmm.dta`:

```
. use https://www.stata-press.com/data/r16/sem_2fmm
(Affective and cognitive arousal)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_2fmm.dta
    obs:             216                  Affective and cognitive arousal
   vars:              10                  25 May 2018 10:11
                                          (_dta has notes)
  ------------------------------------------------------------------------------
  variable name                  variable label
  ------------------------------------------------------------------------------
  a1                             affective arousal 1
  a2                             affective arousal 2
  a3                             affective arousal 3
  a4                             affective arousal 4
  a5                             affective arousal 5
  c1                             cognitive arousal 1
  c2                             cognitive arousal 2
  c3                             cognitive arousal 3
  c4                             cognitive arousal 4
  c5                             cognitive arousal 5
  ------------------------------------------------------------------------------
. notes
_dta:
  1.  Summary statistics data containing published covariances from Thomas O.
      Williams, Ronald C. Eaves, and Cynthia Cox, 2 Apr 2002, "Confirmatory
      factor analysis of an instrument designed to measure affective and
      cognitive arousal", _Educational and Psychological Measurement_, vol. 62
      no. 2, 264-283.
  2.  a1-a5 report scores from 5 miniscales designed to measure affective
      arousal.
  3.  c1-c5 report scores from 5 miniscales designed to measure cognitive
      arousal.
  4.  The series of tests, known as the VST II (Visual Similes Test II) were
      administered to 216 children ages 10 to 12.  The miniscales are sums of
      scores of 5 to 6 items in VST II.
```

See [SEM] **Example 2** to learn how we created this SSD.

Remarks and examples

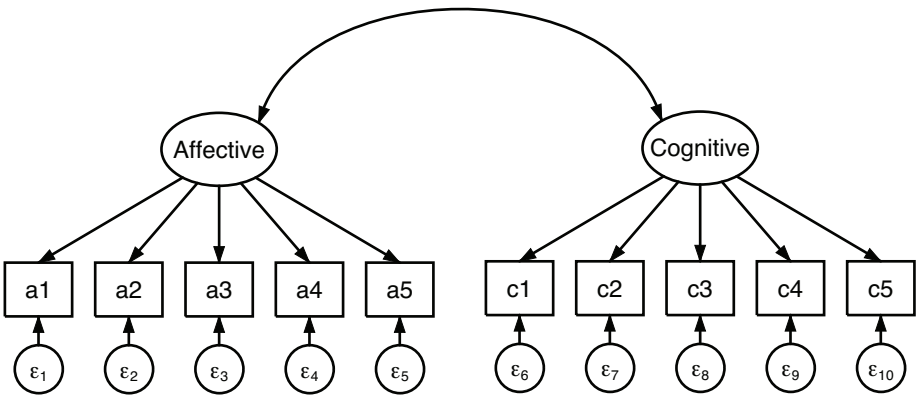
Remarks are presented under the following headings:

Fitting multiple-factor measurement models
Displaying standardized results
Fitting the model with the Builder
Obtaining equation-level goodness of fit by using estat eqgof

See *Multiple-factor measurement models* in [SEM] **Intro 5** for background.

Fitting multiple-factor measurement models

Below we fit the model shown by Kline (2005, 70–74, 184), namely,




```
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5)
Endogenous variables
Measurement:  a1 a2 a3 a4 a5 c1 c2 c3 c4 c5
Exogenous variables
Latent:       Affective Cognitive
Fitting target model:
Iteration 0:   log likelihood = -9542.8803
Iteration 1:   log likelihood = -9539.5505
Iteration 2:   log likelihood = -9539.3856
Iteration 3:   log likelihood = -9539.3851
```

```
Structural equation model                       Number of obs     =        216
Estimation method  = ml
Log likelihood     = -9539.3851
 ( 1)  [a1]Affective = 1
 ( 2)  [c1]Cognitive = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
a1 Affective	1	(constrained)				
a2 Affective	.9758098	.0460752	21.18	0.000	.885504	1.066116
a3 Affective	.8372599	.0355086	23.58	0.000	.7676643	.9068556
a4 Affective	.9640461	.0499203	19.31	0.000	.866204	1.061888
a5 Affective	1.063701	.0435751	24.41	0.000	.9782951	1.149107
c1 Cognitive	1	(constrained)				
c2 Cognitive	1.114702	.0655687	17.00	0.000	.9861901	1.243215
c3 Cognitive	1.329882	.0791968	16.79	0.000	1.174659	1.485105
c4 Cognitive	1.172792	.0711692	16.48	0.000	1.033303	1.312281
c5 Cognitive	1.126356	.0644475	17.48	0.000	1.000041	1.252671
var(e.a1)	384.1359	43.79119			307.2194	480.3095
var(e.a2)	357.3524	41.00499			285.3805	447.4755
var(e.a3)	154.9507	20.09026			120.1795	199.7822
var(e.a4)	496.4594	54.16323			400.8838	614.8214
var(e.a5)	191.6857	28.07212			143.8574	255.4154
var(e.c1)	171.6638	19.82327			136.894	215.2649
var(e.c2)	171.8055	20.53479			135.9247	217.1579
var(e.c3)	276.0144	32.33535			219.3879	347.2569
var(e.c4)	224.1994	25.93412			178.7197	281.2527
var(e.c5)	146.8655	18.5756			114.6198	188.1829
var(Affect~e)	1644.463	193.1032			1306.383	2070.034
var(Cognit~e)	455.9349	59.11245			353.6255	587.8439
cov(Affect~e, Cognitive)	702.0736	85.72272	8.19	0.000	534.0601	870.087

```
LR test of model vs. saturated: chi2(34)  =     88.88, Prob > chi2 = 0.0000
```

Notes:

1. In [SEM] **Example 1**, we ran `sem` on raw data. In this example, we run `sem` on SSD. There are no special `sem` options that we need to specify because of this.
2. The estimated coefficients reported above are unstandardized coefficients or, if you prefer, factor loadings.
3. The coefficients listed at the bottom of the coefficient table that start with `e.` are the estimated error variances. They represent the variance of the indicated measurement that is not measured by the respective latent variables.
4. The above results do not match exactly (Kline 2005, 184). If we specified `sem` option `nm1`, results are more likely to match to 3 or 4 digits. The `nm1` option says to divide by $N - 1$ rather than by N in producing variances and covariances.

Displaying standardized results

The output will be easier to interpret if we display standardized values for paths rather than path coefficients. A standardized value is in standard deviation units. It is the change in one variable given a change in another, both measured in standard deviation units. We can obtain standardized values by specifying `sem`'s `standardized` option, which we can do when we fit the model or when we replay results:

```
. sem, standardized
Structural equation model                       Number of obs     =        216
Estimation method  = ml
Log likelihood     = -9539.3851
 ( 1)  [a1]Affective = 1
 ( 2)  [c1]Cognitive = 1
```

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
a1						
Affective	.9003553	.0143988	62.53	0.000	.8721342	.9285765
a2						
Affective	.9023249	.0141867	63.60	0.000	.8745195	.9301304
a3						
Affective	.9388883	.0097501	96.29	0.000	.9197784	.9579983
a4						
Affective	.8687982	.0181922	47.76	0.000	.8331421	.9044543
a5						
Affective	.9521559	.0083489	114.05	0.000	.9357923	.9685195
c1						
Cognitive	.8523351	.0212439	40.12	0.000	.8106978	.8939725
c2						
Cognitive	.8759601	.0184216	47.55	0.000	.8398544	.9120658
c3						
Cognitive	.863129	.0199624	43.24	0.000	.8240033	.9022547
c4						
Cognitive	.8582786	.0204477	41.97	0.000	.8182018	.8983554
c5						
Cognitive	.8930346	.0166261	53.71	0.000	.8604479	.9256212
var(e.a1)	.1893602	.0259281			.1447899	.2476506
var(e.a2)	.1858097	.0256021			.1418353	.2434179
var(e.a3)	.1184887	.0183086			.0875289	.1603993
var(e.a4)	.2451896	.0316107			.1904417	.3156764
var(e.a5)	.0933991	.015899			.0669031	.1303885
var(e.c1)	.2735248	.0362139			.2110086	.354563
var(e.c2)	.2326939	.0322732			.1773081	.3053806
var(e.c3)	.2550083	.0344603			.1956717	.3323385
var(e.c4)	.2633578	.0350997			.2028151	.3419733
var(e.c5)	.2024893	.0296954			.1519049	.2699183
var(Affect~e)	1	.			.	.
var(Cognit~e)	1	.			.	.
cov(Affect~e, Cognitive)	.8108102	.0268853	30.16	0.000	.758116	.8635045

```
LR test of model vs. saturated: chi2(34)  =     88.88, Prob > chi2 = 0.0000
```

Notes:

1. In addition to obtaining standardized coefficients, the `standardized` option reports estimated error variances as the fraction of the variance that is unexplained. Error variances were previously unintelligible numbers such as 384.136 and 357.352. Now they are 0.189 and 0.186.
2. Also listed in the `sem` output are variances of latent variables. In the previous output, latent variable `Affective` had variance 1,644.46 with standard error 193. In the standardized output, it has variance 1 with standard error missing. The variances of the latent variables are standardized to 1, and obviously, being a normalization, there is no corresponding standard error.
3. We can now see at the bottom of the coefficient table that affective and cognitive arousal are correlated 0.81 because standardized covariances are correlation coefficients.
4. The standardized coefficients for this model can be interpreted as the correlation coefficients between the indicator and the latent variable because each indicator measures only one factor. For instance, the standardized path coefficient `a1<-Affective` is 0.90, meaning the correlation between `a1` and `Affective` is 0.90.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_2fmm
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Change the size of the observed variables' rectangles.

   From the SEM Builder menu, select **Settings > Variables > All observed...**.

   In the resulting dialog box, change the first size to `.38` and click on **OK**.

4. Create the measurement component for affective arousal.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and one-fourth of the way in from the left.

   In the resulting dialog box,

   a. change the *Latent variable name* to `Affective`;

   b. select `a1`, `a2`, `a3`, `a4`, and `a5` by using the *Measurement variables* control;

   c. select `Down` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move this component by clicking on any variable and dragging it.

5. Create the measurement component for cognitive arousal.

   Repeat the process from item 4, but place the measurement component about one-third of the way down from the top and three-fourths of the way in from the left. Label the latent variable `Cognitive`, and select measurement variables `c1`, `c2`, `c3`, `c4`, and `c5`. Drag to reposition if desired.

6. Correlate the latent factors.

   a. Select the Add covariance tool, ⌒.

   b. Click in the upper-right quadrant of the `Affective` oval (it will highlight when you hover over it), and drag a covariance to the upper-left quadrant of the `Cognitive` oval (it will highlight when you can release to connect the covariance).

7. Clean up.

   If you do not like where a covariance has been connected to its variable, use the Select tool, , to click on the covariance, and then simply click on where it connects to an oval and drag the endpoint. You can also change the bow of the covariance by dragging the control point that extends from one end of the selected covariance.

8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

9. Show standardized estimates.

   From the SEM Builder menu, select **View > Standardized estimates**.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_2fmm
```

Obtaining equation-level goodness of fit by using estat eqgof

That the correlation between `a1` and `Affective` is 0.90 implies that the fraction of the variance of `a1` explained by `Affective` is $0.90^2 = 0.81$, and left unexplained is $1 - 0.81 = 0.19$. Instead of manually calculating the proportion of variance explained by indicators, we can use the `estat eqgof` command:

```
. estat eqgof
Equation-level goodness of fit
```

depvars	Variance fitted	Variance predicted	residual	R-squared	mc	mc2
observed						
a1	2028.598	1644.463	384.1359	.8106398	.9003553	.8106398
a2	1923.217	1565.865	357.3524	.8141903	.9023249	.8141903
a3	1307.726	1152.775	154.9507	.8815113	.9388883	.8815113
a4	2024.798	1528.339	496.4594	.7548104	.8687982	.7548104
a5	2052.328	1860.643	191.6857	.9066009	.9521559	.9066009
c1	627.5987	455.9349	171.6638	.7264752	.8523351	.7264752
c2	738.3325	566.527	171.8055	.7673061	.8759601	.7673061
c3	1082.374	806.3598	276.0144	.7449917	.863129	.7449917
c4	851.311	627.1116	224.1994	.7366422	.8582786	.7366422
c5	725.3002	578.4346	146.8655	.7975107	.8930346	.7975107
overall				.9949997		

```
mc  = correlation between depvar and its prediction
mc2 = mc^2 is the Bentler-Raykov squared multiple correlation coefficient
```

Notes:

1. `fitted` reports the fitted variance of each of the endogenous variables, whether observed or latent. In this case, we have observed endogenous variables.
2. `predicted` reports the variance of the predicted value of each endogenous variable.
3. `residual` reports the leftover residual variance.
4. `R-squared` reports R^2, the fraction of variance explained by each indicator. The fraction of the variance of `Affective` explained by `a1` is 0.81, just as we calculated by hand at the beginning of this section. The overall R^2 is also called the coefficient of determination.
5. `mc` stands for multiple correlation, and `mc2` stands for multiple-correlation squared. `R-squared`, `mc`, and `mc2` all report the relatedness of the indicated dependent variable with the model's linear prediction. In recursive models, all three statistics are really the same number. `mc` is equal to the square root of `R-squared`, and `mc2` is equal to `R-squared`.

   In nonrecursive models, these three statistics are different and each can have problems. `R-squared` and `mc` can actually become negative! That does not mean the model has negative predictive power or that it might not even have reasonable predictive power. `mc2` $=$ `mc`^{2} is recommended by Bentler and Raykov (2000) to be used instead of `R-squared` for nonrecursive systems.

In [SEM] **Example 4**, we examine the goodness-of-fit statistics for this model.

In [SEM] **Example 5**, we examine modification indices for this model.

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Bentler, P. M., and T. Raykov. 2000. On measures of explained variance in nonrecursive structural equation models. *Journal of Applied Psychology* 85: 125–131.

Kline, R. B. 2005. *Principles and Practice of Structural Equation Modeling*. 2nd ed. New York: Guilford Press.

Also see

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Example 2** — Creating a dataset from published covariances

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **Example 26** — Fitting a model with data missing at random

[SEM] **Example 31g** — Two-factor measurement model (generalized response)

[SEM] **sem** — Structural equation model estimation command

[SEM] **estat eqgof** — Equation-level goodness-of-fit statistics

Title

Example 4 — Goodness-of-fit statistics

Description Remarks and examples Reference Also see

Description

Here we demonstrate `estat gof`. See [SEM] **Intro 7** and [SEM] **estat gof**.

This example picks up where [SEM] **Example 3** left off:

```
. use https://www.stata-press.com/data/r16/sem_2fmm
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5)
```

Remarks and examples

When we fit this model in [SEM] **Example 3**, at the bottom of the output, we saw

```
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5)
  (output omitted)
LR test of model vs. saturated: chi2(34)   =    88.88, Prob > chi2 = 0.0000
```

Most texts refer to this test against the saturated model as the "model χ^2 test".

These results indicate poor goodness of fit; see [SEM] **Example 1**. The default goodness-of-fit statistic reported by `sem`, however, can be overly influenced by sample size, correlations, variance unrelated to the model, and multivariate nonnormality (Kline 2016, 271).

Goodness of fit in cases of `sem` is a measure of how well you fit the observed moments, which in this case are the covariances between all pairs of `a1`, ..., `a5`, `c1`, ..., `c5`. In a measurement model, the assumed underlying causes are unobserved, and in this example, those unobserved causes are the latent variables `Affective` and `Cognitive`. It may be reasonable to assume that the observed `a1`, ..., `a5`, `c1`, ..., `c5` can be filtered through imagined variables `Affective` and `Cognitive`, but that can be reasonable only if not too much information contained in the original variables is lost. Thus goodness-of-fit statistics are of great interest to those fitting measurement models. Goodness-of-fit statistics are of far less interest when all variables in the model are observed.

Other goodness-of-fit statistics are available.

```
. estat gof, stats(all)
```

Fit statistic	Value	Description
Likelihood ratio		
chi2_ms(34)	88.879	model vs. saturated
p > chi2	0.000	
chi2_bs(45)	2467.161	baseline vs. saturated
p > chi2	0.000	
Population error		
RMSEA	0.087	Root mean squared error of approximation
90% CI, lower bound	0.065	
upper bound	0.109	
pclose	0.004	Probability RMSEA <= 0.05
Information criteria		
AIC	19120.770	Akaike's information criterion
BIC	19191.651	Bayesian information criterion
Baseline comparison		
CFI	0.977	Comparative fit index
TLI	0.970	Tucker-Lewis index
Size of residuals		
SRMR	0.022	Standardized root mean squared residual
CD	0.995	Coefficient of determination

Notes:

1. Desirable values vary from test to test.

2. We asked for all the goodness-of-fit tests. We could have obtained specific tests from the above output by specifying the appropriate option; see [SEM] **estat gof**.

3. Under likelihood ratio, `estat gof` reports two tests. The first is a repeat of the model χ^2 test reported at the bottom of the `sem` output. The saturated model is the model that fits the covariances perfectly. We can reject at the 5% level (or any other level) that the model fits as well as the saturated model.

   The second test is a baseline versus saturated comparison. The baseline model includes the mean and variances of all observed variables plus the covariances of all observed exogenous variables. Different authors define the baseline differently. We can reject at the 5% level (or any other level) that the baseline model fits as well as the saturated model.

4. Under population error, the RMSEA value is reported along with the lower and upper bounds of its 90% confidence interval. Most interpreters of this test check whether the lower bound is below 0.05 or the upper bound is above 0.10. If the lower bound is below 0.05, then they would not reject the hypothesis that the fit is close. If the upper bound is above 0.10, they would not reject the hypothesis that the fit is poor. The logic is to perform one test on each end of the 90% confidence interval and thus have 95% confidence in the result. This model's fit is not close, and its upper limit is just over the bounds of being considered poor.

   Pclose, a commonly used word in reference to this test, is the probability that the RMSEA value is less than 0.05, interpreted as the probability that the predicted moments are close to the moments in the population. This model's fit is not close.

5. Under information criteria are reported AIC and BIC, which contain little information by themselves but are often used to compare models. Smaller values are considered better.
6. Under baseline comparison are reported CFI and TLI, two indices such that a value close to 1 indicates a good fit. TLI is also known as the nonnormed fit index.
7. Under size of residuals is reported the standardized root mean squared residual (SRMR) and the coefficient of determination (CD).

   A perfect fit corresponds to an SRMR of 0, and a good fit corresponds to a "small" value, considered by some to be limited at 0.08. The model fits well by this standard.

   The CD is like an R^2 for the whole model. A value close to 1 indicates a good fit.

`estat gof` provides multiple goodness-of-fit statistics because, across fields, different researchers use different statistics. You should not print them all and look for the one reporting the result you seek.

Reference

Kline, R. B. 2016. *Principles and Practice of Structural Equation Modeling*. 4th ed. New York: Guilford Press.

Also see

[SEM] **Example 3** — Two-factor measurement model

[SEM] **Example 21** — Group-level goodness of fit

[SEM] **estat gof** — Goodness-of-fit statistics

Title

Example 5 — Modification indices

Description Remarks and examples Reference Also see

Description

Here we demonstrate the use of `estat mindices`; see [SEM] **Intro 7** and [SEM] **estat mindices**.

This example picks up where [SEM] **Example 3** left off:

```
. use https://www.stata-press.com/data/r16/sem_2fmm
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5)
```

Remarks and examples

When we fit this model in [SEM] **Example 4**, we allowed the latent variables to be correlated. We typed

```
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5)
```

and by default in the command language, latent exogenous variables are assumed to be correlated unless we specify otherwise. Had we used the Builder, the latent exogenous variables would have been assumed to be uncorrelated unless we had drawn the curved path between them.

The original authors who collected these data analyzed them assuming no covariance, which we could obtain by typing

```
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5), ///
                                          cov(Affective*Cognitive@0)
```

It was Kline (2005, 70–74, 184) who allowed the covariance. Possibly he did that after looking at the modification indices.

The modification indices report statistics on all omitted paths and covariances. Let's begin with the model without the covariance:

```
. sem (Affective -> a1 a2 a3 a4 a5) (Cognitive -> c1 c2 c3 c4 c5),
>                                   cov(Affective*Cognitive@0)
  (output omitted)
. estat mindices
Modification indices
```

	MI	df	P>MI	EPC	Standard EPC
Measurement					
a5					
Cognitive	8.059	1	0.00	.1604476	.075774
c5					
Affective	5.885	1	0.02	.0580897	.087733
cov(e.a1,e.a4)	5.767	1	0.02	84.81133	.1972802
cov(e.a1,e.a5)	7.597	1	0.01	-81.82092	-.2938627
cov(e.a2,e.a4)	14.300	1	0.00	129.761	.3110565
cov(e.a2,e.c4)	4.071	1	0.04	-45.44807	-.1641344
cov(e.a3,e.a4)	21.183	1	0.00	-116.8181	-.4267012
cov(e.a3,e.a5)	25.232	1	0.00	118.4674	.6681337
cov(e.a5,e.c4)	4.209	1	0.04	39.07999	.184049
cov(e.c1,e.c3)	11.326	1	0.00	66.3965	.3098331
cov(e.c1,e.c5)	8.984	1	0.00	-47.31483	-.2931597
cov(e.c3,e.c4)	12.668	1	0.00	-80.98353	-.333871
cov(e.c4,e.c5)	4.483	1	0.03	38.6556	.2116015
cov(Affective,Cognitive)	128.482	1	0.00	704.4469	.8094959

```
EPC = expected parameter change
```

Notes:

1. Four columns of results are reported.
   a. `MI` stands for modification index and is an approximation to the change in the model's goodness-of-fit χ^2 if the path were added.
   b. `df` stands for degrees of freedom and is the number that would be added to d of the $\chi^2(d)$.
   c. `P>MI` is the value of the significance of $\chi^2(\texttt{df})$.
   d. `EPC` stands for expected parameter change and is an approximation to the value of the parameter if it were not constrained to 0. It is reported in unstandardized (column 3) and standardized (column 4) units.
2. There are lots of significant omitted paths and covariances in the above output.
3. Paths and covariances are listed only if the modification index is significant at the 0.05 level, corresponding to $\chi^2(1)$ value 3.8414588. You may specify the `minchi2()` option to use different $\chi^2(1)$ values. Specify `minchi2(0)` if you wish to see all tests.
4. The omitted covariance between `Affective` and `Cognitive` has the largest change in χ^2 observed. Perhaps this is why Kline (2005, 70–74, 184) allowed a covariance between the two latent variables. The standardized `EPC` reports the relaxed-constraint correlation value, which is the value reported for the unconstrained correlation path in [SEM] **Example 3**.

Another way of dealing with this significant result would be to add a direct path between the variables, but that perhaps would have invalidated the theory being proposed. The original authors instead proposed a second-order model postulating that `Affective` and `Cognitive` are themselves measurements of another latent variable that might be called `Arousal`.

Reference

Kline, R. B. 2005. *Principles and Practice of Structural Equation Modeling*. 2nd ed. New York: Guilford Press.

Also see

[SEM] **Example 3** — Two-factor measurement model

[SEM] **estat mindices** — Modification indices

Title

Example 6 — Linear regression

Description Remarks and examples Also see

Description

Linear regression is demonstrated using `auto.dta`:

```
. sysuse auto
(1978 Automobile Data)
```

See *Structural models 1: Linear regression* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

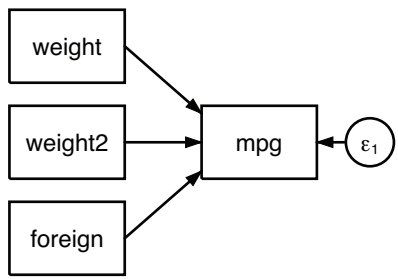
Fitting linear regression models
Displaying standardized results
Fitting the model with the Builder

Fitting linear regression models

The first two examples in [R] **regress** are

```
. regress mpg weight c.weight#c.weight foreign
. regress, beta
```

This model corresponds to



To fit this model with sem, we type

```
. generate weight2 = weight^2
. sem (mpg <- weight weight2 foreign)
Endogenous variables
Observed:  mpg
Exogenous variables
Observed:  weight weight2 foreign
Fitting target model:
Iteration 0:   log likelihood = -1909.8206
Iteration 1:   log likelihood = -1909.8206
Structural equation model                       Number of obs     =         74
Estimation method  = ml
Log likelihood     = -1909.8206
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
mpg						
weight	-.0165729	.0038604	-4.29	0.000	-.0241392	-.0090067
weight2	1.59e-06	6.08e-07	2.62	0.009	4.00e-07	2.78e-06
foreign	-2.2035	1.03022	-2.14	0.032	-4.222695	-.1843056
_cons	56.53884	6.027559	9.38	0.000	44.72504	68.35264
var(e.mpg)	10.19332	1.675772			7.385485	14.06865

```
Note: The LR test of model vs. saturated is not reported because the fitted
      model is not full rank.
```

Notes:

1. We wished to include variable weight^2 in our model. Because sem does not allow Stata's factor-variable notation, we first had to generate new variable weight2.
2. Reported coefficients match those reported by regress.
3. Reported standard errors (SEs) differ slightly from those reported by regress. For instance, the SE for foreign is reported here as 1.03, whereas regress reported 1.06. SEM is an asymptotic estimator, and sem divides variances and covariances by $N = 74$, the number of observations. regress provides unbiased finite-sample estimates and divides by $N-k-1 = 74-3-1 = 70$. Note that $1.03\sqrt{74/70} = 1.06$.
4. sem reports z statistics whereas regress reports t statistics.
5. Reported confidence intervals differ slightly between sem and regress because of the $(N-k-1)/N$ issue.
6. sem reports the point estimate of e.mpg as 10.19332. regress reports the root MSE as 3.2827. And $\sqrt{10.19332 \times 74/70} = 3.2827$.

Displaying standardized results

To obtain standardized coefficients from regress, you specify the beta option. To obtain standardized coefficients from sem, you specify the standardized option.

```
. sem, standardized
Structural equation model                       Number of obs     =         74
Estimation method  = ml
Log likelihood     = -1909.8206
```

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
mpg						
weight	-2.226321	.4950378	-4.50	0.000	-3.196577	-1.256064
weight2	1.32654	.498261	2.66	0.008	.3499662	2.303113
foreign	-.17527	.0810378	-2.16	0.031	-.3341011	-.0164389
_cons	9.839209	.9686872	10.16	0.000	7.940617	11.7378
var(e.mpg)	.308704	.0482719			.2272168	.4194152

```
Note: The LR test of model vs. saturated is not reported because the fitted
      model is not full rank.
```

`regress` simply reports standardized coefficients in an extra column. All other results are reported in unstandardized form. `sem` updates the entire output with the standardized values.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset and create the additional variable `weight2`.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/auto
   . generate weight2 = weight^2
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the regression component for the `mpg` outcome.

   Select the Add regression component tool, , and then click in the center of the diagram.

   In the resulting dialog box,

   a. select `mpg` in the *Dependent variable* control;

   b. select `weight`, `weight2`, and `foreign` by using the *Independent variables* control;

   c. select `Left` in the *Independent variables' direction* control;

   d. click on **OK**.

   If you wish, move this component by clicking on any variable and dragging it.

4. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

5. Show standardized estimates.

   From the SEM Builder menu, select **View > Standardized estimates**.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_regress
```

Also see

[SEM] **Example 12** — Seemingly unrelated regression

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Example 43g** — Tobit regression

[SEM] **Example 44g** — Interval regression

[SEM] **sem** — Structural equation model estimation command

Title

Example 7 — Nonrecursive structural model

Description Remarks and examples References Also see

Description

To demonstrate a nonrecursive structural model with all variables observed, we use data from Duncan, Haller, and Portes (1968):

```
. use https://www.stata-press.com/data/r16/sem_sm1
(Structural model with all observed values)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_sm1.dta
    obs:             329                  Structural model with all obse..
   vars:              10                  25 May 2018 10:13
                                          (_dta has notes)
  ------------------------------------------------------------------------------
  variable name                  variable label
  ------------------------------------------------------------------------------
  r_intel                        respondent's intelligence
  r_parasp                       respondent's parental aspiration
  r_ses                          respondent's family socioeconomic status
  r_occasp                       respondent's occupational aspiration
  r_educasp                      respondent's educational aspiration
  f_intel                        friend's intelligence
  f_parasp                       friend's parental aspiration
  f_ses                          friend's family socioeconomic status
  f_occasp                       friend's occupational aspiration
  f_educasp                      friend's educational aspiration
  ------------------------------------------------------------------------------
. notes
_dta:
  1.  Summary statistics data from Duncan, O.D., Haller, A.O., and Portes, A.,
      1968, "Peer Influences on Aspirations:  A Reinterpretation", _American
      Journal of Sociology_ 74, 119-137.
  2.  The data contain 329 boys with information on five variables and the same
      information for each boy's best friend.
```

If you typed `ssd status`, you would learn that this dataset contains the correlation matrix only. Variances (standard deviations) and means are undefined. Thus we need to use this dataset cautiously. It is always better if you enter the variances and means if you have them.

That these data are the correlations only will not matter for how we will use them.

Remarks and examples

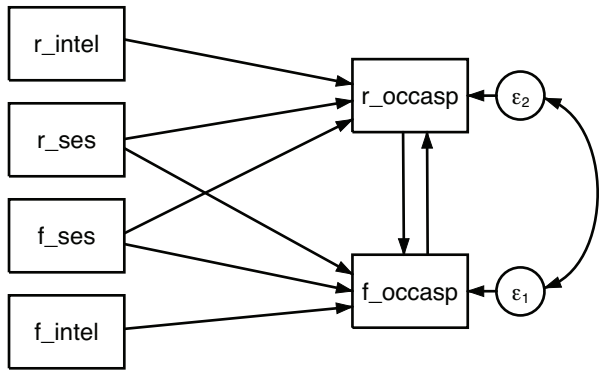
See *Structural models 8: Dependencies between response variables* in [SEM] **Intro 5** for background.

Remarks are presented under the following headings:

Fitting the model
Fitting the model with the Builder
Checking stability with estat stable
Reporting total, direct, and indirect effects with estat teffects

Fitting the model

In the referenced paper above, the authors fit the following model:



```
. sem (r_occasp <- f_occasp r_intel r_ses f_ses)
>     (f_occasp <- r_occasp f_intel f_ses r_ses),
>                               cov(e.r_occasp*e.f_occasp) standardized
Endogenous variables
Observed:  r_occasp f_occasp
Exogenous variables
Observed:  r_intel r_ses f_ses f_intel
Fitting target model:
Iteration 0:   log likelihood = -2617.0489
Iteration 1:   log likelihood = -2617.0489
Structural equation model                       Number of obs     =        329
Estimation method  = ml
Log likelihood     = -2617.0489
```

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
r_occasp						
f_occasp	.2773441	.1281904	2.16	0.031	.0260956	.5285926
r_intel	.2854766	.05	5.71	0.000	.1874783	.3834748
r_ses	.1570082	.0520841	3.01	0.003	.0549252	.2590912
f_ses	.0973327	.060153	1.62	0.106	-.020565	.2152304
f_occasp						
r_occasp	.2118102	.156297	1.36	0.175	-.0945264	.5181467
r_ses	.0794194	.0587732	1.35	0.177	-.0357739	.1946127
f_ses	.1681772	.0537199	3.13	0.002	.062888	.2734663
f_intel	.3693682	.0525924	7.02	0.000	.2662891	.4724474
var(e.r_oc~p)	.6889244	.0399973			.6148268	.7719519
var(e.f_oc~p)	.6378539	.039965			.5641425	.7211964
cov(e.r_oc~p, e.f_occasp)	-.2325666	.2180087	-1.07	0.286	-.6598558	.1947227

```
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

Notes:

1. We specified the `standardized` option, but in this case that did not matter much because these data are based on the correlation coefficients only, so standardized values are equal to unstandardized values. The exception is the correlation between the latent endogenous variables, as reflected in the correlation of their errors, and we wanted to show that results match those in the original paper.

2. Nearly all results match those in the original paper. The authors normalized the errors to have a variance of 1; `sem` normalizes the paths from the errors to have coefficient 1. While you can apply most normalizing constraints any way you wish, `sem` restricts errors to have path coefficients of 1 and this cannot be modified. You could, however, prove to yourself that `sem` would produce the same variances as the authors produced by typing

```
. sem, coeflegend
. display sqrt(_b[/var(e.r_occasp)])
. display sqrt(_b[/var(e.f_occasp)])
```

because the coefficients would be the standard deviations of the errors estimated without the variance-1 constraint. Thus all results match. We replayed results by using the `coeflegend` option so that we would know what to type to refer to the two error variances, namely, `_b[/var(e.r_occasp)]` and `_b[/var(e.f_occasp)]`.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_sm1
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the four independent variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-third of the way in from the left and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the four variables in this order: `r_intel`, `r_ses`, `f_ses`, and `f_intel`;

   c. select `Vertical` in the *Orientation* control;

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

4. Create the two dependent variables.

   Select the Add observed variables set tool, , and then click in the diagram about two-thirds of the way in from the left, vertically aligned with the top of the `f_intel` rectangle.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the two variable names `r_occasp` and `f_occasp`;

   c. select `Vertical` in the *Orientation* control;

   d. select the **Distances** tab;

   e. select `.5` (inch) from the *Distance between variables* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create paths from the independent variables to the dependent variables.

   a. Select the Add path tool, .

   b. Click in the right side of the `r_intel` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `r_occasp` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the ⟶ tool, create the following paths by first clicking in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable.

```
r_ses -> r_occasp
r_ses -> f_occasp
f_ses -> r_occasp
f_ses -> f_occasp
f_intel -> f_occasp
```

6. Create paths for the feedback loop between endogenous variables.

   a. Continue with the Add path tool, ⟶.

   b. Click in the bottom of the `r_occasp` rectangle, slightly to the left of the center, and drag a path to the `f_occasp` rectangle.

   c. Click in the top of the `f_occasp` rectangle, slightly to the right of the center, and drag a path to the `r_occasp` rectangle.

7. Correlate the error terms.

   a. Select the Add covariance tool, ⌒.

   b. Click in the error term for `r_occasp` (the circle labeled ϵ_1), and drag a covariance to the error term for `f_occasp` (the circle labeled ϵ_2).

8. Clean up.

   If you do not like where a path has been connected to its variable, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint. Similarly, you can change where the covariance connects to the error terms by clicking on the covariance and dragging the endpoint. You can also change the bow of the covariance by clicking on the covariance and dragging the control point that extends from one end of the selected covariance.

9. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

10. Show standardized estimates.

    From the SEM Builder menu, select **View > Standardized estimates**.

Tips: When you draw paths that should be exactly horizontal or vertical, such as the two between `r_occasp` and `f_occasp`, holding the *Shift* key as you drag the path will guarantee that the line is perfectly vertical. Also, when drawing paths from the independent variables to the dependent variables, you may find it more convenient to change the automation settings as described in the tips of [SEM] **Example 9**. However, this does not work for the feedback loop between the dependent variables.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_sm1
```

Checking stability with estat stable

```
. estat stable
Stability analysis of simultaneous equation systems
   Eigenvalue stability condition
```

Eigenvalue	Modulus
.2423722	.242372
-.2423722	.242372

```
   stability index =  .2423722
   All the eigenvalues lie inside the unit circle.
   SEM satisfies stability condition.
```

Notes:

1. `estat stable` is for use on nonrecursive models. Recursive models are, by design, stable.
2. Stability concerns whether the parameters of the model are such that the model would blow up if it were operated over and over again. If the results are found not to be stable, then that casts questions about the validity of the model.
3. The stability is the maximum of the moduli, and the moduli are the absolute values of the eigenvalues. Usually, the two eigenvalues are not identical, but it is a property of this model that they are.
4. If the stability index is less than 1, then the reported estimates yield a stable model.

In the next section, we use `estat teffects` to estimate total effects. That is appropriate only if the model is stable, as we find that it is.

Reporting total, direct, and indirect effects with estat teffects

```
. estat teffects
```

Direct effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
r_occasp						
r_occasp	0	(no path)				
f_occasp	.2773441	.1287622	2.15	0.031	.0249748	.5297134
r_intel	.2854766	.0522001	5.47	0.000	.1831662	.3877869
r_ses	.1570082	.052733	2.98	0.003	.0536534	.260363
f_ses	.0973327	.0603699	1.61	0.107	-.0209901	.2156555
f_intel	0	(no path)				
f_occasp						
r_occasp	.2118102	.1563958	1.35	0.176	-.09472	.5183404
f_occasp	0	(no path)				
r_intel	0	(no path)				
r_ses	.0794194	.0589095	1.35	0.178	-.0360411	.1948799
f_ses	.1681772	.0543854	3.09	0.002	.0615838	.2747705
f_intel	.3693682	.0557939	6.62	0.000	.2600142	.4787223

Indirect effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
r_occasp						
r_occasp	.0624106	.0594905	1.05	0.294	-.0541886	.1790097
f_occasp	.0173092	.0220932	0.78	0.433	-.0259928	.0606112
r_intel	.0178168	.0159383	1.12	0.264	-.0134217	.0490552
r_ses	.0332001	.0204531	1.62	0.105	-.0068872	.0732875
f_ses	.0556285	.0292043	1.90	0.057	-.0016109	.112868
f_intel	.1088356	.052243	2.08	0.037	.0064411	.21123
f_occasp						
r_occasp	.0132192	.0215569	0.61	0.540	-.0290316	.05547
f_occasp	.0624106	.0594905	1.05	0.294	-.0541886	.1790097
r_intel	.0642406	.0490164	1.31	0.190	-.0318298	.160311
r_ses	.0402881	.0315496	1.28	0.202	-.021548	.1021242
f_ses	.0323987	.0262124	1.24	0.216	-.0189765	.083774
f_intel	.0230525	.0202112	1.14	0.254	-.0165607	.0626657

Total effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
r_occasp						
r_occasp	.0624106	.0594905	1.05	0.294	-.0541886	.1790097
f_occasp	.2946533	.1468389	2.01	0.045	.0068544	.5824522
r_intel	.3032933	.0509684	5.95	0.000	.2033971	.4031896
r_ses	.1902083	.050319	3.78	0.000	.091585	.2888317
f_ses	.1529612	.050844	3.01	0.003	.0533089	.2526136
f_intel	.1088356	.052243	2.08	0.037	.0064411	.21123
f_occasp						
r_occasp	.2250294	.1770669	1.27	0.204	-.1220154	.5720742
f_occasp	.0624106	.0594905	1.05	0.294	-.0541886	.1790097
r_intel	.0642406	.0490164	1.31	0.190	-.0318298	.160311
r_ses	.1197074	.0483919	2.47	0.013	.0248611	.2145537
f_ses	.2005759	.0488967	4.10	0.000	.10474	.2964118
f_intel	.3924207	.0502422	7.81	0.000	.2939478	.4908936

Note:

1. In the path diagram we drew for this model, you can see that the intelligence of the respondent, `r_intel`, has both direct and indirect effects on the occupational aspiration of the respondent, `r_occasp`. The tables above reveal that

$$0.3033 = 0.2855 + 0.0178$$

where 0.2855 is the direct effect and 0.0178 is the indirect effect.

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Duncan, O. D., A. O. Haller, and A. Portes. 1968. Peer influences on aspirations: A reinterpretation. *American Journal of Sociology* 74: 119–137.

Also see

[SEM] **Example 8** — Testing that coefficients are equal, and constraining them

[SEM] **sem** — Structural equation model estimation command

[SEM] **estat stable** — Check stability of nonrecursive system

[SEM] **estat teffects** — Decomposition of effects into total, direct, and indirect

Title

Example 8 — Testing that coefficients are equal, and constraining them

Description Remarks and examples Also see

Description

This example continues where [SEM] **Example 7** left off, where we typed

```
. use https://www.stata-press.com/data/r16/sem_sm1
. ssd describe
. notes
. sem (r_occasp <- f_occasp r_intel r_ses f_ses)  ///
      (f_occasp <- r_occasp f_intel f_ses r_ses), ///
                  cov(e.r_occasp*e.f_occasp) standardized
. estat stable
. estat teffects
```

Remarks and examples

Remarks are presented under the following headings:

Using test to evaluate adding constraints
Refitting the model with added constraints
Using estat scoretests to test whether constraints can be relaxed

We want to show you how to evaluate potential constraints after estimation, how to fit a model with constraints, and how to evaluate enforced constraints after estimation.

Obviously, in a real analysis, if you evaluated potential constraints after estimation, there would be no reason to evaluate enforced constraints after estimation, and vice versa.

Using test to evaluate adding constraints

In this model of respondents and corresponding friends, it would be surprising if the coefficients relating friends' characteristics to respondents' occupational aspirations and vice versa were not equal. It would also be surprising if coefficients relating a respondent's characteristics to his occupational aspirations were not equal to those of his friends' characteristics to his occupational aspirations. The paths that we suspect should be equal are

```
r_intel  -> r_occasp          f_intel  -> f_occasp
r_ses    -> r_occasp          f_ses    -> f_occasp
f_ses    -> r_occasp          r_ses    -> f_occasp
f_occasp -> r_occasp          r_occasp -> f_occasp
```

You are about to learn that to test whether those paths have equal coefficients, you type

```
. test (_b[r_occasp:r_intel ]==_b[f_occasp:f_intel ])   ///
       (_b[r_occasp:r_ses   ]==_b[f_occasp:f_ses   ])   ///
       (_b[r_occasp:f_ses   ]==_b[f_occasp:r_ses   ])   ///
       (_b[r_occasp:f_occasp]==_b[f_occasp:r_occasp])
```

In Stata, `_b[]` is how one accesses the estimated parameters. It is difficult to remember what the names are. To determine the names of the parameters, replay the `sem` results with the `coeflegend` option:

```
. sem, coeflegend
Structural equation model                       Number of obs     =        329
Estimation method  = ml
Log likelihood     = -2617.0489

------------------------------------------------------------------------------
             |      Coef.  Legend
-------------+----------------------------------------------------------------
Structural   |
  r_occasp   <-
    f_occasp |   .2773441  _b[r_occasp:f_occasp]
     r_intel |   .2854766  _b[r_occasp:r_intel]
       r_ses |   .1570082  _b[r_occasp:r_ses]
       f_ses |   .0973327  _b[r_occasp:f_ses]
  -----------+----------------------------------------------------------------
  f_occasp   <-
    r_occasp |   .2118102  _b[f_occasp:r_occasp]
       r_ses |   .0794194  _b[f_occasp:r_ses]
       f_ses |   .1681772  _b[f_occasp:f_ses]
     f_intel |   .3693682  _b[f_occasp:f_intel]
-------------+----------------------------------------------------------------
var(e.r_oc~p)|   .6868304  _b[/var(e.r_occasp)]
var(e.f_oc~p)|   .6359151  _b[/var(e.f_occasp)]
-------------+----------------------------------------------------------------
cov(e.r_oc~p,|
  e.f_occasp)|  -.1536992  _b[/cov(e.r_occasp,e.f_occasp)]
------------------------------------------------------------------------------
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

With the parameter names at hand, to perform the test, we can type

```
. test (_b[r_occasp:r_intel ]==_b[f_occasp:f_intel ])
>      (_b[r_occasp:r_ses   ]==_b[f_occasp:f_ses   ])
>      (_b[r_occasp:f_ses   ]==_b[f_occasp:r_ses   ])
>      (_b[r_occasp:f_occasp]==_b[f_occasp:r_occasp])
 ( 1)  [r_occasp]r_intel - [f_occasp]f_intel = 0
 ( 2)  [r_occasp]r_ses - [f_occasp]f_ses = 0
 ( 3)  [r_occasp]f_ses - [f_occasp]r_ses = 0
 ( 4)  [r_occasp]f_occasp - [f_occasp]r_occasp = 0

           chi2(  4) =    1.61
         Prob > chi2 =    0.8062
```

We cannot reject the constraint, just as we expected.

Refitting the model with added constraints

We could refit the model with these constraints by typing

```
. sem (r_occasp <- f_occasp@b1  r_intel@b2  r_ses@b3  f_ses@b4)
>     (f_occasp <- r_occasp@b1  f_intel@b2  f_ses@b3  r_ses@b4),
>                              cov(e.r_occasp*e.f_occasp)
Endogenous variables
Observed:  r_occasp f_occasp
Exogenous variables
Observed:  r_intel r_ses f_ses f_intel
Fitting target model:
Iteration 0:   log likelihood = -2617.8735
Iteration 1:   log likelihood = -2617.8705
Iteration 2:   log likelihood = -2617.8705
Structural equation model                       Number of obs     =        329
Estimation method  = ml
Log likelihood     = -2617.8705
 ( 1)  [r_occasp]f_occasp - [f_occasp]r_occasp = 0
 ( 2)  [r_occasp]r_intel - [f_occasp]f_intel = 0
 ( 3)  [r_occasp]r_ses - [f_occasp]f_ses = 0
 ( 4)  [r_occasp]f_ses - [f_occasp]r_ses = 0
```

	OIM					
	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
r_occasp						
f_occasp	.2471578	.1024504	2.41	0.016	.0463588	.4479568
r_intel	.3271847	.0407973	8.02	0.000	.2472234	.4071459
r_ses	.1635056	.0380582	4.30	0.000	.0889129	.2380984
f_ses	.088364	.0427106	2.07	0.039	.0046529	.1720752
f_occasp						
r_occasp	.2471578	.1024504	2.41	0.016	.0463588	.4479568
r_ses	.088364	.0427106	2.07	0.039	.0046529	.1720752
f_ses	.1635056	.0380582	4.30	0.000	.0889129	.2380984
f_intel	.3271847	.0407973	8.02	0.000	.2472234	.4071459
var(e.r_oc~p)	.6884513	.0538641			.5905757	.8025477
var(e.f_oc~p)	.6364713	.0496867			.5461715	.7417005
cov(e.r_oc~p, e.f_occasp)	-.1582175	.1410111	-1.12	0.262	-.4345942	.1181592

```
LR test of model vs. saturated: chi2(4)   =      1.64, Prob > chi2 = 0.8010
```

Using estat scoretests to test whether constraints can be relaxed

```
. estat scoretests
(no score tests to report; all chi2 values less than 3.841458820694123)
```

No tests were reported because no tests were individually significant at the 5% level. We can obtain all the individual tests by adding the `minchi2(0)` option, which we can abbreviate to `min(0)`:

```
. estat scoretests, min(0)
Score tests for linear constraints
  ( 1)  [r_occasp]f_occasp - [f_occasp]r_occasp = 0
  ( 2)  [r_occasp]r_intel - [f_occasp]f_intel = 0
  ( 3)  [r_occasp]r_ses - [f_occasp]f_ses = 0
  ( 4)  [r_occasp]f_ses - [f_occasp]r_ses = 0
```

	chi2	df	P>chi2
(1)	0.014	1	0.91
(2)	1.225	1	0.27
(3)	0.055	1	0.81
(4)	0.136	1	0.71

Notes:

1. When we began this example, we used `test` to evaluate potential constraints that we were considering. We obtained an overall $\chi^2(4)$ statistic of 1.61 and thus could not reject the constraints at any reasonable level.
2. We then refit the model with those constraints.
3. For pedantic reasons, now we use `estat scoretests` to evaluate relaxing constraints included in the model. `estat scoretests` does not report a joint test. You cannot sum the χ^2 values to obtain a joint test statistic. Thus we learn only that the individual constraints should not be relaxed at reasonable confidence levels.
4. Thus when evaluating multiple constraints, it is better to fit the model without the constraints and use `test` to evaluate them jointly.

Also see

[SEM] **Example 7** — Nonrecursive structural model

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **estat scoretests** — Score tests

[SEM] **test** — Wald test of linear hypotheses

Title

Example 9 — Structural model with measurement component

Description Remarks and examples References Also see

Description

To demonstrate a structural model with a measurement component, we use data from Wheaton et al. (1977):

```
. use https://www.stata-press.com/data/r16/sem_sm2
(Structural model with measurement component)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_sm2.dta
    obs:             932                  Structural model with measurem..
   vars:              13                  25 May 2018 11:45
                                          (_dta has notes)
```

variable name	variable label
educ66	Education, 1966
occstat66	Occupational status, 1966
anomia66	Anomia, 1966
pwless66	Powerlessness, 1966
socdist66	Latin American social distance, 1966
occstat67	Occupational status, 1967
anomia67	Anomia, 1967
pwless67	Powerlessness, 1967
socdist67	Latin American social distance, 1967
occstat71	Occupational status, 1971
anomia71	Anomia, 1971
pwless71	Powerlessness, 1971
socdist71	Latin American social distance, 1971

```
. notes
_dta:
  1.  Summary statistics data from Wheaton, B., Muthen B., Alwin, D., &
      Summers, G., 1977, "Assessing reliability and stability in panel models",
      in D. R. Heise (Ed.), _Sociological Methodology 1977_ (pp. 84-136), San
      Francisco: Jossey-Bass, Inc.
  2.  Four indicators each measured in 1966, 1967, and 1981, plus another
      indicator (educ66) measured only in 1966.
  3.  Intended use: Create structural model relating Alienation in 1971,
      Alienation in 1967, and SES in 1966.
```

See *Structural models 9: Unobserved inputs, outputs, or both* in [SEM] **Intro 5** for background.

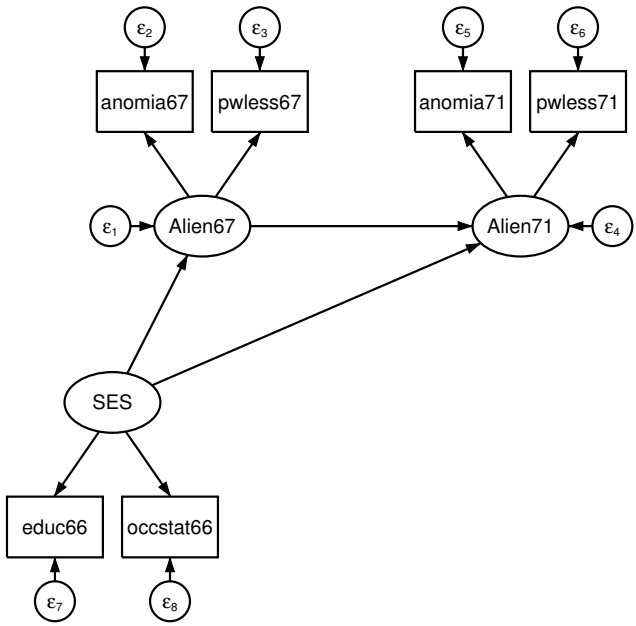
Remarks and examples

Remarks are presented under the following headings:

Fitting the model
Fitting the model with the Builder
Evaluating omitted paths with estat mindices
Refitting the model

Fitting the model

Simplified versions of the model fit by the authors of the referenced paper appear in many SEM software manuals. A simplified model is




```
. sem                                   ///
>     (anomia67 pwless67 <- Alien67)    /// measurement piece
>     (anomia71 pwless71 <- Alien71)    /// measurement piece
>     (Alien67 <- SES)                  /// structural  piece
>     (Alien71 <- Alien67 SES)          /// structural  piece
>     (    SES -> educ66 occstat66)     //  measurement piece
Endogenous variables
Measurement:  anomia67 pwless67 anomia71 pwless71 educ66 occstat66
Latent:       Alien67 Alien71
Exogenous variables
Latent:       SES
Fitting target model:
Iteration 0:   log likelihood = -15249.988
Iteration 1:   log likelihood = -15246.584
Iteration 2:   log likelihood = -15246.469
Iteration 3:   log likelihood = -15246.469
```

```
Structural equation model                       Number of obs     =        932
Estimation method  = ml
Log likelihood     = -15246.469
 ( 1)  [anomia67]Alien67 = 1
 ( 2)  [anomia71]Alien71 = 1
 ( 3)  [educ66]SES = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
Alien67						
SES	-.6140404	.0562407	-10.92	0.000	-.7242701	-.5038107
Alien71						
Alien67	.7046342	.0533512	13.21	0.000	.6000678	.8092007
SES	-.1744153	.0542489	-3.22	0.001	-.2807413	-.0680894
Measurement						
anomia67						
Alien67	1	(constrained)				
_cons	13.61	.1126205	120.85	0.000	13.38927	13.83073
pwless67						
Alien67	.8884887	.0431565	20.59	0.000	.8039034	.9730739
_cons	14.67	.1001798	146.44	0.000	14.47365	14.86635
anomia71						
Alien71	1	(constrained)				
_cons	14.13	.1158943	121.92	0.000	13.90285	14.35715
pwless71						
Alien71	.8486022	.0415205	20.44	0.000	.7672235	.9299808
_cons	14.9	.1034537	144.03	0.000	14.69723	15.10277
educ66						
SES	1	(constrained)				
_cons	10.9	.1014894	107.40	0.000	10.70108	11.09892
occstat66						
SES	5.331259	.4307503	12.38	0.000	4.487004	6.175514
_cons	37.49	.6947112	53.96	0.000	36.12839	38.85161
var(e.ano~67)	4.009921	.3582978			3.365724	4.777416
var(e.pwl~67)	3.187468	.283374			2.677762	3.794197
var(e.ano~71)	3.695593	.3911512			3.003245	4.54755
var(e.pwl~71)	3.621531	.3037908			3.072483	4.268693
var(e.educ66)	2.943819	.5002527			2.109908	4.107319
var(e.occ~66)	260.63	18.24572			227.2139	298.9605
var(e.Ali~67)	5.301416	.483144			4.434225	6.338201
var(e.Ali~71)	3.737286	.3881546			3.048951	4.581019
var(SES)	6.65587	.6409484			5.511067	8.038482

```
LR test of model vs. saturated: chi2(6)   =     71.62, Prob > chi2 = 0.0000
```

Notes:

1. Measurement component: In both 1967 and 1971, anomia and powerlessness are used to measure endogenous latent variables representing alienation for the same two years. Education and occupational status are used to measure the exogenous latent variable SES.

2. Structural component: `SES->Alien67` and `SES->Alien71`, and `Alien67->Alien71`.
3. The model versus saturated χ^2 test indicates that the model is a poor fit.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_sm2
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for alienation in 1967.

   Select the Add measurement component tool, , and then click in the diagram about halfway down from the top and one-third of the way in from the left.

   In the resulting dialog box,

   a. change the *Latent variable name* to `Alien67`;

   b. select `anomia67` and `pwless67` by using the *Measurement variables* control;

   c. select `Up` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

4. Create the measurement component for alienation in 1971.

   Repeat the process from item 3, but place the measurement component about halfway down from the top and two-thirds of the way in from the left. Label the latent variable `Alien71`, and select measurement variables `anomia71` and `pwless71`. Again drag to reposition if desired.

5. Create the measurement component for socioeconomic status.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way up from the bottom and one-fourth of the way in from the left.

   In the resulting dialog box,

   a. change the *Latent variable name* to `SES`;

   b. select `educ66` and `occstat66` by using the *Measurement variables* control;

   c. select `Down` in the *Measurement direction* control;

   d. click on **OK**.

6. Create a path from `Alien67` to `Alien71`.

   a. Select the Add path tool, .

   b. Click in the right half of the `Alien67` oval (it will highlight when you hover over it), and drag a path to the left half of the `Alien71` oval (it will highlight when you can release to connect the path).

7. Create a path from `SES` to `Alien67`.

   Continuing with the tool, click in the upper portion of the `SES` oval, and drag a path to the lower-left quadrant of the `Alien67` oval.

8. Create a path from `SES` to `Alien71`.

   Continuing with the tool, click in the upper-right quadrant of the `SES` oval, and drag a path to the lower-left quadrant of the `Alien71` oval.

9. Clean up the direction of the errors.

   The error on `Alien67` may have been created under the path from `Alien67` to `Alien71`. If so, choose the Select tool, , and then click in the `Alien67` oval. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is where you want it.

10. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

Tips: We took some care above to click in specific quadrants of the ovals when creating paths. This is because, by default, the Builder places the ends of paths on the boundary of the ovals and rectangles closest to where we click. When you create paths, it is often convenient to change the automation settings (**Settings > Automation...**) so that *Connection attachment points* is *Attach based on position of variables* rather than *Attach nearest to drop point*. With the *Connection attachment points* setting, the paths connect nicely regardless of where we click or drop in the variables' ovals and rectangles.

Note that when you create covariances with the Covariances tool, , *Attach nearest to drop point* is sometimes preferred.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_sm2
```

Evaluating omitted paths with estat mindices

That the model is a poor fit leads us to look at the modification indices:

```
. estat mindices
Modification indices
```

		MI	df	P>MI	EPC	Standard EPC
Measurement						
anomia67						
	anomia71	51.977	1	0.00	.3906425	.4019984
	pwless71	32.517	1	0.00	-.2969297	-.2727609
	educ66	5.627	1	0.02	.0935048	.0842631
pwless67						
	anomia71	41.618	1	0.00	-.3106995	-.3594367
	pwless71	23.622	1	0.00	.2249714	.2323233
	educ66	6.441	1	0.01	-.0889042	-.0900664
anomia71						
	anomia67	58.768	1	0.00	.429437	.4173061
	pwless67	38.142	1	0.00	-.3873066	-.3347904
pwless71						
	anomia67	46.188	1	0.00	-.3308484	-.3601641
	pwless67	27.760	1	0.00	.2871709	.2780833
educ66						
	anomia67	4.415	1	0.04	.1055965	.1171781
	pwless67	6.816	1	0.01	-.1469371	-.1450411
cov(e.anomia67,e.anomia71)		63.786	1	0.00	1.951578	.5069627
cov(e.anomia67,e.pwless71)		49.892	1	0.00	-1.506704	-.3953794
cov(e.anomia67,e.educ66)		6.063	1	0.01	.5527612	.1608845
cov(e.pwless67,e.anomia71)		49.876	1	0.00	-1.534199	-.4470094
cov(e.pwless67,e.pwless71)		37.357	1	0.00	1.159123	.341162
cov(e.pwless67,e.educ66)		7.752	1	0.01	-.5557802	-.1814365

EPC = expected parameter change

Notes:

1. There are lots of statistically significant paths we could add to the model.
2. Some of those statistically significant paths also make theoretical sense.
3. Two in particular that make theoretical sense are the covariances between `e.anomia67` and `e.anomia71` and between `e.pwless67` and `e.pwless71`.

Refitting the model

Let's refit the model and include those two previously excluded covariances:

```
. sem                                                   ///
>     (anomia67 pwless67 <- Alien67)                    /// measurement piece
>     (anomia71 pwless71 <- Alien71)                    /// measurement piece
>     (Alien67 <- SES)                                  /// structural  piece
>     (Alien71 <- Alien67 SES)                          /// structural  piece
>     (    SES -> educ66 occstat66)                     /// measurement piece
>                     , cov(e.anomia67*e.anomia71) ///
>                       cov(e.pwless67*e.pwless71)
Endogenous variables
Measurement:  anomia67 pwless67 anomia71 pwless71 educ66 occstat66
Latent:       Alien67 Alien71
Exogenous variables
Latent:       SES
Fitting target model:
Iteration 0:   log likelihood = -15249.988
Iteration 1:   log likelihood =  -15217.95
Iteration 2:   log likelihood = -15213.126
Iteration 3:   log likelihood = -15213.046
Iteration 4:   log likelihood = -15213.046
```

```
Structural equation model                       Number of obs     =        932
Estimation method  = ml
Log likelihood     = -15213.046

 ( 1)  [anomia67]Alien67 = 1
 ( 2)  [anomia71]Alien71 = 1
 ( 3)  [educ66]SES = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
Alien67						
SES	-.5752228	.057961	-9.92	0.000	-.6888244	-.4616213
Alien71						
Alien67	.606954	.0512305	11.85	0.000	.5065439	.707364
SES	-.2270301	.0530773	-4.28	0.000	-.3310596	-.1230006
Measurement						
anomia67						
Alien67	1	(constrained)				
_cons	13.61	.1126143	120.85	0.000	13.38928	13.83072
pwless67						
Alien67	.9785952	.0619825	15.79	0.000	.8571117	1.100079
_cons	14.67	.1001814	146.43	0.000	14.47365	14.86635
anomia71						
Alien71	1	(constrained)				
_cons	14.13	.1159036	121.91	0.000	13.90283	14.35717
pwless71						
Alien71	.9217508	.0597225	15.43	0.000	.8046968	1.038805
_cons	14.9	.1034517	144.03	0.000	14.69724	15.10276
educ66						
SES	1	(constrained)				
_cons	10.9	.1014894	107.40	0.000	10.70108	11.09892
occstat66						
SES	5.22132	.425595	12.27	0.000	4.387169	6.055471
_cons	37.49	.6947112	53.96	0.000	36.12839	38.85161
var(e.ano~67)	4.728874	.456299			3.914024	5.713365
var(e.pwl~67)	2.563413	.4060733			1.879225	3.4967
var(e.ano~71)	4.396081	.5171156			3.490904	5.535966
var(e.pwl~71)	3.072085	.4360333			2.326049	4.057398
var(e.educ66)	2.803674	.5115854			1.960691	4.009091
var(e.occ~66)	264.5311	18.22483			231.1177	302.7751
var(e.Ali~67)	4.842059	.4622537			4.015771	5.838364
var(e.Ali~71)	4.084249	.4038995			3.364613	4.957802
var(SES)	6.796014	.6524866			5.630283	8.203105
cov(e.ano~67, e.anomia71)	1.622024	.3154267	5.14	0.000	1.003799	2.240249
cov(e.pwl~67, e.pwless71)	.3399961	.2627541	1.29	0.196	-.1749925	.8549847

```
LR test of model vs. saturated: chi2(4)   =      4.78, Prob > chi2 = 0.3111
```

Notes:

1. We find the covariance between `e.anomia67` and `e.anomia71` to be significant ($Z = 5.14$).
2. We find the covariance between `e.pwless67` and `e.pwless71` to be insignificant at the 5% level ($Z = 1.29$).
3. The model versus saturated χ^2 test indicates that the model is a good fit.

References

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Wheaton, B., B. Muthén, D. F. Alwin, and G. F. Summers. 1977. Assessing reliability and stability in panel models. In *Sociological Methodology 1977*, ed. D. R. Heise, 84–136. San Francisco: Jossey-Bass.

Also see

[SEM] **Example 32g** — Full structural equation model (generalized response)

[SEM] **estat mindices** — Modification indices

[SEM] **test** — Wald test of linear hypotheses

Title

Example 10 — MIMIC model

Description Remarks and examples Reference Also see

Description

To demonstrate a MIMIC model, we use the following summary statistics data:

```
. use https://www.stata-press.com/data/r16/sem_mimic1
(Multiple indicators and multiple causes)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_mimic1.dta
    obs:             432                  Multiple indicators and multip..
   vars:               5                  31 Oct 2018 08:29
                                          (_dta has notes)
-------------------------------------------------------------------------------
  variable name                  variable label
-------------------------------------------------------------------------------
  occpres                        occupational prestige, two-digit Dunca..
  income                         total family income in units of $2000,..
  s_occpres                      subjective occupational prestige
  s_income                       subjective income
  s_socstat                      subjective overall social status
-------------------------------------------------------------------------------
. notes
_dta:
  1.  Summary statistics data from Kluegel, J. R., R. Singleton, Jr., and C. E.
      Starnes, 1977, "Subjective class identification:  A multiple indicator
      approach", _American Sociological Review_, 42: 599-611.
  2.  Data is also analyzed in Bollen, K. A. 1989, _Structural Equations with
      Latent Variables_, New York: John Wiley & Sons, Inc.
  3.  The summary statistics represent 432 white adults included in the sample
      for the 1969 Gary Area Project for the Institute of Social Research at
      Indiana University.
  4.  The three subjective variables are measures of socioeconomic status based
      on an individuals perception of their own income, occupational prestige,
      and social status.
  5.  The income and occpres variables are objective measures of income and
      occupational prestige, respectively.
```

See *Structural models 10: MIMIC models* in [SEM] **Intro 5** for background.

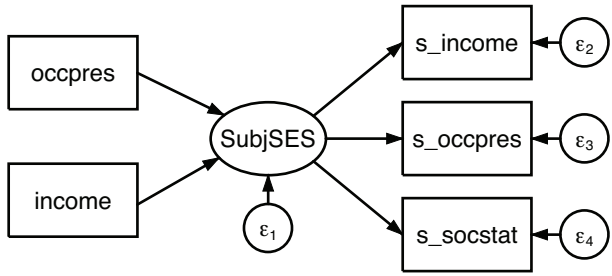
Remarks and examples

Remarks are presented under the following headings:

Fitting the MIMIC model
Fitting the MIMIC model with the Builder
Evaluating the residuals with estat residuals
Performing likelihood-ratio tests with lrtest

Fitting the MIMIC model

Based on the data referenced above, Bollen (1989, 397–399) fits a MIMIC model, the path diagram of which is



Bollen includes paths that he constrains and we do not show. Our model is nonetheless equivalent to the one he shows. In his textbook, Bollen illustrates various ways the same model can be written.

```
. sem (SubjSES -> s_income s_occpres s_socstat) (SubjSES <- income occpres)
Endogenous variables
Measurement:  s_income s_occpres s_socstat
Latent:       SubjSES
Exogenous variables
Observed:     income occpres
Fitting target model:
Iteration 0:   log likelihood = -4252.1834  (not concave)
Iteration 1:   log likelihood = -4022.8854  (not concave)
Iteration 2:   log likelihood = -3994.3867
Iteration 3:   log likelihood = -3978.7751  (not concave)
Iteration 4:   log likelihood = -3974.6636
Iteration 5:   log likelihood = -3972.0269
Iteration 6:   log likelihood = -3971.9238
Iteration 7:   log likelihood = -3971.9236
Structural equation model                       Number of obs     =        432
Estimation method  = ml
Log likelihood     = -3971.9236
 ( 1)  [s_income]SubjSES = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
SubjSES						
income	.082732	.0138498	5.97	0.000	.0555869	.1098772
occpres	.0046275	.0012464	3.71	0.000	.0021847	.0070704
Measurement						
s_income						
SubjSES	1	(constrained)				
_cons	.9612091	.0794151	12.10	0.000	.8055583	1.11686
s_occpres						
SubjSES	.7301352	.0832915	8.77	0.000	.5668869	.8933835
_cons	1.114563	.0656195	16.99	0.000	.9859512	1.243175
s_socstat						
SubjSES	.9405161	.0934855	10.06	0.000	.7572878	1.123744
_cons	1.002113	.0706576	14.18	0.000	.863627	1.1406
var(e.s_in~e)	.2087546	.0254098			.1644474	.2649996
var(e.s_oc~s)	.2811852	.0228914			.2397153	.3298291
var(e.s_so~t)	.1807129	.0218405			.1425987	.2290146
var(e.Subj~S)	.1860097	.0270476			.1398822	.2473481

```
LR test of model vs. saturated: chi2(4)   =     26.65, Prob > chi2 = 0.0000
```

Notes:

1. In this model, there are three observed variables that record the respondents' ideas of their perceived socioeconomic status (SES). One is the respondent's general idea of his or her SES (`s_socstat`); another is based on the respondent's income (`s_income`); and the last is based on the respondent's occupational prestige (`s_occpres`). Those three variables form the latent variable `SubjSES`.
2. The other two observed variables are the respondent's income (`income`) and occupation, the latter measured by the two-digit Duncan SEI scores for occupations (`occpres`). These two variables are treated as predictors of `SubjSES`.

3. In the model, item 1 is viewed as subjective and item 2 is viewed as objective.
4. All variables are statistically significant at the 5% level, but the model versus saturated test suggests that we are not modeling the covariances well.

Fitting the MIMIC model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_mimic1
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for subject socioeconomic status.

   Select the Add measurement component tool, , and then click near the center of the diagram.

   In the resulting dialog box,

   a. change the *Latent variable name* to `SubjSES`;

   b. select `s_income`, `s_occpress`, and `s_socstat` by using the *Measurement variables* control;

   c. select `Right` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move this component by clicking on any variable and dragging it.

4. Create the two variables for the formative indicators of socioeconomic status.

   a. Select the Add observed variable tool, , and then click in the diagram to add the new variable slightly above and to the left of `SubjSES`. After adding it, you can click inside the rectangle and move the variable if you wish.

   b. In the Contextual Toolbar, select `occpress` by using the *Variable* control.

   c. Continuing with the tool, click in the diagram to add another new variable below the `occpress` variable.

   d. In the Contextual Toolbar, select `income` by using the *Variable* control.

5. Create the paths for the formative indicators of socioeconomic status.

   a. Select the Add path tool, .

   b. Click in the lower-right quadrant of the `occpress` rectangle (it will highlight when you hover over it), and drag a path to the upper-left quadrant of the `SubjSES` oval (it will highlight when you can release to connect the path).

   c. Continuing with the tool, click in the upper-right quadrant of the `income` rectangle, and drag a path to the lower-left quadrant of the `SubjSES` oval.

6. Clean up the direction of the errors.

   The error on SubjSES is likely to have been created atop one of the existing paths. Choose the Select tool, , and then click in the SubjSES oval. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is where you want it.

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

Tip: See the tips of [SEM] **Example 9** to make creating paths somewhat easier than described above.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_mimic1
```

Evaluating the residuals with estat residuals

Remember that SEM fits covariances and means. Residuals in the SEM sense thus refer to covariances and means. If we are not fitting well, we can examine the residuals.

```
. estat residuals, normalized
Residuals of observed variables
  Mean residuals
```

	s_income	s_occpres	s_socstat	income	occpres
raw	0.000	0.000	0.000	0.000	0.000
normalized	0.000	0.000	0.000	0.000	0.000

  Covariance residuals

	s_income	s_occpres	s_socstat	income	occpres
s_income	0.000				
s_occpres	-0.009	0.000			
s_socstat	0.000	0.008	0.000		
income	0.101	-0.079	-0.053	0.000	
occpres	-0.856	1.482	0.049	0.000	0.000

  Normalized covariance residuals

	s_income	s_occpres	s_socstat	income	occpres
s_income	0.000				
s_occpres	-0.425	0.000			
s_socstat	0.008	0.401	0.000		
income	1.362	-1.137	-0.771	0.000	
occpres	-1.221	2.234	0.074	0.000	0.000

Notes:

1. The residuals can be partitioned into two subsets: mean residuals and covariance residuals.
2. The normalized option caused the normalized residuals to be displayed.
3. Concerning mean residuals, the raw residuals and the normalized residuals are shown on a separate line of the first table.

4. Concerning covariance residuals, the raw residuals and the normalized residuals are shown in separate tables.
5. Distinguish between normalized residuals and standardized residuals. Both are available from `estat residuals`; if we wanted standardized residuals, we would have specified the `standardized` option instead of or along with `normalized`.
6. Both normalized and standardized residuals attempt to adjust the residuals in the same way. The normalized residuals are always valid, but they do not follow a standard normal distribution. The standardized residuals do follow a standard normal distribution if they can be calculated; otherwise, they will equal missing values. When both can be calculated (equivalent to both being appropriate), the normalized residuals will be a little smaller than the standardized residuals.
7. The normalized covariance residuals between `income` and `s_income` and between `occpres` and `s_occpres` are large.

Performing likelihood-ratio tests with lrtest

Thus Bollen (1989, 397–399) suggests adding a direct path from the objective measures to the corresponding subjective measures. We are about to fit the model

```
(SubjSES -> s_income s_occpres s_socstat)   ///
(SubjSES <- income occpres)                 ///
(s_income <- income)                        ///  <- new
(s_occpres <- occpres)                      //   <- new
```

For no other reason than we want to demonstrate the likelihood-ratio test, we will then use `lrtest` rather than `test` to test the joint significance of the new paths. `lrtest` compares the likelihood values of two fitted models. Thus we will use `lrtest` to compare this new model with the one above. To do that, we must plan ahead and store in memory the currently fit model:

```
. estimates store mimic1
```

Alternatively, we could skip that and calculate the joint significance of the two new paths by using a Wald test and the `test` command.

In any case, having stored the current estimates under the name `mimic1`, we can now fit our new model:

```
. sem (SubjSES -> s_income s_occpres s_socstat)
>     (SubjSES <- income occpres)
>     (s_income <- income)
>     (s_occpres <- occpres)
Endogenous variables
Observed:     s_income s_occpres
Measurement:  s_socstat
Latent:       SubjSES
Exogenous variables
Observed:     income occpres
Fitting target model:
Iteration 0:   log likelihood = -4267.0974  (not concave)
Iteration 1:   log likelihood = -4022.6745  (not concave)
Iteration 2:   log likelihood = -3977.0648
Iteration 3:   log likelihood = -3962.9229
Iteration 4:   log likelihood = -3962.1604
Iteration 5:   log likelihood = -3960.8404
Iteration 6:   log likelihood = -3960.7133
Iteration 7:   log likelihood = -3960.7111
Iteration 8:   log likelihood = -3960.7111
```

```
Structural equation model                       Number of obs     =        432
Estimation method  = ml
Log likelihood     = -3960.7111
 ( 1)  [s_income]SubjSES = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
s_income						
SubjSES	1	(constrained)				
income	.0532425	.0142862	3.73	0.000	.025242	.081243
_cons	.8825314	.0781685	11.29	0.000	.7293239	1.035739
s_occpres						
SubjSES	.783781	.1011457	7.75	0.000	.585539	.982023
occpres	.0045201	.0013552	3.34	0.001	.0018641	.0071762
_cons	1.06586	.0696058	15.31	0.000	.9294353	1.202285
SubjSES						
income	.0538025	.0129158	4.17	0.000	.028488	.0791171
occpres	.0034324	.0011217	3.06	0.002	.0012339	.0056309
Measurement						
s_socstat						
SubjSES	1.195539	.1582735	7.55	0.000	.8853282	1.505749
_cons	1.07922	.078323	13.78	0.000	.9257097	1.23273
var(e.s_in~e)	.2292697	.0248905			.1853261	.2836329
var(e.s_oc~s)	.2773786	.0223972			.2367783	.3249407
var(e.s_so~t)	.1459009	.028228			.0998556	.2131785
var(e.Subj~S)	.1480275	.0278381			.1023918	.2140029

```
LR test of model vs. saturated: chi2(2)   =      4.22, Prob > chi2 = 0.1211
```

Now we can perform the likelihood-ratio test:

```
. lrtest mimic1 .
Likelihood-ratio test                                 LR chi2(2)  =     22.42
(Assumption: mimic1 nested in .)                      Prob > chi2 =    0.0000
```

Notes:

1. The syntax of `lrtest` is `lrtest` *modelname1 modelname2*. We specified the first model name as `mimic1`, the model we previously stored. We specified the second model name as period (`.`), meaning the model most recently fit. The order in which we specify the names is irrelevant.
2. We find the two added paths to be extremely significant.

Reference

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Also see

[SEM] **Example 36g** — MIMIC model (generalized response)

[SEM] **sem** — Structural equation model estimation command

[SEM] **estat residuals** — Display mean and covariance residuals

[SEM] **lrtest** — Likelihood-ratio test of linear hypothesis

Title

Example 11 — estat framework

Description Remarks and examples Also see

Description

To demonstrate `estat framework`, which displays results in Bentler–Weeks form, we continue where [SEM] **Example 10** left off:

```
. use https://www.stata-press.com/data/r16/sem_mimic1
. ssd describe
. notes
. sem (SubjSES -> s_income s_occpres s_socstat)   ///
      (SubjSES <- income occpres)
. estat residuals, normalized
. estimates store mimic1
. sem (SubjSES -> s_income s_occpres s_socstat)   ///
      (SubjSES <- income occpres)                 ///
      (s_income <- income)                        ///
      (s_occpres <- occpres)
. lrtest mimic1 .
```

See *Structural models 10: MIMIC models* in [SEM] **Intro 5** for background.

Remarks and examples

If you prefer to see SEM results reported in Bentler–Weeks form, type `estat framework` after estimating with `sem`. Many people find Bentler–Weeks form helpful in understanding how the model is fit.

[SEM] **Example 10** ended by fitting

```
. sem (SubjSES   -> s_income s_occpres s_socstat)   ///
      (SubjSES   <- income occpres)                 ///
      (s_income  <- income)                         ///
      (s_occpres <- occpres)
```

In Bentler–Weeks form, the output appears as

```
. estat framework, fitted
Endogenous variables on endogenous variables
```

Beta	observed s_income	s_occpres	s_socstat	latent SubjSES
observed				
s_income	0	0	0	1
s_occpres	0	0	0	.783781
s_socstat	0	0	0	1.195539
latent				
SubjSES	0	0	0	0

Exogenous variables on endogenous variables

Gamma	observed income	occpres
observed		
s_income	.0532425	0
s_occpres	0	.0045201
s_socstat	0	0
latent		
SubjSES	.0538025	.0034324

Covariances of error variables

Psi	observed e.s_inc~e	e.s_occ~s	e.s_soc~t	latent e.SubjSES
observed				
e.s_income	.2292697			
e.s_occpres	0	.2773786		
e.s_socstat	0	0	.1459009	
latent				
e.SubjSES	0	0	0	.1480275

Intercepts of endogenous variables

alpha	observed s_income	s_occpres	s_socstat	latent SubjSES
_cons	.8825314	1.06586	1.07922	0

Covariances of exogenous variables

Phi	observed income	occpres
observed		
income	4.820021	
occpres	13.62431	451.6628

Means of exogenous variables

kappa	observed income	occpres
mean	5.04	36.698

```
Fitted covariances of observed and latent variables
```

Sigma	observed s_income	s_occpres	s_socstat	latent SubjSES	observed income
observed					
s_income	.4478609				
s_occpres	.1614446	.4086519			
s_socstat	.225515	.1738222	.392219		
latent					
SubjSES	.1886304	.1453924	.2060311	.1723333	
observed					
income	.5627232	.3014937	.3659463	.3060932	4.820021
occpres	3.008694	3.831184	2.729776	2.283302	13.62431

Sigma	observed occpres
observed	
occpres	451.6628

```
Fitted means of observed and latent variables
```

mu	observed s_income	s_occpres	s_socstat	latent SubjSES	observed income
mu	1.548	1.543	1.554	.3971264	5.04

mu	observed occpres
mu	36.698

Notes:

1. Bentler–Weeks form is a vector and matrix notation for the estimated parameters of the model. The matrices are known as $\boldsymbol{\beta}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Psi}$, $\boldsymbol{\alpha}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\kappa}$. Those Greek names are spelled out in the labels, along with a header stating what each contains.
2. We specified `estat framework` option `fitted`. That caused `estat framework` to list one more matrix and one more vector at the end: $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$. These two results are especially interesting to those wishing to see the ingredients of the residuals reported by `estat residuals`.
3. One of the more useful results reported by `estat framework, fitted` is the $\boldsymbol{\Sigma}$ matrix, which reports all estimated covariances in a readable format and includes the model-implied covariances that do not appear in `sem`'s ordinary output.
4. `estat framework` also allows the `standardized` option if you want standardized output.

Also see

[SEM] **Example 10** — MIMIC model

[SEM] **estat framework** — Display estimation results in modeling framework

Title

Example 12 — Seemingly unrelated regression

Description Remarks and examples Also see

Description

`sem` can be used to estimate seemingly unrelated regression. We will use `auto.dta`, which surely needs no introduction:

```
. sysuse auto
(1978 Automobile Data)
```

See *Structural models 11: Seemingly unrelated regression (SUR)* in [SEM] **Intro 5**.

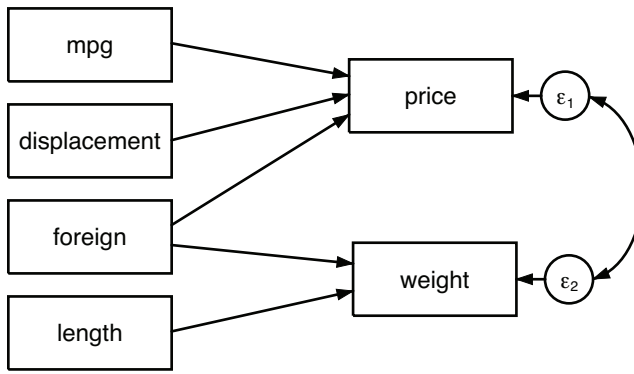
Remarks and examples

Remarks are presented under the following headings:

Fitting the seemingly unrelated regression model
Fitting the model with the Builder

Fitting the seemingly unrelated regression model

We fit the following model:




```
. sem  (price  <- foreign mpg displacement)
>      (weight <- foreign length),
>                                cov(e.price*e.weight)
Endogenous variables
Observed:  price weight
Exogenous variables
Observed:  foreign mpg displacement length
Fitting target model:
Iteration 0:   log likelihood = -2150.9983
Iteration 1:   log likelihood = -2138.5739
Iteration 2:   log likelihood = -2133.3461
Iteration 3:   log likelihood = -2133.1979
Iteration 4:   log likelihood = -2133.1956
Iteration 5:   log likelihood = -2133.1956
```

```
Structural equation model                       Number of obs     =         74
Estimation method  = ml
Log likelihood     = -2133.1956
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
price						
foreign	2940.929	724.7311	4.06	0.000	1520.482	4361.376
mpg	-105.0163	57.93461	-1.81	0.070	-218.566	8.53347
displace~t	17.22083	4.5941	3.75	0.000	8.216558	26.2251
_cons	4129.866	1984.253	2.08	0.037	240.8022	8018.931
weight						
foreign	-153.2515	76.21732	-2.01	0.044	-302.6347	-3.868275
length	30.73507	1.584743	19.39	0.000	27.62903	33.84111
_cons	-2711.096	312.6813	-8.67	0.000	-3323.94	-2098.252
var(e.price)	4732491	801783.1			3395302	6596312
var(e.weight)	60253.09	9933.316			43616.45	83235.44
cov(e.price, e.weight)	209268	73909.54	2.83	0.005	64407.92	354128

```
LR test of model vs. saturated: chi2(3)   =     38.86, Prob > chi2 = 0.0000
```

Notes:

1. Point estimates are the same as reported by

   ```
   . sureg (price foreign mpg displ) (weight foreign length), isure
   ```

   `sureg`'s `isure` option is required to make `sureg` iterate to the maximum likelihood estimate.

2. If you wish to compare the estimated variances and covariances after estimation by `sureg`, type

   ```
   . matrix list e(Sigma)
   ```

   `sureg` does not estimate standard errors on variances and covariances.

3. Standard errors will be different between `sem` and `sureg`. In this case, there is no reason to prefer one set of standard errors over the other, and standard errors are asymptotically equivalent. This is a case of exogenous variables only on the right-hand side. When the model being fit is recursive, standard errors produced by `sem` are better than those from `sureg`, both asymptotically and in finite samples.

4. One reason you might want to use `sem` is that `sem` will provide robust standard errors whereas `sureg` does not.

5. Multivariate regression can be viewed as seemingly unrelated regression. You just need to specify the same regressors for each equation. In that case, standard errors reported by `sem` will be the same as those reported by `mvreg` if one applies the multiplicative $\sqrt{(N-p-1)/N}$ degree-of-freedom adjustment.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/auto
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Change the size of the observed variables' rectangles.

   From the SEM Builder menu, select **Settings > Variables > All observed...**.

   In the resulting dialog box, change the first size to `.85` and click on **OK**.

4. Create the four independent variables.

   Select the Add observed variables set tool, ▫▫▫, and then click in the diagram about one-fourth of the way in from the left and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the four variables in this order: `mpg`, `displacement`, `foreign`, and `length`;

   c. select `Vertical` in the *Orientation* control;

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the two dependent variables.

   Select the Add observed variables set tool, ▫▫▫, and then click about two-thirds of the way in from the left and vertically aligned with the top of the `length` rectangle.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the variables `price` and `weight`;

   c. select `Vertical` in the *Orientation* control;

   d. select the **Distances** tab;

   e. select `.5` (inch) from the *Distance between variables* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

6. Create paths from the independent variables to the dependent variables.

   a. Select the Add path tool, →.

   b. Click in the right side of the `mpg` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `price` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the → tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable:

```
displacement -> price
foreign -> price
foreign -> weight
length -> weight
```

7. Correlate the error terms.

   a. Select the Add covariance tool, ⌒.

   b. Click in the ϵ_1 circle (it will highlight when you hover over it), and drag a covariance to the ϵ_2 circle (it will highlight when you can release to connect the covariance).

8. Clean up.

   If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint. Similarly, you can change where the covariance connects to the error terms by clicking on the covariance and dragging the endpoint. You can also change the bow of the covariance by clicking on the covariance and dragging the control point that extends from one end of the selected covariance.

9. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_sureg
```

Also see

[SEM] **Example 13** — Equation-level Wald test

[SEM] **sem** — Structural equation model estimation command

Title

Example 13 — Equation-level Wald test

Description Remarks and examples Also see

Description

We demonstrate `estat eqtest`. See [SEM] **Intro 7** and [SEM] **estat eqtest**.

This example picks up where [SEM] **Example 12** left off:

```
. use https://www.stata-press.com/data/r16/auto
. sem  (price  <- foreign mpg displacement)              ///
       (weight <- foreign length),                       ///
                      cov(e.price*e.weight)
```

Remarks and examples

We have fit a two-equation model with equations for endogenous variables `price` and `weight`. There happen to be two equations, the model happens to be a seemingly unrelated regression, and the endogenous variables happen to be observed, but none of that is important right now.

`estat eqtest` displays equation-by-equation Wald tests that all coefficients excluding the intercepts are 0.

```
. estat eqtest
Wald tests for equations
```

	chi2	df	p
observed			
price	36.43	3	0.0000
weight	633.34	2	0.0000

Note:

1. The null hypothesis for this test is that the coefficients other than the intercepts are 0. We can reject that null hypothesis for each equation.

Also see

[SEM] **Example 12** — Seemingly unrelated regression

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **estat eqtest** — Equation-level tests that all coefficients are zero

Title

Example 14 — Predicted values

Description Remarks and examples Also see

Description

We demonstrate the use of `predict`. See [SEM] **Intro 7** and [SEM] **predict after sem**.

This example picks up where the first part of [SEM] **Example 1** left off:

```
. use https://www.stata-press.com/data/r16/sem_1fmm
. sem (x1 x2 x3 x4 <- X)
```

Remarks and examples

`predict` can create new variables containing predicted values of 1) observed endogenous variables, 2) latent variables, whether endogenous or exogenous, and 3) latent endogenous variables. In the case of latent variables, item 2 corresponds to the factor score and item 3 is the linear prediction.

Below we demonstrate 1 and 2:

```
. predict x1hat x2hat, xb(x1 x2)
. predict Xhat, latent(X)
```

You specify options on `predict` to specify what you want predicted and how. Because of the differing options, the two commands could not have been combined into one command.

Our dataset now contains three new variables. Below we compare the three variables with the original `x1` and `x2` by using first `summarize` and then `correlate`:

```
. summarize x1 x1hat x2 x2hat Xhat
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	500	99.518	14.35402	60	137
x1hat	500	99.518	9.363112	71.45533	126.7325
x2	500	99.954	14.1939	52	140
x2hat	500	99.954	9.674426	70.95827	128.0733
Xhat	500	1.03e-08	9.363112	-28.06267	27.21449

Notes:

1. Means of `x1hat` and `x1` are identical; means of `x2hat` and `x2` are identical.
2. The standard deviation of `x1hat` is less than that of `x1`; the standard deviation of `x2hat` is less than that of `x2`. Some of the variation in `x1` and `x2` is not explained by the model.
3. Standard deviations of `x1hat` and `Xhat` are equal. This is because in

$$x_1 = b_0 + b_1 X + e_1$$

coefficient b_1 was constrained to be equal to 1 because of the anchoring normalization constraint; see *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.

The mean of `Xhat` in the model above is 1.03e–08 rather than 0. Had we typed

```
. predict double Xhat, latent(X)
```

the mean would have been −1.17e–16.

```
. correlate x1 x1hat x2 x2hat Xhat
(obs=500)
```

	x1	x1hat	x2	x2hat	Xhat
x1	1.0000				
x1hat	0.6705	1.0000			
x2	0.4537	0.7007	1.0000		
x2hat	0.6705	1.0000	0.7007	1.0000	
Xhat	0.6705	1.0000	0.7007	1.0000	1.0000

Notes:

1. Both `x1hat` and `x2hat` correlate 1 with `Xhat`. That is because both are linear functions of `Xhat` alone.
2. That `x1hat` and `x2hat` correlate 1 is implied by item 1, directly above.
3. That `Xhat`, `x1hat`, and `x2hat` all have the same correlation with `x1` and with `x2` is also implied by item 1, directly above.

Also see

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **predict after sem** — Factor scores, linear predictions, etc.

Title

Example 15 — Higher-order CFA

Description Remarks and examples Reference Also see

Description

`sem` can be used to estimate higher-order confirmatory factor analysis models.

```
. use https://www.stata-press.com/data/r16/sem_hcfa1
(Higher-order CFA)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_hcfa1.dta
    obs:             251                  Higher-order CFA
   vars:              16                  25 May 2018 11:26
                                          (_dta has notes)
```

variable name	variable label
phyab1	Physical ability 1
phyab2	Physical ability 2
phyab3	Physical ability 3
phyab4	Physical ability 4
appear1	Appearance 1
appear2	Appearance 2
appear3	Appearance 3
appear4	Appearance 4
peerrel1	Relationship w/ peers 1
peerrel2	Relationship w/ peers 2
peerrel3	Relationship w/ peers 3
peerrel4	Relationship w/ peers 4
parrel1	Relationship w/ parent 1
parrel2	Relationship w/ parent 2
parrel3	Relationship w/ parent 3
parrel4	Relationship w/ parent 4

```
. notes
_dta:
  1.  Summary statistics data from Marsh, H. W. and Hocevar, D., 1985,
      "Application of confirmatory factor analysis to the study of
      self-concept: First- and higher order factor models and their invariance
      across groups", _Psychological Bulletin_, 97: 562-582.
  2.  Summary statistics based on 251 students from Sydney, Australia in Grade
      5.
  3.  Data collected using the Self-Description Questionnaire and includes
      sixteen subscales designed to measure nonacademic traits: four intended
      to measure physical ability, four intended to measure physical
      appearance, four intended to measure relations with peers, and four
      intended to measure relations with parents.
```

See *Higher-order CFA models* in [SEM] **Intro 5** for background.

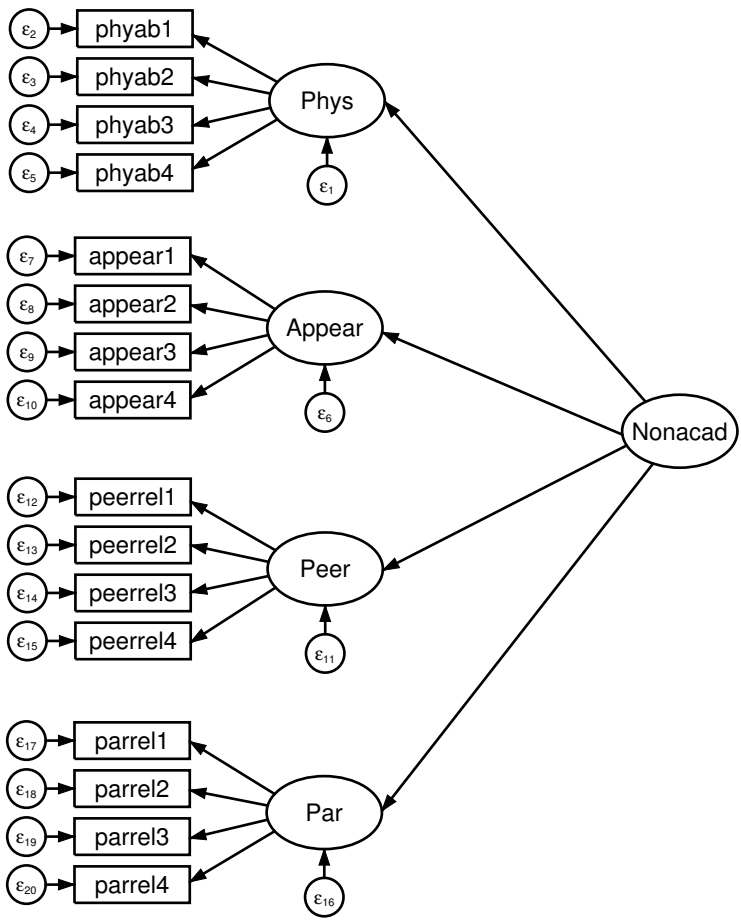
Remarks and examples

Remarks are presented under the following headings:

Fitting the model
Fitting the model with the Builder

Fitting the model

We fit the following model:



```
. sem (Phys -> phyab1 phyab2 phyab3 phyab4)
>     (Appear -> appear1 appear2 appear3 appear4)
>     (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)
>     (Par -> parrel1 parrel2 parrel3 parrel4)
>     (Nonacad -> Phys Appear Peer Par)

Endogenous variables

Measurement:  phyab1 phyab2 phyab3 phyab4 appear1 appear2 appear3 appear4
              peerrel1 peerrel2 peerrel3 peerrel4 parrel1 parrel2 parrel3
              parrel4
Latent:       Phys Appear Peer Par

Exogenous variables

Latent:       Nonacad

Fitting target model:

Iteration 0:   log likelihood = -7686.6699  (not concave)
Iteration 1:   log likelihood = -7643.7387  (not concave)
Iteration 2:   log likelihood = -7616.2966  (not concave)
Iteration 3:   log likelihood = -7597.6133
Iteration 4:   log likelihood = -7588.9515
Iteration 5:   log likelihood = -7585.3162
Iteration 6:   log likelihood = -7584.8125
Iteration 7:   log likelihood = -7584.7885
Iteration 8:   log likelihood = -7584.7881

Structural equation model                       Number of obs     =        251
Estimation method  = ml
Log likelihood     = -7584.7881

 ( 1)  [phyab1]Phys = 1
 ( 2)  [appear1]Appear = 1
 ( 3)  [peerrel1]Peer = 1
 ( 4)  [parrel1]Par = 1
 ( 5)  [Phys]Nonacad = 1
```

	OIM					
	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
Phys						
Nonacad	1	(constrained)				
Appear						
Nonacad	2.202491	.3975476	5.54	0.000	1.423312	2.98167
Peer						
Nonacad	1.448035	.2921383	4.96	0.000	.8754549	2.020616
Par						
Nonacad	.569956	.1382741	4.12	0.000	.2989437	.8409683
Measurement						
phyab1						
Phys	1	(constrained)				
_cons	8.2	.1159065	70.75	0.000	7.972827	8.427173
phyab2						
Phys	.9332477	.1285726	7.26	0.000	.68125	1.185245
_cons	8.23	.122207	67.34	0.000	7.990479	8.469521

phyab3						
Phys	1.529936	.1573845	9.72	0.000	1.221468	1.838404
_cons	8.17	.1303953	62.66	0.000	7.91443	8.42557
phyab4						
Phys	1.325641	.1338053	9.91	0.000	1.063387	1.587894
_cons	8.56	.1146471	74.66	0.000	8.335296	8.784704
appear1						
Appear	1	(constrained)				
_cons	7.41	.1474041	50.27	0.000	7.121093	7.698907
appear2						
Appear	1.0719	.0821893	13.04	0.000	.9108121	1.232988
_cons	7	.1644123	42.58	0.000	6.677758	7.322242
appear3						
Appear	1.035198	.0893075	11.59	0.000	.8601581	1.210237
_cons	7.17	.1562231	45.90	0.000	6.863808	7.476192
appear4						
Appear	.9424492	.0860848	10.95	0.000	.7737262	1.111172
_cons	7.4	.1474041	50.20	0.000	7.111093	7.688907
peerrel1						
Peer	1	(constrained)				
_cons	8.81	.1077186	81.79	0.000	8.598875	9.021125
peerrel2						
Peer	1.214379	.1556051	7.80	0.000	.9093989	1.51936
_cons	7.94	.1215769	65.31	0.000	7.701714	8.178286
peerrel3						
Peer	1.667829	.190761	8.74	0.000	1.293944	2.041714
_cons	7.52	.1373248	54.76	0.000	7.250848	7.789152
peerrel4						
Peer	1.363627	.159982	8.52	0.000	1.050068	1.677186
_cons	8.29	.1222066	67.84	0.000	8.050479	8.529521
parrel1						
Par	1	(constrained)				
_cons	9.35	.0825215	113.30	0.000	9.188261	9.511739
parrel2						
Par	1.159754	.184581	6.28	0.000	.7979822	1.521527
_cons	9.13	.0988998	92.32	0.000	8.93616	9.32384
parrel3						
Par	2.035143	.2623826	7.76	0.000	1.520882	2.549403
_cons	8.67	.1114983	77.76	0.000	8.451467	8.888533
parrel4						
Par	1.651802	.2116151	7.81	0.000	1.237044	2.06656
_cons	9	.0926003	97.19	0.000	8.818507	9.181493
var(e.phyab1)	2.07466	.2075636			1.705244	2.524103
var(e.phyab2)	2.618638	.252693			2.167386	3.163841
var(e.phyab3)	1.231013	.2062531			.8864333	1.70954
var(e.phyab4)	1.019261	.1600644			.7492262	1.386621
var(e.appe~1)	1.986955	.2711164			1.520699	2.596169

```
var(e.appe~2)   2.801673   .3526427                      2.189162    3.585561
var(e.appe~3)    2.41072    .300262                      1.888545    3.077276
var(e.appe~4)   2.374508   .2872554                      1.873267    3.009868
var(e.peer~1)   1.866632    .18965                       1.529595    2.277933
var(e.peer~2)   2.167766   .2288099                      1.762654    2.665984
var(e.peer~3)   1.824346   .2516762                      1.392131    2.390749
var(e.peer~4)   1.803918    .212599                      1.431856    2.272659
var(e.parr~1)   1.214141   .1195921                      1.000982    1.472692
var(e.parr~2)   1.789125   .1748043                      1.477322    2.166738
var(e.parr~3)   1.069717   .1767086                      .7738511    1.478702
var(e.parr~4)   .8013735    .121231                      .5957527    1.077963
  var(e.Phys)    .911538   .1933432                      .6014913    1.381403
var(e.Appear)    1.59518   .3704939                      1.011838    2.514828
  var(e.Peer)   .2368108   .1193956                      .0881539    .6361528
   var(e.Par)   .3697854   .0915049                      .2276755     .600597
 var(Nonacad)   .3858166   .1237638                      .2057449    .7234903
------------------------------------------------------------------------------
LR test of model vs. saturated: chi2(100) =    219.48, Prob > chi2 = 0.0000
```

Notes:

1. The idea behind this model is that physical ability, appearance, and relationships with peers and parents may be determined by a latent variable containing nonacademic traits. This model was suggested by Bollen (1989, 315).
2. `sem` automatically provided normalization constraints for the first-order factors `Phys`, `Appear`, `Peer`, and `Par`. Their path coefficients were set to 1.
3. `sem` automatically provided a normalization constraint for the second-order factor `Nonacad`. Its path coefficient was set to 1.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_hcfa1
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Enlarge the size of the canvas to accommodate the length of the diagram.

   Click on the **Adjust canvas size** button, , in the Standard Toolbar, change the second size to `7` (inches), and then click on **OK**.

4. Change the size of the observed variables' rectangles.

   a. In the SEM Builder menu, select **Settings > Variables > All observed...**.

   b. In the resulting dialog box, change the second size to `.25` and click on **OK**.

5. Create the measurement component for physical ability.

   Select the Add measurement component tool, . Then using the darker one-inch grid lines in the background as a guide, click in the diagram about two inches in from the left and one inch down from the top.

In the resulting dialog box,

a. change the *Latent variable name* to `Phys`;

b. select `phyab1`, `phyab2`, `phyab3`, and `phyab4` by using the *Measurement variables* control;

c. select `Left` in the *Measurement direction* control;

d. click on **OK**.

If you wish, move the component by clicking on any variable and dragging it.

6. Create the remaining first-order measurement components.

   a. Repeat the process from item 5, but place the measurement component on the grid line two inches in from the left and about two and one-half inches down from the top. Label the latent variable `Appear`, and select measurement variables `appear1`, `appear2`, `appear3`, and `appear4`.

   b. Repeat the process from item 5, but place the measurement component on the grid line two inches in from the left and about four inches down from the top. Label the latent variable `Peer`, and select measurement variables `peerrel1`, `peerrel2`, `peerrel3`, and `peerrel4`.

   c. Repeat the process from item 5, but place the measurement component on the grid line two inches in from the left and about five and one-half inches down from the top. Label the latent variable `Par`, and select measurement variables `parrel1`, `parrel2`, `parrel3`, and `parrel4`.

7. Create the second-order latent variable.

   a. Select the Add latent variable tool, ⬭, and then click in the diagram about two inches in from the right and vertically centered between the `Appear` and `Peer` latent variables.

   b. In the Contextual Toolbar, type `Nonacad` in the *Name* control and press *Enter*.

8. Create paths from `Nonacad` to each of the first-order latent variables.

   a. Select the Add path tool, →.

   b. Click in the upper-left quadrant of the `Nonacad` oval (it will highlight when you hover over it), and drag a path to the lower-left quadrant of the `Phys` oval (it will highlight when you can release to connect the path).

   c. Continuing with the → tool, create the following paths by clicking first on the left side of the `Nonacad` variable and dragging to the right side of the first-order latent variable.

```
Nonacad -> Appear
Nonacad -> Peer
Nonacad -> Par
```

9. Clean up the direction of the errors.

   We want the errors for each of the latent variables to be below the latent variable. The errors for `Phys`, `Appear`, and `Peer` are likely to have been created in other directions.

   a. Choose the Select tool, .

   b. Click in the `Phys` oval.

   c. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is below the latent variable.

   Repeat this for all errors on latent variables that are not below the latent variable.

10. Clean up the paths.

    If you do not like where a path has been connected to its variable, use the Select tool, , to click on the path, and then simply click on where it connects to an oval and drag the endpoint.

11. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

Tip: See the tips of [SEM] **Example 9** to make creating paths somewhat easier than described above.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_hcfa1
```

Reference

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Also see

[SEM] **sem** — Structural equation model estimation command

Title

Example 16 — Correlation

Description Remarks and examples Also see

Description

`sem` can be used to produce correlations or covariances between exogenous variables. The advantages of using `sem` over Stata's `correlate` command are that you can perform statistical tests on the results and that you can handle missing values in a more elegant way.

To demonstrate these features, we use

```
. use https://www.stata-press.com/data/r16/census13
(1980 Census data by state)
. describe
Contains data from https://www.stata-press.com/data/r16/census13.dta
  obs:            50                          1980 Census data by state
 vars:             9                          2 Dec 2018 14:01
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
state           byte    %13.0g     state1     State
brate           int     %10.0g                Birth rate
pop             long    %12.0gc               Population
medage          float   %9.2f                 Median age
division        byte    %8.0g      division   Census division
region          byte    %-8.0g     cenreg     Census region
mrgrate         float   %9.0g                 Marriage rate
dvcrate         float   %9.0g                 Divorce rate
medagesq        float   %9.0g
-------------------------------------------------------------------------------
Sorted by:
```

See *Correlations* in [SEM] **Intro 5** for background.

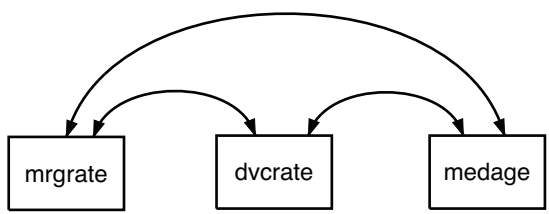
Remarks and examples

Remarks are presented under the following headings:

Using sem to obtain correlation matrices
Fitting the model with the Builder
Testing correlations with estat stdize and test

Using sem to obtain correlation matrices

We fit the following model:



This model does nothing more than estimate the covariances (correlations), something we could obtain from the `correlate` command by typing

```
. correlate mrgrate dvcrate medage
(obs=50)
```

	mrgrate	dvcrate	medage
mrgrate	1.0000		
dvcrate	0.7700	1.0000	
medage	-0.0177	-0.2229	1.0000

```
. correlate mrgrate dvcrate medage, covariance
(obs=50)
```

	mrgrate	dvcrate	medage
mrgrate	.000662		
dvcrate	.000063	1.0e-05	
medage	-.000769	-.001191	2.86775

As explained in *Correlations* in [SEM] **Intro 5**, to see results presented as correlations rather than as covariances, we specify `sem`'s `standardized` option:

```
. sem ( <- mrgrate dvcrate medage), standardized
Exogenous variables
Observed:  mrgrate dvcrate medage
Fitting target model:
Iteration 0:   log likelihood =  258.58985
Iteration 1:   log likelihood =  258.58985
Structural equation model                       Number of obs     =         50
Estimation method  = ml
Log likelihood     =  258.58985
```

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
mean(mrgrate)	.7332509	.1593002	4.60	0.000	.4210282	1.045474
mean(dvcrate)	2.553791	.291922	8.75	0.000	1.981634	3.125947
mean(medage)	17.62083	1.767749	9.97	0.000	14.15611	21.08556
var(mrgrate)	1	.			.	.
var(dvcrate)	1	.			.	.
var(medage)	1	.			.	.
cov(mrgrate, dvcrate)	.7699637	.0575805	13.37	0.000	.6571079	.8828195
cov(mrgrate, medage)	-.0176541	.1413773	-0.12	0.901	-.2947485	.2594403
cov(dvcrate, medage)	-.222932	.1343929	-1.66	0.097	-.4863373	.0404732

```
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

Note:

1. The correlations reported are

	sem	correlate
mrgrate and dvcrate	0.7699637	0.7700
mrgrate and medage	−0.0176541	−0.0177
dvcrate and medage	−0.222932	−0.2229

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/census13
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the set of observed variables.

   Select the Add observed variables set tool, ▫▫▫, and then click in the diagram about halfway down from the top and a quarter of the way in from the left.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the variables in this order: `mrgrate`, `dvcrate`, and `medage`;

   c. select `Horizontal` in the *Orientation* control;

   d. select the **Distances** tab;

   e. select `.5` (inch) in the *Distance between variables* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

   Be sure you select the observed variables in the order indicated above; otherwise, the instructions below for creating covariances will not be correct.

4. Correlate each pair of variables.

   a. Select the Add covariance tool, ⌒.

   b. Click in the top of the `mrgrate` rectangle, slightly to the right of the center (it will highlight when you hover over it), and drag a path to the top of the `dvcrate` rectangle, slightly to the left of the center (it will highlight when you can release to connect the covariance).

   c. Click in the top of the `dvcrate` rectangle, slightly to the right of the center, and drag a path to the top of the `medage` rectangle, slightly to the left of the center.

   d. Click in the top of the `mrgrate` rectangle, slightly to the left of the center, and drag a path to the top of the `medage` rectangle, slightly to the right of the center.

5. Clean up.

   If you do not like where a covariance has been connected to its variable, use the Select tool, , to click on the covariance, and then simply click on where it connects to an oval and drag the endpoint. You can also change the bow of the covariance by dragging the control point that extends from one end of the selected covariance.

6. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

7. Show standardized estimates.

   From the SEM Builder menu, select **View > Standardized estimates**.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_corr
```

Testing correlations with estat stdize and test

We can test whether the correlations between median age and marriage and divorce rates are equal with `test` by typing

```
. estat stdize:  ///
        test _b[/cov(medage,mrgrate)] = _b[/cov(medage,dvcrate)]
```

We must prefix `test` with `estat stdize` because otherwise we would be testing equality of covariances; see *Displaying other results, statistics, and tests (sem and gsem)* in [SEM] **Intro 7** and see [SEM] **estat stdize**.

That we refer to the two correlations (covariances) by typing `_b[/cov(medage,mrgrate)]` and `_b[/cov(medage,dvcrate)]` is something nobody remembers and that we remind ourselves of by redisplaying `sem` results with the `coeflegend` option:

```
. sem, coeflegend
Structural equation model                       Number of obs     =         50
Estimation method  = ml
Log likelihood     =  258.58985

------------------------------------------------------------------------------
             |      Coef.  Legend
-------------+----------------------------------------------------------------
mean(mrgrate)|   .0186789  _b[/mean(mrgrate)]
mean(dvcrate)|   .0079769  _b[/mean(dvcrate)]
 mean(medage)|      29.54  _b[/mean(medage)]
-------------+----------------------------------------------------------------
 var(mrgrate)|   .0006489  _b[/var(mrgrate)]
 var(dvcrate)|  9.76e-06  _b[/var(dvcrate)]
  var(medage)|     2.8104  _b[/var(medage)]
-------------+----------------------------------------------------------------
 cov(mrgrate,|
     dvcrate)|   .0000613  _b[/cov(mrgrate,dvcrate)]
 cov(mrgrate,|
      medage)|  -.0007539  _b[/cov(mrgrate,medage)]
 cov(dvcrate,|
      medage)|  -.0011674  _b[/cov(dvcrate,medage)]
------------------------------------------------------------------------------
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

We can now obtain the test:

```
. estat stdize:
>         test _b[/cov(medage,mrgrate)] = _b[/cov(medage,dvcrate)]
 ( 1)  [/]cov(mrgrate,medage) - [/]cov(dvcrate,medage) = 0

           chi2(  1) =    4.78
         Prob > chi2 =    0.0288
```

Note:

1. We can reject the test at the 5% level.

Also see

[SEM] **estat stdize** — Test standardized parameters

[SEM] **test** — Wald test of linear hypotheses

[R] **correlate** — Correlations of variables

Title

Example 17 — Correlated uniqueness model

Description Remarks and examples Reference Also see

Description

To demonstrate a correlated uniqueness model, we use the following summary statistics data:

```
. use https://www.stata-press.com/data/r16/sem_cu1
(Correlated uniqueness)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_cu1.dta
    obs:             500                  Correlated uniqueness
   vars:               9                  18 Jan 2019 09:34
                                          (_dta has notes)
  ------------------------------------------------------------------------------
  variable name                  variable label
  ------------------------------------------------------------------------------
  par_i                          self-report inventory for paranoid
  szt_i                          self-report inventory for schizotypal
  szd_i                          self-report inventory for schizoid
  par_c                          clinical interview rating for paranoid
  szt_c                          clinical interview rating for schizoty..
  szd_c                          clinical interview rating for schizoid
  par_o                          observer rating for paranoid
  szt_o                          observer rating for schizotypal
  szd_o                          observer rating for schizoid
  ------------------------------------------------------------------------------
. notes
_dta:
  1.  Summary statistic data for Multitrait-Multimethod matrix (a specific kind
      of correlation matrix) and standard deviations from Brown, Timothy A.,
      2015, _Confirmatory Factor Analysis for Applied Research, 2nd ed._, New
      York, NY: The Guilford Press.
  2.  Summary statistics represent a sample of 500 patients who were evaluated
      for three personality disorders using three different methods.
  3.  The personality disorders include paranoid, schizotypal, and schizoid.
  4.  The methods of evaluation include a self-report inventory, ratings from a
      clinical interview, and observational ratings.
```

See *Correlated uniqueness model* in [SEM] **Intro 5** for background.

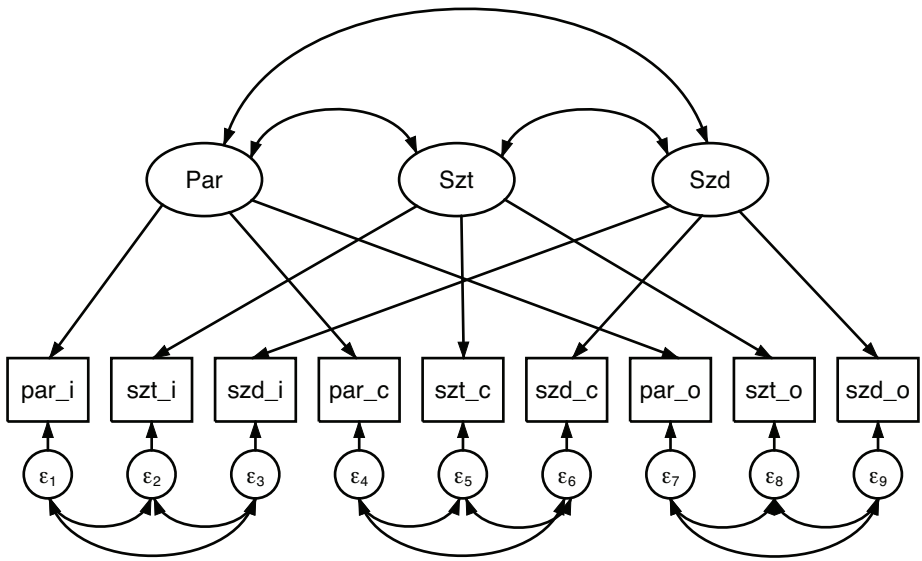
Remarks and examples

Remarks are presented under the following headings:

Fitting the model
Fitting the model with the Builder

Fitting the model

We fit the following model:




```
. sem (Par -> par_i par_c par_o)
>     (Szt -> szt_i szt_c szt_o)
>     (Szd -> szd_i szd_c szd_o),
>                covstr(e.par_i e.szt_i e.szd_i, unstructured)
>                covstr(e.par_c e.szt_c e.szd_c, unstructured)
>                covstr(e.par_o e.szt_o e.szd_o, unstructured)
>                standardized
Endogenous variables
Measurement:  par_i par_c par_o szt_i szt_c szt_o szd_i szd_c szd_o
Exogenous variables
Latent:       Par Szt Szd
Fitting target model:
Iteration 0:   log likelihood =  -10210.31  (not concave)
Iteration 1:   log likelihood = -10040.188  (not concave)
Iteration 2:   log likelihood = -9971.4015
Iteration 3:   log likelihood = -9918.0037
Iteration 4:   log likelihood = -9883.6368
Iteration 5:   log likelihood = -9880.0242
Iteration 6:   log likelihood = -9879.9961
Iteration 7:   log likelihood = -9879.9961
```

```
Structural equation model                       Number of obs     =        500
Estimation method  = ml
Log likelihood     = -9879.9961
 ( 1)  [par_i]Par = 1
 ( 2)  [szt_i]Szt = 1
 ( 3)  [szd_i]Szd = 1
```

Standardized	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
par_i <-						
Par	.7119709	.0261858	27.19	0.000	.6606476	.7632941
par_c <-						
Par	.8410183	.0242205	34.72	0.000	.7935469	.8884897
par_o <-						
Par	.7876062	.0237685	33.14	0.000	.7410209	.8341916
szt_i <-						
Szt	.7880887	.0202704	38.88	0.000	.7483594	.8278179
szt_c <-						
Szt	.7675732	.0244004	31.46	0.000	.7197493	.8153972
szt_o <-						
Szt	.8431662	.0181632	46.42	0.000	.807567	.8787653
szd_i <-						
Szd	.7692321	.0196626	39.12	0.000	.7306942	.80777
szd_c <-						
Szd	.8604596	.0179455	47.95	0.000	.8252871	.8956321
szd_o <-						
Szd	.8715597	.0155875	55.91	0.000	.8410086	.9021107
var(e.par_i)	.4930975	.0372871			.4251739	.5718722
var(e.par_c)	.2926882	.0407398			.2228049	.3844905
var(e.par_o)	.3796764	.0374404			.3129503	.4606295
var(e.szt_i)	.3789163	.0319498			.3211966	.4470082
var(e.szt_c)	.4108313	.0374582			.3436006	.4912169
var(e.szt_o)	.2890708	.0306291			.2348623	.3557912
var(e.szd_i)	.408282	.0302501			.3530966	.4720922
var(e.szd_c)	.2596093	.0308827			.2056187	.3277766
var(e.szd_o)	.2403837	.027171			.192616	.2999976
var(Par)	1	.			.	.
var(Szt)	1	.			.	.
var(Szd)	1	.			.	.

cov(e.par_i, e.szt_i)	.2166732	.0535966	4.04	0.000	.1116258	.3217207
cov(e.par_i, e.szd_i)	.4411039	.0451782	9.76	0.000	.3525563	.5296515
cov(e.par_c, e.szt_c)	-.1074802	.0691107	-1.56	0.120	-.2429348	.0279743
cov(e.par_c, e.szd_c)	-.2646125	.0836965	-3.16	0.002	-.4286546	-.1005705
cov(e.par_o, e.szt_o)	.4132457	.0571588	7.23	0.000	.3012165	.5252749
cov(e.par_o, e.szd_o)	.3684402	.0587572	6.27	0.000	.2532781	.4836022
cov(e.szt_i, e.szd_i)	.7456394	.0351079	21.24	0.000	.6768292	.8144496
cov(e.szt_c, e.szd_c)	-.3296552	.0720069	-4.58	0.000	-.4707861	-.1885244
cov(e.szt_o, e.szd_o)	.4781276	.0588923	8.12	0.000	.3627009	.5935544
cov(Par,Szt)	.3806759	.045698	8.33	0.000	.2911095	.4702422
cov(Par,Szd)	.3590146	.0456235	7.87	0.000	.2695941	.4484351
cov(Szt,Szd)	.3103837	.0466126	6.66	0.000	.2190246	.4017428

```
LR test of model vs. saturated: chi2(15)  =     14.37, Prob > chi2 = 0.4976
```

Notes:

1. We use the correlated uniqueness model fit above to analyze a multitrait–multimethod (MTMM) matrix. The MTMM matrix was developed by Campbell and Fiske (1959) to evaluate construct validity of measures. Each trait is measured using different methods, and the correlation matrix produced is used to evaluate whether measures that are related in theory are related in fact (convergent validity) and whether measures that are not intended to be related are not related in fact (discriminant validity).

   In this example, the traits are the latent variables `Par`, `Szt`, and `Szd`.

   The observed variables are the method–trait combinations.

   The observed traits are the personality disorders paranoid (`par`), schotypal (`szt`), and schizoid (`szd`). The methods used to measure them are self-report (`_i`), clinical interview (`_c`), and observer rating (`_o`). Thus variable `par_i` is paranoid (`par`) measured by self-report (`_i`).

2. Note our use of the `covstructure()` option, which we abbreviated to `covstr()`. We used this option instead of `cov()` to save typing; see *Correlated uniqueness model* in [SEM] **Intro 5**.

3. Large values of the factor loadings (path coefficients) indicate convergent validity.

4. Small correlations between latent variables indicate discriminant validity.

Fitting the model with the Builder

Use the diagram above for reference.

There are many ways to draw this diagram. This one produces the cleanest diagram most easily.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_cu1
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Change the size of the observed variables' rectangles.

   a. In the SEM Builder menu, select **Settings > Variables > All observed...**.

   b. In the resulting dialog box, change the first size to `.44` and click on **OK**.

4. Create the nine measurement variables.

   Select the Add observed variables set tool, , and then click in the far left of the diagram, about one-third of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the nine variables in this order: `par_i`, `szt_i`, `szd_i`, `par_c`, `szt_c`, `szd_c`, `par_o`, `szt_o`, and `szd_o`;

   c. select `Horizontal` in the *Orientation* control;

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

   Be sure you select the observed variables in the order indicated above; otherwise, the instructions below for creating covariances among the errors will not be correct.

5. Create the set of three latent variables.

   Select the Add latent variables set tool, , and then click above the `szt_i` rectangle, about one-third of the way down from the top of the diagram.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. type the three latent variable names `Par`, `Szt`, and `Szd` into the *Variables* control;

   c. select `Horizontal` in the *Orientation* control;

   d. select the **Distances** tab;

   e. select `1` (inch) from the *Distance between variables* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

   We could have forgone specifying the distances between variables and simply dragged each variable where we wished after it was created.

6. Create the factor-loading paths.

   a. Select the Add path tool, .

   b. Click in the bottom of the `Par` oval (it will highlight when you hover over it), and drag a path to the top of the `par_i` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the ⟶ tool, create the following paths by clicking first in the bottom of the latent variable and dragging it to the top of the observed (measurement) variable:

```
Szt -> szt_i
Szd -> szd_i
Par -> par_c
Szt -> szt_c
Szd -> szd_c
Par -> par_o
Szt -> szt_o
Szd -> szd_o
```

7. Clean up the direction of the errors.

We want all the errors to be below the measurement variables.

a. Choose the Select tool, .

b. Click on the rectangle of any measurement variable whose associated error is not below it.

c. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is below the measurement variable.

Repeat this for all errors that are not below the measurement variables.

8. Correlate the errors within the self-report, clinical interview, and observer rating groups.

a. Select the Add covariance tool, ⌒.

b. Click in the ϵ_2 circle (it will highlight when you hover over it), and drag a covariance to the ϵ_1 circle (it will highlight when you can release to connect the covariance).

c. Continue with the Add covariance tool, ⌒, to create eight more covariances by clicking the first-listed error and dragging it to the second-listed error.

ϵ_3 -> ϵ_2
ϵ_3 -> ϵ_1
ϵ_5 -> ϵ_4
ϵ_6 -> ϵ_5
ϵ_6 -> ϵ_4
ϵ_8 -> ϵ_7
ϵ_9 -> ϵ_8
ϵ_9 -> ϵ_7

The order in which we create the covariances is unimportant. We dragged each covariance from right to left because the bow of the covariance is outward when we drag in a clockwise direction and inward when we drag in a counterclockwise direction. Had we connected the opposite way, we would have needed to use the Contextual Toolbar to mirror the bow of the covariances.

9. Correlate the latent factors.

a. Select the Add covariance tool, ⌒.

b. Click in the `Par` oval and drag a covariance to the `Szt` oval.

c. Click in the `Szt` oval and drag a covariance to the `Szd` oval.

d. Click in the `Par` oval and drag a covariance to the `Szd` oval.

10. Clean up.

    If you do not like where a covariance has been connected to its variable, use the Select tool, , to click on the covariance, and then simply click on where it connects to an oval and drag the endpoint. You can also change the bow of the covariance by dragging the control point that extends from one end of the selected covariance.

11. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

12. Show standardized estimates.

    From the SEM Builder menu, select **View > Standardized estimates**.

Reference

Campbell, D. T., and D. W. Fiske. 1959. Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological Bulletin* 56: 81–105.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem and gsem option covstructure( )** — Specifying covariance restrictions

Title

Example 18 — Latent growth model

Description Remarks and examples Reference Also see

Description

To demonstrate a latent growth model, we use the following data:

```
. use https://www.stata-press.com/data/r16/sem_lcm
. describe
Contains data from https://www.stata-press.com/data/r16/sem_lcm.dta
  obs:           359
 vars:             4                          25 May 2018 11:08
                                               (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
lncrime0        float   %9.0g                 ln(crime rate) in Jan & Feb
lncrime1        float   %9.0g                 ln(crime rate) in Mar & Apr
lncrime2        float   %9.0g                 ln(crime rate) in May & Jun
lncrime3        float   %9.0g                 ln(crime rate) in Jul & Aug
-------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Data used in Bollen, Kenneth A. and Patrick J. Curran, 2006, _Latent
      Curve Models: A Structural Equation Perspective_. Hoboken, New Jersey:
      John Wiley & Sons
  2.  Data from 1995 Uniform Crime Reports for 359 communities in New York
      state.
```

See *Latent growth models* in [SEM] **Intro 5** for background.

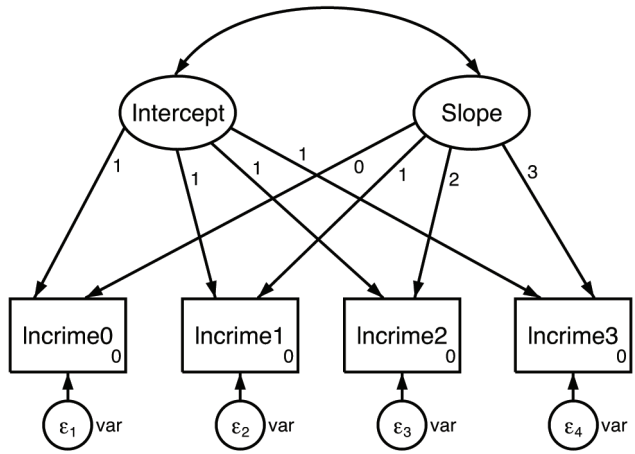
Remarks and examples

Remarks are presented under the following headings:

Fitting the model
Fitting the model with the Builder

Fitting the model

We fit the following model:




```
. sem (lncrime0 <- Intercept@1 Slope@0 _cons@0)
>     (lncrime1 <- Intercept@1 Slope@1 _cons@0)
>     (lncrime2 <- Intercept@1 Slope@2 _cons@0)
>     (lncrime3 <- Intercept@1 Slope@3 _cons@0),
>             latent(Intercept Slope)
>             var(e.lncrime0@var e.lncrime1@var
>                 e.lncrime2@var e.lncrime3@var)
>             means(Intercept Slope)
Endogenous variables
Measurement:  lncrime0 lncrime1 lncrime2 lncrime3
Exogenous variables
Latent:       Intercept Slope
Fitting target model:
Iteration 0:   log likelihood = -1034.1038
Iteration 1:   log likelihood = -1033.9044
Iteration 2:   log likelihood = -1033.9037
Iteration 3:   log likelihood = -1033.9037
Structural equation model                       Number of obs     =        359
Estimation method  = ml
Log likelihood     = -1033.9037
 ( 1)  [lncrime0]Intercept = 1
 ( 2)  [lncrime1]Intercept = 1
 ( 3)  [lncrime1]Slope = 1
 ( 4)  [lncrime2]Intercept = 1
 ( 5)  [lncrime2]Slope = 2
 ( 6)  [lncrime3]Intercept = 1
 ( 7)  [lncrime3]Slope = 3
 ( 8)  [/]var(e.lncrime0) - [/]var(e.lncrime3) = 0
 ( 9)  [/]var(e.lncrime1) - [/]var(e.lncrime3) = 0
 (10)  [/]var(e.lncrime2) - [/]var(e.lncrime3) = 0
 (11)  [lncrime0]_cons = 0
 (12)  [lncrime1]_cons = 0
 (13)  [lncrime2]_cons = 0
 (14)  [lncrime3]_cons = 0
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
lncrime0						
Intercept	1	(constrained)				
_cons	0	(constrained)				
lncrime1						
Intercept	1	(constrained)				
Slope	1	(constrained)				
_cons	0	(constrained)				
lncrime2						
Intercept	1	(constrained)				
Slope	2	(constrained)				
_cons	0	(constrained)				
lncrime3						
Intercept	1	(constrained)				
Slope	3	(constrained)				
_cons	0	(constrained)				
mean(Inter~t)	5.337915	.0407501	130.99	0.000	5.258047	5.417784
mean(Slope)	.1426952	.0104574	13.65	0.000	.1221992	.1631912
var(e.lncr~0)	.0981956	.0051826			.0885457	.1088972
var(e.lncr~1)	.0981956	.0051826			.0885457	.1088972
var(e.lncr~2)	.0981956	.0051826			.0885457	.1088972
var(e.lncr~3)	.0981956	.0051826			.0885457	.1088972
var(Interc~t)	.527409	.0446436			.4467822	.6225858
var(Slope)	.0196198	.0031082			.0143829	.0267635
cov(Interc~t, Slope)	-.034316	.0088848	-3.86	0.000	-.0517298	-.0169022

LR test of model vs. saturated: chi2(8) = 16.25, Prob > chi2 = 0.0390

Notes:

1. In this example, we have repeated measures of the crime rate in 1995. We will assume that the underlying rate grows linearly.
2. As explained in *Latent growth models* in [SEM] **Intro 5**, we assume

$$\texttt{lncrime}_i = \texttt{Intercept} + i \times \texttt{Slope}$$

3. `sem` does not usually report the means of latent exogenous variables because `sem` automatically includes the identifying constraint that the means are 0; see *How sem (gsem) solves the problem for you* in [SEM] **Intro 4** and see *Default normalization constraints* in [SEM] **sem**.

   In this case, `sem` did not constrain the means to be 0 because we specified `sem`'s `means()` option. In particular, we specified `means(Intercept Slope)`, which said not to constrain the means of those two exogenous latent variables and to report the estimated result.

   Our model was identified even without the usual 0 constraints on `Intercept` and `Slope` because we specified enough other constraints.
4. We estimate the `Intercept` to have mean 5.34 and the mean `Slope` to be 0.14 per two months. Remember, we have measured crime rates as log base e crime rates.

5. It might help some to think of this as a mixed model:

```
. generate id = _n
. reshape long lncrime, i(id) j(year)
. mixed lncrime year || id:year, cov(unstructured) mle var
```

The mean `Intercept` and `Slope` are what `mixed` would refer to as the coefficients in the fixed-effects part of the model.

Acock (2013, chap. 4) discusses the use of `sem` to fit latent growth-curve models in more detail. Acock demonstrates extensions to the basic model we fit here, such as including time-varying and time-invariant covariates in the model.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

```
. use https://www.stata-press.com/data/r16/sem_lcm
```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the four repeated measurements for log crime rate.

   Select the Add observed variables set tool, ▫▫▫, and then click in the diagram about one-fourth of the way in from the left and about one-third of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the variables in this order: `lncrime0`, `lncrime1`, `lncrime2`, and `lncrime3`;

   c. select `Horizontal` in the *Orientation* control;

   d. select the **Distances** tab;

   e. select `.25` (inch) from the *Distance between variables* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

4. Create the two latent variables.

   Select the Add latent variables set tool, ◦◦◦, and then click in the diagram about one-third of the way down from the top and above the left side of the `lncrime0` rectangle.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. type the latent variable names `Intercept` and `Slope` into the *Variables* control;

   c. select `Horizontal` in the *Orientation* control;

   d. select the **Distances** tab;

e. select `1` (inch) from the *Distance between variables* control;

f. click on **OK**.

If you wish, move the set of variables by clicking on any variable and dragging it.

We could have forgone specifying the distances between variables and simply dragged each variable where we wished after it was created.

5. Create the factor-loading paths.

   a. Select the Add path tool, ⟶.

   b. Click in the bottom-left quadrant of the `Intercept` oval (it will highlight when you hover over it), and drag a path to the top of the `lncrime0` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the ⟶ tool, create the following paths by clicking first in the bottom of the latent variable and dragging it to the top of the observed (measurement) variable:

```
Intercept -> lncrime1
Intercept -> lncrime2
Intercept -> lncrime3
Slope -> lncrime0
Slope -> lncrime1
Slope -> lncrime2
Slope -> lncrime3
```

6. Clean up the direction of the errors.

   We want all the errors to be below the measurement variables.

   a. Choose the Select tool, .

   b. Click on the rectangle of any measurement variable whose associated error is not below it.

   c. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is below the measurement variable.

   Repeat this for all errors that are not below the measurement variables.

7. Create the covariance between `Intercept` and `Slope`.

   a. Select the Add covariance tool, ⌒.

   b. Click in the top-right quadrant of the `Intercept` oval, and drag a covariance to the top left of the `Slope` oval.

8. Clean up paths and covariance.

   If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint. Similarly, you can change where the covariance connects to the latent variables by clicking on the covariance and dragging the endpoint. You can also change the bow of the covariance by clicking on the covariance and dragging the control point that extends from one end of the selected covariance.

9. Constrain the intercepts of the measurements to 0.

   a. Choose the Select tool, .

b. Click on the rectangle for `lncrime0`. In the Contextual Toolbar, type `0` in the α box and press *Enter*.

c. Repeat this process to add the `0` constraint on the intercept for `lncrime1`, `lncrime2`, and `lncrime3`.

10. Set constraints on the paths from `Intercept` to the measurements.

a. Continue with the Select tool, .

b. Click on the path from `Intercept` to `lncrime0`. In the Contextual Toolbar, type `1` in the β box and press *Enter*.

c. Repeat this process to add the `1` constraint on the following paths:

```
Intercept -> lncrime1
Intercept -> lncrime2
Intercept -> lncrime3
```

11. Set constraints on the paths from `Slope` to the measurements.

a. Continue with the Select tool, .

b. Click on the path from `Slope` to `lncrime0`. In the Contextual Toolbar, type `0` in the β box and press *Enter*.

c. Click on the path from `Slope` to `lncrime1`. In the Contextual Toolbar, type `1` in the β box and press *Enter*.

d. Click on the path from `Slope` to `lncrime2`. In the Contextual Toolbar, type `2` in the β box and press *Enter*.

e. Click on the path from `Slope` to `lncrime3`. In the Contextual Toolbar, type `3` in the β box and press *Enter*.

12. Set equality constraints on the error variances.

a. Continue with the Select tool, .

b. Click in the ϵ_1 circle, which is the error term for `lncrime0`. In the Contextual Toolbar, type `var` in the σ^2 box and press *Enter*.

c. Repeat this process to add the `var` constraint on the three remaining error variances: ϵ_2, ϵ_3, and ϵ_4.

13. Clean up placement of the constraints.

From the SEM Builder menu, select **Settings > Connections > Paths...**.

In the resulting dialog box, do the following:

a. Click on the **Results** tab.

b. Click on the **Result 1...** button at the bottom left.

c. In the *Appearance of result 1 - paths* dialog box that opens, choose `20` (%) in the *Distance between nodes* control.

d. Click on **OK** on the *Appearance of result 1 - paths* dialog box.

e. Click on **OK** on the *Connection settings - paths* dialog box.

14. Specify that the means of `Intercept` and `Slope` are to be estimated.

    a. Choose the Select tool, .

    b. Double-click on the `Intercept` oval.

    c. In the resulting dialog box, check the *Estimate mean* box.

    d. Click on **OK**.

    e. In the same manner, double-click on `Slope` and check the *Estimate mean* box.

15. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_lcm
```

Reference

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Also see

[SEM] **sem** — Structural equation model estimation command

Title

Example 19 — Creating multiple-group summary statistics data

Description Remarks and examples Reference Also see

Description

The data analyzed in [SEM] **Example 20** are summary statistics data (SSD) and contain summary statistics on two groups of subjects, those from grade 4 and those from grade 5. Below we show how we created this summary statistics dataset.

See [SEM] **Intro 11** for background on SSD.

Remarks and examples

See [SEM] **Example 2** for creating a single-group dataset from published covariances. In this example, we will create a two-group dataset from published correlations, standard deviations, and means.

Marsh and Hocevar (1985) publish lots of SSD, of which we will enter the data for students in grade 4 and grade 5 found on pages 579–581. In that source, the authors published the correlations, standard deviations, and means of their variables.

We will set the data for the first group, declare that we have groups and wish to add another, and set the data for the second group.

Starting with the first group, we will issue the following commands:

```
. ssd init       variable names
. ssd set obs    values
. ssd set means  values
. ssd set sd     values
. ssd set corr   values
```

We will first set the end-of-line delimiter to a semicolon because we are going to have some long lines. We will be entering SSD for 16 variables!

```
. #delimit ;
delimiter now ;
. ssd init phyab1   phyab2   phyab3   phyab4
>          appear1  appear2  appear3  appear4
>          peerrel1 peerrel2 peerrel3 peerrel4
>          parrel1  parrel2  parrel3  parrel4 ;
Summary statistics data initialized.  Next use, in any order,
    ssd set observations (required)
        It is best to do this first.
    ssd set means (optional)
        Default setting is 0.
    ssd set variances or ssd set sd (optional)
        Use this only if you have set or will set correlations and, even
        then, this is optional but highly recommended.  Default setting is 1.
    ssd set covariances or ssd set correlations (required)
```

```
. ssd set obs 134 ;
  (value set)
    Status:
                        observations:    set
                               means:  unset
                    variances or sd:  unset
         covariances or correlations:  unset (required to be set)
. ssd set means
>     8.34 8.34 8.37 8.40 7.51 7.22 7.03 7.13
>     8.44 7.62 7.06 7.89 9.32 9.39 8.69 9.13 ;
  (values set)
    Status:
                        observations:    set
                               means:    set
                    variances or sd:  unset
         covariances or correlations:  unset (required to be set)
. ssd set sd
>     1.90 1.75 2.06 1.88 2.30 2.63 2.71 2.42
>     2.05 2.22 2.38 2.12 1.21 1.21 1.71 1.32 ;
  (values set)
    Status:
                        observations:    set
                               means:    set
                    variances or sd:    set
         covariances or correlations:  unset (required to be set)
. ssd set corr
>     1.0 \
>     .50 1.0 \
>     .59 .46 1.0 \
>     .58 .43 .66 1.0 \
>     .30 .27 .35 .46 1.0 \
>     .32 .34 .38 .39 .71 1.0 \
>     .38 .41 .43 .53 .68 .67 1.0 \
>     .23 .29 .33 .43 .61 .63 .73 1.0 \
>     .43 .32 .40 .42 .36 .34 .45 .42 1.0 \
>     .38 .40 .38 .49 .53 .61 .69 .59 .59 1.0 \
>     .27 .24 .41 .37 .43 .46 .57 .57 .61 .59 1.0 \
>     .43 .41 .37 .47 .51 .45 .63 .61 .59 .58 .65 1.0 \
>     .20 .14 .15 .18 .22 .21 .13 .03 .15 .19 .12 .14 1.0 \
>     .29 .18 .26 .20 .25 .29 .17 .25 .35 .23 .23 .28 .25 1.0 \
>     .37 .14 .34 .37 .34 .34 .35 .33 .42 .36 .39 .39 .53 .50 1.0 \
>     .13 .10 .16 .21 .33 .28 .23 .22 .23 .25 .23 .28 .46 .43 .59 1.0 ;
  (values set)
    Status:
                        observations:    set
                               means:    set
                    variances or sd:    set
         covariances or correlations:    set
. #delimit cr
delimiter now cr
```

We have now entered the data for the first group, and `ssd` reports that we have a fully set dataset.

Next we are going to add a second group by typing

```
. ssd addgroup grade
  (new group grade==2 added)
    The ssd set commands now modify the new group grade==2.  If you need to
    modify data for grade==1, place a 1 right after the set.  For example,
        . ssd set 1 means ...
    would modify the means for group grade==1.
```

The `ssd set` command now modifies the new group `grade==2`. If we needed to modify data for `grade==1`, we would place a 1 right after the `set`. For example,

```
. ssd set 1 means ...
```

We are not modifying data; however, we are now adding data for the second group. The procedure for entering the second group is the same as the procedure for entering the first group:

```
. ssd set obs    values
. ssd set means  values
. ssd set sd     values
. ssd set corr   values
```

We do that below.

```
. #delimit ;
delimiter now ;
. ssd set obs 251 ;
  (value set for group grade==2)
    Status for group grade==2:
                      observations:    set
                             means:  unset
                  variances or sd:  unset
        covariances or correlations:  unset (required to be set)
. ssd set corr
>     1.0 \
>     .31 1.0 \
>     .52 .45 1.0 \
>     .54 .46 .70 1.0 \
>     .15 .33 .22 .21 1.0 \
>     .14 .28 .21 .13 .72 1.0 \
>     .16 .32 .35 .31 .59 .56 1.0 \
>     .23 .29 .43 .36 .55 .51 .65 1.0 \
>     .24 .13 .24 .23 .25 .24 .24 .30 1.0 \
>     .19 .26 .22 .18 .34 .37 .36 .32 .38 1.0 \
>     .16 .24 .36 .30 .33 .29 .44 .51 .47 .50 1.0 \
>     .16 .21 .35 .24 .31 .33 .41 .39 .47 .47 .55 1.0 \
>     .08 .18 .09 .12 .19 .24 .08 .21 .21 .19 .19 .20 1.0 \
>     .01 -.01 .03 .02 .10 .13 .03 .05 .26 .17 .23 .26 .33 1.0 \
>     .06 .19 .22 .22 .23 .24 .20 .26 .16 .23 .38 .24 .42 .40 1.0 \
>     .04 .17 .10 .07 .26 .24 .12 .26 .16 .22 .32 .17 .42 .42 .65 1.0 ;
  (values set for group grade==2)
    Status for group grade==2:
                      observations:    set
                             means:  unset
                  variances or sd:  unset
        covariances or correlations:    set
```

```
. ssd set sd     1.84 1.94 2.07 1.82 2.34 2.61 2.48 2.34
>                1.71 1.93 2.18 1.94 1.31 1.57 1.77 1.47 ;
  (values set for group grade==2)
    Status for group grade==2:
                        observations:    set
                               means:  unset
                    variances or sd:    set
         covariances or correlations:    set
. ssd set means  8.20 8.23 8.17 8.56 7.41 7.00 7.17 7.40
>                8.81 7.94 7.52 8.29 9.35 9.13 8.67 9.00 ;
  (values set for group grade==2)
    Status for group grade==2:
                        observations:    set
                               means:    set
                    variances or sd:    set
         covariances or correlations:    set
. #delimit cr
delimiter now cr
```

We could stop here and save the data in a Stata dataset. We might type

```
. save sem_2fmmby
```

However, we intend to use these data as an example in this manual and online. Here is what you would see if you typed `ssd describe`:

```
. ssd describe
  Summary statistics data
    obs:             385
   vars:              16
  ------------------------------------------------------------------------------
  variable name                  variable label
  ------------------------------------------------------------------------------
  phyab1
  phyab2
  phyab3
  phyab4
  appear1
  appear2
  appear3
  appear4
  peerrel1
  peerrel2
  peerrel3
  peerrel4
  parrel1
  parrel2
  parrel3
  parrel4
  ------------------------------------------------------------------------------
  Group variable:  grade  (2 groups)
   Obs. by group:  134, 251
```

We are going to label these data so that `ssd describe` can provide more information:

```
. label data "two-factor CFA"
. label var phyab1   "Physical ability 1"
. label var phyab2   "Physical ability 2"
. label var phyab3   "Physical ability 3"
. label var phyab4   "Physical ability 4"
. label var appear1   "Appearance 1"
. label var appear2   "Appearance 2"
. label var appear3   "Appearance 3"
. label var appear4   "Appearance 4"
. label var peerrel1  "Relationship w/ peers 1"
. label var peerrel2  "Relationship w/ peers 2"
. label var peerrel3  "Relationship w/ peers 3"
. label var peerrel4  "Relationship w/ peers 4"
. label var parrel1   "Relationship w/ parent 1"
. label var parrel2   "Relationship w/ parent 2"
. label var parrel3   "Relationship w/ parent 3"
. label var parrel4   "Relationship w/ parent 4"
. #delimit ;
delimiter now ;
. notes: Summary statistics data from
>        Marsh, H. W. and Hocevar, D., 1985,
>        "Application of confirmatory factor analysis to the study of
>        self-concept: First- and higher order factor models and their
>        invariance across groups", _Psychological Bulletin_, 97: 562-582. ;
. notes: Summary statistics based on
>        134 students in grade 4 and
>        251 students in grade 5
>        from Sydney, Australia. ;
. notes: Group 1 is grade 4, group 2 is grade 5. ;
. notes: Data collected using the Self-Description Questionnaire
>        and includes sixteen subscales designed to measure
>        nonacademic traits:  four intended to measure physical
>        ability, four intended to measure physical appearance,
>        four intended to measure relations with peers, and four
>        intended to measure relations with parents. ;
. #delimit cr
delimiter now cr
```

We would now `save` the dataset.

To see `ssd describe`'s output with the data labeled, see [SEM] **Example 20**.

Reference

Marsh, H. W., and D. Hocevar. 1985. Application of confirmatory factor analysis to the study of self-concept: First- and higher order factor models and their invariance across groups. *Psychological Bulletin* 97: 562–582.

Also see

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **ssd** — Making summary statistics data (sem only)

Title

Example 20 — Two-factor measurement model by group

Description Remarks and examples Reference Also see

Description

Below we demonstrate `sem`'s `group()` option, which allows fitting models in which path coefficients and covariances differ across groups of the data, such as for males and females. We use the following data:

```
. use https://www.stata-press.com/data/r16/sem_2fmmby
(two-factor CFA)
. ssd describe
  Summary statistics data from
  https://www.stata-press.com/data/r16/sem_2fmmby.dta
    obs:             385                  two-factor CFA
   vars:              16                  25 May 2018 11:11
                                          (_dta has notes)
-------------------------------------------------------------------------------
  variable name                  variable label
-------------------------------------------------------------------------------
  phyab1                         Physical ability 1
  phyab2                         Physical ability 2
  phyab3                         Physical ability 3
  phyab4                         Physical ability 4
  appear1                        Appearance 1
  appear2                        Appearance 2
  appear3                        Appearance 3
  appear4                        Appearance 4
  peerrel1                       Relationship w/ peers 1
  peerrel2                       Relationship w/ peers 2
  peerrel3                       Relationship w/ peers 3
  peerrel4                       Relationship w/ peers 4
  parrel1                        Relationship w/ parent 1
  parrel2                        Relationship w/ parent 2
  parrel3                        Relationship w/ parent 3
  parrel4                        Relationship w/ parent 4
-------------------------------------------------------------------------------
  Group variable:  grade  (2 groups)
   Obs. by group:  134, 251
. notes
_dta:
  1.  Summary statistics data from Marsh, H. W. and Hocevar, D., 1985,
      "Application of confirmatory factor analysis to the study of
      self-concept: First- and higher order factor models and their invariance
      across groups", _Psychological Bulletin_, 97: 562-582.
  2.  Summary statistics based on 134 students in grade 4 and 251 students in
      grade 5 from Sydney, Australia.
  3.  Group 1 is grade 4, group 2 is grade 5.
  4.  Data collected using the Self-Description Questionnaire and includes
      sixteen subscales designed to measure nonacademic status: four intended
      to measure physical ability, four intended to measure physical
      appearance, four intended to measure relations with peers, and four
      intended to measure relations with parents.
```

Remarks and examples

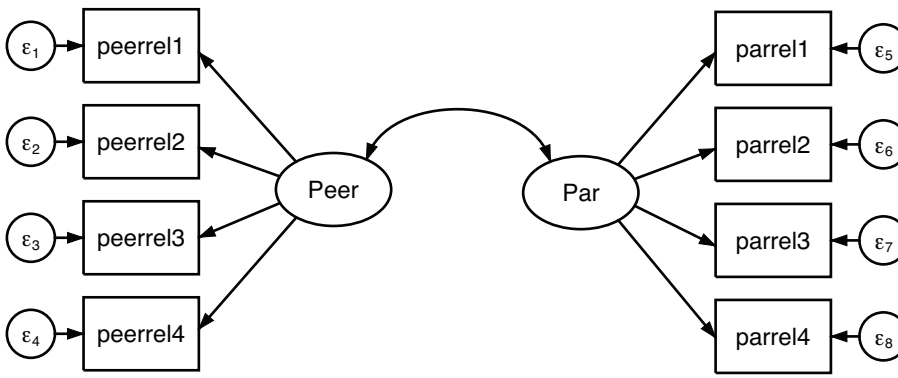
Remarks are presented under the following headings:

Background
Fitting the model with all the data
Fitting the model with the group() option
Fitting the model with the Builder

Background

See [SEM] **Intro 6** for background on `sem`'s `group()` option.

We will fit the model



which in command syntax can be written as

```
(Peer -> peerrel1 peerrel2 peerrel3 peerrel4)   ///
(Par  -> parrel1  parrel2  parrel3  parrel4)
```

We are using the same data used in [SEM] **Example 15**, but we are using more of the data and fitting a different model. To remind you, those data were collected from students in grade 5. The dataset we are using, however, has data for students from grade 4 and from grade 5, which was created in [SEM] **Example 19**. We have the following observed variables:

1. Four measures of physical ability.
2. Four measures of appearance.
3. Four measures of quality of relationship with peers.
4. Four measures of quality of relationship with parents.

In this example, we will consider solely the measurement problem, and include only the measurement variables for the two kinds of relationship quality. We are going to treat quality of relationship with peers as measures of underlying factor `Peer` and quality of relationship with parents as measures of underlying factor `Par`.

Below we will

1. Fit the model with all the data. This amounts to assuming that the students in grades 4 and 5 are identical in terms of this measurement problem.
2. Fit the model with `sem`'s `group()` option, which will constrain some parameters to be the same for students in grades 4 and 5 and leave free of constraint the others.

Fitting the model with all the data

Throughout this example, we want you to appreciate that we are using SSD and that matters not at all. Not one command would have a different syntax or option, or produce a different result, if we had the real data.

We begin by fitting the model with all the data:

```
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)
>     (Par  -> parrel1  parrel2  parrel3  parrel4)

Endogenous variables

Measurement:  peerrel1 peerrel2 peerrel3 peerrel4 parrel1 parrel2 parrel3 parrel4

Exogenous variables

Latent:       Peer Par

Fitting target model:

Iteration 0:   log likelihood =  -5559.545
Iteration 1:   log likelihood =  -5558.609
Iteration 2:   log likelihood = -5558.6017
Iteration 3:   log likelihood = -5558.6017
```

```
Structural equation model                       Number of obs     =        385
Estimation method  = ml
Log likelihood     = -5558.6017
 ( 1)  [peerrel1]Peer = 1
 ( 2)  [parrel1]Par = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
peerrel1						
Peer	1	(constrained)				
_cons	8.681221	.0937197	92.63	0.000	8.497534	8.864908
peerrel2						
Peer	1.113865	.09796	11.37	0.000	.9218666	1.305863
_cons	7.828623	.1037547	75.45	0.000	7.625268	8.031979
peerrel3						
Peer	1.42191	.114341	12.44	0.000	1.197806	1.646014
_cons	7.359896	.1149905	64.00	0.000	7.134519	7.585273
peerrel4						
Peer	1.204146	.0983865	12.24	0.000	1.011312	1.39698
_cons	8.150779	.1023467	79.64	0.000	7.950183	8.351375
parrel1						
Par	1	(constrained)				
_cons	9.339558	.0648742	143.96	0.000	9.212407	9.46671
parrel2						
Par	1.112383	.1378687	8.07	0.000	.8421655	1.382601
_cons	9.220494	.0742356	124.21	0.000	9.074994	9.365993
parrel3						
Par	2.037924	.204617	9.96	0.000	1.636882	2.438966
_cons	8.676961	.088927	97.57	0.000	8.502667	8.851255
parrel4						
Par	1.52253	.1536868	9.91	0.000	1.221309	1.82375
_cons	9.045247	.0722358	125.22	0.000	8.903667	9.186826
var(e.peer~1)	1.809309	.1596546			1.521956	2.150916
var(e.peer~2)	2.193804	.194494			1.843884	2.610129
var(e.peer~3)	1.911874	.214104			1.535099	2.381126
var(e.peer~4)	1.753037	.1749613			1.441575	2.131792
var(e.parr~1)	1.120333	.0899209			.9572541	1.311193
var(e.parr~2)	1.503003	.1200739			1.285162	1.757769
var(e.parr~3)	.9680081	.1419777			.7261617	1.290401
var(e.parr~4)	.8498834	.0933687			.685245	1.054078
var(Peer)	1.572294	.2255704			1.186904	2.082822
var(Par)	.5000022	.093189			.3469983	.7204709
cov(Peer,Par)	.4226706	.0725253	5.83	0.000	.2805236	.5648176

```
LR test of model vs. saturated: chi2(19)  =     28.19, Prob > chi2 = 0.0798
```

Note:

1. We are using SSD with data for two separate groups. There is no hint of that in the output above because `sem` combined the summary statistics and produced overall results just as if we had the real data.

Fitting the model with the group() option

```
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)
>     (Par  -> parrel1  parrel2  parrel3  parrel4), group(grade)
Endogenous variables
Measurement:  peerrel1 peerrel2 peerrel3 peerrel4 parrel1 parrel2 parrel3 parrel4
Exogenous variables
Latent:       Peer Par
Fitting target model:
Iteration 0:   log likelihood =  -13049.77  (not concave)
Iteration 1:   log likelihood = -10819.682  (not concave)
Iteration 2:   log likelihood = -8873.4568  (not concave)
Iteration 3:   log likelihood = -6119.7114  (not concave)
Iteration 4:   log likelihood =  -5949.354  (not concave)
Iteration 5:   log likelihood = -5775.6085  (not concave)
Iteration 6:   log likelihood = -5713.9178  (not concave)
Iteration 7:   log likelihood = -5638.1208  (not concave)
Iteration 8:   log likelihood = -5616.6335  (not concave)
Iteration 9:   log likelihood = -5595.7507  (not concave)
Iteration 10:  log likelihood = -5589.9802  (not concave)
Iteration 11:  log likelihood = -5578.8701  (not concave)
Iteration 12:  log likelihood = -5574.0162  (not concave)
Iteration 13:  log likelihood = -5568.0786
Iteration 14:  log likelihood = -5551.7349
Iteration 15:  log likelihood = -5544.0052
Iteration 16:  log likelihood = -5542.7113
Iteration 17:  log likelihood = -5542.6775
Iteration 18:  log likelihood = -5542.6774
Structural equation model                       Number of obs     =        385
Grouping variable  = grade                      Number of groups  =          2
Estimation method  = ml
Log likelihood     = -5542.6774
 ( 1)  [peerrel1]1bn.grade#c.Peer = 1
 ( 2)  [peerrel2]1bn.grade#c.Peer - [peerrel2]2.grade#c.Peer = 0
 ( 3)  [peerrel3]1bn.grade#c.Peer - [peerrel3]2.grade#c.Peer = 0
 ( 4)  [peerrel4]1bn.grade#c.Peer - [peerrel4]2.grade#c.Peer = 0
 ( 5)  [parrel1]1bn.grade#c.Par = 1
 ( 6)  [parrel2]1bn.grade#c.Par - [parrel2]2.grade#c.Par = 0
 ( 7)  [parrel3]1bn.grade#c.Par - [parrel3]2.grade#c.Par = 0
 ( 8)  [parrel4]1bn.grade#c.Par - [parrel4]2.grade#c.Par = 0
 ( 9)  [peerrel1]1bn.grade - [peerrel1]2.grade = 0
 (10)  [peerrel2]1bn.grade - [peerrel2]2.grade = 0
 (11)  [peerrel3]1bn.grade - [peerrel3]2.grade = 0
 (12)  [peerrel4]1bn.grade - [peerrel4]2.grade = 0
 (13)  [parrel1]1bn.grade - [parrel1]2.grade = 0
 (14)  [parrel2]1bn.grade - [parrel2]2.grade = 0
 (15)  [parrel3]1bn.grade - [parrel3]2.grade = 0
 (16)  [parrel4]1bn.grade - [parrel4]2.grade = 0
 (17)  [peerrel1]2.grade#c.Peer = 1
 (18)  [parrel1]2.grade#c.Par = 1
 (19)  [/]mean(Peer)#1bn.grade = 0
 (20)  [/]mean(Par)#1bn.grade = 0
```

Group : 1 Number of obs = 134

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
peerrel1						
Peer	1	(constrained)				
_cons	8.466539	.1473448	57.46	0.000	8.177748	8.755329
peerrel2						
Peer	1.109234	.0975279	11.37	0.000	.9180833	1.300385
_cons	7.589872	.1632145	46.50	0.000	7.269977	7.909766
peerrel3						
Peer	1.409361	.1138314	12.38	0.000	1.186256	1.632467
_cons	7.056996	.1964299	35.93	0.000	6.672001	7.441992
peerrel4						
Peer	1.195982	.0980272	12.20	0.000	1.003852	1.388112
_cons	7.89358	.169158	46.66	0.000	7.562036	8.225123
parrel1						
Par	1	(constrained)				
_cons	9.368654	.0819489	114.32	0.000	9.208037	9.529271
parrel2						
Par	1.104355	.1369365	8.06	0.000	.8359649	1.372746
_cons	9.287629	.0903296	102.82	0.000	9.110587	9.464672
parrel3						
Par	2.05859	.2060583	9.99	0.000	1.654723	2.462457
_cons	8.741898	.136612	63.99	0.000	8.474144	9.009653
parrel4						
Par	1.526706	.1552486	9.83	0.000	1.222424	1.830987
_cons	9.096609	.1061607	85.69	0.000	8.888538	9.30468
mean(Peer)	0	(constrained)				
mean(Par)	0	(constrained)				
var(e.peer~1)	1.824193	.2739446			1.359074	2.448489
var(e.peer~2)	2.236974	.3310875			1.673699	2.989817
var(e.peer~3)	1.907009	.3383293			1.346908	2.700023
var(e.peer~4)	1.639881	.272764			1.18367	2.271925
var(e.parr~1)	.9669121	.1302489			.7425488	1.259067
var(e.parr~2)	.9683878	.133192			.7395628	1.268012
var(e.parr~3)	.8377567	.1986089			.526407	1.333258
var(e.parr~4)	.8343032	.1384649			.6026352	1.15503
var(Peer)	2.039297	.3784544			1.41747	2.933912
var(Par)	.4492996	.1011565			.2889976	.6985183
cov(Peer,Par)	.5012091	.1193333	4.20	0.000	.2673201	.7350982

Group : 2 Number of obs = 251

	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
peerrel1						
Peer	1	(constrained)				
_cons	8.466539	.1473448	57.46	0.000	8.177748	8.755329
peerrel2						
Peer	1.109234	.0975279	11.37	0.000	.9180833	1.300385
_cons	7.589872	.1632145	46.50	0.000	7.269977	7.909766
peerrel3						
Peer	1.409361	.1138314	12.38	0.000	1.186256	1.632467
_cons	7.056996	.1964299	35.93	0.000	6.672001	7.441992
peerrel4						
Peer	1.195982	.0980272	12.20	0.000	1.003852	1.388112
_cons	7.89358	.169158	46.66	0.000	7.562036	8.225123
parrel1						
Par	1	(constrained)				
_cons	9.368654	.0819489	114.32	0.000	9.208037	9.529271
parrel2						
Par	1.104355	.1369365	8.06	0.000	.8359649	1.372746
_cons	9.287629	.0903296	102.82	0.000	9.110587	9.464672
parrel3						
Par	2.05859	.2060583	9.99	0.000	1.654723	2.462457
_cons	8.741898	.136612	63.99	0.000	8.474144	9.009653
parrel4						
Par	1.526706	.1552486	9.83	0.000	1.222424	1.830987
_cons	9.096609	.1061607	85.69	0.000	8.888538	9.30468
mean(Peer)	.3296841	.1570203	2.10	0.036	.02193	.6374382
mean(Par)	-.0512439	.0818255	-0.63	0.531	-.211619	.1091313
var(e.peer~1)	1.773813	.1889104			1.439644	2.185549
var(e.peer~2)	2.165228	.2321565			1.75484	2.671589
var(e.peer~3)	1.950679	.2586196			1.504298	2.529516
var(e.peer~4)	1.822448	.2151827			1.445942	2.296992
var(e.parr~1)	1.213159	.1192634			1.000547	1.470949
var(e.parr~2)	1.79031	.1747374			1.478596	2.167739
var(e.parr~3)	1.015707	.1713759			.7297073	1.4138
var(e.parr~4)	.8599648	.1165865			.6592987	1.121706
var(Peer)	1.307976	.2061581			.9603661	1.781406
var(Par)	.5201696	.1029353			.3529413	.7666329
cov(Peer,Par)	.3867156	.079455	4.87	0.000	.2309867	.5424445

LR test of model vs. saturated: chi2(50) = 61.91, Prob > chi2 = 0.1204

Notes:

1. In *Which parameters vary by default, and which do not* in [SEM] **Intro 6**, we wrote that, generally speaking, when we specify `group(`*groupvar*`)`, the measurement part of the model is constrained by default to be the same across the groups, whereas the remaining parts will have separate parameters for each group.

More precisely, we revealed that `sem` classifies each parameter into one of nine classes, which are the following:

Class description	Class name	
1. structural coefficients	`scoef`	
2. structural intercepts	`scons`	
3. measurement coefficients	`mcoef`	
4. measurement intercepts	`mcons`	
5. covariances of structural errors	`serrvar`	
6. covariances of measurement errors	`merrvar`	
7. covariances between structural and measurement errors	`smerrcov`	
8. means of exogenous variables	`meanex`	(*)
9. covariances of exogenous variables	`covex`	(*)
10. all the above	`all`	(*)
11. none of the above	`none`	

(*) Be aware that classes 8, 9, and 10 (`meanex`, `covex`, and `all`) exclude the observed exogenous variables—include only the latent exogenous variables—unless you specify option `noxconditional` or the `noxconditional` option is otherwise implied; see [SEM] **sem option noxconditional**. This is what you would desire in most cases.

By default, classes 3 and 4 are constrained to be equal and the rest are allowed to vary.

2. Thus you might expect that most of the parameters of our model would have been left unconstrained until you remember that we are fitting a measurement model. That is why `sem` listed 20 constraints at the top of the estimation results. Some of the constraints are substantive and some are normalization.

3. In the output, we have a separate table of parameter estimates for each level of `grade`.

   In our data, group 1 corresponds with students in grade 4, and group 2 corresponds with students in grade 5.

4. It may surprise you that the output contains estimates for the means of the latent variables. Usually, `sem` does not report this.

   Usually, you are running on only one group of data and those means cannot be estimated, at least not without additional identifying constraints. When you are running on two or more groups, the means for all the groups except one can be estimated.

In [SEM] **Example 21**, we use `estat ggof` to evaluate goodness of fit group by group.

In [SEM] **Example 22**, we use `estat ginvariant` to test whether parameters that are constrained across groups should not be and whether parameters that are not constrained could be.

In [SEM] **Example 23**, we show how to constrain the parameters we choose to be equal across groups.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/sem_2fmmby
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for relationships with peers.

   Select the Add measurement component tool, , and then click in the diagram about halfway down from the top and about one-third of the way in from the left.

   In the resulting dialog box,

   a. change the *Latent variable name* to `Peer`;

   b. select `peerrel1`, `peerrel2`, `peerrel3`, and `peerrel4` by using the *Measurement variables* control;

   c. select `Left` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

4. Create the measurement component for relationships with parents.

   Select the Add measurement component tool, , and then click in the diagram about halfway down from the top and about one-third of the way in from the right.

   In the resulting dialog box,

   a. change the *Latent variable name* to `Par`;

   b. select `parrel1`, `parrel2`, `parrel3`, and `parrel4` by using the *Measurement variables* control;

   c. select `Right` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Correlate the latent variables.

   a. Select the Add covariance tool, .

   b. Click in the upper-right quadrant of the `Peer` oval (it will highlight when you hover over it), and drag a covariance to the upper-left quadrant of the `Par` oval (it will highlight when you can release to connect the covariance).

6. Clean up.

   If you do not like where a covariance has been connected to its variable, use the Select tool, , to click on the covariance, and then simply click on where it connects to an oval and drag the endpoint. You can also change the bow of the covariance by dragging the control point that extends from one end of the selected covariance.

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar.

   In the resulting dialog box, do the following:

   a. Select the **Group** tab.

   b. Select the *Group analysis* radio button. The variable `grade` should appear in the *Group variable* control.

   c. Click on **OK**.

   d. In the Standard Toolbar, use the *Group* control to toggle between results for group 1 and group 2.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_2fmmby
```

Reference

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Also see

[SEM] **Example 3** — Two-factor measurement model

[SEM] **Example 19** — Creating multiple-group summary statistics data

[SEM] **Example 21** — Group-level goodness of fit

[SEM] **Example 22** — Testing parameter equality across groups

[SEM] **Example 23** — Specifying parameter constraints across groups

[SEM] **Intro 6** — Comparing groups

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem group options** — Fitting models on different groups

Title

Example 21 — Group-level goodness of fit

Description Remarks and examples Also see

Description

Below we demonstrate the `estat ggof` command, which may be used after `sem` with the `group()` option. `estat ggof` displays group-by-group goodness-of-fit statistics.

We pick up where [SEM] **Example 20** left off:

```
. use https://www.stata-press.com/data/r16/sem_2fmmby
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)   ///
      (Par  -> parrel1  parrel2  parrel3  parrel4), group(grade)
```

Remarks and examples

```
. estat ggof
Group-level fit statistics
```

	N	SRMR	CD
grade			
1	134	0.063	0.969
2	251	0.047	0.955

```
Note: Group-level chi-squared statistics
      are not reported because of
      constraints between groups.
```

Notes:

1. Reported are the goodness-of-fit tests that `estat gof, stats(residuals)` would report. The difference is that they are reported for each group rather than overall.
2. If the fit is good, then SRMR (standardized root mean squared residual) will be close to 0 and CD (the coefficient of determination) will be near 1.

It is also appropriate to run `estat gof` to obtain overall results:

```
. estat gof, stats(residuals)
```

Fit statistic	Value	Description
Size of residuals		
SRMR	0.056	Standardized root mean squared residual
CD	0.958	Coefficient of determination

Also see

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **Example 4** — Goodness-of-fit statistics

[SEM] **estat ggof** — Group-level goodness-of-fit statistics

[SEM] **estat gof** — Goodness-of-fit statistics

Title

Example 22 — Testing parameter equality across groups

Description Remarks and examples Also see

Description

Below we demonstrate `estat ginvariant` to test parameters across groups.

We pick up where [SEM] **Example 20** left off:

```
. use https://www.stata-press.com/data/r16/sem_2fmmby
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)   ///
      (Par  -> parrel1  parrel2  parrel3  parrel4), group(grade)
```

Remarks and examples

We use `estat ginvariant` to test whether parameters that are constrained to be equal across groups should not be and whether parameters that are not constrained across groups could be.

```
. estat ginvariant

Tests for group invariance of parameters
```

	Wald Test			Score Test		
	chi2	df	p>chi2	chi2	df	p>chi2
Measurement						
peerrel1						
Peer	.	.	.	2.480	1	0.1153
_cons	.	.	.	0.098	1	0.7537
peerrel2						
Peer	.	.	.	0.371	1	0.5424
_cons	.	.	.	0.104	1	0.7473
peerrel3						
Peer	.	.	.	2.004	1	0.1568
_cons	.	.	.	0.002	1	0.9687
peerrel4						
Peer	.	.	.	0.239	1	0.6246
_cons	.	.	.	0.002	1	0.9611
parrel1						
Par	.	.	.	0.272	1	0.6019
_cons	.	.	.	0.615	1	0.4329
parrel2						
Par	.	.	.	0.476	1	0.4903
_cons	.	.	.	3.277	1	0.0703
parrel3						
Par	.	.	.	3.199	1	0.0737
_cons	.	.	.	1.446	1	0.2291
parrel4						
Par	.	.	.	2.969	1	0.0849
_cons	.	.	.	0.397	1	0.5288
var(e.peer~1)	0.024	1	0.8772	.	.	.
var(e.peer~2)	0.033	1	0.8565	.	.	.
var(e.peer~3)	0.011	1	0.9152	.	.	.
var(e.peer~4)	0.294	1	0.5879	.	.	.
var(e.parr~1)	1.981	1	0.1593	.	.	.
var(e.parr~2)	14.190	1	0.0002	.	.	.
var(e.parr~3)	0.574	1	0.4486	.	.	.
var(e.parr~4)	0.022	1	0.8813	.	.	.
var(Peer)	4.583	1	0.0323	.	.	.
var(Par)	0.609	1	0.4350	.	.	.
cov(Peer,Par)	0.780	1	0.3772	.	.	.

Notes:

1. In the output above, score tests are reported for parameters that were constrained. The null hypothesis is that the constraint is valid. None of the tests reject a valid constraint.

2. Wald tests are reported for parameters that were not constrained. The null hypothesis is that a constraint would be valid. Only in two cases does it appear that grade 4 differs from grade 5, namely, the variance of `e.parrel2` and the variance of `Peer`.

3. We remind you that these tests are marginal tests. That is, each test is intended to be interpreted separately. These are not joint tests of simultaneous imposition or relaxation of constraints. If you want simultaneous tests, you must do them yourself by using, for instance, the `test` command.

   If joint tests of parameter classes are desired, the `class` option can be used.

These results imply that none of the constraints we impose should be relaxed, and that perhaps we could constrain all the variances and covariances to be equal across groups except for the variances of `e.parrel2` and `Peer`. We do that in [SEM] **Example 23**.

Also see

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **Example 23** — Specifying parameter constraints across groups

[SEM] **estat ginvariant** — Tests for invariance of parameters across groups

Title

Example 23 — Specifying parameter constraints across groups

Description Remarks and examples Also see

Description

Below we demonstrate how to constrain the parameters we want constrained to be equal across groups when using `sem` with the `group()` option.

We pick up where [SEM] **Example 22** left off:

```
. use https://www.stata-press.com/data/r16/sem_2fmmby
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)   ///
      (Par  -> parrel1  parrel2  parrel3  parrel4), group(grade)
. estat ginvariant
```

The `estat ginvariant` command implied that perhaps we could constrain all the variances and covariances to be equal across groups except for the variances of `e.parrel2` and `Peer`.

Remarks and examples

Remarks are presented under the following headings:

Background
Fitting the constrained model

Background

We can specify which parameters we wish to allow to vary. Remember that `sem`'s `group()` option classifies the parameters of the model as follows:

Class description	Class name	
1. structural coefficients	`scoef`	
2. structural intercepts	`scons`	
3. measurement coefficients	`mcoef`	
4. measurement intercepts	`mcons`	
5. covariances of structural errors	`serrvar`	
6. covariances of measurement errors	`merrvar`	
7. covariances between structural and measurement errors	`smerrcov`	
8. means of exogenous variables	`meanex`	(*)
9. covariances of exogenous variables	`covex`	(*)
10. all the above	`all`	(*)
11. none of the above	`none`	

(*) Exogenous variables means just the latent exogenous variables unless you specify `sem` option `noxconditional` or you specify option `method(mlmv)` (which implies option `noxconditional`); see [SEM] **sem option noxconditional**.

When fitting a model with the `group()` option,

```
. sem ..., ... group(varname)
```

you may also specify the `ginvariant()` option:

```
. sem ..., ... group(varname) ginvariant(class names)
```

You may specify any of the class names as being `ginvariant()`. You may specify as many class names as you wish. When you specify `ginvariant()`, `sem` cancels its default actions on which parameters vary and which do not, and uses the information you specify. All classes that you do not mention as being `ginvariant()` are allowed to vary across groups.

By using `ginvariant()`, you can constrain, or free by your silence, whole classes of parameters. For instance, you could type

```
. sem ..., group(mygroup) ginvariant(mcoef mcons serrvar)
```

and you are constraining those parameters to be equal across groups and leaving unconstrained `scoef`, `scons`, `merrvar`, `smerrcov`, `meanex`, and `covex`.

In addition, if a class is constrained, you can still unconstrain individual coefficients. Consider the model

```
. sem ... (x1<-L) ...
```

If you typed

```
. sem ... (1: x1<-L@a1) (2: x1<-L@a2) ..., group(mygroup) ginvariant(all)
```

then all estimated parameters would be the same across groups except for the path `x1<-L`, and it would be free to vary in groups 1 and 2.

By the same token, if a class is unconstrained, you can still constrain individual coefficients. If you typed

```
. sem ... (1: x1<-L@a) (2: x1<-L@a) ..., group(mygroup) ginvariant(none)
```

then you would leave unconstrained all parameters except the path `x1<-L`, and it would be constrained to be equal in groups 1 and 2.

This is all discussed in [SEM] **Intro 6**, including how to constrain and free variance and covariance parameters.

Fitting the constrained model

In our case, we wish to fit our model:

```
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)   ///
      (Par  -> parrel1  parrel2  parrel3  parrel4),   ///
                  group(grade)
```

We impose constraints on all parameters except the variances of `e.parrel2` and `Peer`. We can do that by typing

```
. sem (Peer -> peerrel1 peerrel2 peerrel3 peerrel4)
>     (Par  -> parrel1  parrel2  parrel3  parrel4),
>                   group(grade)
>                   ginvariant(all)
>                   byparm
>                   var(1: e.parrel2@v1)
>                   var(2: e.parrel2@v2)
>                   var(1: Peer@v3)
>                   var(2: Peer@v4)

Endogenous variables

Measurement:  peerrel1 peerrel2 peerrel3 peerrel4 parrel1 parrel2 parrel3
              parrel4

Exogenous variables

Latent:       Peer Par

Fitting target model:

Iteration 0:   log likelihood = -5560.9934
Iteration 1:   log likelihood = -5552.3122
Iteration 2:   log likelihood = -5549.5391
Iteration 3:   log likelihood = -5549.3528
Iteration 4:   log likelihood = -5549.3501
Iteration 5:   log likelihood = -5549.3501

Structural equation model                       Number of obs     =        385
Grouping variable  = grade                      Number of groups  =          2
Estimation method  = ml
Log likelihood     = -5549.3501

 ( 1)  [peerrel1]1bn.grade#c.Peer = 1
 ( 2)  [peerrel2]1bn.grade#c.Peer - [peerrel2]2.grade#c.Peer = 0
 ( 3)  [peerrel3]1bn.grade#c.Peer - [peerrel3]2.grade#c.Peer = 0
 ( 4)  [peerrel4]1bn.grade#c.Peer - [peerrel4]2.grade#c.Peer = 0
 ( 5)  [parrel1]1bn.grade#c.Par = 1
 ( 6)  [parrel2]1bn.grade#c.Par - [parrel2]2.grade#c.Par = 0
 ( 7)  [parrel3]1bn.grade#c.Par - [parrel3]2.grade#c.Par = 0
 ( 8)  [parrel4]1bn.grade#c.Par - [parrel4]2.grade#c.Par = 0
 ( 9)  [/]var(e.peerrel1)#1bn.grade - [/]var(e.peerrel1)#2.grade = 0
 (10)  [/]var(e.peerrel2)#1bn.grade - [/]var(e.peerrel2)#2.grade = 0
 (11)  [/]var(e.peerrel3)#1bn.grade - [/]var(e.peerrel3)#2.grade = 0
 (12)  [/]var(e.peerrel4)#1bn.grade - [/]var(e.peerrel4)#2.grade = 0
 (13)  [/]var(e.parrel1)#1bn.grade - [/]var(e.parrel1)#2.grade = 0
 (14)  [/]var(e.parrel3)#1bn.grade - [/]var(e.parrel3)#2.grade = 0
 (15)  [/]var(e.parrel4)#1bn.grade - [/]var(e.parrel4)#2.grade = 0
 (16)  [/]cov(Peer,Par)#1bn.grade - [/]cov(Peer,Par)#2.grade = 0
 (17)  [/]var(Par)#1bn.grade - [/]var(Par)#2.grade = 0
 (18)  [peerrel1]1bn.grade - [peerrel1]2.grade = 0
 (19)  [peerrel2]1bn.grade - [peerrel2]2.grade = 0
 (20)  [peerrel3]1bn.grade - [peerrel3]2.grade = 0
 (21)  [peerrel4]1bn.grade - [peerrel4]2.grade = 0
 (22)  [parrel1]1bn.grade - [parrel1]2.grade = 0
 (23)  [parrel2]1bn.grade - [parrel2]2.grade = 0
 (24)  [parrel3]1bn.grade - [parrel3]2.grade = 0
 (25)  [parrel4]1bn.grade - [parrel4]2.grade = 0
 (26)  [peerrel1]2.grade#c.Peer = 1
 (27)  [parrel1]2.grade#c.Par = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
peerrel1						
Peer						
[*]	1	(constrained)				
_cons						
[*]	8.708274	.0935844	93.05	0.000	8.524852	8.891696
peerrel2						
Peer						
[*]	1.112225	.0973506	11.42	0.000	.9214217	1.303029
_cons						
[*]	7.858713	.1035989	75.86	0.000	7.655663	8.061763
peerrel3						
Peer						
[*]	1.416486	.113489	12.48	0.000	1.194052	1.638921
_cons						
[*]	7.398217	.1147474	64.47	0.000	7.173316	7.623118
peerrel4						
Peer						
[*]	1.196494	.0976052	12.26	0.000	1.005191	1.387796
_cons						
[*]	8.183148	.1021513	80.11	0.000	7.982936	8.383361
parrel1						
Par						
[*]	1	(constrained)				
_cons						
[*]	9.339558	.0648742	143.96	0.000	9.212407	9.46671
parrel2						
Par						
[*]	1.100315	.1362999	8.07	0.000	.8331722	1.367458
_cons						
[*]	9.255299	.0725417	127.59	0.000	9.11312	9.397478
parrel3						
Par						
[*]	2.051278	.2066714	9.93	0.000	1.64621	2.456347
_cons						
[*]	8.676961	.088927	97.57	0.000	8.502667	8.851255
parrel4						
Par						
[*]	1.529938	.154971	9.87	0.000	1.2262	1.833675
_cons						
[*]	9.045247	.0722358	125.22	0.000	8.903667	9.186826

```
------------------------------------------------------------------------------
var(e.peer~1)|
         [*] |   1.799133   .159059                      1.512898    2.139523
var(e.peer~2)|
         [*] |   2.186953   .193911                       1.838086    2.602035
var(e.peer~3)|
         [*] |   1.915661   .2129913                      1.54056     2.382094
var(e.peer~4)|
         [*] |   1.767354   .1746104                      1.45622     2.144965
var(e.parr~1)|
         [*] |   1.125082   .0901338                      .9615942    1.316366
var(e.parr~2)|
           1 |   .9603043    .13383                       .730775     1.261927
           2 |   1.799668   .1747351                      1.487807    2.176898
var(e.parr~3)|
         [*] |   .9606889   .1420406                      .7190021    1.283617
var(e.parr~4)|
         [*] |   .8496935   .0933448                      .6850966    1.053835
    var(Peer)|
           1 |   1.951555   .3387796                      1.388727    2.742489
           2 |   1.361431   .2122853                      1.002927    1.848084
     var(Par)|
         [*] |   .4952527   .0927994                      .3430288    .7150281
-------------+----------------------------------------------------------------
cov(Peer,Par)|
         [*] |   .4096197   .0708726     5.78   0.000     .2707118    .5485275
------------------------------------------------------------------------------
Note: [*] identifies parameter estimates constrained to be equal across
      groups.
LR test of model vs. saturated: chi2(61)  =     75.25, Prob > chi2 = 0.1037
```

Notes:

1. In [SEM] **Example 20**, we previously fit this model by typing

   ```
   . sem (...) (...), group(grade)
   ```

   This time, we typed

   ```
   . sem (...) (...), group(grade)         ///
                      ginvariant(all)      ///
                      byparm               ///
                      var(1: e.parrel2@v1) ///
                      var(2: e.parrel2@v2) ///
                      var(1: Peer@v3)      ///
                      var(2: Peer@v4)
   ```

2. We specified the `byparm` option so that the results are sorted by parameters rather than groups. With this option, `sem` will report a single value for a parameter estimate if it is constrained to be equal across the groups. Because most of the parameters are constrained to be equal across groups, the output is much shorter than the default output. The default output produces a separate table for each group.

3. Previously, `sem, group()` mentioned 20 constraints that it imposed because of normalization or because of assumed `ginvariant(mcoef mcons)`.

   This time, `sem, group()` mentioned 27 constraints. It applied more constraints because we specified `ginvariant(all)`.

4. After the `ginvariant(all)` option, we relaxed the following constraints:

```
var(1: e.parrel2@v1)
var(2: e.parrel2@v2)
var(1: Peer@v3)
var(2: Peer@v4)
```

`ginvariant(all)` specified, among other constraints, that

```
var(1: e.parrel2) == var(2: e.parrel2)
var(1: Peer) == var(2: Peer)
```

`ginvariant(all)` did that by secretly issuing the options

```
var(1: e.parrel2@secretname1)
var(2: e.parrel2@secretname1)
var(1: Peer@secretname2)
var(2: Peer@secretname2)
```

because that is how you impose equality constraints with the path notation. When we specified

```
var(1: e.parrel2@v1)
var(2: e.parrel2@v2)
var(1: Peer@v3)
var(2: Peer@v4)
```

our new constraints overrode the secretly issued constraints. It would not have worked to leave off the symbolic names; see [SEM] **sem path notation extensions**. We specified the symbolic names `v1`, `v2`, `v3`, and `v4`. `v1` and `v2` overrode *secretname1*, and thus the constraint that `var(e.parrel2)` be equal across the two groups was relaxed. `v3` and `v4` overrode *secretname2*, and thus the constraint that `var(Peer)` be equal across groups was relaxed.

Also see

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **Example 22** — Testing parameter equality across groups

[SEM] **Intro 6** — Comparing groups

[SEM] **sem group options** — Fitting models on different groups

Title

Example 24 — Reliability

Description Remarks and examples Also see

Description

Below we demonstrate `sem`'s `reliability()` option with the following data:

```
. use https://www.stata-press.com/data/r16/sem_rel
(measurement error with known reliabilities)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
y	1,234	701.081	71.79378	487	943
x1	1,234	100.278	14.1552	51	149
x2	1,234	100.2066	14.50912	55	150

```
. notes
_dta:
  1.  Fictional data.
  2.  Variables x1 and x2 each contain a test score designed to measure X.  The
      test is scored to have mean 100.
  3.  Variables x1 and x2 are both known to have reliability 0.5.
  4.  Variable y is the outcome, believed to be related to X.
```

See [SEM] **sem and gsem option reliability( )** for background.

Remarks and examples

Remarks are presented under the following headings:

Baseline model (reliability ignored)
Model with reliability
Model with two measurement variables and reliability

Baseline model (reliability ignored)

```
. sem (y <- x1)
Endogenous variables
Observed:  y
Exogenous variables
Observed:  x1
Fitting target model:
Iteration 0:   log likelihood = -11629.745
Iteration 1:   log likelihood = -11629.745
Structural equation model                       Number of obs     =      1,234
Estimation method  = ml
Log likelihood     = -11629.745
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
y						
x1	3.54976	.1031254	34.42	0.000	3.347637	3.751882
_cons	345.1184	10.44365	33.05	0.000	324.6492	365.5876
var(e.y)	2627.401	105.7752			2428.053	2843.115

```
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

Notes:

1. In these data, variable `x1` is measured with error.
2. If we ignore that, we obtain a path coefficient for `y<-x1` of 3.55.
3. We also ran this model for `y<-x2`. We obtained a path coefficient of 3.48.

Model with reliability

```
. sem (x1<-X) (y<-X), reliability(x1 .5)
Endogenous variables
Measurement:  x1 y
Exogenous variables
Latent:       X
Fitting target model:
Iteration 0:   log likelihood = -11745.845
Iteration 1:   log likelihood = -11661.626
Iteration 2:   log likelihood = -11631.469
Iteration 3:   log likelihood = -11629.755
Iteration 4:   log likelihood = -11629.745
Iteration 5:   log likelihood = -11629.745
Structural equation model                       Number of obs     =      1,234
Estimation method  = ml
Log likelihood     = -11629.745
 ( 1)  [x1]X = 1
 ( 2)  [/]var(e.x1) = 100.1036
```

	OIM				
	Coef.	Std. Err.	z	P>|z|	[95% Conf. Interval]
Measurement					
x1					
X	1	(constrained)			
_cons	100.278	.4027933	248.96	0.000	99.4885 101.0674
y					
X	7.09952	.352463	20.14	0.000	6.408705 7.790335
_cons	701.081	2.042929	343.17	0.000	697.077 705.0851
var(e.x1)	100.1036	(constrained)			
var(e.y)	104.631	207.3381			2.152334 5086.411
var(X)	100.1036	8.060038			85.48963 117.2157

```
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

Notes:

1. We wish to estimate the effect of `y<-x1` when `x1` is measured with error (0.50 reliability). To do that, we introduce latent variable `X` and write our model as `(x1<-X) (y<-X)`.
2. When we ignored the measurement error of `x1`, we obtained a path coefficient for `y<-x1` of 3.55. Taking into account the measurement error, we obtain a coefficient of 7.1.

Model with two measurement variables and reliability

```
. sem (x1 x2<-X) (y<-X), reliability(x1 .5  x2 .5)
Endogenous variables
Measurement:  x1 x2 y
Exogenous variables
Latent:       X
Fitting target model:
Iteration 0:   log likelihood = -16258.636
Iteration 1:   log likelihood = -16258.401
Iteration 2:   log likelihood =   -16258.4
Structural equation model                       Number of obs     =      1,234
Estimation method  = ml
Log likelihood     =   -16258.4
 ( 1)  [x1]X = 1
 ( 2)  [/]var(e.x1) = 100.1036
 ( 3)  [/]var(e.x2) = 105.1719
```

	OIM					
	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
x1						
X	1	(constrained)				
_cons	100.278	.4037851	248.34	0.000	99.48655	101.0694
x2						
X	1.030101	.0417346	24.68	0.000	.9483029	1.1119
_cons	100.2066	.4149165	241.51	0.000	99.39342	101.0199
y						
X	7.031299	.2484176	28.30	0.000	6.544409	7.518188
_cons	701.081	2.042928	343.17	0.000	697.077	705.0851
var(e.x1)	100.1036	(constrained)				
var(e.x2)	105.1719	(constrained)				
var(e.y)	152.329	105.26			39.31868	590.1553
var(X)	101.0907	7.343656			87.67509	116.5591

```
LR test of model vs. saturated: chi2(2)   =      0.59, Prob > chi2 = 0.7430
```

Notes:

1. We wish to estimate the effect of `y<-X`. We have two measures of `X`—`x1` and `x2`—both measured with error (0.50 reliability).
2. In the previous section, we used just `x1`. We obtained path coefficient 7.1 with standard error 0.4. Using both `x1` and `x2`, we obtain path coefficient 7.0 and standard error 0.2.
3. We at StataCorp created these fictional data. The true coefficient is 7.

Also see

[SEM] **sem and gsem option reliability( )** — Fraction of variance not due to measurement error

[SEM] **Example 1** — Single-factor measurement model

Title

Example 25 — Creating summary statistics data from raw data

Description Remarks and examples Also see

Description

Below we show how to create summary statistics data (SSD) from raw data. We use `auto2.dta`:

```
. use https://www.stata-press.com/data/r16/auto2
(1978 Automobile Data)
. describe
  (output omitted)
. summarize
  (output omitted)
```

Remarks and examples

Remarks are presented under the following headings:

Preparing data for conversion
Converting to summary statistics form
Publishing SSD
Creating SSD with multiple groups

We are going to create SSD containing the variables `price`, `mpg`, `weight`, `displacement`, and `foreign`.

Preparing data for conversion

Before building the SSD, prepare the data to be converted:

1. Drop variables that you do not intend to include in the SSD. Dropping variables is not a requirement, but it will be easier to spot problems if you begin by eliminating the irrelevant variables.
2. Verify that you have no string variables in the resulting data. Summary statistics datasets cannot contain string values.
3. Verify that there are no missing values. If there are, be aware that observations containing one or more variables with missing values will be omitted from the SSD.
4. Verify that all variables are on a reasonable scale. We recommend that the means of variables be only 3 or 4 orders of magnitude different from each other. This will help to preserve numerical accuracy when the SSD are used.
5. Create any new variables containing transformations of existing variables that might be useful later. Once the data are converted to summary statistics form, you will not be able to create such variables.
6. Place the variables in a logical order. That will help the user of the SSD understand the data.
7. Save the resulting prepared data. Probably you will never need the prepared data, but one never knows for sure.

We take our own advice below:

```
. * ------------------------------------------------------------------------------
. * Suggestion 1:  Keep relevant variables:
. *
. keep price mpg weight displacement foreign

.
. * ------------------------------------------------------------------------------
. * Suggestion 2: Check for string variables
. * Suggestion 3: Verify no missing values
. * Suggestion 4: Verify variables on a reasonable scale:
. *
. summarize
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41
      weight |         74    3019.459    777.1936       1760       4840
displacement |         74    197.2973    91.83722         79        425
     foreign |         74    .2972973    .4601885          0          1

.
. * We will rescale weight and price:
. replace   weight = weight/1000
variable weight was int now float
(74 real changes made)
. replace   price  =  price/1000
variable price was int now float
(74 real changes made)
. label var weight "Weight (1000s lbs.)"
. label var price  "Price  ($1,000s)"
. * and now we check our work:
. *
. summarize
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6.165257    2.949496     3.291     15.906
         mpg |         74     21.2973    5.785503         12         41
      weight |         74    3.019459    .7771936      1.76       4.84
displacement |         74    197.2973    91.83722         79        425
     foreign |         74    .2972973    .4601885          0          1

.
. * ------------------------------------------------------------------------------
. * Suggestion 5: Create useful transformations:
. *
. generate gpm    = 1/mpg
. label var gpm    "Gallons per mile"
.
```

```
. * ------------------------------------------------------------------------
. * Suggestion 6: Place variables in logical order:
. *
. order price mpg gpm
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	74	6.165257	2.949496	3.291	15.906
mpg	74	21.2973	5.785503	12	41
gpm	74	.0501928	.0127986	.0243902	.0833333
weight	74	3.019459	.7771936	1.76	4.84
displacement	74	197.2973	91.83722	79	425
foreign	74	.2972973	.4601885	0	1

```
.
. * ------------------------------------------------------------------------
. * Suggestion 7: save prepared data
. *
. save auto_raw
file auto_raw.dta saved
. * ------------------------------------------------------------------------
```

Converting to summary statistics form

To create the summary statistics dataset, you just need to type `sud build` and the names of the variables to be included. If you have previously kept the relevant variables, you can type `ssd build _all`.

We recommend the following steps:

1. Convert data to summary statistics form:

```
. ssd build _all
```

2. Review the result:

```
. ssd describe
. notes
. ssd list
```

3. Digitally sign the data:

```
. datasignature set
```

4. Save the data:

```
. save auto_ss
```

We follow our advice below. After that, we will show you the advantages of digitally signing the data.

```
. * ------------------------------------------------------------------------
. * Convert data:
. *
. ssd build _all
 (data in memory now summary statistics data; you can use ssd describe and
  ssd list to describe and list results.)
```

```
. * ------------------------------------------------------------------------
. * Review results:
. *
. ssd describe
  Summary statistics data
    obs:             74
   vars:              6
                                        (_dta has notes)
  ------------------------------------------------------------------------
  variable name                 variable label
  ------------------------------------------------------------------------
  price                         Price  ($1,000s)
  mpg                           Mileage (mpg)
  gpm                           Gallons per mile
  weight                        Weight (1000s lbs.)
  displacement                  Displacement (cu. in.)
  foreign                       Car type
  ------------------------------------------------------------------------

. notes
_dta:
  1.  summary statistics data built from 'auto_raw.dta' on 25 Feb 2019 17:30:30
      using -ssd build _all-
. ssd list
  Observations = 74
  Means:
         price           mpg           gpm        weight  displacement
     6.1652567     21.297297      .0501928     3.0194595      197.2973
       foreign
      .2972973
  Variances implicitly defined; they are the diagonal of the covariance
  matrix.
  Covariances:
         price           mpg           gpm        weight  displacement
     8.6995258
    -7.9962828     33.472047
     .02178417    -.06991586      .0001638
     1.2346748    -3.6294262     .00849897     .60402985
     134.06705    -374.92521     .90648519      63.87345     8434.0748
     .06612809     1.0473899    -.00212897    -.21202888    -25.938912
       foreign
     .21177342
. * ------------------------------------------------------------------------
. * Digitally sign:
. *
. datasignature set
  8:8(102846):1914186416:2867097560       (data signature set)
. * ------------------------------------------------------------------------
. * Save:
. *
. save auto_ss
file auto_ss.dta saved
. * ------------------------------------------------------------------------
```

We recommend digitally signing the data. This way, anyone can verify later that the data are unchanged:

```
. datasignature confirm
  (data unchanged since 29may2018 15:18)
```

Let us show you what would happen if the data had changed:

```
. replace mpg = mpg+.0001 in 5
(1 real change made)
. datasignature confirm
  data have changed since 29may2018 15:18
r(9);
```

There is no reason for you or anyone else to change the SSD after it has been created, so we recommend that you digitally sign the data. With regular datasets, users do make changes, if only by adding variables.

Be aware that the data signature is a function of the variable names, so if you rename a variable—something you are allowed to do—the signature will change and `datasignature` will report, for example, "data have changed since 29may2018 15:18". Solutions to that problem are discussed in [SEM] **ssd**.

Publishing SSD

The summary statistics dataset you have just created can obviously be sent to and used by any Stata user. If you wish to publish your data in printed form, use `ssd describe` and `ssd list` to describe and list the data.

Creating SSD with multiple groups

The process for creating SSD containing multiple groups is nearly the same as for creating single-group data. The only differences are that you do not drop the group variable during preparation and that rather than typing

```
. ssd build _all
```

you type

```
. ssd build _all, group(varname)
```

Below we build the automobile SSD again, but this time, we specify `group(rep78)`:

```
. ssd build _all, group(rep78)
```

If you think carefully about this, you may be worried that `_all` includes `rep78` and thus we will be including the grouping variable among the summary statistics. `ssd build` knows to omit the group variable:

```
. * ------------------------------------------------------------------------
. * Suggestion 1:  Keep relevant variables:
. *
. webuse auto2, clear
(1978 Automobile Data)
. keep price mpg weight displacement foreign rep78
```

```
. * ------------------------------------------------------------------------
. * Suggestion 2: Check for string variables
. * Suggestion 3: Verify no missing values
. * Suggestion 4: Verify variables on a reasonable scale:
. *
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	74	6165.257	2949.496	3291	15906
mpg	74	21.2973	5.785503	12	41
rep78	69	3.405797	.9899323	1	5
weight	74	3019.459	777.1936	1760	4840
displacement	74	197.2973	91.83722	79	425
foreign	74	.2972973	.4601885	0	1

```
. drop if rep78 >= .
(5 observations deleted)

. * We will rescale weight and price:
. replace   weight = weight/1000
variable weight was int now float
(69 real changes made)

. replace   price  =  price/1000
variable price was int now float
(69 real changes made)

. label var weight "Weight (1000s lbs.)"

. label var price  "Price  ($1,000s)"

. * and now we check our work:
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	69	6.146043	2.91244	3.291	15.906
mpg	69	21.28986	5.866408	12	41
rep78	69	3.405797	.9899323	1	5
weight	69	3.032029	.7928515	1.76	4.84
displacement	69	198	93.14789	79	425
foreign	69	.3043478	.4635016	0	1

```
. * ------------------------------------------------------------------------
. * Suggestion 5: Create useful transformations:
. *
. generate gpm    = 1/mpg

. label var gpm    "Gallons per mile"

. * ------------------------------------------------------------------------
. * Suggestion 6: Place variables in logical order:
. *
. order price mpg gpm

. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	69	6.146043	2.91244	3.291	15.906
mpg	69	21.28986	5.866408	12	41
gpm	69	.0502584	.0128353	.0243902	.0833333
rep78	69	3.405797	.9899323	1	5
weight	69	3.032029	.7928515	1.76	4.84
displacement	69	198	93.14789	79	425
foreign	69	.3043478	.4635016	0	1

```
. * ------------------------------------------------------------------------------
. * Suggestion 7: save prepared data
. *
. save auto_group_raw
file auto_group_raw.dta saved
. * ------------------------------------------------------------------------------
. * ------------------------------------------------------------------------------
. * Convert data:
. *
. ssd build _all, group(rep78)
 (data in memory now summary statistics data; you can use ssd describe and
  ssd list to describe and list results.)
. * ------------------------------------------------------------------------------
. * Review results:
. *
. ssd describe
  Summary statistics data
    obs:              69
   vars:               6
                                        (_dta has notes)
  ------------------------------------------------------------------------------
  variable name                  variable label
  ------------------------------------------------------------------------------
  price                          Price  ($1,000s)
  mpg                            Mileage (mpg)
  gpm                            Gallons per mile
  weight                         Weight (1000s lbs.)
  displacement                   Displacement (cu. in.)
  foreign                        Car type
  ------------------------------------------------------------------------------
  Group variable:  rep78  (5 groups)
   Obs. by group:  2, 8, 30, 18, 11
. notes
_dta:
  1.  summary statistics data built from 'auto_group_raw.dta' on 25 Feb 2019
      17:30:30 using -ssd build _all, group(rep78)-
. ssd list
-------------------------------------------------------------------------------
Group rep78==Poor:
  (output omitted)
-------------------------------------------------------------------------------
Group rep78==Fair:
  (output omitted)
-------------------------------------------------------------------------------
Group rep78==Average:
  (output omitted)
-------------------------------------------------------------------------------
Group rep78==Good:
  (output omitted)
-------------------------------------------------------------------------------
Group rep78==Excellent:
  Observations = 11
  Means:
         price           mpg           gpm        weight  displacement
         5.913     27.363636     .04048131     2.3227273     111.09091
       foreign
     .81818182
```

```
  Variances implicitly defined; they are the diagonal of the covariance
  matrix.
  Covariances:
         price           mpg           gpm        weight  displacement
     6.8422143
    -15.608899      76.254545
     .02750797      -.1184875      .00019114
       .956802     -3.0610912      .00510833     .16856184
     55.493298     -201.03636      .33150758     9.9577283     648.09091
     .34169998     -.92727273      .00182175     .07254547     3.8181818
       foreign
     .16363636
```

```
. * ------------------------------------------------------------------------
. * Digitally sign:
. *
. datasignature set
  40:8(34334):2679920516:3596756814       (data signature set)
. * ------------------------------------------------------------------------
. * Save:
. *
. save auto_group_ss
file auto_group_ss.dta saved
. * ------------------------------------------------------------------------
```

Also see

[SEM] **ssd** — Making summary statistics data (sem only)

Title

Example 26 — Fitting a model with data missing at random

Description Remarks and examples Also see

Description

`sem method(mlmv)` is demonstrated using

```
. use https://www.stata-press.com/data/r16/cfa_missing
(CFA MAR data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
id	500	250.5	144.4818	1	500
test1	406	97.37475	13.91442	56.0406	136.5672
test2	413	98.04501	13.84145	62.25496	129.3881
test3	443	100.9699	13.4862	65.51753	137.3046
test4	417	99.56815	14.25438	53.8719	153.9779
taken	500	3.358	.6593219	2	4

```
. notes
_dta:
  1.  Fictional data on 500 subjects taking four tests.
  2.  Tests results M.A.R. (missing at random).
  3.  230 took all 4 tests
  4.  219 took 3 of the 4 tests
  5.  51 took 2 of the 4 tests
  6.  All tests have expected mean 100, s.d. 14.
```

See [SEM] **Intro 4** for background.

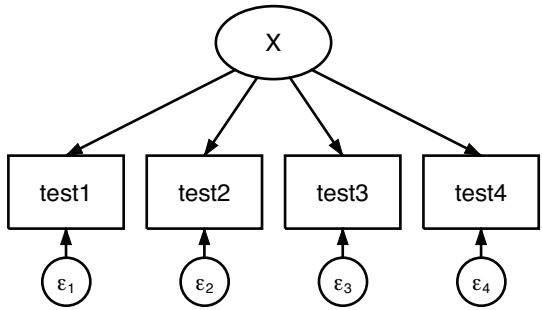
Remarks and examples

Remarks are presented under the following headings:

Fitting the model with method(ml)
Fitting the model with method(mlmv)
Fitting the model with the Builder

Fitting the model with method(ml)

We fit a single-factor measurement model.




```
. sem (test1 test2 test3 test4 <- X), nolog
(270 observations with missing values excluded)
Endogenous variables
Measurement:  test1 test2 test3 test4
Exogenous variables
Latent:       X
Structural equation model                       Number of obs     =        230
Estimation method  = ml
Log likelihood     = -3464.3099
 ( 1)  [test1]X = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
test1						
X	1	(constrained)				
_cons	96.76907	.8134878	118.96	0.000	95.17467	98.36348
test2						
X	1.021885	.1183745	8.63	0.000	.789875	1.253895
_cons	92.41248	.8405189	109.95	0.000	90.7651	94.05987
test3						
X	.5084673	.0814191	6.25	0.000	.3488889	.6680457
_cons	94.12958	.7039862	133.71	0.000	92.7498	95.50937
test4						
X	.5585651	.0857772	6.51	0.000	.3904449	.7266853
_cons	92.2556	.7322511	125.99	0.000	90.82042	93.69079
var(e.test1)	55.86083	10.85681			38.16563	81.76028
var(e.test2)	61.88092	11.50377			42.985	89.08338
var(e.test3)	89.07839	8.962574			73.13566	108.4965
var(e.test4)	93.26508	9.504276			76.37945	113.8837
var(X)	96.34453	16.28034			69.18161	134.1725

```
LR test of model vs. saturated: chi2(2)   =      0.39, Prob > chi2 = 0.8212
```

Notes:

1. This model was fit using 230 of the 500 observations in the dataset. Unless you use `sem`'s `method(mlmv)`, observations are casewise omitted, meaning that if there is a single variable with a missing value among the variables being used, the observation is ignored.
2. The coefficients for `test3` and `test4` are 0.51 and 0.56. Because we at StataCorp manufactured these data, we can tell you that the true coefficients are 1.
3. The error variance for `e.test1` and `e.test2` are understated. These data were manufactured with an error variance of 100.
4. These data are missing at random (MAR), not missing completely at random (MCAR). In MAR data, which values are missing can be a function of the observed values in the data. MAR data can produce biased estimates if the missingness is ignored, as we just did. MCAR data do not bias estimates.

Fitting the model with method(mlmv)

```
. sem (test1 test2 test3 test4 <- X), method(mlmv) nolog
Endogenous variables
Measurement:  test1 test2 test3 test4
Exogenous variables
Latent:       X
  (output omitted)
Structural equation model                       Number of obs     =        500
Estimation method  = mlmv
Log likelihood     = -6592.9961
 ( 1)  [test1]X = 1
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
test1						
X	1	(constrained)				
_cons	98.94386	.6814418	145.20	0.000	97.60826	100.2795
test2						
X	1.069952	.1079173	9.91	0.000	.8584378	1.281466
_cons	99.84218	.6911295	144.46	0.000	98.48759	101.1968
test3						
X	.9489025	.0896098	10.59	0.000	.7732706	1.124534
_cons	101.0655	.6256275	161.54	0.000	99.83928	102.2917
test4						
X	1.021626	.0958982	10.65	0.000	.8336687	1.209583
_cons	99.64509	.6730054	148.06	0.000	98.32603	100.9642
var(e.test1)	101.1135	10.1898			82.99057	123.1941
var(e.test2)	95.45572	10.79485			76.47892	119.1413
var(e.test3)	95.14847	9.053014			78.9611	114.6543
var(e.test4)	101.0943	10.0969			83.12124	122.9536
var(X)	94.04629	13.96734			70.29508	125.8225

```
LR test of model vs. saturated: chi2(2)   =      2.27, Prob > chi2 = 0.3209
```

Notes:

1. The model is now fit using all 500 observations in the dataset.
2. The coefficients for `test3` and `test4`—previously 0.51 and 0.56—are now 0.95 and 1.02.
3. Error variance estimates are now consistent with the true value of 100.
4. Standard errors of path coefficients are mostly smaller than reported in the previous model.
5. `method(mlmv)` requires that the data be MCAR or MAR.
6. `method(mlmv)` requires that the data be multivariate normal.

Fitting the model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/cfa_missing
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for `X`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and about halfway in from the left.

   In the resulting dialog box,

   a. change the *Latent variable name* to `X`;
   b. select `test1`, `test2`, `test3`, and `test4` by using the *Measurement variables* control;
   c. select `Down` in the *Measurement direction* control;
   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

4. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar. In the resulting dialog box,

   a. select the **Model** tab;
   b. select the *Maximum likelihood with missing values* radio button;
   c. click on **OK**.

You can open a completed diagram in the Builder by typing

```
. webgetsem cfa_missing
```

Also see

[SEM] **Intro 4** — Substantive concepts

[SEM] **sem option method()** — Specifying method and calculation of VCE

Title

Example 27g — Single-factor measurement model (generalized response)

Description Remarks and examples Also see

Description

This is the first example in the *g* series. The *g* means that the example focuses exclusively on the `gsem` command. If you are interested primarily in standard linear SEMs, you may want to skip the remaining examples. If you are especially interested in generalized SEMs, we suggest you read the remaining examples in order.

`gsem` provides three features not provided by `sem`: the ability to fit SEMs containing generalized linear response variables, the ability to fit multilevel mixed SEMs, and the ability to fit models with categorical latent variables.

Generalized response variables means that the response variables can be specifications from the generalized linear model (GLM). These include probit, logistic regression, ordered probit and logistic regression, multinomial logistic regression, and more. We use generalized linear response variables in this example.

Multilevel mixed models refer to the simultaneous handling of group-level effects, which can be nested or crossed. Thus you can include unobserved and observed effects for subjects, subjects within group, group within subgroup, . . . , or for subjects, group, subgroup, ... See [SEM] **Example 30g** for an example of a multilevel model.

Categorical latent variables are latent variables with categories that correspond to groups in the population. The categories are called classes, and we do not observe which individuals belong to which class. Instead, we estimate the probability of being in each class. `gsem` does not allow categorical latent variables and continuous latent variables (observation level or multilevel) to be present in the same model. See [SEM] **Example 50g** for an example with a categorical latent variable.

Below we demonstrate a single-factor measure model with pass/fail (binary outcome) responses rather than continuous responses. We use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_1fmm
(single-factor pass/fail measurement model)
. summarize
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
          x1 |        123    .4065041    .4931897         0          1
          x2 |        123    .4065041    .4931897         0          1
          x3 |        123    .4227642    .4960191         0          1
          x4 |        123    .3495935    .4787919         0          1
          s4 |        123    690.9837    77.50737       481        885
. notes
_dta:
  1.  Fictional data.
  2.  The variables x1, x2, x3, and x4 record 1=pass, 0=fail.
  3.  Pass/fail for x1, x2, x3: score > 100
  4.  Pass/fail for x4: score > 725
  5.  Variable s4 contains actual score for test 4.
```

See *Single-factor measurement models* in [SEM] **Intro 5** for background.

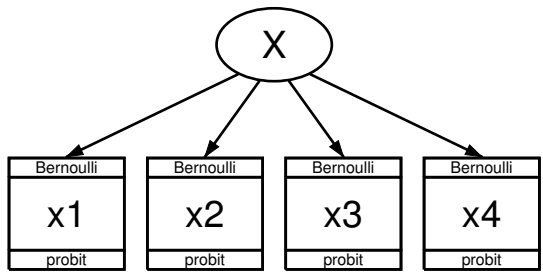
Remarks and examples

Remarks are presented under the following headings:

Single-factor pass/fail measurement model
Single-factor pass/fail + continuous measurement model
Fitting the model with the Builder

Single-factor pass/fail measurement model

Below we fit the following model:



The measurement variables we have (`x1`, ..., `x4`) are not continuous. They are pass/fail, coded as 1 (pass) and 0 (fail). To account for that, we use probit (also known as family Bernoulli, link probit). The equations for this model are

$$\Pr(x_1 = 1) = \Phi(\alpha_1 + X\beta_1)$$

$$\Pr(x_2 = 1) = \Phi(\alpha_2 + X\beta_2)$$

$$\Pr(x_3 = 1) = \Phi(\alpha_3 + X\beta_3)$$

$$\Pr(x_4 = 1) = \Phi(\alpha_4 + X\beta_4)$$

where $\Phi(\cdot)$ is the $N(0,1)$ cumulative distribution.

One way to think about this is to imagine a test that is scored on a continuous scale. Let's imagine the scores were s_1, s_2, s_3, and s_4 and distributed $N(\mu_i, \sigma_i^2)$, as test scores often are. Let's further imagine that for each test, a cutoff c_i is chosen and the student passes the test if $s_i \geq c_i$.

If we had the test scores, we would fit this as a linear model. We would posit

$$s_i = \gamma_i + X\delta_i + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma_i^2)$. However, we do not have test scores in our data; we have only the pass/fail results $x_i = 1$ if $s_i > c_i$.

So let's consider the pass/fail problem. The probability that a student passes test i is determined by the probability that the student scores above the cutoff:

$$\begin{aligned}
\Pr(s_i > c_i) &= \Pr(\gamma_i + X\delta_i + \epsilon_i > c_i) \\
&= \Pr\{\, \epsilon_i > c_i - (\gamma_i + X\delta_i) \,\} \\
&= \Pr\{\, -\epsilon_i \le -c_i + (\gamma_i + X\delta_i) \,\} \\
&= \Pr\{\, -\epsilon_i \le (\gamma_i - c_i) + X\delta_i \,\} \\
&= \Pr\{\, -\epsilon_i/\sigma_i \le (\gamma_i - c_i)/\sigma_i + X\delta_i/\sigma_i \} \\
&= \Phi\{\, (\gamma_i - c_i)/\sigma_i + X\delta_i/\sigma_i \,\}
\end{aligned}$$

The last equation is the probit model. In fact, we just derived the probit model, and now we know the relationship between the parameters we will be able to estimate with our pass/fail data: α_i and β_i. We also now know the parameters we could have estimated if we had the continuous test scores: γ_i and δ_i. The relationship is

$$\begin{aligned}
\alpha_i &= (\gamma_i - c_i)/\sigma_i \\
\beta_i &= \delta_i/\sigma_i
\end{aligned}$$

Notice that the right-hand sides of both equations are divided by σ_i, the standard deviation of the error term from the linear model for the ith test score. In pass/fail data, we lose the original scale of the score, and the slope coefficient we will be able to estimate is the slope coefficient from the linear model divided by the standard deviation of the error term. Meanwhile, the intercept we will be able to estimate is just the difference of the continuous model's intercept and the cutoff for passing the test, sans scale.

The command to fit the model and the results are

```
. gsem (x1 x2 x3 x4 <-X), probit

Fitting fixed-effects model:

Iteration 0:   log likelihood = -329.82091
Iteration 1:   log likelihood = -329.57665
Iteration 2:   log likelihood = -329.57664

Refining starting values:

Grid node 0:   log likelihood = -273.75437

Fitting full model:

Iteration 0:   log likelihood = -273.75437
Iteration 1:   log likelihood =  -264.3035
Iteration 2:   log likelihood = -263.37815
Iteration 3:   log likelihood =   -262.305
Iteration 4:   log likelihood = -261.69025
Iteration 5:   log likelihood = -261.42132
Iteration 6:   log likelihood = -261.35508
Iteration 7:   log likelihood =  -261.3224
Iteration 8:   log likelihood =  -261.3133
Iteration 9:   log likelihood = -261.30783
Iteration 10:  log likelihood = -261.30535
Iteration 11:  log likelihood = -261.30405
Iteration 12:  log likelihood = -261.30337
Iteration 13:  log likelihood = -261.30302
Iteration 14:  log likelihood = -261.30283
Iteration 15:  log likelihood = -261.30272
Iteration 16:  log likelihood = -261.30267
Iteration 17:  log likelihood = -261.30264
Iteration 18:  log likelihood = -261.30263
```

```
Generalized structural equation model           Number of obs     =        123
Response       : x1
Family         : Bernoulli
Link           : probit
Response       : x2
Family         : Bernoulli
Link           : probit
Response       : x3
Family         : Bernoulli
Link           : probit
Response       : x4
Family         : Bernoulli
Link           : probit
Log likelihood = -261.30263
 ( 1)  [x1]X = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1						
X	1	(constrained)				
_cons	-.3666763	.1896773	-1.93	0.053	-.738437	.0050844
x2						
X	1.33293	.4686743	2.84	0.004	.4143455	2.251515
_cons	-.4470271	.2372344	-1.88	0.060	-.911998	.0179438
x3						
X	.6040478	.1908343	3.17	0.002	.2300195	.9780761
_cons	-.2276709	.1439342	-1.58	0.114	-.5097767	.0544349
x4						
X	9.453342	5.151819	1.83	0.067	-.6440375	19.55072
_cons	-4.801027	2.518038	-1.91	0.057	-9.736291	.1342372
var(X)	2.173451	1.044885			.847101	5.576536

Notes:

1. In the path diagrams, `x1`, ..., `x4` are shown as being family Bernoulli, link probit. On the command line, we just typed `probit` although we could have typed `family(bernoulli) link(probit)`. In the command language, `probit` is a synonym for `family(bernoulli) link(probit)`.
2. Variable `X` is latent exogenous and thus needs a normalizing constraint. The variable is anchored to the first observed variable, `x1`, and thus the path coefficient is constrained to be 1. See *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.
3. The path coefficients for `X->x1`, `X->x2`, and `X->x3` are 1, 1.33, and 0.60. Meanwhile, the path coefficient for `X->x4` is 9.45. This is not unexpected; we at StataCorp generated these fictional data, and we made the `x4` effect large and less precisely estimable.

Single-factor pass/fail + continuous measurement model

In the above example, all equations were probit. Different equations within a single SEM can have a different family and link.

Below we refit our model with the continuous measure for test 4 (variable `s4`) in place of the pass/fail measure (variable `x4`). We continue to use the pass/fail measures for tests 1, 2, and 3.

```
. gsem (x1 x2 x3 <-X, probit) (s4<-X)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -959.23492
Iteration 1:   log likelihood = -959.09499
Iteration 2:   log likelihood = -959.09499
Refining starting values:
Grid node 0:   log likelihood = -905.14944
Fitting full model:
Iteration 0:   log likelihood = -905.14944  (not concave)
Iteration 1:   log likelihood = -872.33773
Iteration 2:   log likelihood = -869.83144
Iteration 3:   log likelihood = -869.69578
Iteration 4:   log likelihood = -869.68928
Iteration 5:   log likelihood =  -869.6892
Generalized structural equation model           Number of obs     =        123
Response       : x1
Family         : Bernoulli
Link           : probit
Response       : x2
Family         : Bernoulli
Link           : probit
Response       : x3
Family         : Bernoulli
Link           : probit
Response       : s4
Family         : Gaussian
Link           : identity
Log likelihood =  -869.6892
 ( 1)  [x1]X = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1						
X	1	(constrained)				
_cons	-.4171085	.1964736	-2.12	0.034	-.8021896	-.0320274
x2						
X	1.298311	.3280144	3.96	0.000	.6554142	1.941207
_cons	-.4926357	.2387179	-2.06	0.039	-.9605142	-.0247573
x3						
X	.682969	.1747328	3.91	0.000	.3404989	1.025439
_cons	-.2942021	.1575014	-1.87	0.062	-.6028992	.0144949
s4						
X	55.24829	12.19904	4.53	0.000	31.3386	79.15798
_cons	690.9837	6.960106	99.28	0.000	677.3422	704.6253
var(X)	1.854506	.7804393			.812856	4.230998
var(e.s4)	297.8565	408.64			20.24012	4383.299

Notes:

1. We obtain similar coefficients for `x1`, . . . , `x3`.
2. We removed `x4` (a pass/fail variable) and substituted `s4` (the actual test score). `s4` turns out to be more significant than `x4`. This suggests a poor cutoff was set for "passing" test 4.

3. The log-likelihood values for the two models we have fit are strikingly different: −261 in the previous model and −870 in the current model. The difference has no meaning. Log-likelihood values are dependent on the model specified. We changed the fourth equation from a probit specification to a continuous (linear-regression) specification, and just doing that changes the metric of the log-likelihood function. Comparisons of log-likelihood values are only meaningful when they share the same metric.

Fitting the model with the Builder

Use the diagram in *Single-factor pass/fail measurement model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_1fmm
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `X`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `X`;

   b. select `x1`, `x2`, `x3`, and `x4` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Bernoulli, Probit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

6. To fit the model in *Single-factor pass/fail + continuous measurement model*, modify the diagram created in the previous steps.

   a. Use the Select tool, , to click on the `x4` rectangle.

   b. In the Contextual Toolbar, select `s4` in the *Variable* control.

   c. In the Contextual Toolbar, select `Gaussian, Identity` in the *Family/Link* control.

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_1fmm
```

Also see

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Example 28g** — One-parameter logistic IRT (Rasch) model

[SEM] **Example 29g** — Two-parameter logistic IRT model

[SEM] **Example 30g** — Two-level measurement model (multilevel, generalized response)

[SEM] **Example 31g** — Two-factor measurement model (generalized response)

[SEM] **Example 50g** — Latent class model

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 5** — Tour of models

Title

Example 28g — One-parameter logistic IRT (Rasch) model

Description Remarks and examples References Also see

Description

To demonstrate a one-parameter logistic IRT (Rasch) model, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_cfa
(Fictional math abilities data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
school	500	10.5	5.772056	1	20
id	500	50681.71	29081.41	71	100000
q1	500	.506	.5004647	0	1
q2	500	.394	.4891242	0	1
q3	500	.534	.4993423	0	1
q4	500	.424	.4946852	0	1
q5	500	.49	.5004006	0	1
q6	500	.434	.4961212	0	1
q7	500	.52	.5001002	0	1
q8	500	.494	.5004647	0	1
att1	500	2.946	1.607561	1	5
att2	500	2.948	1.561465	1	5
att3	500	2.84	1.640666	1	5
att4	500	2.91	1.566783	1	5
att5	500	3.086	1.581013	1	5
test1	500	75.548	5.948653	55	93
test2	500	80.556	4.976786	65	94
test3	500	75.572	6.677874	50	94
test4	500	74.078	8.845587	43	96

```
. notes
_dta:
  1.  Fictional data on math ability and attitudes of 500 students from 20
      schools.
  2.  Variables q1-q8 are incorrect/correct (0/1) on individual math questions.
  3.  Variables att1-att5 are items from a Likert scale measuring each
      student's attitude toward math.
  4.  Variables test1-test4 are test scores from tests of four different
      aspects of mathematical abilities.  Range of scores: 0-100.
```

These data record results from a fictional instrument measuring mathematical ability. Variables `q1` through `q8` are the items from the instrument.

For discussions of Rasch models, IRT models, and their extensions, see Embretson and Reise (2000), van der Linden and Hambleton (1997), Skrondal and Rabe-Hesketh (2004), Andrich (1988), Bond and Fox (2015), and Fischer and Molenaar (1995). The standard one-parameter logistic model can be fit using the `irt 1pl` command; see [IRT] **irt 1pl**. This example demonstrates how to fit this model. With `gsem`, we can build on this model to fit many of the extensions to basic IRT models discussed in these books.

See *Item response theory (IRT) models* in [SEM] **Intro 5** for background.

Remarks and examples

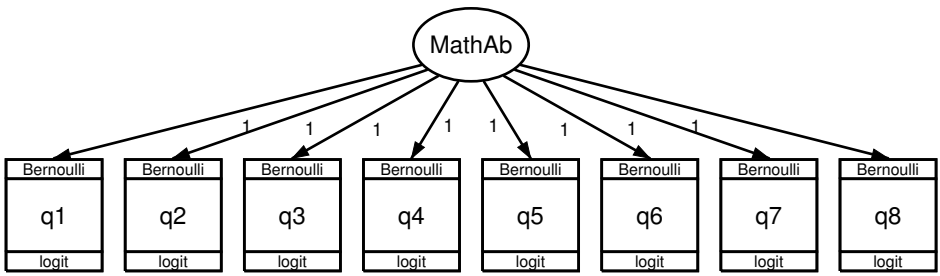
Remarks are presented under the following headings:

1-PL IRT model with unconstrained variance
1-PL IRT model with variance constrained to 1
Obtaining item characteristic curves
Fitting the model with the Builder

1-PL IRT model with unconstrained variance

Mechanically speaking, one-parameter logistic (1-PL) IRT models are similar to the probit measurement model we demonstrated in [SEM] **Example 27g**. The differences are that we will use logit rather than probit and that we will place various constraints on the logit model to obtain results that will allow us to judge the difficulty of the individual questions.

The model we wish to fit is



In the 1-PL model, we place constraints that all coefficients, the factor loadings, are equal to 1. The negative of the intercept for each question will then represent the difficulty of the question:

```
. gsem (MathAb -> (q1-q8)@1), logit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2750.3114
Iteration 1:   log likelihood = -2749.3709
Iteration 2:   log likelihood = -2749.3708
Refining starting values:
Grid node 0:   log likelihood = -2653.2353
Fitting full model:
Iteration 0:   log likelihood = -2653.2353
Iteration 1:   log likelihood = -2651.2171
Iteration 2:   log likelihood = -2650.9117
Iteration 3:   log likelihood = -2650.9116
Generalized structural equation model           Number of obs     =        500
Response       : q1
Family         : Bernoulli
Link           : logit
Response       : q2
Family         : Bernoulli
Link           : logit
Response       : q3
Family         : Bernoulli
Link           : logit
Response       : q4
Family         : Bernoulli
Link           : logit
Response       : q5
Family         : Bernoulli
Link           : logit
Response       : q6
Family         : Bernoulli
Link           : logit
Response       : q7
Family         : Bernoulli
Link           : logit
Response       : q8
Family         : Bernoulli
Link           : logit
Log likelihood = -2650.9116
 ( 1)  [q1]MathAb = 1
 ( 2)  [q2]MathAb = 1
 ( 3)  [q3]MathAb = 1
 ( 4)  [q4]MathAb = 1
 ( 5)  [q5]MathAb = 1
 ( 6)  [q6]MathAb = 1
 ( 7)  [q7]MathAb = 1
 ( 8)  [q8]MathAb = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
MathAb	1	(constrained)				
_cons	.0293252	.1047674	0.28	0.780	-.1760152	.2346656
q2						
MathAb	1	(constrained)				
_cons	-.5025012	.1068768	-4.70	0.000	-.7119759	-.2930264
q3						
MathAb	1	(constrained)				
_cons	.1607425	.104967	1.53	0.126	-.044989	.3664739
q4						
MathAb	1	(constrained)				
_cons	-.3574951	.105835	-3.38	0.001	-.564928	-.1500623
q5						
MathAb	1	(constrained)				
_cons	-.0456599	.1047812	-0.44	0.663	-.2510274	.1597075
q6						
MathAb	1	(constrained)				
_cons	-.3097521	.1055691	-2.93	0.003	-.5166637	-.1028404
q7						
MathAb	1	(constrained)				
_cons	.09497	.1048315	0.91	0.365	-.1104959	.300436
q8						
MathAb	1	(constrained)				
_cons	-.0269104	.1047691	-0.26	0.797	-.232254	.1784332
var(MathAb)	.7929701	.1025406			.6154407	1.02171

Notes:

1. We had to use `gsem` and not `sem` to fit this model because the response variables were 0/1 and not continuous and because we wanted to use logit and not a continuous model.
2. To place the constraints that all coefficients are equal to 1, in the diagram we placed 1s along the path from the underlying latent factor `MathAb` to each of the questions. In the command language, we added `@1` to our command:

   ```
   gsem (MathAb -> (q1-q8)@1), logit
   ```

   Had we omitted the `@1`, we would have obtained coefficients about how well each question measured math ability.

   There are several ways we could have asked that the model above be fit. They include the following:

   ```
   gsem (MathAb -> q1@1 q2@1 q3@1 q4@1 q5@1 q6@1 q7@1 q8@1), logit
   gsem (MathAb -> (q1 q2 q3 q4 q5 q6 q7 q8)@1), logit
   gsem (MathAb -> (q1-q8)@1), logit
   ```

   Similarly, for the shorthand `logit`, we could have typed `family(bernoulli) link(logit)`.
3. The negative of the reported intercept is proportional to the difficulty of the item. The most difficult is `q2`, and the least difficult is `q3`.

1-PL IRT model with variance constrained to 1

The goal of the 1-PL model is in fact to constrain the loadings to be equal. In the previous model, that was achieved by constraining them to be 1 and letting the variance of the latent variable float. An alternative with perhaps easier-to-interpret results would constrain the variance of the latent variable to be 1—giving it a standard-normal interpretation—and constrain the loadings to be merely equal:

```
. gsem (MathAb -> (q1-q8)@b), logit var(MathAb@1) nodvheader
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2750.3114
Iteration 1:   log likelihood = -2749.3709
Iteration 2:   log likelihood = -2749.3708
Refining starting values:
Grid node 0:   log likelihood = -2645.8536
Fitting full model:
Iteration 0:   log likelihood = -2656.1973
Iteration 1:   log likelihood = -2650.9139
Iteration 2:   log likelihood = -2650.9116
Iteration 3:   log likelihood = -2650.9116
Generalized structural equation model           Number of obs     =        500
Log likelihood = -2650.9116
 ( 1)  [q1]MathAb - [q8]MathAb = 0
 ( 2)  [q2]MathAb - [q8]MathAb = 0
 ( 3)  [q3]MathAb - [q8]MathAb = 0
 ( 4)  [q4]MathAb - [q8]MathAb = 0
 ( 5)  [q5]MathAb - [q8]MathAb = 0
 ( 6)  [q6]MathAb - [q8]MathAb = 0
 ( 7)  [q7]MathAb - [q8]MathAb = 0
 ( 8)  [/]var(MathAb) = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	.0293253	.1047674	0.28	0.780	-.1760151	.2346657
q2						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	-.5025011	.1068768	-4.70	0.000	-.7119758	-.2930264
q3						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	.1607425	.104967	1.53	0.126	-.044989	.366474
q4						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	-.3574951	.105835	-3.38	0.001	-.5649279	-.1500622
q5						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	-.0456599	.1047812	-0.44	0.663	-.2510273	.1597076
q6						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	-.309752	.1055691	-2.93	0.003	-.5166637	-.1028403
q7						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	.0949701	.1048315	0.91	0.365	-.1104959	.300436

q8						
MathAb	.8904887	.0575755	15.47	0.000	.7776429	1.003335
_cons	-.0269103	.1047691	-0.26	0.797	-.232254	.1784333
var(MathAb)	1	(constrained)				

Notes:

1. The log-likelihood values of both models is −2650.9116. The models are equivalent.
2. Intercepts are unchanged.

Obtaining item characteristic curves

Item characteristic curves graph the conditional probability of a particular response given the latent trait. In our case, this simply amounts to graphing the probability of a correct answer against math ability. After estimation, we can obtain the predicted probabilities of a correct answer by typing

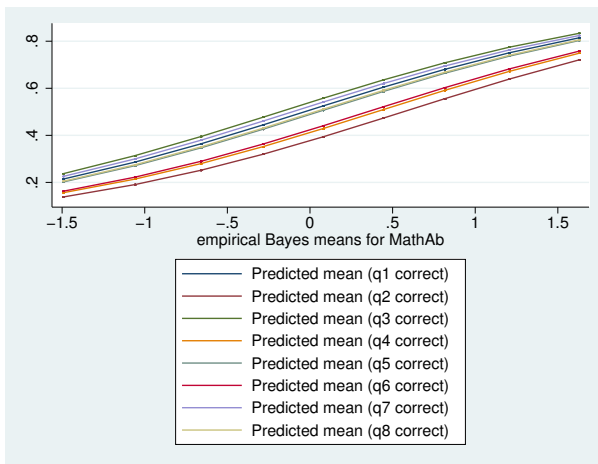
```
. predict pr*, pr
(option conditional(ebmeans) assumed)
(using 7 quadrature points)
```

We can obtain the predicted value of the latent variable by typing

```
. predict ability, latent(MathAb)
(option ebmeans assumed)
(using 7 quadrature points)
```

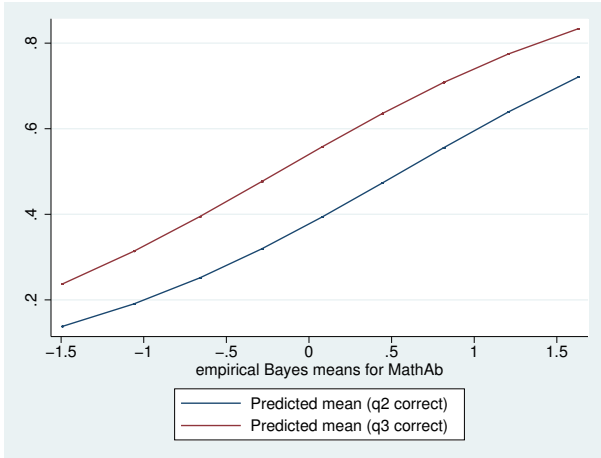
and thus we can obtain the item characteristic curves for all eight questions by typing

```
. twoway line pr1 pr2 pr3 pr4 pr5 pr6 pr7 pr8 ability, sort xlabel(-1.5(.5)1.5)
```



A less busy graph might show merely the most difficult and least difficult questions:

```
. twoway line pr2 pr3 ability, sort xlabel(-1.5(.5)1.5) legend(col(1))
```



The slopes of each curve are identical because we have constrained them to be identical. Thus we just see the shift between difficulties with the lower items having higher levels of difficulty.

Fitting the model with the Builder

Use the diagram in *1-PL IRT model with unconstrained variance* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cfa
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `MathAb`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAb`;

   b. select `q1`, `q2`, `q3`, `q4`, `q5`, `q6`, `q7`, and `q8` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Bernoulli, Logit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Constrain all path coefficients to 1.

   a. Choose the Select tool, .

   b. Click on the path from `MathAb` to `q1`. In the Contextual Toolbar, type `1` in the β box and press *Enter*.

   c. Repeat this process to add the `1` constraint on the paths from `MathAb` to each of the other measurement variables.

6. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

7. To fit the model in *1-PL IRT model with variance constrained to 1*, change the constraints in the diagram created above.

   a. From the SEM Builder menu, select **Estimation > Clear estimates** to clear results from the previous model.

   b. Choose the Select tool, .

   c. Click on the path from `MathAb` to `q1`. In the Contextual Toolbar, type `b` in the β box and press *Enter*.

   d. Repeat this process to add the `b` constraint on the paths from `MathAb` to each of the other measurement variables.

   e. With , click on the oval for `MathAb`. In the Contextual Toolbar, type `1` in the σ^2 box and press *Enter*.

8. Estimate again.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder for the first model by typing

```
. webgetsem gsem_irt1
```

You can open a completed diagram in the Builder for the second model by typing

```
. webgetsem gsem_irt2
```

References

Andrich, D. 1988. *Rasch Models for Measurement.* Newbury Park, CA: SAGE.

Bond, T. G., and C. M. Fox. 2015. *Applying the Rasch Model: Fundamental Measurement in the Human Sciences.* 3rd ed. New York: Routledge.

Embretson, S. E., and S. P. Reise. 2000. *Item Response Theory for Psychologists.* Mahwah, NJ: Lawrence Erlbaum.

Fischer, G. H., and I. W. Molenaar, ed. 1995. *Rasch Models: Foundations, Recent Developments, and Applications.* New York: Springer.

Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests.* Copenhagen: Danish Institute of Educational Research.

——. 1980. *Probabilistic Models for Some Intelligence and Attainment Tests (Expanded ed.).* Chicago: University of Chicago Press.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models.* Boca Raton, FL: Chapman & Hall/CRC.

van der Linden, W. J., and R. K. Hambleton, ed. 1997. *Handbook of Modern Item Response Theory*. New York: Springer.

Also see

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

[SEM] **Example 29g** — Two-parameter logistic IRT model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **predict after gsem** — Generalized linear predictions, etc.

[IRT] **irt 1pl** — One-parameter logistic model

Title

Example 29g — Two-parameter logistic IRT model

Description Remarks and examples References Also see

Description

We demonstrate a two-parameter logistic (2-PL) IRT model with the same data used in [SEM] **Example 28g**:

```
. use https://www.stata-press.com/data/r16/gsem_cfa
(Fictional math abilities data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
school	500	10.5	5.772056	1	20
id	500	50681.71	29081.41	71	100000
q1	500	.506	.5004647	0	1
q2	500	.394	.4891242	0	1
q3	500	.534	.4993423	0	1
q4	500	.424	.4946852	0	1
q5	500	.49	.5004006	0	1
q6	500	.434	.4961212	0	1
q7	500	.52	.5001002	0	1
q8	500	.494	.5004647	0	1
att1	500	2.946	1.607561	1	5
att2	500	2.948	1.561465	1	5
att3	500	2.84	1.640666	1	5
att4	500	2.91	1.566783	1	5
att5	500	3.086	1.581013	1	5
test1	500	75.548	5.948653	55	93
test2	500	80.556	4.976786	65	94
test3	500	75.572	6.677874	50	94
test4	500	74.078	8.845587	43	96

```
. notes
_dta:
  1.  Fictional data on math ability and attitudes of 500 students from 20
      schools.
  2.  Variables q1-q8 are incorrect/correct (0/1) on individual math questions.
  3.  Variables att1-att5 are items from a Likert scale measuring each
      student's attitude toward math.
  4.  Variables test1-test4 are test scores from tests of four different
      aspects of mathematical abilities.  Range of scores: 0-100.
```

These data record results from a fictional instrument measuring mathematical ability. Variables `q1` through `q8` are the items from the instrument.

For discussions of IRT models and their extensions, see Embretson and Reise (2000), van der Linden and Hambleton (1997), Skrondal and Rabe-Hesketh (2004), and Rabe-Hesketh, Skrondal, and Pickles (2004). The two-parameter logistic model can be fit using the `irt 2pl` command; see [IRT] **irt 2pl**. This example demonstrates how to fit this model. With `gsem`, we can build on this model to fit many of the extensions to basic IRT models discussed in these books.

See *Item response theory (IRT) models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

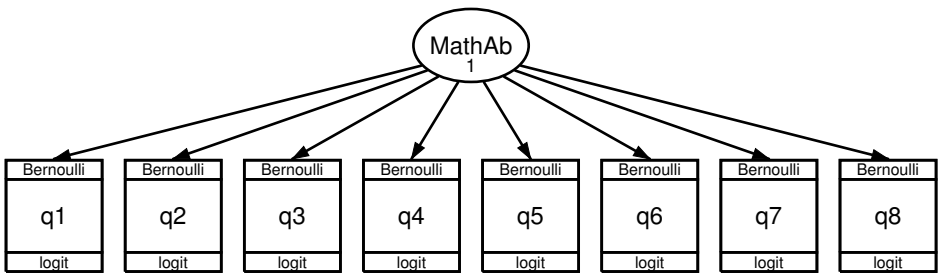
Fitting the 2-PL IRT model
Obtaining predicted difficulty and discrimination
Using coeflegend to obtain the symbolic names of the parameters
Graphing item characteristic curves
Fitting the model with the Builder

Fitting the 2-PL IRT model

When we fit the 1-PL model, we commented that it was similar to the probit measure model we demonstrated in [SEM] **Example 27g**. The 1-PL model differed in that it used logit rather than probit, and it placed constraints on the loadings to judge the difficulty of the individual questions.

The 2-PL model is even more similar to [SEM] **Example 27g**. We still substitute logit for probit, but we only constrain the variance (the latent variable) to be 1—we leave the loadings unconstrained—and we constrain the variance to be 1 merely to aid interpretation. Compared with the 1-PL example, this time we will measure not just difficulty but discrimination as well.

The model we wish to fit is



The results are

```
. gsem (MathAb -> q1-q8), logit var(MathAb@1)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2750.3114
Iteration 1:   log likelihood = -2749.3709
Iteration 2:   log likelihood = -2749.3708
Refining starting values:
Grid node 0:   log likelihood = -2645.8536
Fitting full model:
Iteration 0:   log likelihood = -2645.8536
Iteration 1:   log likelihood = -2637.4315
Iteration 2:   log likelihood = -2637.3761
Iteration 3:   log likelihood = -2637.3759
Generalized structural equation model           Number of obs     =        500
Response       : q1
Family         : Bernoulli
Link           : logit
Response       : q2
Family         : Bernoulli
Link           : logit
Response       : q3
Family         : Bernoulli
Link           : logit
Response       : q4
Family         : Bernoulli
Link           : logit
Response       : q5
Family         : Bernoulli
Link           : logit
Response       : q6
Family         : Bernoulli
Link           : logit
Response       : q7
Family         : Bernoulli
Link           : logit
Response       : q8
Family         : Bernoulli
Link           : logit
Log likelihood = -2637.3759
 ( 1)  [/]var(MathAb) = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
MathAb	1.466636	.2488104	5.89	0.000	.9789765	1.954296
_cons	.0373363	.1252274	0.30	0.766	-.208105	.2827776
q2						
MathAb	.5597118	.1377584	4.06	0.000	.2897102	.8297134
_cons	-.4613391	.0989722	-4.66	0.000	-.6553211	-.2673571
q3						
MathAb	.73241	.1486818	4.93	0.000	.440999	1.023821
_cons	.1533363	.1006072	1.52	0.127	-.0438503	.3505228

q4						
MathAb	.4839501	.1310028	3.69	0.000	.2271893	.7407109
_cons	-.3230667	.0957984	-3.37	0.001	-.5108281	-.1353054
q5						
MathAb	1.232244	.2075044	5.94	0.000	.8255426	1.638945
_cons	-.0494684	.1163093	-0.43	0.671	-.2774304	.1784937
q6						
MathAb	.946535	.1707729	5.54	0.000	.6118262	1.281244
_cons	-.3147231	.1083049	-2.91	0.004	-.5269969	-.1024493
q7						
MathAb	1.197317	.2029485	5.90	0.000	.7995449	1.595088
_cons	.1053405	.1152979	0.91	0.361	-.1206393	.3313203
q8						
MathAb	.8461858	.1588325	5.33	0.000	.5348799	1.157492
_cons	-.026705	.1034396	-0.26	0.796	-.2294429	.1760329
var(MathAb)	1	(constrained)				

Notes:

1. In the above model, we constrain the variance `MathAb` to be 1 by typing `var(MathAb@1)`.
2. Had we not constrained `var(MathAb@1)`, the path coefficient from `MathAb` to `q1` would have automatically constrained to be 1 to set the latent variable's scale. When we applied `var(MathAb@1)`, the automatic constraint was automatically released. Setting the variance of a latent variable is another way of setting its scale.
3. We set `var(MathAb@1)` to ease interpretation. Our latent variable, `MathAb`, is now $N(0, 1)$.
4. Factor loadings, which are the slopes, are estimated above for each question.
5. The slopes reveal how discriminating each question is in regard to mathematical ability. Question 1 is the most discriminating, and question 4 is the least discriminating.
6. In the 1-PL model, the negative of the intercept is a measure of difficulty if we constrain the slopes to be equal to each other. To measure difficulty in the 2-PL model, we divide the negative of the intercept by the unconstrained slope. If you do the math, you will discover that question 2 is the most difficult and question 3 is the least difficult. It will be easier, however, merely to continue reading; in the next section, we show an easy way to calculate the discrimination and difficulty for all the questions.

Obtaining predicted difficulty and discrimination

For each question, discrimination is defined as the question's slope coefficient.

For each question, difficulty is defined as the negative of the question's intercept divided by its slope.

Here is how we quickly obtain all the discrimination and difficulty values in a single, easy-to-read table:

```
. preserve
. drop _all
. set obs 8
number of observations (_N) was 0, now 8
. generate str question = "q" + strofreal(_n)
. generate diff =  .
(8 missing values generated)
. generate disc    =  .
(8 missing values generated)
. forvalues i = 1/8 {
  2.   replace diff = -_b[q‘i’:_cons] / _b[q‘i’:MathAb] in ‘i’
  3.   replace disc =  _b[q‘i’:MathAb] in ‘i’
  4. }
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
(1 real change made)
. format diff disc %9.4f
. egen rank_diff = rank(diff)
. egen rank_disc = rank(disc)
. list
```

	question	diff	disc	rank_d~f	rank_d~c
1.	q1	-0.0255	1.4666	3	8
2.	q2	0.8242	0.5597	8	2
3.	q3	-0.2094	0.7324	1	3
4.	q4	0.6676	0.4840	7	1
5.	q5	0.0401	1.2322	5	7
6.	q6	0.3325	0.9465	6	5
7.	q7	-0.0880	1.1973	2	6
8.	q8	0.0316	0.8462	4	4

```
. restore
```

Notes:

1. Our goal in the Stata code above is to create a dataset containing one observation for each question. The dataset will contain the following variables: `question` containing `q1`, `q2`, ...; `diff` and `disc` containing each question's difficulty and discrimination values; and `rank_disc` and `rank_diff` containing the ranks of those discrimination and difficulty values.

2. We first `preserve`d the current data before tossing out the data in memory. Later, after making and displaying our table, we `restore`d the original contents.
3. We then made an 8-observation, 0-variable dataset (`set obs 8`) and added variables to it. We created string variable `question` containing `q1`, `q2`, ...
4. We were ready to create variables `diff` and `disc`. They are defined in terms of estimated coefficients, and we had no idea what the names of those coefficients were. To find out, we typed `gsem, coeflegend` (output shown below). We quickly learned that the slope coefficients had names like `_b[q1:MathAb]`, `_b[q2:MathAb]`, ..., and the intercepts had names like `_b[/q1]`, `_b[/q2]`, ...
5. We created new variables `diff` and `disc` containing missing values and then created a `forvalues` loop to fill in the new variables. Notice the odd-looking `‘i’` inside the loop. `‘i’` is the way that you say "substitute the value of (local macro) `i` here".
6. We put a display format on new variables `diff` and `disc` so that when we listed them, they would be easier to read.
7. We created the rank of each variable by using the `egen` command.
8. We `list`ed the results. So now you do not have to do the math to see that question 2 is the most difficult (it has `rank_diff` $= 8$) and question 3 is the least (it has `rank_diff` $= 1$).
9. We typed `restore`, bringing our original data back into memory and leaving ourselves in a position to continue with this example.

Using coeflegend to obtain the symbolic names of the parameters

In the section above, we did not retype coefficient values to obtain discrimination and difficulty. After estimation, coefficient values are stored in `_b[`*name*`]`. To find out what the names are, type `gsem, coeflegend`. Here are the results:

```
. gsem, coeflegend
Generalized structural equation model           Number of obs     =        500
  (output omitted)
Log likelihood = -2637.3759
 ( 1)  [/]var(MathAb) = 1
```

	Coef.	Legend
q1		
MathAb	1.466636	_b[q1:MathAb]
_cons	.0373363	_b[q1:_cons]
q2		
MathAb	.5597118	_b[q2:MathAb]
_cons	-.4613391	_b[q2:_cons]
q3		
MathAb	.73241	_b[q3:MathAb]
_cons	.1533363	_b[q3:_cons]
q4		
MathAb	.4839501	_b[q4:MathAb]
_cons	-.3230667	_b[q4:_cons]
q5		
MathAb	1.232244	_b[q5:MathAb]
_cons	-.0494684	_b[q5:_cons]
q6		
MathAb	.946535	_b[q6:MathAb]
_cons	-.3147231	_b[q6:_cons]
q7		
MathAb	1.197317	_b[q7:MathAb]
_cons	.1053405	_b[q7:_cons]
q8		
MathAb	.8461858	_b[q8:MathAb]
_cons	-.026705	_b[q8:_cons]
var(MathAb)	1	_b[/var(MathAb)]

Graphing item characteristic curves

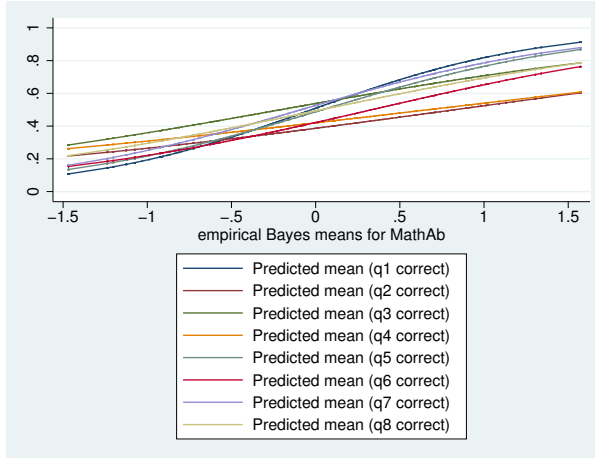
We showed you the item characteristic curves in [SEM] **Example 28g**, so we will show them to you again. Graphs of item characteristic curves plot the probability of a correct answer against the latent trait, which in this case is math ability.

We obtain the probabilities of a correct answer (the values of the latent variable) just as we did previously,

```
. predict pr2pl*, pr
(option conditional(ebmeans) assumed)
(using 7 quadrature points)
. predict ability2pl, latent(MathAb)
(option ebmeans assumed)
(using 7 quadrature points)
```

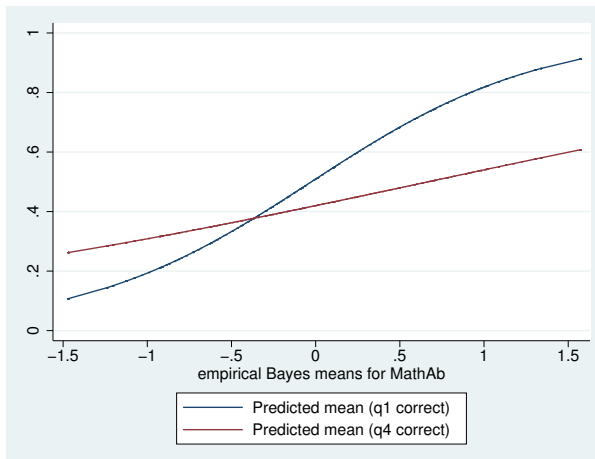
and we graph the curves just as we did previously, too. Here are all eight curves on one graph:

```
. twoway line pr2pl* ability2pl, sort xlabel(-1.5(.5)1.5) legend(col(1))
```



In [SEM] **Example 28g**, we showed a graph for the most and least difficult questions. This time we show a graph for the most and least discriminating questions:

```
. twoway line pr2pl1 pr2pl4 ability2pl, sort xlabel(-1.5(.5)1.5) legend(col(1))
```



Here the curves are not parallel because the discrimination has not been constrained to be equal across the questions. Question 1 has a steeper slope, so it is more discriminating.

Fitting the model with the Builder

Use the diagram in *Fitting the 2-PL IRT model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cfa
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `MathAb`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAb`;

   b. select `q1`, `q2`, `q3`, `q4`, `q5`, `q6`, `q7`, and `q8` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Bernoulli, Logit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Constrain the variance of `MathAb` to 1.

   a. Choose the Select tool, .

   b. Click on the oval for `MathAb`. In the Contextual Toolbar, type `1` in the σ^2 box and press *Enter*.

6. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_irt3
```

References

Embretson, S. E., and S. P. Reise. 2000. *Item Response Theory for Psychologists*. Mahwah, NJ: Lawrence Erlbaum.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

van der Linden, W. J., and R. K. Hambleton, ed. 1997. *Handbook of Modern Item Response Theory*. New York: Springer.

Also see

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

[SEM] **Example 28g** — One-parameter logistic IRT (Rasch) model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **predict after gsem** — Generalized linear predictions, etc.

[IRT] **irt 2pl** — Two-parameter logistic model

Title

Example 30g — Two-level measurement model (multilevel, generalized response)

Description Remarks and examples References Also see

Description

We demonstrate a multilevel measurement model with the same data used in [SEM] **Example 29g**:

```
. use https://www.stata-press.com/data/r16/gsem_cfa
(Fictional math abilities data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
school	500	10.5	5.772056	1	20
id	500	50681.71	29081.41	71	100000
q1	500	.506	.5004647	0	1
q2	500	.394	.4891242	0	1
q3	500	.534	.4993423	0	1
q4	500	.424	.4946852	0	1
q5	500	.49	.5004006	0	1
q6	500	.434	.4961212	0	1
q7	500	.52	.5001002	0	1
q8	500	.494	.5004647	0	1
att1	500	2.946	1.607561	1	5
att2	500	2.948	1.561465	1	5
att3	500	2.84	1.640666	1	5
att4	500	2.91	1.566783	1	5
att5	500	3.086	1.581013	1	5
test1	500	75.548	5.948653	55	93
test2	500	80.556	4.976786	65	94
test3	500	75.572	6.677874	50	94
test4	500	74.078	8.845587	43	96

```
. notes
_dta:
  1.  Fictional data on math ability and attitudes of 500 students from 20
      schools.
  2.  Variables q1-q8 are incorrect/correct (0/1) on individual math questions.
  3.  Variables att1-att5 are items from a Likert scale measuring each
      student's attitude toward math.
  4.  Variables test1-test4 are test scores from tests of four different
      aspects of mathematical abilities.  Range of scores: 0-100.
```

These data record results from a fictional instrument measuring mathematical ability. Variables `q1` through `q8` are the items from the instrument.

For discussions of multilevel measurement models, including extensions beyond the example we present here, see Mehta and Neale (2005) and Skrondal and Rabe-Hesketh (2004).

See *Single-factor measurement models* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

Remarks and examples

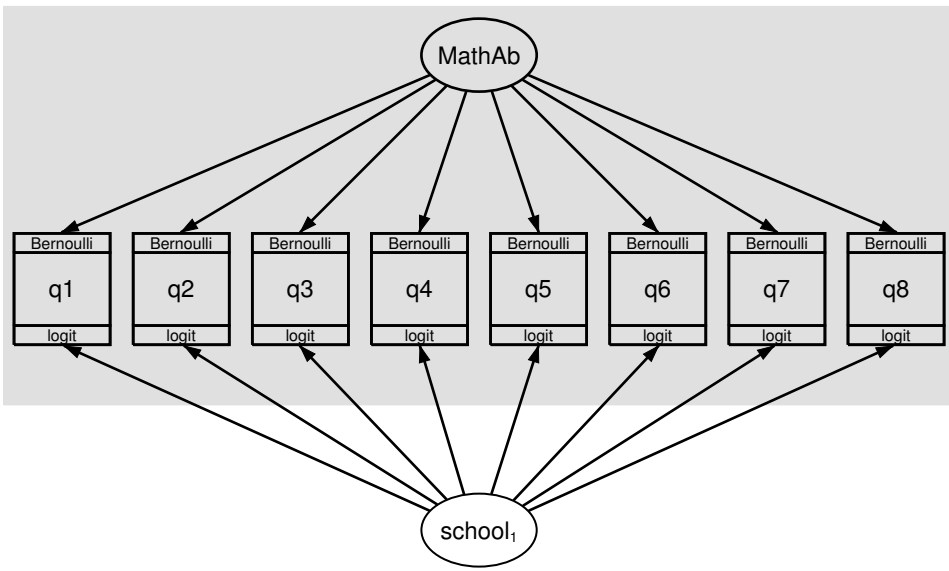
Remarks are presented under the following headings:

Fitting the two-level model
Fitting the variance-components model
Fitting the model with the Builder

Fitting the two-level model

We extend the measurement model fit in [SEM] **Example 29g** to better account for our (fictional) data. In the data, students are nested within school, but we have ignored that so far. In this example, we include a latent variable at the school level to account for possible school-by-school effects.

The model we wish to fit is



The double-ringed `school`$_1$ is new. That new component of the path diagram is saying, "I am a latent variable at the `school` level—meaning I am constant within school and vary across schools—and I correspond to a latent variable named `M1`"; see *Specifying generalized SEMs: Multilevel mixed effects (2 levels)* in [SEM] **Intro 2**. This new variable will account for the effect, if any, of the identity of the school.

To fit this model without this new, school-level component in it, we would type

```
. gsem (MathAb -> q1-q8), logit
```

To include the new school-level component, we add `M1[school]` to the exogenous variables:

```
. gsem (MathAb M1[school] -> q1-q8), logit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2750.3114
Iteration 1:   log likelihood = -2749.3709
Iteration 2:   log likelihood = -2749.3708
Refining starting values:
Grid node 0:   log likelihood = -2649.0033
Fitting full model:
Iteration 0:   log likelihood = -2649.0033  (not concave)
Iteration 1:   log likelihood = -2645.0613  (not concave)
Iteration 2:   log likelihood = -2641.9755  (not concave)
Iteration 3:   log likelihood = -2634.3857
Iteration 4:   log likelihood = -2631.1111
Iteration 5:   log likelihood = -2630.7898
Iteration 6:   log likelihood = -2630.2477
Iteration 7:   log likelihood = -2630.2402
Iteration 8:   log likelihood = -2630.2074
Iteration 9:   log likelihood = -2630.2063
Iteration 10:  log likelihood = -2630.2063
Generalized structural equation model           Number of obs     =        500
Response       : q1
Family         : Bernoulli
Link           : logit
Response       : q2
Family         : Bernoulli
Link           : logit
Response       : q3
Family         : Bernoulli
Link           : logit
Response       : q4
Family         : Bernoulli
Link           : logit
Response       : q5
Family         : Bernoulli
Link           : logit
Response       : q6
Family         : Bernoulli
Link           : logit
Response       : q7
Family         : Bernoulli
Link           : logit
Response       : q8
Family         : Bernoulli
Link           : logit
Log likelihood = -2630.2063
 ( 1)  [q1]M1[school] = 1
 ( 2)  [q2]MathAb = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
M1[school]	1	(constrained)				
MathAb	2.807515	.9468682	2.97	0.003	.9516878	4.663343
_cons	.0388021	.1608489	0.24	0.809	-.276456	.3540602
q2						
M1[school]	.6673925	.3058328	2.18	0.029	.0679712	1.266814
MathAb	1	(constrained)				
_cons	-.4631159	.1201227	-3.86	0.000	-.698552	-.2276798
q3						
M1[school]	.3555867	.3043548	1.17	0.243	-.2409377	.9521111
MathAb	1.455529	.5187786	2.81	0.005	.4387416	2.472316
_cons	.1537831	.1070288	1.44	0.151	-.0559894	.3635556
q4						
M1[school]	.7073241	.3419273	2.07	0.039	.037159	1.377489
MathAb	.8420897	.3528195	2.39	0.017	.1505762	1.533603
_cons	-.3252735	.1202088	-2.71	0.007	-.5608784	-.0896686
q5						
M1[school]	.7295553	.3330652	2.19	0.028	.0767595	1.382351
MathAb	2.399529	.8110973	2.96	0.003	.8098079	3.989251
_cons	-.0488674	.1378015	-0.35	0.723	-.3189533	.2212185
q6						
M1[school]	.484903	.2844447	1.70	0.088	-.0725983	1.042404
MathAb	1.840627	.5934017	3.10	0.002	.6775813	3.003673
_cons	-.3139302	.1186624	-2.65	0.008	-.5465042	-.0813563
q7						
M1[school]	.3677241	.2735779	1.34	0.179	-.1684787	.903927
MathAb	2.444023	.8016872	3.05	0.002	.8727449	4.015301
_cons	.1062164	.1220796	0.87	0.384	-.1330552	.3454881
q8						
M1[school]	.5851299	.3449508	1.70	0.090	-.0909612	1.261221
MathAb	1.606287	.5367614	2.99	0.003	.5542541	2.65832
_cons	-.0261962	.1189835	-0.22	0.826	-.2593995	.2070071
var(						
M1[school])	.2121216	.1510032			.052558	.8561121
var(MathAb)	.2461246	.1372513			.0825055	.7342217

Notes:

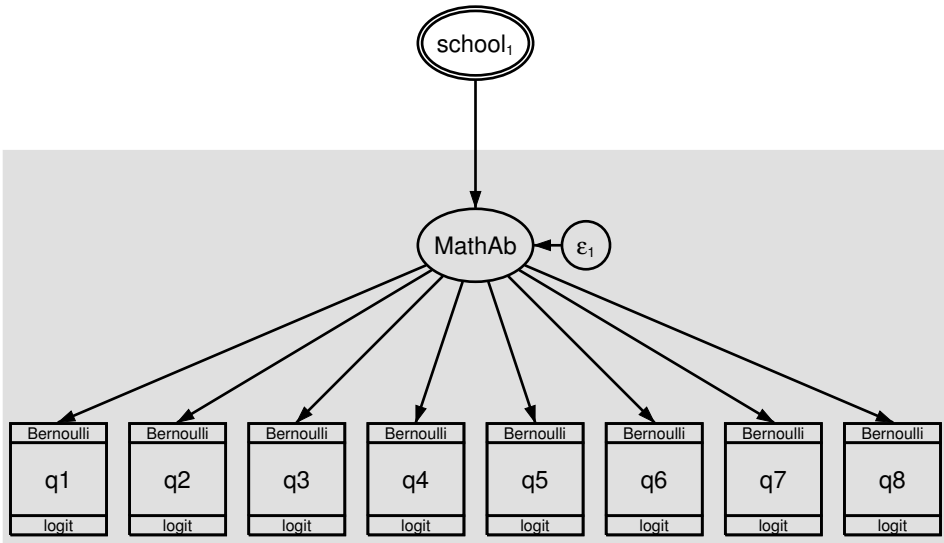
1. The variance of `M1[school]` is estimated to be 0.21.
2. So how important is `M1[school]`? The variance of `MathAb` is estimated to be 0.25, so math ability and school have roughly the same variance, and both of course have mean 0. The math ability coefficients, meanwhile, are larger—often much larger—than the school coefficients in every case, so math ability is certainly more important than school in explaining whether questions were answered correctly. At this point, we are merely exploring the magnitude of effect.
3. You could also include a school-level latent variable for each question. For instance, you could type

```
. gsem (MathAb M1[school] N1[school] -> q1) ///
       (MathAb M1[school] N2[school] -> q2) ///
       (MathAb M1[school] N3[school] -> q3) ///
       (MathAb M1[school] N4[school] -> q4) ///
       (MathAb M1[school] N5[school] -> q5) ///
       (MathAb M1[school] N6[school] -> q6) ///
       (MathAb M1[school] N7[school] -> q7) ///
       (MathAb M1[school] N8[school] -> q8), logit
```

   You will sometimes see such effects included in multilevel measurement models in theoretical discussions of models. Be aware that estimation of models with many latent variables is problematic, requiring both time and luck.

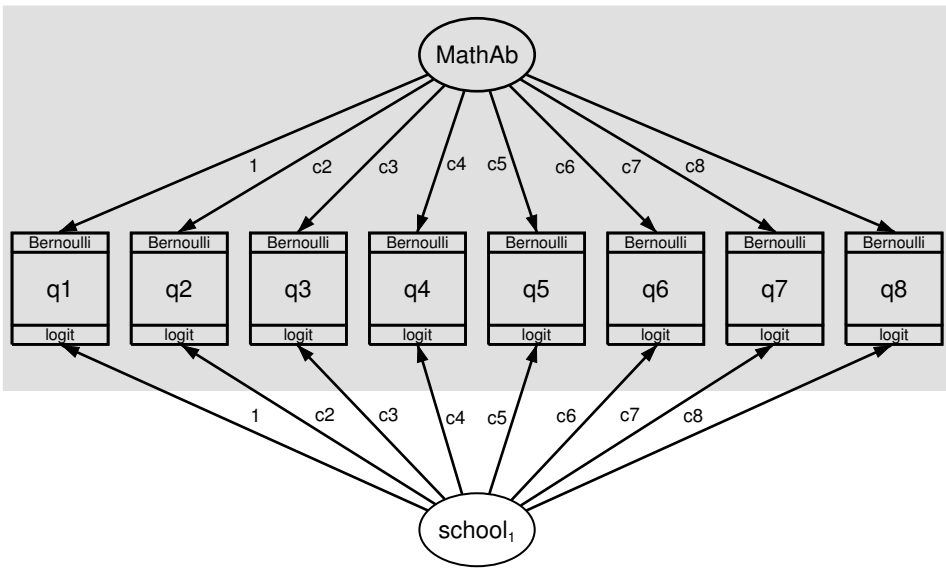
Fitting the variance-components model

In a variance-components model, school would affect math ability which would affect correctness of answers to questions. The model might be drawn like this:



The above is a great way to draw the model. Sadly, `gsem` cannot understand it. The problem from `gsem`'s perspective is that one latent variable is affecting another and the two latent variables are at different levels.

So we have to draw the model differently:



The models may look different, but constraining the coefficients along the paths from math ability and from school to each question is identical in effect to the model above.

The result of fitting the model is

```
. gsem (MathAb M1[school] ->
>      q1@1  q2@c2  q3@c3  q4@c4  q5@c5  q6@c6  q7@c7 q8@c8), logit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2750.3114
Iteration 1:   log likelihood = -2749.3709
Iteration 2:   log likelihood = -2749.3708
Refining starting values:
Grid node 0:   log likelihood = -2642.8248
Fitting full model:
Iteration 0:   log likelihood = -2651.7239  (not concave)
Iteration 1:   log likelihood = -2644.4937
Iteration 2:   log likelihood =   -2634.92
Iteration 3:   log likelihood = -2633.9336
Iteration 4:   log likelihood = -2633.5924
Iteration 5:   log likelihood = -2633.5922
Generalized structural equation model           Number of obs     =        500
  (output omitted)
Log likelihood = -2633.5922
 ( 1)  [q1]M1[school] = 1
 ( 2)  [q1]MathAb = 1
 ( 3)  [q2]M1[school] - [q2]MathAb = 0
 ( 4)  [q3]M1[school] - [q3]MathAb = 0
 ( 5)  [q4]M1[school] - [q4]MathAb = 0
 ( 6)  [q5]M1[school] - [q5]MathAb = 0
 ( 7)  [q6]M1[school] - [q6]MathAb = 0
 ( 8)  [q7]M1[school] - [q7]MathAb = 0
 ( 9)  [q8]M1[school] - [q8]MathAb = 0
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
M1[school]	1	(constrained)				
MathAb	1	(constrained)				
_cons	.0385522	.1556214	0.25	0.804	-.2664601	.3435646
q2						
M1[school]	.3876281	.1156823	3.35	0.001	.1608951	.6143612
MathAb	.3876281	.1156823	3.35	0.001	.1608951	.6143612
_cons	-.4633143	.1055062	-4.39	0.000	-.6701028	-.2565259
q3						
M1[school]	.4871164	.1295515	3.76	0.000	.2332001	.7410328
MathAb	.4871164	.1295515	3.76	0.000	.2332001	.7410328
_cons	.1533212	.1098068	1.40	0.163	-.0618962	.3685386
q4						
M1[school]	.3407151	.1058542	3.22	0.001	.1332446	.5481856
MathAb	.3407151	.1058542	3.22	0.001	.1332446	.5481856
_cons	-.3246936	.1011841	-3.21	0.001	-.5230108	-.1263763

q5						
M1[school]	.8327426	.1950955	4.27	0.000	.4503624	1.215123
MathAb	.8327426	.1950955	4.27	0.000	.4503624	1.215123
_cons	-.0490579	.1391324	-0.35	0.724	-.3217524	.2236365
q6						
M1[school]	.6267415	.1572247	3.99	0.000	.3185868	.9348962
MathAb	.6267415	.1572247	3.99	0.000	.3185868	.9348962
_cons	-.3135398	.1220389	-2.57	0.010	-.5527317	-.074348
q7						
M1[school]	.7660343	.187918	4.08	0.000	.3977219	1.134347
MathAb	.7660343	.187918	4.08	0.000	.3977219	1.134347
_cons	.1039102	.1330652	0.78	0.435	-.1568927	.3647131
q8						
M1[school]	.5600833	.1416542	3.95	0.000	.2824462	.8377203
MathAb	.5600833	.1416542	3.95	0.000	.2824462	.8377203
_cons	-.0264193	.1150408	-0.23	0.818	-.2518951	.1990565
var(						
M1[school])	.1719347	.1150138			.0463406	.6379187
var(MathAb)	2.062489	.6900045			1.070589	3.973385

1. Note that for each question, the coefficient on `MathAb` is identical to the coefficient on `M1[school]`.
2. We estimate separate variances for `M1[school]` and `MathAb`. They are 0.17 and 2.06. Now that the coefficients are the same on school and ability, we can directly compare these variances. We see that math ability has a much larger affect than does school.

Fitting the model with the Builder

Use the diagram in *Fitting the two-level model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cfa
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `MathAb`.

   Select the Add measurement component tool, , and then click in the diagram about one-fourth of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAb`;

   b. select `q1`, `q2`, `q3`, `q4`, `q5`, `q6`, `q7`, and `q8` by using the *Measurement variables* control;

c. check *Make measurements generalized*;

d. select `Bernoulli, Logit` in the *Family/Link* control;

e. select `Down` in the *Measurement direction* control;

f. click on **OK**.

If you wish, move the component by clicking on any variable and dragging it.

5. Create the school-level latent variable.

a. Select the Add multilevel latent variable tool, , and click about one-fourth of the way up from the bottom and slightly left of the center.

b. In the Contextual Toolbar, click on the button.

c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `school` > *Observations* in the next control.

d. Specify `M1` as the *Base name*.

e. Click on **OK**.

6. Create the factor-loading paths for the multilevel latent variable.

a. Select the Add path tool, .

b. Click in the top-left quadrant of the double oval for $\texttt{school}_1$ (it will highlight when you hover over it), and drag a path to the bottom of the `q1` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the tool, draw paths from $\texttt{school}_1$ to each of the remaining rectangles.

7. Clean up paths.

If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle or oval and drag the endpoint.

8. Estimate.

Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

9. To fit the model in *Fitting the variance-components model*, add constraints to the diagram created above.

a. From the SEM Builder menu, select **Estimation > Clear estimates** to clear results from the previous model.

b. Choose the Select tool, .

c. Click on the path from `MathAb` to `q1`. In the Contextual Toolbar, type `1` in the box and press *Enter*.

d. Click on the path from $\texttt{school}_1$ to `q1`. In the Contextual Toolbar, type `1` in the box and press *Enter*.

c. Click on the path from `MathAb` to `q2`. In the Contextual Toolbar, type `c2` in the box and press *Enter*.

d. Click on the path from $\texttt{school}_1$ to `q2`. In the Contextual Toolbar, type `c2` in the box and press *Enter*.

e. Repeat this process to add the `c3` constraint on both paths to `q3`, the `c4` constraint on both paths to `q4`, . . . , and the `c8` constraint on both paths to `q8`.

10. Estimate again.

Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder for the first model by typing

```
. webgetsem gsem_mlcfa1
```

You can open a completed diagram in the Builder for the second model by typing

```
. webgetsem gsem_mlcfa2
```

References

Mehta, P. D., and M. C. Neale. 2005. People are variables too: Multilevel structural equations modeling. *Psychological Methods* 10: 259–284.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Also see

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

[SEM] **Example 29g** — Two-parameter logistic IRT model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 31g — Two-factor measurement model (generalized response)

Description Remarks and examples Also see

Description

We demonstrate a two-factor generalized linear measurement model with the same data used in [SEM] **Example 29g**:

```
. use https://www.stata-press.com/data/r16/gsem_cfa
(Fictional math abilities data)

. describe

Contains data from https://www.stata-press.com/data/r16/gsem_cfa.dta
  obs:           500                          Fictional math abilities data
 vars:            19                          21 Mar 2018 10:38
                                              (_dta has notes)
```

variable name	storage type	display format	value label	variable label
school	byte	%9.0g		School id
id	long	%9.0g		Student id
q1	byte	%9.0g	result	q1 correct
q2	byte	%9.0g	result	q2 correct
q3	byte	%9.0g	result	q3 correct
q4	byte	%9.0g	result	q4 correct
q5	byte	%9.0g	result	q5 correct
q6	byte	%9.0g	result	q6 correct
q7	byte	%9.0g	result	q7 correct
q8	byte	%9.0g	result	q8 correct
att1	float	%26.0g	agree	Skills taught in math class will help me get a better job.
att2	float	%26.0g	agree	Math is important in everyday life
att3	float	%26.0g	agree	Working math problems makes me anxious.
att4	float	%26.0g	agree	Math has always been my worst subject.
att5	float	%26.0g	agree	I am able to learn new math concepts easily.
test1	byte	%9.0g		Score, math test 1
test2	byte	%9.0g		Score, math test 2
test3	byte	%9.0g		Score, math test 3
test4	byte	%9.0g		Score, math test 4

```
Sorted by:

. notes

_dta:
  1.  Fictional data on math ability and attitudes of 500 students from 20
      schools.
  2.  Variables q1-q8 are incorrect/correct (0/1) on individual math questions.
  3.  Variables att1-att5 are items from a Likert scale measuring each
      student's attitude toward math.
  4.  Variables test1-test4 are test scores from tests of four different
      aspects of mathematical abilities.  Range of scores: 0-100.
```

These data record results from a fictional instrument measuring mathematical ability. Variables `q1` through `q8` are the items from the instrument.

In this example, we will also be using variables `att1` through `att5`. These are five Likert-scale questions measuring each student's attitude toward math.

See *Multiple-factor measurement models* in [SEM] **Intro 5** for background.

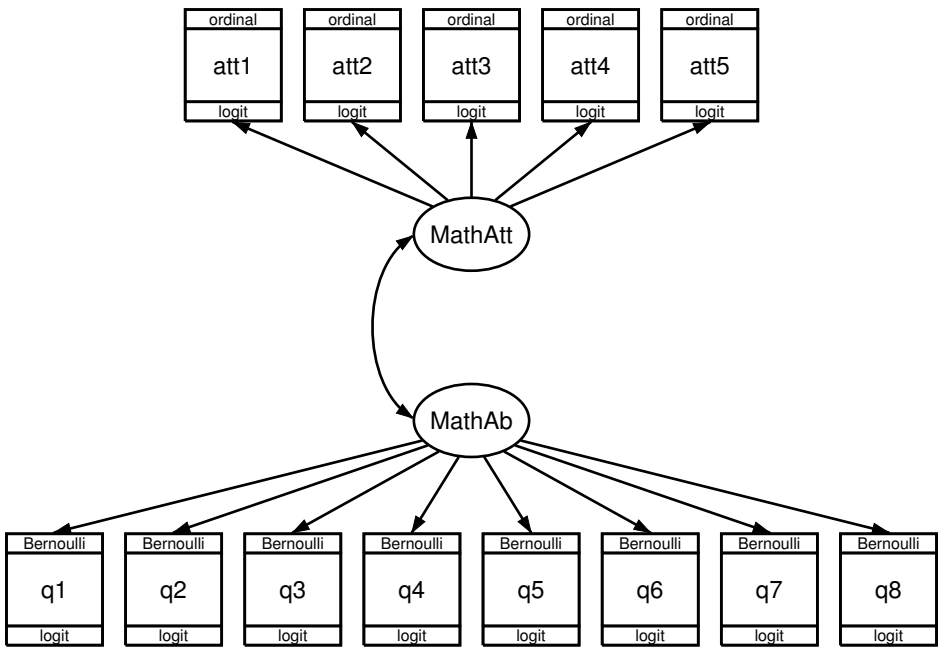
Remarks and examples

Remarks are presented under the following headings:

Fitting the two-factor model
Fitting the model with the Builder

Fitting the two-factor model

We extend the measurement model fit in [SEM] **Example 29g** from one factor, math ability, to two factors, math ability and attitude. The model we wish to fit is



In this model, mathematical ability affects the correctness of the answers to the items just as previously. The new component, attitude toward mathematics, is correlated with math ability. We expect this correlation to be positive, but that is yet to be determined.

What is important about the attitudinal questions is that the responses are ordinal, that is, the ordering of the possible answers is significant. In other cases, we might have a categorical variable taking on, say, five values; even if the values are 1, 2, 3, 4, and 5, there is no case in which answer 5 is greater than answer 4, answer 4 is greater than answer 3, and so on.

For our attitude measures, however, response 5 signifies strong agreement with a statement and 1 signifies strong disagreement. We handle the ordinal property by specifying that the attitudinal responses are family ordinal, link logit, also known as ordered logit or ordinal logistic regression, and also known in Stata circles as ologit.

In the command language, to fit a one-factor measurement model with math ability, we would type

```
gsem (MathAb -> q1-q8), logit
```

To include the second factor, attitude correlated with math ability, we would type

```
gsem (MathAb  -> q1-q8,      logit)     ///
     (MathAtt -> att1-att5, ologit)
```

The covariance between `MathAtt` and `MathAb` does not even appear in the command! That is because latent exogenous variables are assumed to be correlated in the command language unless you specify otherwise; in path diagrams, such variables are correlated only if a curved path is drawn between them.

There is another, minor difference in syntax between the one-factor and two-factor models that is worth your attention. Notice that the `logit` was outside the parentheses in the command to fit the one-factor model, but it is inside the parentheses in the command to fit the two-factor model. Actually, `logit` could have appeared inside the parentheses to fit the one-factor model. When options appear inside parentheses, they affect only what is specified inside the parentheses. When they appear outside parentheses, they affect all parenthetical specifications.

To obtain the estimates of the two-factor model, we type

```
. gsem (MathAb -> q1-q8, logit)
>      (MathAtt -> att1-att5, ologit)

Fitting fixed-effects model:

Iteration 0:   log likelihood = -6629.7253
Iteration 1:   log likelihood = -6628.7848
Iteration 2:   log likelihood = -6628.7848

Refining starting values:

Grid node 0:   log likelihood = -6457.4584

Fitting full model:

Iteration 0:   log likelihood = -6457.4584
Iteration 1:   log likelihood = -6437.9594
Iteration 2:   log likelihood = -6400.2731
Iteration 3:   log likelihood = -6396.3795
Iteration 4:   log likelihood = -6394.5787
Iteration 5:   log likelihood = -6394.4019
Iteration 6:   log likelihood = -6394.3923
Iteration 7:   log likelihood = -6394.3923

Generalized structural equation model           Number of obs     =        500

Response       : q1
Family         : Bernoulli
Link           : logit

Response       : q2
Family         : Bernoulli
Link           : logit

Response       : q3
Family         : Bernoulli
Link           : logit

Response       : q4
Family         : Bernoulli
Link           : logit
```

```
Response       : q5
Family         : Bernoulli
Link           : logit

Response       : q6
Family         : Bernoulli
Link           : logit

Response       : q7
Family         : Bernoulli
Link           : logit

Response       : q8
Family         : Bernoulli
Link           : logit

Response       : att1
Family         : ordinal
Link           : logit

Response       : att2
Family         : ordinal
Link           : logit

Response       : att3
Family         : ordinal
Link           : logit

Response       : att4
Family         : ordinal
Link           : logit

Response       : att5
Family         : ordinal
Link           : logit

Log likelihood = -6394.3923

 ( 1)  [q1]MathAb = 1
 ( 2)  [att1]MathAtt = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
MathAb	1	(constrained)				
_cons	.0446118	.1272964	0.35	0.726	-.2048845	.2941082
q2						
MathAb	.3446081	.1050264	3.28	0.001	.1387601	.5504562
_cons	-.4572215	.0979965	-4.67	0.000	-.6492911	-.265152
q3						
MathAb	.5445245	.1386993	3.93	0.000	.272679	.8163701
_cons	.1591406	.1033116	1.54	0.123	-.0433464	.3616276
q4						
MathAb	.2858874	.0948553	3.01	0.003	.0999743	.4718004
_cons	-.3196648	.0947684	-3.37	0.001	-.5054075	-.1339222
q5						
MathAb	.8174803	.1867024	4.38	0.000	.4515504	1.18341
_cons	-.04543	.116575	-0.39	0.697	-.2739127	.1830527
q6						
MathAb	.6030448	.1471951	4.10	0.000	.3145478	.8915419
_cons	-.309992	.1070853	-2.89	0.004	-.5198753	-.1001086

q7						
MathAb	.72084	.1713095	4.21	0.000	.3850796	1.056601
_cons	.1047265	.1116494	0.94	0.348	-.1141023	.3235552
q8						
MathAb	.5814761	.1426727	4.08	0.000	.3018428	.8611094
_cons	-.0250442	.1045134	-0.24	0.811	-.2298868	.1797983
att1						
MathAtt	1	(constrained)				
att2						
MathAtt	.3788714	.0971223	3.90	0.000	.1885152	.5692276
att3						
MathAtt	-1.592717	.3614859	-4.41	0.000	-2.301216	-.8842173
att4						
MathAtt	-.8100107	.153064	-5.29	0.000	-1.11001	-.5100108
att5						
MathAtt	.5225423	.1170141	4.47	0.000	.2931988	.7518858
/att1						
cut1	-1.10254	.1312272			-1.359741	-.8453396
cut2	-.2495339	.1160385			-.4769651	-.0221027
cut3	.2983261	.1164414			.0701052	.5265471
cut4	1.333053	.1391907			1.060244	1.605861
/att2						
cut1	-1.055791	.1062977			-1.264131	-.8474513
cut2	-.1941211	.0941435			-.378639	-.0096032
cut3	.3598488	.0952038			.1732528	.5464448
cut4	1.132624	.1082204			.9205156	1.344732
/att3						
cut1	-1.053519	.1733999			-1.393377	-.7136614
cut2	-.0491073	.1442846			-.3318999	.2336853
cut3	.5570671	.1538702			.2554871	.8586471
cut4	1.666859	.2135554			1.248298	2.08542
/att4						
cut1	-1.07378	.1214071			-1.311734	-.8358264
cut2	-.2112462	.1076501			-.4222366	-.0002559
cut3	.406347	.1094847			.191761	.620933
cut4	1.398185	.1313327			1.140778	1.655593
/att5						
cut1	-1.244051	.1148443			-1.469142	-1.018961
cut2	-.336135	.0986678			-.5295203	-.1427498
cut3	.2137776	.0978943			.0219084	.4056468
cut4	.9286849	.107172			.7186316	1.138738
var(MathAb)	2.300652	.7479513			1.216527	4.350909
var(MathAtt)	1.520854	.4077674			.8992196	2.572228
cov(MathAb, MathAtt)	.8837681	.2204606	4.01	0.000	.4516733	1.315863

Notes:

1. The estimated covariance between math attitude and ability is 0.88.
2. There is something new in the output, namely, things labeled `cut1`, ..., `cut4`. These appear for each of the five attitudinal measures. These are the ordered logit's cutpoints, the values on the logit's distribution that separate attitude 1 from attitude 2, attitude 2 from attitude 3, and so on. The four cutpoints map the continuous distribution into five ordered, categorical groups.
3. There's something interesting hiding in the `MathAtt` coefficients: the coefficients for two of the paths, `att3 att4 <- MathAtt`, are negative! If you look back to the description of the data, you will find that the sense of these two questions was reversed from those of the other questions. Strong agreement on these two questions was agreement with a negative feeling about mathematics.

`estat sd` displays the fitted variance components as standard deviations and correlations. From the following, we see that the estimated correlation between attitude and ability is 0.4725.

```
. estat sd
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sd(MathAb)	1.51679	.2465573			1.102963	2.085883
sd(MathAtt)	1.233229	.1653251			.9482719	1.603817
corr(MathAb, MathAtt)	.4724644	.0649541	7.27	0.000	.3451567	.5997721

Fitting the model with the Builder

Use the diagram in *Fitting the two-factor model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cfa
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `MathAb`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way up from the bottom and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAb`;

   b. select `q1`, `q2`, `q3`, `q4`, `q5`, `q6`, `q7`, and `q8` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Bernoulli, Logit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Create the measurement component for `MathAtt`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAtt`;

   b. select `att1`, `att2`, `att3`, `att4`, and `att5` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Ordinal, Logit` in the *Family/Link* control;

   e. select `Up` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

6. Create the covariance between `MathAtt` and `MathAb`.

   a. Select the Add covariance tool, .

   b. Click in the top-left quadrant of the `MathAb` oval, and drag a covariance to the bottom left of the `MathAtt` oval.

7. Clean up.

   If you do not like where a covariance has been connected to its variable, use the Select tool, , to simply click on the covariance, and then click on where it connects to an oval and drag the endpoint. You can also change the bow of the covariance by dragging the control point that extends from one end of the selected covariance.

8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_2fmm
```

Also see

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

[SEM] **Example 29g** — Two-parameter logistic IRT model

[SEM] **Example 32g** — Full structural equation model (generalized response)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat sd** — Display variance components as standard deviations and correlations

Title

Example 32g — Full structural equation model (generalized response)

Description Remarks and examples Also see

Description

To demonstrate a structural model with a measurement component, we use the same data used in [SEM] **Example 31g**:

```
. use https://www.stata-press.com/data/r16/gsem_cfa
(Fictional math abilities data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
school	500	10.5	5.772056	1	20
id	500	50681.71	29081.41	71	100000
q1	500	.506	.5004647	0	1
q2	500	.394	.4891242	0	1
q3	500	.534	.4993423	0	1
q4	500	.424	.4946852	0	1
q5	500	.49	.5004006	0	1
q6	500	.434	.4961212	0	1
q7	500	.52	.5001002	0	1
q8	500	.494	.5004647	0	1
att1	500	2.946	1.607561	1	5
att2	500	2.948	1.561465	1	5
att3	500	2.84	1.640666	1	5
att4	500	2.91	1.566783	1	5
att5	500	3.086	1.581013	1	5
test1	500	75.548	5.948653	55	93
test2	500	80.556	4.976786	65	94
test3	500	75.572	6.677874	50	94
test4	500	74.078	8.845587	43	96

```
. notes
_dta:
  1.  Fictional data on math ability and attitudes of 500 students from 20
      schools.
  2.  Variables q1-q8 are incorrect/correct (0/1) on individual math questions.
  3.  Variables att1-att5 are items from a Likert scale measuring each
      student's attitude toward math.
  4.  Variables test1-test4 are test scores from tests of four different
      aspects of mathematical abilities.  Range of scores: 0-100.
```

These data record results from a fictional instrument measuring mathematical ability. Variables `q1` through `q8` are the items from the instrument.

In this example, we will also be using variables `att1` through `att5`. These are five Likert-scale questions measuring each student's attitude toward math.

See *Structural models 9: Unobserved inputs, outputs, or both* in [SEM] **Intro 5** for background.

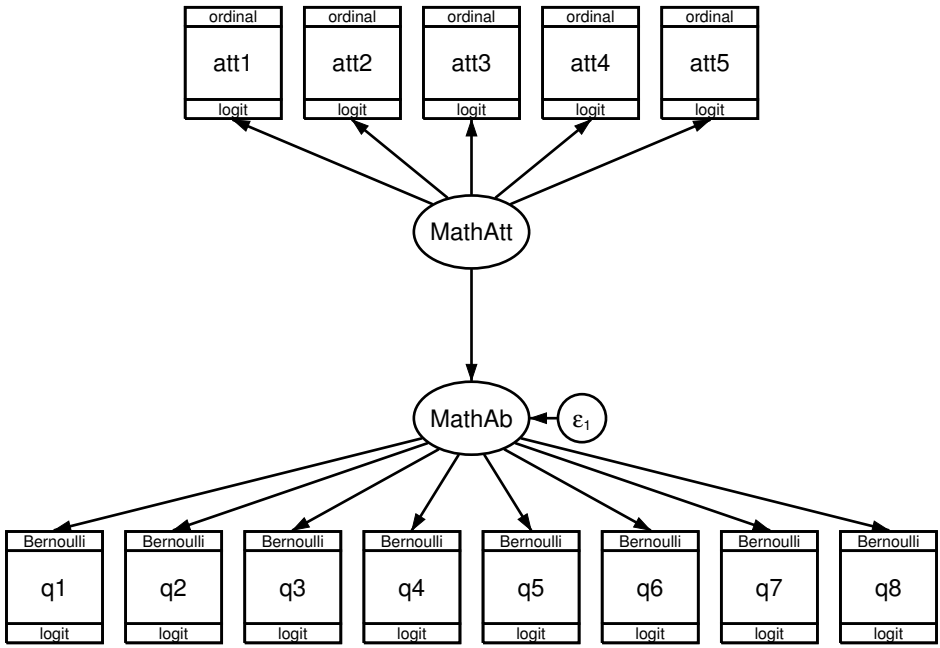
Remarks and examples

Remarks are presented under the following headings:

Structural model with measurement component
Fitting the model with the Builder

Structural model with measurement component

We wish to fit the following model:



This is the same model we fit in [SEM] **Example 31g**, except that rather than a correlation (curved path) between `MathAtt` and `MathAb`, this time we assume a direct effect and so allow a straight path.

If you compare the two path diagrams, in addition to the new substitution of the direct path for the curved path signifying correlation, there is now an error variable on `MathAb`. In the previous diagram, `MathAb` was exogenous. In this diagram, it is endogenous and thus requires an error term. In the Builder, the error term is added automatically.

To fit this model in the command language, we type

```
. gsem (MathAb -> q1-q8, logit)
>      (MathAtt -> att1-att5, ologit)
>      (MathAtt -> MathAb)

Fitting fixed-effects model:

Iteration 0:   log likelihood = -6629.7253
Iteration 1:   log likelihood = -6628.7848
Iteration 2:   log likelihood = -6628.7848

Refining starting values:

Grid node 0:   log likelihood = -6429.1636
```

```
Fitting full model:

Iteration 0:   log likelihood = -6429.1636
Iteration 1:   log likelihood = -6396.7471
Iteration 2:   log likelihood = -6394.6197
Iteration 3:   log likelihood = -6394.3949
Iteration 4:   log likelihood = -6394.3923
Iteration 5:   log likelihood = -6394.3923

Generalized structural equation model           Number of obs     =        500

Response       : q1
Family         : Bernoulli
Link           : logit

Response       : q2
Family         : Bernoulli
Link           : logit

Response       : q3
Family         : Bernoulli
Link           : logit

Response       : q4
Family         : Bernoulli
Link           : logit

Response       : q5
Family         : Bernoulli
Link           : logit

Response       : q6
Family         : Bernoulli
Link           : logit

Response       : q7
Family         : Bernoulli
Link           : logit

Response       : q8
Family         : Bernoulli
Link           : logit

Response       : att1
Family         : ordinal
Link           : logit

Response       : att2
Family         : ordinal
Link           : logit

Response       : att3
Family         : ordinal
Link           : logit

Response       : att4
Family         : ordinal
Link           : logit

Response       : att5
Family         : ordinal
Link           : logit

Log likelihood = -6394.3923

 ( 1)  [q1]MathAb = 1
 ( 2)  [att1]MathAtt = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
q1						
MathAb	1	(constrained)				
_cons	.044612	.1272967	0.35	0.726	-.204885	.294109
q2						
MathAb	.3446066	.1050261	3.28	0.001	.1387593	.550454
_cons	-.4572215	.0979965	-4.67	0.000	-.6492911	-.2651519
q3						
MathAb	.5445222	.1386992	3.93	0.000	.2726767	.8163677
_cons	.1591406	.1033116	1.54	0.123	-.0433465	.3616276
q4						
MathAb	.2858862	.0948549	3.01	0.003	.099974	.4717984
_cons	-.3196648	.0947684	-3.37	0.001	-.5054075	-.1339222
q5						
MathAb	.8174769	.1867022	4.38	0.000	.4515473	1.183406
_cons	-.04543	.116575	-0.39	0.697	-.2739129	.1830528
q6						
MathAb	.6030423	.1471949	4.10	0.000	.3145457	.8915389
_cons	-.3099919	.1070853	-2.89	0.004	-.5198754	-.1001085
q7						
MathAb	.7208369	.171309	4.21	0.000	.3850774	1.056597
_cons	.1047264	.1116494	0.94	0.348	-.1141024	.3235553
q8						
MathAb	.5814736	.1426725	4.08	0.000	.3018406	.8611067
_cons	-.0250443	.1045135	-0.24	0.811	-.2298869	.1797984
att1						
MathAtt	1	(constrained)				
att2						
MathAtt	.3788715	.0971234	3.90	0.000	.1885131	.5692299
att3						
MathAtt	-1.592717	.3614956	-4.41	0.000	-2.301236	-.8841989
att4						
MathAtt	-.8100108	.1530675	-5.29	0.000	-1.110017	-.510004
att5						
MathAtt	.5225425	.1170166	4.47	0.000	.2931942	.7518907
MathAb						
MathAtt	.581103	.14776	3.93	0.000	.2914987	.8707072
/att1						
cut1	-1.10254	.131228			-1.359742	-.8453377
cut2	-.2495339	.1160385			-.4769652	-.0221025
cut3	.2983261	.1164415			.070105	.5265472
cut4	1.333052	.1391919			1.060241	1.605864

/att2				
cut1	-1.055791	.1062977	-1.264131	-.8474513
cut2	-.1941211	.0941435	-.378639	-.0096032
cut3	.3598488	.0952038	.1732528	.5464448
cut4	1.132624	.1082204	.9205156	1.344732
/att3				
cut1	-1.053519	.1734001	-1.393377	-.7136612
cut2	-.0491074	.1442846	-.3319	.2336853
cut3	.5570672	.1538702	.2554871	.8586472
cut4	1.666859	.2135557	1.248297	2.08542
/att4				
cut1	-1.07378	.1214071	-1.311734	-.8358264
cut2	-.2112462	.1076501	-.4222365	-.0002559
cut3	.406347	.1094847	.191761	.620933
cut4	1.398185	.1313327	1.140778	1.655593
/att5				
cut1	-1.244051	.1148443	-1.469142	-1.018961
cut2	-.336135	.0986678	-.5295203	-.1427498
cut3	.2137776	.0978943	.0219084	.4056468
cut4	.9286849	.107172	.7186316	1.138738
var(e.MathAb)	1.787117	.5974753	.9280606	3.441357
var(MathAtt)	1.520854	.4077885	.8991947	2.572298

Notes:

1. In the model fit in [SEM] **Example 31g**, we estimated a correlation between `MathAtt` and `MathAb` of 0.4725.
2. Theoretically speaking, the model fit above and the model in [SEM] **Example 31g** are equivalent. Both posit a linear relationship between the latent variables and merely choose to parameterize the relationship differently. In [SEM] **Example 31g**, it was parameterized as a covariance. In this example, it is parameterized as causal. People often use structural equation modeling to confirm a proposed hypothesis. It is important that the causal model you specify be based on theory or that you have some other justification. You need something other than empirical results to rule out competing but equivalent models such as the covariance model. Distinguishing causality from correlation is always problematic.
3. Practically speaking, note that the log-likelihood values for this model and the model in [SEM] **Example 31g** are equal at −6394.3923. Also note that the estimated variances of math attitude, `var(MathAtt)`, are also equal at 1.520854.

Fitting the model with the Builder

Use the diagram in *Structural model with measurement component* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cfa
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.
4. Create the measurement component for `MathAb`.

   Select the Add Measurement Component tool, , and then click in the diagram about one-third of the way up from the bottom and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAb`;

   b. select `q1`, `q2`, `q3`, `q4`, `q5`, `q6`, `q7`, and `q8` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Bernoulli, Logit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.
5. Create the measurement component for `MathAtt`.

   Select the Add Measurement Component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `MathAtt`;

   b. select `att1`, `att2`, `att3`, `att4`, and `att5` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Ordinal, Logit` in the *Family/Link* control;

   e. select `Up` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.
6. Create path from `MathAtt` to `MathAb`.

   a. Select the Add Path tool, .

   b. Click in the bottom of the `MathAtt` oval (it will highlight when you hover over it), and drag a path to the top of the `MathAb` oval (it will highlight when you can release to connect the path).
7. Clean up the direction of the error.

   The error on `MathAb` has likely been created below the oval instead of to the right of the oval. If so, choose the Select tool, , and then click in the `MathAb` oval. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is where you want it.
8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_sem
```

Also see

[SEM] **Example 9** — Structural model with measurement component

[SEM] **Example 31g** — Two-factor measurement model (generalized response)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 33g — Logistic regression

Description Remarks and examples Reference Also see

Description

In this example, we demonstrate with **gsem** how to fit a standard logistic regression, which is often referred to as the logit model in generalized linear model (GLM) framework.

```
. use https://www.stata-press.com/data/r16/gsem_lbw
(Hosmer & Lemeshow data)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_lbw.dta
  obs:           189                          Hosmer & Lemeshow data
 vars:            11                          21 Mar 2018 12:28
                                              (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
id              int     %8.0g                 subject id
low             byte    %8.0g                 birth weight < 2500g
age             byte    %8.0g                 age of mother
lwt             int     %8.0g                 weight, last menstrual period
race            byte    %8.0g      race       race
smoke           byte    %9.0g      smoke      smoked during pregnancy
ptl             byte    %8.0g                 premature labor history (count)
ht              byte    %8.0g                 has history of hypertension
ui              byte    %8.0g                 presence, uterine irritability
ftv             byte    %8.0g                 # physician visits, 1st trimester
bwt             int     %8.0g                 birth weight (g)
-------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Data from Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013.
      "Applied Logistic Regression". 3rd ed. Hoboken, NJ: Wiley.
  2.  Data from a study of risk factors associated with low birth weights.
```

See *Structural models 3: Binary-outcome models* in [SEM] **Intro 5** for background.

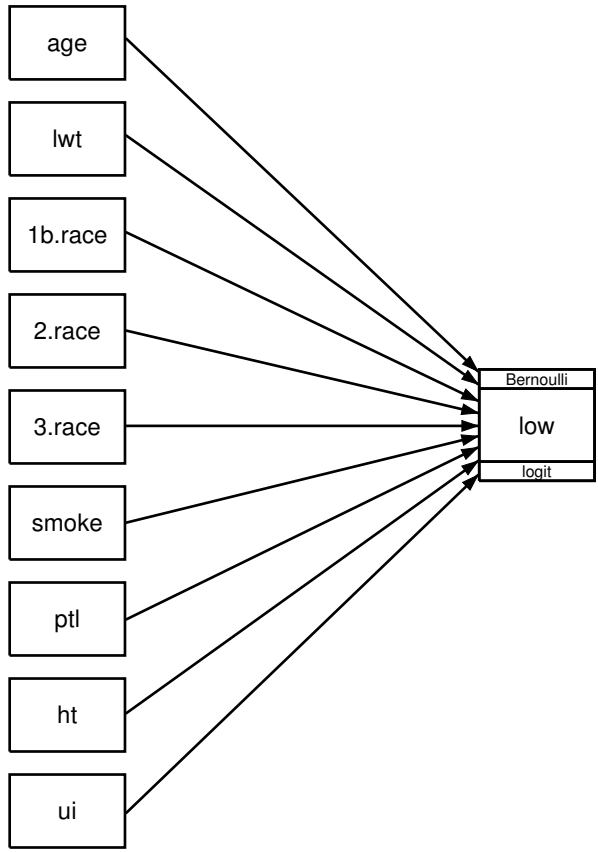
Remarks and examples

Remarks are presented under the following headings:

Fitting the logit model
Obtaining odds ratios
Fitting the model with the Builder

Fitting the logit model

The model we wish to fit is



That is, we wish to fit a model in which low birthweight is determined by a history of hypertension (`ht`), mother's age (`age`), mother's weight at last menstrual period (`lwt`), mother's race (white, black, or other; `race`), whether the mother smoked during pregnancy (`smoke`), the number of premature babies previously born to the mother (`ptl`), and whether the mother has suffered from the presence of uterine irritability (`ui`).

The path diagram matches the variable names listed in parentheses above except for `race`, where the path diagram contains not one but three boxes filled in with `1b.race`, `2.race`, and `3.race`. This is because in our dataset, race is coded 1, 2, or 3, meaning white, black, or other. We want to include indicator variables for `race` so that we have a separate coefficient for each race. Thus we need three boxes.

In Stata, `1.race` means "an indicator for race equaling 1". Thus it should not surprise you if you filled in the boxes with `1.race`, `2.race`, and `3.race`, and that is almost what we did. The difference is that we filled in the first box with `1b.race` rather than `1.race`. We use the `b` to specify the base category, which we specified as white. If we wanted the base category to be black, we would have specified `2b.race` and left `1.race` alone.

The above is called factor-variable notation. See [SEM] **Intro 3** for details on using factor-variable notation with `gsem`.

In the command language, we could type

```
. gsem (low <- age lwt 1b.race 2.race 3.race smoke ptl ht ui), logit
```

to fit the model. Written that way, there is a one-to-one correspondence to what we would type and what we would draw in the Builder. The command language, however, has a feature that will allow us to type `i.race` instead of `1b.race 2.race 3.race`. To fit the model, we could type

```
. gsem (low <- age lwt i.race smoke ptl ht ui), logit
```

`i.`*varname* is a command-language shorthand for specifying indicators for all the levels of a variable and using the first level as the base category. You can use `i.`*varname* in the command language but not in path diagrams because boxes can contain only one variable. In the Builder, however, you will discover a neat feature so that you can type `i.race` and the Builder will create however many boxes are needed for you, filled in, and with the first category marked as the base. We will explain below how you do that.

The result of typing our estimation command is

```
. gsem (low <- age lwt i.race smoke ptl ht ui), logit
Iteration 0:   log likelihood =  -101.0213
Iteration 1:   log likelihood = -100.72519
Iteration 2:   log likelihood =   -100.724
Iteration 3:   log likelihood =   -100.724
Generalized structural equation model           Number of obs     =        189
Response       : low
Family         : Bernoulli
Link           : logit
Log likelihood =   -100.724
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
low						
age	-.0271003	.0364504	-0.74	0.457	-.0985418	.0443412
lwt	-.0151508	.0069259	-2.19	0.029	-.0287253	-.0015763
race						
black	1.262647	.5264101	2.40	0.016	.2309024	2.294392
other	.8620792	.4391532	1.96	0.050	.0013548	1.722804
smoke	.9233448	.4008266	2.30	0.021	.137739	1.708951
ptl	.5418366	.346249	1.56	0.118	-.136799	1.220472
ht	1.832518	.6916292	2.65	0.008	.4769494	3.188086
ui	.7585135	.4593768	1.65	0.099	-.1418484	1.658875
_cons	.4612239	1.20459	0.38	0.702	-1.899729	2.822176

Obtaining odds ratios

Some of you are looking at the output above, nodding your heads, and thinking to yourselves, "Yes, that's right." Others are shaking your heads sadly and thinking, "Where are the exponentiated coefficients, the odds ratios?" Researchers from different backgrounds are used to seeing logit results presented in two different ways.

If you want to see the odds ratios, type `estat eform` after fitting the model:

```
. estat eform
```

	exp(b)	Std. Err.	z	P>|z|	[95% Conf.	Interval]
low						
age	.9732636	.0354759	-0.74	0.457	.9061578	1.045339
lwt	.9849634	.0068217	-2.19	0.029	.9716834	.9984249
race						
black	3.534767	1.860737	2.40	0.016	1.259736	9.918406
other	2.368079	1.039949	1.96	0.050	1.001356	5.600207
smoke	2.517698	1.00916	2.30	0.021	1.147676	5.523162
ptl	1.719161	.5952579	1.56	0.118	.8721455	3.388787
ht	6.249602	4.322408	2.65	0.008	1.611152	24.24199
ui	2.1351	.9808153	1.65	0.099	.8677528	5.2534
_cons	1.586014	1.910496	0.38	0.702	.1496092	16.8134

Whichever way you look at the results above, they are identical to the results that would be produced by typing

```
. logit low age lwt i.race smoke ptl ht ui
```

or

```
. logistic low age lwt i.race smoke ptl ht ui
```

which are two other ways that Stata can fit logit models. `logit`, like `gsem`, reports coefficients by default. `logistic` reports odds ratios by default.

Fitting the model with the Builder

Use the diagram in *Fitting the logit model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_lbw
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Enlarge the size of the canvas to accommodate the height of the diagram.

   Click on the **Adjust canvas size** button, , in the Standard Toolbar, change the second size to `5` (inches), and then click on **OK**.

5. Create the logistic regression component for `low`.

   Select the Add regression component tool, , and then click in the diagram about one-third of the way in from the left and halfway down.

   In the resulting dialog box,

   a. select `low` in the *Dependent variable* control;

b. check *Make measurements generalized*;

c. select `Bernoulli, Logit` in the *Family/Link* control;

d. select the *Select variables* radio button (it may already be selected);

e. use the *Independent variables* control to select the variables `age` and `lwt`;

f. include the levels of the factor variable `race` by clicking on the "" button next to the *Independent variables* control. In the resulting dialog box, select the *Factor variable* radio button, select `Main effect` in the *Specification* control, and select `race` in the *Variables* control for *Variable 1*. Click on **Add to varlist**, and then click on **OK**;

g. continue with the *Independent variables* control to select the variables `smoke`, `ptl`, `ht`, and `ui`;

h. select `Left` in the *Independent variables' direction* control;

i. click on **OK**.

If you wish, move the component by clicking on any variable and dragging it.

6. Clean up.

   The box for `low` is created closer to the independent variables than it is in the example diagram. Use the Select tool, , and click on the box for `low`. Drag it to the right to allow more space for results along the paths.

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_logit
```

Reference

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

Also see

[SEM] **Example 34g** — Combined models (generalized responses)

[SEM] **Example 35g** — Ordered probit and ordered logit

[SEM] **Example 37g** — Multinomial logistic regression

[SEM] **Intro 3** — Learning the language: Factor-variable notation (gsem only)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat eform** — Display exponentiated coefficients

Title

Example 34g — Combined models (generalized responses)

Description Remarks and examples Reference Also see

Description

We demonstrate how to fit a combined model with one Poisson regression and one logit regression by using the following data:

```
. use https://www.stata-press.com/data/r16/gsem_lbw
(Hosmer & Lemeshow data)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_lbw.dta
  obs:           189                          Hosmer & Lemeshow data
 vars:            11                          21 Mar 2018 12:28
                                              (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
id              int     %8.0g                 subject id
low             byte    %8.0g                 birth weight < 2500g
age             byte    %8.0g                 age of mother
lwt             int     %8.0g                 weight, last menstrual period
race            byte    %8.0g      race       race
smoke           byte    %9.0g      smoke      smoked during pregnancy
ptl             byte    %8.0g                 premature labor history (count)
ht              byte    %8.0g                 has history of hypertension
ui              byte    %8.0g                 presence, uterine irritability
ftv             byte    %8.0g                 # physician visits, 1st trimester
bwt             int     %8.0g                 birth weight (g)
------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Data from Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013.
      "Applied Logistic Regression". 3rd ed. Hoboken, NJ: Wiley.
  2.  Data from a study of risk factors associated with low birth weights.
```

See *Structural models 8: Dependencies between response variables* in [SEM] **Intro 5** for background.

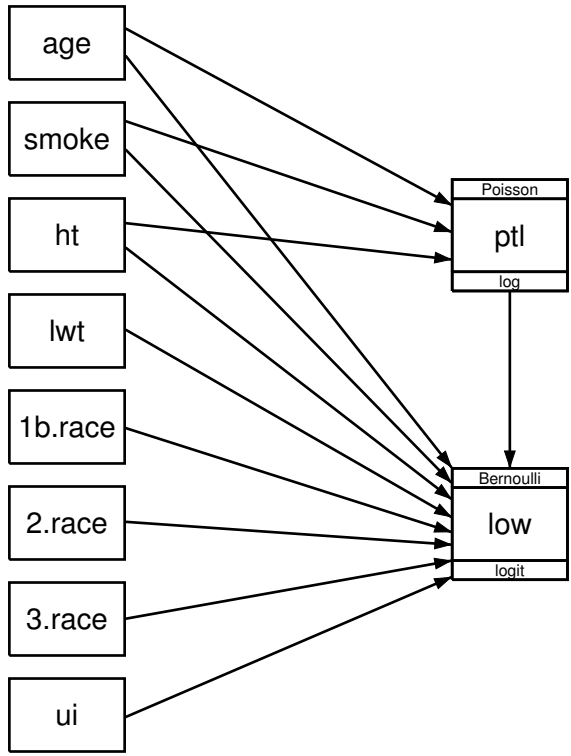
Remarks and examples

Remarks are presented under the following headings:

Fitting the combined model
Obtaining odds ratios and incidence-rate ratios
Fitting the model with the Builder

Fitting the combined model

The model we wish to fit is



This model has one logit equation and one Poisson regression equation, with the Poisson response variable also being an explanatory variable in the logit equation.

Because the two equations are recursive, it is not necessary to fit these models together. We could draw separate diagrams for each equation and fit each separately. Even so, many researchers often do fit recursive models together, and sometimes, it is just the first step before placing constraints across models or introducing a common latent variable. The latter might be likely in this case because neither generalized linear response has an error that could be correlated and so the only way to correlate these two responses in `gsem` is to add a shared latent variable affecting each.

Our purpose here is to show that you can mix models with generalized response variables of different types.

To fit the model in the command language, we type

```
. gsem (low <- ptl age smoke ht lwt i.race ui, logit)
>      (ptl <- age smoke ht, poisson)
Iteration 0:   log likelihood = -322.96738
Iteration 1:   log likelihood =  -200.5818
Iteration 2:   log likelihood = -198.58086
Iteration 3:   log likelihood = -198.56179
Iteration 4:   log likelihood = -198.56178
Generalized structural equation model           Number of obs     =        189
Response       : low
Family         : Bernoulli
Link           : logit
Response       : ptl
Family         : Poisson
Link           : log
Log likelihood = -198.56178
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
low						
ptl	.5418366	.346249	1.56	0.118	-.136799	1.220472
age	-.0271003	.0364504	-0.74	0.457	-.0985418	.0443412
smoke	.9233448	.4008266	2.30	0.021	.137739	1.708951
ht	1.832518	.6916292	2.65	0.008	.4769494	3.188086
lwt	-.0151508	.0069259	-2.19	0.029	-.0287253	-.0015763
race						
black	1.262647	.5264101	2.40	0.016	.2309023	2.294392
other	.8620791	.4391532	1.96	0.050	.0013548	1.722803
ui	.7585135	.4593768	1.65	0.099	-.1418484	1.658875
_cons	.4612238	1.20459	0.38	0.702	-1.899729	2.822176
ptl						
age	.0370598	.0298752	1.24	0.215	-.0214946	.0956142
smoke	.9602534	.3396867	2.83	0.005	.2944796	1.626027
ht	-.1853501	.7271851	-0.25	0.799	-1.610607	1.239906
_cons	-2.985512	.7842174	-3.81	0.000	-4.52255	-1.448474

Obtaining odds ratios and incidence-rate ratios

As mentioned in [SEM] **Example 33g**, some researchers prefer to see exponentiated coefficients. In both odds ratios and incidence-rate ratios, exponentiation is meaningful. Exponentiated logit coefficients are odds ratios, and exponentiated Poisson regression coefficients are incidence-rate ratios. To obtain exponentiated coefficients for both equations, we type

```
. estat eform low ptl
```

	exp(b)	Std. Err.	z	P>|z|	[95% Conf.	Interval]
low						
ptl	1.719161	.5952579	1.56	0.118	.8721455	3.388787
age	.9732636	.0354759	-0.74	0.457	.9061578	1.045339
smoke	2.517698	1.00916	2.30	0.021	1.147676	5.523162
ht	6.249602	4.322407	2.65	0.008	1.611152	24.24199
lwt	.9849634	.0068217	-2.19	0.029	.9716834	.9984249
race						
black	3.534767	1.860737	2.40	0.016	1.259736	9.918406
other	2.368079	1.039949	1.96	0.050	1.001356	5.600207
ui	2.1351	.9808153	1.65	0.099	.8677528	5.2534
_cons	1.586014	1.910496	0.38	0.702	.1496092	16.8134
ptl						
age	1.037755	.0310032	1.24	0.215	.9787348	1.100334
smoke	2.612358	.8873835	2.83	0.005	1.342428	5.083638
ht	.8308134	.6041551	-0.25	0.799	.1997664	3.45529
_cons	.0505137	.0396137	-3.81	0.000	.0108613	.2349286

Had we merely typed `estat eform` without the two equation names, we would have obtained exponentiated coefficients for the first equation only.

Equation names are easily found on the output or the path diagrams. Equations are named after the dependent variable.

Fitting the model with the Builder

Use the diagram in *Fitting the combined model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_lbw
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the independent variables.

   Select the Add observed variables set tool, , and then click at the bottom of the diagram about one-third of the way in from the left.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the variables `age`, `smoke`, `ht`, and `lwt` in this order;

   c. include the levels of the factor variable `race` by clicking on the button next to the *Variables* control. In the resulting dialog box, select the *Factor variable* radio button, select `Main effect` in the *Specification* control, and select `race` in the *Variables* control for *Variable 1*. Click on **Add to varlist**, and then click on **OK**;

d. continue with the *Variables* control and select the variable `ui`;

e. select `Vertical` in the *Orientation* control;

f. click on **OK**.

If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the generalized response for premature labor history.

   a. Select the Add generalized response variable tool, ▭.

   b. Click about one-third of the way in from the right side of the diagram, to the right of `ht`.

   c. In the Contextual Toolbar, select `Poisson, Log` in the *Family/Link* control.

   d. In the Contextual Toolbar, select `ptl` in the *Variable* control.

6. Create the generalized response for low birthweight.

   a. Select the Add generalized response variable tool, ▭.

   b. Click about one-third of the way in from the right side of the diagram, to the right of `2.race`.

   c. In the Contextual Toolbar, select `Bernoulli, Logit` in the *Family/Link* control.

   d. In the Contextual Toolbar, select `low` in the *Variable* control.

7. Create paths from the independent variables to the dependent variables.

   a. Select the Add path tool, →.

   b. Click in the right side of the `age` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `ptl` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the → tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable:

```
smoke -> ptl
ht -> ptl
age -> low
smoke -> low
ht -> low
lwt -> low
1b.race -> low
2.race -> low
3.race -> low
ui -> low
```

   d. Continuing with the → tool, create the path from `ptl` to `low` by clicking in the bottom of the `ptl` rectangle and dragging the path to the top of the `low` rectangle.

8. Clean up.

   If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.

9. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_comb
```

Reference

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

Also see

[SEM] **Example 33g** — Logistic regression

[SEM] **Example 45g** — Heckman selection model

[SEM] **Example 46g** — Endogenous treatment-effects model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat eform** — Display exponentiated coefficients

Title

Example 35g — Ordered probit and ordered logit

Description Remarks and examples Reference Also see

Description

Below we demonstrate ordered probit and ordered logit in a measurement-model context. We are not going to illustrate every family/link combination. Ordered probit and logit, however, are unique in that a single equation is able to predict a set of ordered outcomes. The unordered alternative, `mlogit`, requires $k - 1$ equations to fit k (unordered) outcomes.

To demonstrate ordered probit and ordered logit, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_issp93
(Selection from ISSP 1993)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_issp93.dta
  obs:           871                          Selection for ISSP 1993
 vars:             8                          21 Mar 2018 16:03
                                               (_dta has notes)
```

variable name	storage type	display format	value label	variable label
id	int	%9.0g		respondent identifier
y1	byte	%26.0g	agree5	too much science, not enough feelings & faith
y2	byte	%26.0g	agree5	science does more harm than good
y3	byte	%26.0g	agree5	any change makes nature worse
y4	byte	%26.0g	agree5	science will solve environmental problems
sex	byte	%9.0g	sex	sex
age	byte	%9.0g	age	age (6 categories)
edu	byte	%20.0g	edu	education (6 categories)

```
Sorted by:
. notes
_dta:
  1.  Data from Greenacre, M. and J Blasius, 2006, _Multiple Correspondence
      Analysis and Related Methods_, pp. 42-43, Boca Raton: Chapman & Hall.
      Data is a subset of the International Social Survey Program (ISSP) 1993.
  2.  Full text of y1: We believe too often in science, and not enough in
      feelings and faith.
  3.  Full text of y2: Overall, modern science does more harm than good.
  4.  Full text of y3: Any change humans cause in nature, no matter how
      scientific, is likely to make things worse.
  5.  Full text of y4: Modern science will solve our environmental problems
      with little change to our way of life.
```

See *Structural models 5: Ordinal models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

Ordered probit
Ordered logit
Fitting the model with the Builder

Ordered probit

For the measurement model, we focus on variables `y1` through `y4`. Each variable contains 1–5, with 1 meaning strong disagreement and 5 meaning strong agreement with a statement about science.

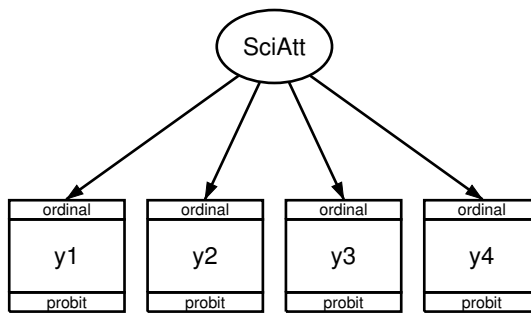
Ordered probit produces predictions about the probabilities that a respondent gives response 1, response 2, ..., response k. It does this by dividing up the domain of an $N(0,1)$ distribution into k categories defined by $k-1$ cutpoints, c_1, c_2, ..., c_{k-1}. Individual respondents are assumed to have a score $s = X\beta + \epsilon$, where $\epsilon \sim N(0,1)$, and then that score is used along with the cutpoints to produce probabilities for each respondent producing response 1, 2, ..., k.

$$\Pr(\text{response is } i \,|\, X) = \Pr(c_{i-1} < X\beta + \epsilon \leq c_i)$$

where $c_0 = -\infty$; $c_k = +\infty$; and c_1, c_2, ..., c_{k-1} and β are parameters of the model to be fit. This ordered probit model has long been known in Stata circles as oprobit.

We have a set of four questions designed to determine the respondent's attitude toward science, each question with $k = 5$ possible answers ranging on a Likert scale from 1 to 5. With ordered probit in hand, we have a way to take a continuous variable, say, a latent variable we will call `SciAtt`, and produce predicted categorical responses.

The measurement model we want to fit is



We fit the model in the command language by typing

```
. gsem (y1 y2 y3 y4 <- SciAtt), oprobit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -5227.8743
Iteration 1:   log likelihood = -5227.8743
Refining starting values:
Grid node 0:   log likelihood = -5230.8106
Fitting full model:
Iteration 0:   log likelihood = -5230.8106  (not concave)
Iteration 1:   log likelihood = -5132.1849  (not concave)
Iteration 2:   log likelihood = -5069.5037
Iteration 3:   log likelihood = -5040.4779
Iteration 4:   log likelihood = -5040.2397
Iteration 5:   log likelihood = -5039.8242
Iteration 6:   log likelihood =  -5039.823
Iteration 7:   log likelihood =  -5039.823
Generalized structural equation model           Number of obs     =        871
Response       : y1
Family         : ordinal
Link           : probit
Response       : y2
Family         : ordinal
Link           : probit
Response       : y3
Family         : ordinal
Link           : probit
Response       : y4
Family         : ordinal
Link           : probit
Log likelihood =  -5039.823
 ( 1)  [y1]SciAtt = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1						
SciAtt	1	(constrained)				
y2						
SciAtt	1.424366	.2126574	6.70	0.000	1.007565	1.841167
y3						
SciAtt	1.283359	.1797557	7.14	0.000	.931044	1.635674
y4						
SciAtt	-.0322354	.0612282	-0.53	0.599	-.1522405	.0877697
/y1						
cut1	-1.343148	.0726927			-1.485623	-1.200673
cut2	.0084719	.0521512			-.0937426	.1106863
cut3	.7876538	.0595266			.6709837	.9043238
cut4	1.989985	.0999181			1.794149	2.18582

/y2				
cut1	-1.997245	.1311972	-2.254387	-1.740104
cut2	-.8240241	.0753839	-.9717738	-.6762743
cut3	.0547025	.0606036	-.0640784	.1734834
cut4	1.419923	.1001258	1.22368	1.616166
/y3				
cut1	-1.271915	.0847483	-1.438019	-1.105812
cut2	.1249493	.0579103	.0114472	.2384515
cut3	.9752553	.0745052	.8292277	1.121283
cut4	2.130661	.1257447	1.884206	2.377116
/y4				
cut1	-1.484063	.0646856	-1.610844	-1.357281
cut2	-.4259356	.0439145	-.5120065	-.3398647
cut3	.1688777	.0427052	.0851771	.2525782
cut4	.9413113	.0500906	.8431356	1.039487
var(SciAtt)	.5265523	.0979611	.3656637	.7582305

Notes:

1. The cutpoints $c_1, \ldots, c_4$ are labeled `cut1`, ..., `cut4` in the output. We have a separate cutpoint for each of the four questions `y1`, ..., `y4`. Look at the estimated cutpoints for `y1`, which are -1.343, 0.008, 0.788, and 1.99. The probabilities that a person with `SciAtt` $= 0$ (its mean) would give the various responses are

$$\Pr(\text{response } 1) = \texttt{normal(-1.343)} = 0.090$$
$$\Pr(\text{response } 2) = \texttt{normal(0.008)} - \texttt{normal}(-1.343) = 0.414$$
$$\Pr(\text{response } 3) = \texttt{normal(0.788)} - \texttt{normal}(0.008) = 0.281$$
$$\Pr(\text{response } 4) = \texttt{normal(1.99)} - \texttt{normal}(0.788) = 0.192$$
$$\Pr(\text{response } 5) = 1 - \texttt{normal(1.99)} = 0.023$$

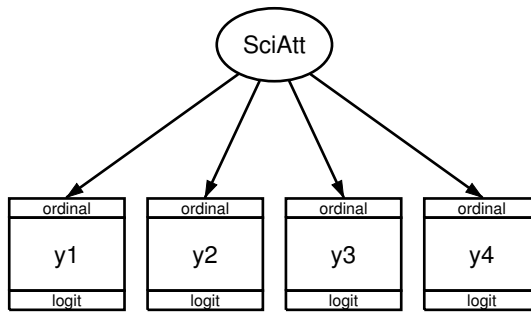
2. The path coefficients `(y1 y2 y3 y4 <- SciAtt)` measure the effect of the latent variable we called science attitude on each of the responses.
3. The estimated path coefficients are 1, 1.42, 1.28, and -0.03 for the four questions.
4. If you read the questions—they are listed above—you will find that in all but the fourth question, agreement signifies a negative attitude toward science. Thus `SciAtt` measures a negative attitude toward science because the loadings on negative questions are positive and the loading on the single positive question is negative.
5. The direction of the meanings of latent variables is always a priori indeterminate and is set by the identifying restrictions we apply. We applied—or more correctly, `gsem` applied for us—the constraint that `y1 <- SciAtt` has path coefficient 1. Because statement 1 was a negative statement about science, that was sufficient to set the direction of `SciAtt` to be the opposite of what we hoped for.

   The direction does not matter. You simply must remember to interpret the latent variable correctly when reading results based on it. In the models we fit, including more complicated models, the signs of the coefficients will work themselves out to adjust for the direction of the variable.

Ordered logit

The description of the ordered logit model is identical to that of the ordered probit model except that where we assumed a normal distribution in our explanation above, we now assume a logit distribution. The distributions are similar.

To fit an ordered logit (ologit) model, the link function shown in the boxes merely changes from probit to logit:



We can fit the model in the command language by typing

```
. gsem (y1 y2 y3 y4 <- SciAtt), ologit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -5227.8743
Iteration 1:   log likelihood = -5227.8743
Refining starting values:
Grid node 0:   log likelihood = -5127.9026
Fitting full model:
Iteration 0:   log likelihood = -5127.9026  (not concave)
Iteration 1:   log likelihood = -5065.4679
Iteration 2:   log likelihood = -5035.9766
Iteration 3:   log likelihood = -5035.0943
Iteration 4:   log likelihood = -5035.0353
Iteration 5:   log likelihood = -5035.0352
Generalized structural equation model           Number of obs     =        871
Response       : y1
Family         : ordinal
Link           : logit
Response       : y2
Family         : ordinal
Link           : logit
Response       : y3
Family         : ordinal
Link           : logit
Response       : y4
Family         : ordinal
Link           : logit
Log likelihood = -5035.0352
 ( 1)  [y1]SciAtt = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1						
SciAtt	1	(constrained)				
y2						
SciAtt	1.394767	.2065479	6.75	0.000	.9899406	1.799593
y3						
SciAtt	1.29383	.1845113	7.01	0.000	.9321939	1.655465
y4						
SciAtt	-.0412446	.0619936	-0.67	0.506	-.1627498	.0802606
/y1						
cut1	-2.38274	.1394292			-2.656016	-2.109464
cut2	-.0088393	.0889718			-.1832207	.1655422
cut3	1.326292	.106275			1.117997	1.534587
cut4	3.522017	.1955535			3.138739	3.905295
/y2						
cut1	-3.51417	.2426595			-3.989774	-3.038566
cut2	-1.421711	.135695			-1.687669	-1.155754
cut3	.0963154	.1046839			-.1088612	.3014921
cut4	2.491459	.1840433			2.130741	2.852178
/y3						
cut1	-2.263557	.1618806			-2.580838	-1.946277
cut2	.2024798	.1012122			.0041075	.400852
cut3	1.695997	.1393606			1.422855	1.969138
cut4	3.828154	.2464566			3.345108	4.3112
/y4						
cut1	-2.606013	.1338801			-2.868413	-2.343613
cut2	-.6866159	.0718998			-.8275369	-.5456949
cut3	.268862	.0684577			.1346874	.4030366
cut4	1.561921	.0895438			1.386419	1.737424
var(SciAtt)	1.715641	.3207998			1.189226	2.475077

Note:

1. Results are nearly identical to those reported for ordered probit.

Fitting the model with the Builder

Use the diagram in *Ordered probit* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_issp93
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the measurement component for `SciAtt`.

   Select the Add measurement component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `SciAtt`;

   b. select `y1`, `y2`, `y3`, and `y4` by using the *Measurement variables* control;

   c. check *Make measurements generalized*;

   d. select `Ordinal, Probit` in the *Family/Link* control;

   e. select `Down` in the *Measurement direction* control;

   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

6. To fit the model in *Ordered logit*, change the type of generalized response for each of the measurement variables.

   a. Choose the Select tool, .

   b. Click on the `y1` rectangle. In the Contextual Toolbar, select `Ordinal, Logit` in the *Family/Link* control.

   c. Repeat this process to change the family and link to `Ordinal, Logit` for `y2`, `y3`, and `y4`.

7. Estimate again.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram for the ordered probit model in the Builder by typing

```
. webgetsem gsem_oprobit
```

You can open a completed diagram for the ordered logit model in the Builder by typing

```
. webgetsem gsem_ologit
```

Reference

Greenacre, M. J. 2006. From simple to multiple correspondence analysis. In *Multiple Correspondence Analysis and Related Methods*, ed. M. J. Greenacre and J. Blasius. Boca Raton, FL: Chapman & Hall.

Also see

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Example 27g** — Single-factor measurement model (generalized response)

[SEM] **Example 33g** — Logistic regression

[SEM] **Example 36g** — MIMIC model (generalized response)

[SEM] **Example 37g** — Multinomial logistic regression

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 36g — MIMIC model (generalized response)

Description Remarks and examples Reference Also see

Description

To demonstrate a multiple-indicators multiple-causes (MIMIC) model with generalized indicators, we use the same data used in [SEM] **Example 35g**:

```
. use https://www.stata-press.com/data/r16/gsem_issp93
(Selection from ISSP 1993)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_issp93.dta
  obs:           871                          Selection for ISSP 1993
 vars:             8                          21 Mar 2018 16:03
                                               (_dta has notes)
```

variable name	storage type	display format	value label	variable label
id	int	%9.0g		respondent identifier
y1	byte	%26.0g	agree5	too much science, not enough feelings & faith
y2	byte	%26.0g	agree5	science does more harm than good
y3	byte	%26.0g	agree5	any change makes nature worse
y4	byte	%26.0g	agree5	science will solve environmental problems
sex	byte	%9.0g	sex	sex
age	byte	%9.0g	age	age (6 categories)
edu	byte	%20.0g	edu	education (6 categories)

```
Sorted by:
. notes
_dta:
  1.  Data from Greenacre, M. and J Blasius, 2006, _Multiple Correspondence
      Analysis and Related Methods_, pp. 42-43, Boca Raton: Chapman & Hall.
      Data is a subset of the International Social Survey Program (ISSP) 1993.
  2.  Full text of y1: We believe too often in science, and not enough in
      feelings and faith.
  3.  Full text of y2: Overall, modern science does more harm than good.
  4.  Full text of y3: Any change humans cause in nature, no matter how
      scientific, is likely to make things worse.
  5.  Full text of y4: Modern science will solve our environmental problems
      with little change to our way of life.
```

See *Structural models 10: MIMIC models* in [SEM] **Intro 5** for background.

Remarks and examples

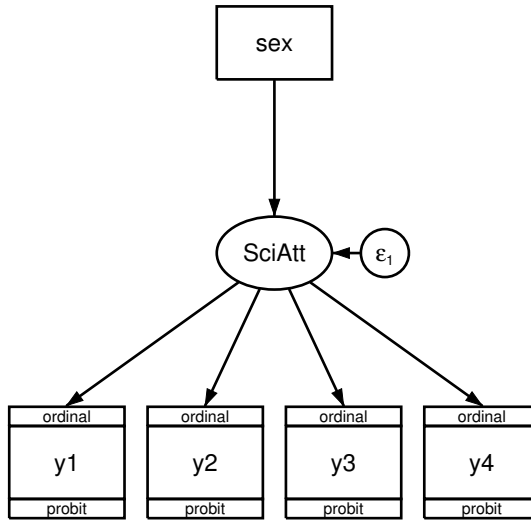
Remarks are presented under the following headings:

Fitting the MIMIC model
Fitting the model with the Builder

Fitting the MIMIC model

For a refresher on MIMIC models, see [SEM] **Example 10**. In the data above, we have ordered categorical indicators. For an explanation on how we are about to treat these indicators with ordered probit, see [SEM] **Example 35g**.

We wish to fit the following model:



We fit the model in the command language by typing

```
. gsem (y1 y2 y3 y4 <- SciAtt) (SciAtt <- sex), oprobit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -5227.8743
Iteration 1:   log likelihood = -5227.8743
Refining starting values:
Grid node 0:   log likelihood = -5230.8106
Fitting full model:
Iteration 0:   log likelihood = -5230.8106  (not concave)
Iteration 1:   log likelihood = -5132.1065  (not concave)
Iteration 2:   log likelihood = -5066.8481
Iteration 3:   log likelihood = -5033.2807
Iteration 4:   log likelihood = -5032.7884
Iteration 5:   log likelihood = -5032.3837
Iteration 6:   log likelihood = -5032.3778
Iteration 7:   log likelihood = -5032.3778
```

```
Generalized structural equation model           Number of obs     =        871
Response       : y1
Family         : ordinal
Link           : probit
Response       : y2
Family         : ordinal
Link           : probit
Response       : y3
Family         : ordinal
Link           : probit
Response       : y4
Family         : ordinal
Link           : probit
Log likelihood = -5032.3778
 ( 1)  [y1]SciAtt = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1						
SciAtt	1	(constrained)				
y2						
SciAtt	1.405732	.2089672	6.73	0.000	.9961641	1.8153
y3						
SciAtt	1.246449	.1710771	7.29	0.000	.911144	1.581754
y4						
SciAtt	-.0345517	.0602017	-0.57	0.566	-.1525449	.0834415
SciAtt						
sex	-.2337427	.0644245	-3.63	0.000	-.3600124	-.1074729
/y1						
cut1	-1.469615	.0855651			-1.63732	-1.301911
cut2	-.10992	.0615897			-.2306336	.0107937
cut3	.6729334	.0644695			.5465755	.7992914
cut4	1.879901	.0996675			1.684557	2.075246
/y2						
cut1	-2.16739	.1480596			-2.457582	-1.877199
cut2	-.9912152	.0943091			-1.176058	-.8063727
cut3	-.1118914	.075311			-.2594982	.0357154
cut4	1.252164	.0983918			1.05932	1.445008
/y3						
cut1	-1.412372	.0977772			-1.604012	-1.220733
cut2	-.0230879	.0687432			-.1578221	.1116464
cut3	.8209522	.0771653			.6697109	.9721935
cut4	1.966042	.1196586			1.731515	2.200568
/y4						
cut1	-1.47999	.0650596			-1.607505	-1.352476
cut2	-.4218768	.0443504			-.508802	-.3349516
cut3	.172995	.0432394			.0882473	.2577427
cut4	.9454906	.0507422			.8460376	1.044944
var(e.SciAtt)	.5283629	.0978703			.3675036	.7596315

Notes:

1. Our latent variable measures a negative attitude toward science, just as it did in [SEM] **Example 35g**.
2. In this MIMIC model, we allow males and females to have different underlying attitudes toward science.
3. The coefficient for `SciAtt <- sex` is -0.234. Variable `sex` is equal to 1 for females, thus females have a lower mean value for `SciAtt` by 0.234. Because our `SciAtt` measure is reversed, this means that females have a more positive attitude toward science. The effect is significant at better than the 1% level.
4. The difference between males and females in `SciAtt` is -0.234. Is that big or small, practically speaking?

   In the ordered probit specification, predicted agreement with questions is determined by an index into a $N(0,1)$ distribution. The value of the index is then compared with the cutpoints to determine the probability that the response is 1, 2, 3, 4, or 5.

   For statement 1, the loading on `SciAtt` is 1, and therefore the average difference in the probit index for males and females is $-0.234 \times 1 = -0.234$ standard deviation units. Females are 0.234 standard deviations to the left of males on average.

   For statement 2, the loading on `SciAtt` is 1.41, and therefore the average difference in the probit index is $-0.234 \times 1.41 = -0.33$ standard deviation units.

   You can work out the effect size for the other statements. We would say that the effect is medium sized.

Fitting the model with the Builder

Use the diagram in *Fitting the MIMIC model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_issp93
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.
4. Create the measurement component for `SciAtt`.

   Select the Add measurement component tool, , and then click in the diagram halfway down and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `SciAtt`;
   b. select `y1`, `y2`, `y3`, and `y4` by using the *Measurement variables* control;
   c. check *Make measurements generalized*;
   d. select `Ordinal, Probit` in the *Family/Link* control;
   e. select `Down` in the *Measurement direction* control;
   f. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Create the variable for the formative indicator of `SciAtt`.
   a. Select the Add observed variable tool, □, and then click in the diagram directly above the oval for `SciAtt` and about one-fourth of the way down from the top. After adding it, you can click inside the rectangle to move the variable if you wish.
   b. In the Contextual Toolbar, select `sex` with the *Variable* control.
6. Create the path from the formative indicator to `SciAtt`.
   a. Select the Add path tool, →.
   b. Click in the bottom of the `sex` rectangle (it will highlight when you hover over it), and drag a path to the top of the `SciAtt` oval (it will highlight when you can release to connect the path).
7. Clean up the direction of the error.

   The error on `SciAtt` is likely to have been created below the oval for `SciAtt`. Choose the Select tool, , and then click in the `SciAtt` oval. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is where you want it.
8. Clean up the location of the path.

   If you do not like where the path between `sex` and `SciAtt` has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle or oval and drag the endpoint.
9. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_mimic
```

Reference

Greenacre, M. J. 2006. From simple to multiple correspondence analysis. In *Multiple Correspondence Analysis and Related Methods*, ed. M. J. Greenacre and J. Blasius. Boca Raton, FL: Chapman & Hall.

Also see

[SEM] **Example 10** — MIMIC model

[SEM] **Example 35g** — Ordered probit and ordered logit

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 37g — Multinomial logistic regression

Description Remarks and examples Reference Also see

Description

With the data below, we demonstrate multinomial logistic regression, also known as multinomial logit, mlogit, and family multinomial, link logit:

```
. use https://www.stata-press.com/data/r16/gsem_sysdsn1
(Health insurance data)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_sysdsn1.dta
  obs:           644                          Health insurance data
 vars:            12                          28 Mar 2018 13:46
                                              (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
site            byte    %9.0g                 study site (1-3)
patid           float   %9.0g                 patient id
insure          byte    %9.0g      insure     insurance type
age             float   %10.0g                NEMC (ISCNRD-IBIRTHD)/365.25
male            byte    %8.0g                 NEMC PATIENT MALE
nonwhite        byte    %9.0g                 race
noinsur0        byte    %8.0g                 no insurance at baseline
noinsur1        byte    %8.0g                 no insurance at year 1
noinsur2        byte    %8.0g                 no insurance at year 2
ppd0            byte    %8.0g                 prepaid at baseline
ppd1            byte    %8.0g                 prepaid at year 1
ppd2            byte    %8.0g                 prepaid at year 2
-------------------------------------------------------------------------------
Sorted by: patid
. notes
_dta:
  1.  Data on health insurance available to 644 psychologically depressed
      subjects.
  2.  Data from Tarlov, A.R., et al., 1989, "The Medical Outcomes Study. An
      application of methods for monitoring the results of medical care." _J.
      of the American Medical Association_ 262, pp. 925-930.
  3.  insure: 1=indemnity, 2=prepaid, 3=uninsured.
```

See *Structural models 6: Multinomial logistic regression* in [SEM] **Intro 5** for background.

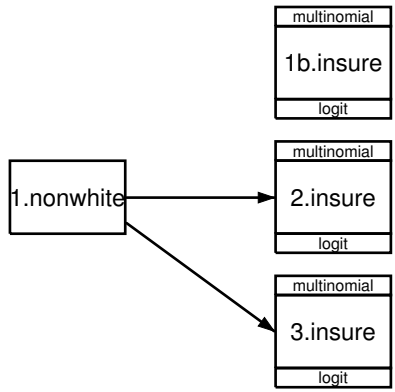
Remarks and examples

Remarks are presented under the following headings:

Simple multinomial logistic regression model
Multinomial logistic regression model with constraints
Fitting the simple multinomial logistic model with the Builder
Fitting the multinomial logistic model with constraints with the Builder

Simple multinomial logistic regression model

In a multinomial logistic regression model, there are multiple unordered outcomes. In our case, these outcomes are recorded in variable `insure`. This variable records three different outcomes—indemnity, prepaid, and uninsured—recorded as 1, 2, and 3. The model we wish to fit is



The response variables are `1.insure`, `2.insure`, and `3.insure`, meaning `insure` = 1 (code for indemnity), `insure` = 2 (code for prepaid), and `insure` = 3 (code for uninsured). We specified that `insure` = 1 be treated as the mlogit base category by placing a `b` on `1.insure` to produce `1b.insure` in the variable box.

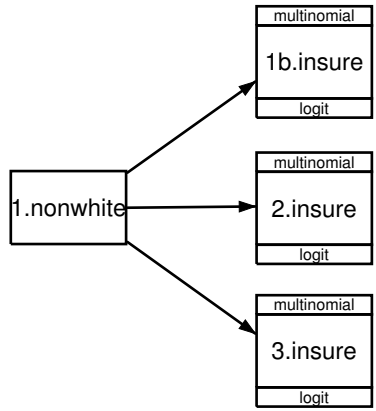
Notice that there are no paths into `1b.insure`. We could just as well have diagrammed the model with a path arrow from the explanatory variable into `1b.insure`. It would have made no difference.

In one sense, omitting the path is more mathematically appropriate, because multinomial logistic base levels are defined by having all coefficients constrained to be 0.

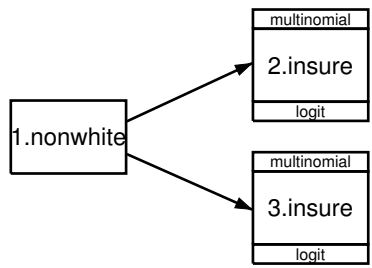
In another sense, drawing the path would be more appropriate because, even with `insure` = 1 as the base level, we are not assuming that outcome `insure` = 1 is unaffected by the explanatory variables. The probabilities of the three possible outcomes must sum to 1, and so any predictor that increases one probability of necessity causes the sum of the remaining probabilities to decrease. If a predictor x has positive effects (coefficients) for both `2.insure` and `3.insure`, then increases in x must cause the probability of `1.insure` to fall.

The choice of base outcome specifies that the coefficients associated with the other outcomes are to be measured relative to that base. In multinomial logistic regression, the coefficients are logs of the probability of the category divided by the probability of the base category, a mouthful also known as the log of the relative-risk ratio.

We drew the diagram one way, but we could just as well have drawn it like this:



In fact, we could just as well have chosen to indicate the base category by omitting it entirely from our diagram, like this:



Going along with that, we could type three different commands, each exactly corresponding to one of the three diagrams:

```
. gsem (1b.insure) (2.insure 3.insure <- i.nonwhite), mlogit
. gsem (1b.insure 2.insure 3.insure <- i.nonwhite), mlogit
. gsem (2.insure 3.insure <- i.nonwhite), mlogit
```

In the command language, however, we would probably just type

```
. gsem (i.insure <- i.nonwhite), mlogit
```

See [SEM] **Intro 3** for a complete description of factor-variable notation. It makes no difference which diagram we draw or which command we type.

This model can be fit using the command syntax by typing

```
. gsem (i.insure <- i.nonwhite), mlogit
Iteration 0:   log likelihood = -556.59502
Iteration 1:   log likelihood = -551.78935
Iteration 2:   log likelihood = -551.78348
Iteration 3:   log likelihood = -551.78348
Generalized structural equation model           Number of obs     =        616
Response       : insure
Base outcome   : 1
Family         : multinomial
Link           : logit
Log likelihood = -551.78348
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.insure	(base outcome)					
2.insure						
1.nonwhite	.6608212	.2157321	3.06	0.002	.2379942	1.083648
_cons	-.1879149	.0937644	-2.00	0.045	-.3716896	-.0041401
3.insure						
1.nonwhite	.3779586	.407589	0.93	0.354	-.4209011	1.176818
_cons	-1.941934	.1782185	-10.90	0.000	-2.291236	-1.592632

Notes:

1. The above results say that nonwhites are more likely to have `insure` $= 2$ relative to 1 than whites, and that nonwhites are more likely to have `insure` $= 3$ relative to 1 than whites, which obviously implies that whites are more likely to have `insure` $= 1$.

2. For a three-outcome multinomial logistic regression model with the first outcome set to be the base level, the probability of each outcome is

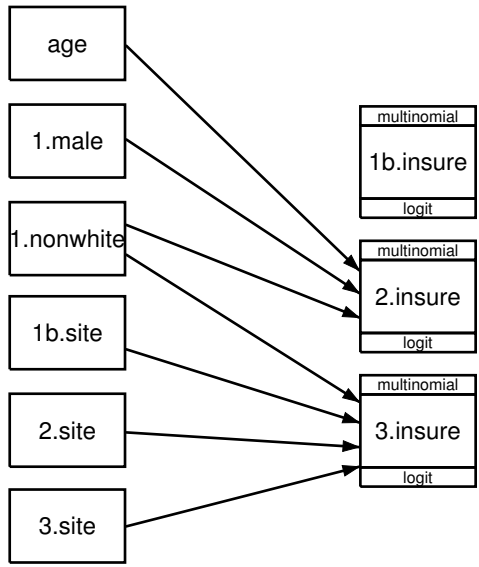
$$\Pr(y = 1) = 1/D$$
$$\Pr(y = 2) = \exp(\mathbf{X}_2\boldsymbol{\beta}_2)/D$$
$$\Pr(y = 2) = \exp(\mathbf{X}_3\boldsymbol{\beta}_3)/D$$

   where $D = 1 + \exp(\mathbf{X}_2\boldsymbol{\beta}_2) + \exp(\mathbf{X}_3\boldsymbol{\beta}_3)$.

3. For whites—that is, for `1.nonwhite` $= 0$—we have $\mathbf{X}_2\boldsymbol{\beta}_2 = -0.1879$ and $\mathbf{X}_3\boldsymbol{\beta}_3 = -1.9419$. Thus $D = 1.9721$, and the probabilities for each outcome are 0.5071, 0.4202, and 0.0727. Those probabilities sum to 1. You can make the similar calculations for nonwhites—that is, for `1.nonwhite` $= 1$—for yourself.

Multinomial logistic regression model with constraints

Using the same data, we wish to fit the following model:



In the above, `insure` = 2 and `insure` = 3 have paths pointing to them from different sets of predictors. They share predictor `1.nonwhite`, but `insure` = 2 also has paths from `age` and `1.male`, whereas `insure` = 3 also has paths from the `site` variables. When we fit this model, we will not obtain estimates of the coefficients on `age` and `1.male` in the equation for `insure` = 3. This is equivalent to constraining the coefficients for `age` and `1.male` to 0 in this equation. In other words, we are placing a constraint that the relative risk of choosing `insure` = 3 rather than `insure` = 1 is the same for males and females and is the same for all ages.

This model can be fit using command syntax by typing

```
. gsem (2.insure <- i.nonwhite age i.male)
>      (3.insure <- i.nonwhite i.site), mlogit
Iteration 0:   log likelihood = -555.85446
Iteration 1:   log likelihood = -541.20487
Iteration 2:   log likelihood = -540.85219
Iteration 3:   log likelihood = -540.85164
Iteration 4:   log likelihood = -540.85164
Generalized structural equation model           Number of obs     =        615
Response       : insure
Base outcome   : 1
Family         : multinomial
Link           : logit
Log likelihood = -540.85164
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.insure	(base outcome)					
2.insure						
1.nonwhite	.7219663	.2184994	3.30	0.001	.2937153	1.150217
age	-.0101291	.0058972	-1.72	0.086	-.0216874	.0014292
1.male	.5037961	.1912717	2.63	0.008	.1289104	.8786818
_cons	.1249932	.2743262	0.46	0.649	-.4126763	.6626627
3.insure						
1.nonwhite	.0569646	.4200407	0.14	0.892	-.7663001	.8802293
site						
2	-1.273576	.4562854	-2.79	0.005	-2.167879	-.3792728
3	.0434253	.3470773	0.13	0.900	-.6368337	.7236843
_cons	-1.558258	.2540157	-6.13	0.000	-2.056119	-1.060396

We could have gotten identical results from Stata's `mlogit` command for both this example and the previous one. To fit the first example, we would have typed

```
. mlogit insure i.nonwhite
```

To obtain the results for this second example, we would have been required to type a bit more:

```
. constraint 1 [Uninsure]age = 0
. constraint 2 [Uninsure]1.male = 0
. constraint 3 [Prepaid]2.site = 0
. constraint 4 [Prepaid]3.site = 0
. mlogit insure i.nonwhite age i.male i.site, constraints(1/4)
```

Having `mlogit` embedded in `gsem`, of course, also provides the advantage that we can combine the mlogit model with measurement models, multilevel models, and more. See [SEM] **Example 41g** for a two-level multinomial logistic regression with random effects.

Fitting the simple multinomial logistic model with the Builder

Use the first diagram in *Simple multinomial logistic regression model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_sysdsn1
   ```

2. Open a new builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the rectangles for each possible outcome of the multinomial endogenous variable.

   Select the Add observed variables set tool, , and then click in the diagram about one-third of the way in from the right and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. check *Make variables generalized responses*;

   c. select `Multinomial, Logit` in the *Family/Link* control;

   d. select `insure` in the *Variable* control;

   e. select `Vertical` in the *Orientation* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the independent variable.

   a. Select the Add observed variable tool, , and then click in the diagram to the left of `2.insure`.

   b. In the Contextual Toolbar, type `1.nonwhite` in the *Variable* control and press *Enter*.

6. Create the paths from the independent variable to the rectangles for outcomes `insure` = 2 and `insure` = 3.

   a. Select the Add path tool, .

   b. Click in the right side of the `1.nonwhite` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `2.insure` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the tool, click in the right side of the `1.nonwhite` rectangle and drag a path to the left side of the `3.insure` rectangle.

7. Clean up the location of the paths.

   If you do not like where the paths have been connected to the rectangles, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.

8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_mlogit1
```

Fitting the multinomial logistic model with constraints with the Builder

Use the diagram in *Multinomial logistic regression model with constraints* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_sysdsn1
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the rectangles for each possible outcome of the multinomial endogenous variable.

   Select the Add observed variables set tool, , and then click in the diagram about one-third of the way in from the right and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. check *Make variables generalized responses*;

   c. select `Multinomial, Logit` in the *Family/Link* control;

   d. select `insure` in the *Variable* control;

   e. select `Vertical` in the *Orientation* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the independent variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-third from the left and one-fourth from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. uncheck *Make variables generalized responses*;

   c. use the *Variables* control and select `age`;

   d. type `1.male 1.nonwhite` in the *Variables* control after `age` (typing `1.varname` rather than using the button to create them as `i.varname` factor variables prevents rectangles corresponding to the base categories for these binary variables from being created);

   e. include the levels of the factor variable `site` by clicking on the button next to the *Variables* control. In the resulting dialog box, select the *Factor variable* radio button, select `Main effect` in the *Specification* control, and select `site` in the *Variables* control for *Variable 1*. Click on **Add to varlist**, and then click on **OK**;

f. select `Vertical` in the *Orientation* control;

g. click on **OK**.

6. Create the paths from the independent variables to the rectangles for outcomes `insure` = 2 and `insure` = 3.

   a. Select the Add path tool, .

   b. Click in the right side of the `age` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `2.insure` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the given outcome of the dependent variable:

```
1.male -> 2.insure
1.nonwhite -> 2.insure
1.nonwhite -> 3.insure
1b.site -> 3.insure
2.site -> 3.insure
3.site -> 3.insure
```

7. Clean up the location of the paths.

   If you do not like where the paths have been connected to the rectangles, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.

8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_mlogit2
```

Reference

Tarlov, A. R., J. E. Ware, Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930.

Also see

[SEM] **Example 35g** — Ordered probit and ordered logit

[SEM] **Example 41g** — Two-level multinomial logistic regression (multilevel)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 38g — Random-intercept and random-slope models (multilevel)

Description Remarks and examples Reference Also see

Description

Below we discuss random-intercept and random-slope models in the context of multilevel models, and specifically, 2-level models, although we could just as well use higher-level models (see [SEM] **Example 39g**). Some people refer to these models as random-effects models and as mixed-effects models.

To demonstrate random-intercept and random-slope models, we will use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_nlsy
(NLSY 1968)

. describe

Contains data from https://www.stata-press.com/data/r16/gsem_nlsy.dta
  obs:         2,763                          NLSY 1968
 vars:            21                          29 Mar 2018 11:30
                                              (_dta has notes)
```

variable name	storage type	display format	value label	variable label
idcode	int	%8.0g		NLS ID
year	int	%8.0g		interview year
birth_yr	byte	%8.0g		birth year
age	byte	%8.0g		age in current year
race	byte	%8.0g	racelbl	race
msp	byte	%8.0g		1 if married, spouse present
nev_mar	byte	%8.0g		1 if never married
grade	byte	%8.0g		current grade completed
collgrad	byte	%8.0g		1 if college graduate
not_smsa	byte	%8.0g		1 if not SMSA
c_city	byte	%8.0g		1 if central city
south	byte	%8.0g		1 if south
ind_code	byte	%8.0g		industry of employment
occ_code	byte	%8.0g		occupation
union	byte	%8.0g		1 if union
wks_ue	byte	%8.0g		weeks unemployed last year
ttl_exp	float	%9.0g		total work experience
tenure	float	%9.0g		job tenure, in years
hours	int	%8.0g		usual hours worked
wks_work	int	%8.0g		weeks worked last year
ln_wage	float	%9.0g		ln(wage/GNP deflator)

```
Sorted by: idcode  year

. notes

_dta:
  1.  Data from National Longitudinal Survey of Young Women 14-27 years of age
      in 1968 (NLSY), Center for Human Resource Research, Ohio State
      University, first released in 1989.
  2.  This data was subsetted for purposes of demonstration.
```

These 2-level data are recorded in long form, that is, each observation corresponds to a year within a subject and the full set of data is spread across repeated observations.

```
. list id year ln_wage union grade in 1/20, sepby(idcode)
```

	idcode	year	ln_wage	union	grade
1.	1	1970	1.451214	.	12
2.	1	1971	1.02862	.	12
3.	1	1972	1.589977	1	12
4.	1	1973	1.780273	.	12
5.	1	1975	1.777012	.	12
6.	1	1977	1.778681	0	12
7.	1	1978	2.493976	.	12
8.	1	1980	2.551715	1	12
9.	1	1983	2.420261	1	12
10.	1	1985	2.614172	1	12
11.	1	1987	2.536374	1	12
12.	1	1988	2.462927	1	12
13.	2	1971	1.360348	0	12
14.	2	1972	1.206198	.	12
15.	2	1973	1.549883	.	12
16.	2	1975	1.832581	.	12
17.	2	1977	1.726721	1	12
18.	2	1978	1.68991	1	12
19.	2	1980	1.726964	1	12
20.	2	1982	1.808289	1	12

In the repeated observations for a subject, some variables vary (they are at the observation level, such as `ln_wage`) and other variables do not vary (they are at the subject level, such as `grade`).

When using `gsem`, multilevel data must be recorded in the long form except in one case. The exception is latent growth curve models, which can be fit in the long or wide form. In the wide form, there is one physical observation for each subject and multiple variables within subject, such as `ln_wage_1970`, `ln_wage_1971`, and so on. Researchers from a structural equation modeling background think about latent growth models in the wide form; see [SEM] **Example 18**.

In all other cases, if your data are in the wide form, use Stata's `reshape` command (see [D] **reshape**) to convert the data to long form.

See *Structural models 1: Linear regression* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

Random-intercept model, single-equation formulation
Random-intercept model, within-and-between formulation
Random-slope model, single-equation formulation
Random-slope model, within-and-between formulation
Fitting the random-intercept model with the Builder
Fitting the random-slope model with the Builder

Random-intercept model, single-equation formulation

There are two formulations of the random-intercept model, which we call the single-equation formulation and the within-and-between formulation. Results from both formulations are identical unless you have observations with missing values, in which case the within-and-between formulation will sometimes use more of the data.

We will show you both formulations, but the single-equation formulation makes a good starting point. The model we wish to fit is

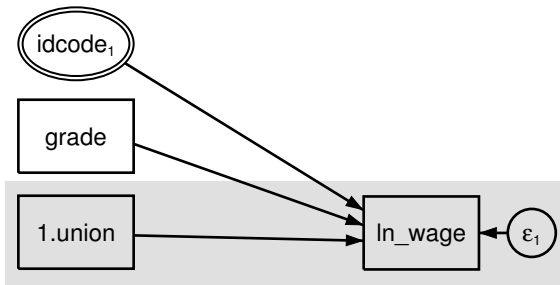


Figure 1

We use factor-variable notation in the diagram above; see [SEM] **Example 37g**.

We are using multilevel data. `ln_wage` and `union` (union membership) vary at the observation level, while `grade` (school completion) varies at the subject level. We have used shading to emphasize that.

In this model, we are including a random intercept (a random effect) at the subject level. Double-ringed `idcode` is saying, "I am a latent variable at the `idcode` level—meaning I am constant within identification codes and vary across identification codes—and I correspond to a latent variable named `M1`." The `M1` part of the statement came from the subscript 1; the `M` part is fixed.

Double-ringed `idcode` indicates a latent variable constant within `idcode`—a random effect. And the fact that the path from the latent variable is pointing to a box and not to another path means that the latent variable is used as a random intercept rather than a random slope. By the way, variable `idcode` in the data contains each subject's identification number.

Using command syntax, we can fit this model by typing

```
. gsem (ln_wage <- i.union grade M1[idcode])
Fitting fixed-effects model:
Iteration 0:   log likelihood = -925.06629
Iteration 1:   log likelihood = -925.06629
Refining starting values:
Grid node 0:   log likelihood =  -763.3769
Fitting full model:
Iteration 0:   log likelihood =  -763.3769
Iteration 1:   log likelihood = -622.04625  (backed up)
Iteration 2:   log likelihood = -613.54948
Iteration 3:   log likelihood = -607.56242
Iteration 4:   log likelihood = -607.49246
Iteration 5:   log likelihood = -607.49233
Iteration 6:   log likelihood = -607.49233
```

```
Generalized structural equation model           Number of obs     =      1,904
Response       : ln_wage
Family         : Gaussian
Link           : identity
Log likelihood = -607.49233
 ( 1)  [ln_wage]M1[idcode] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ln_wage						
1.union	.1637408	.0227254	7.21	0.000	.1191998	.2082818
grade	.0767919	.0067923	11.31	0.000	.0634791	.0901046
M1[idcode]	1	(constrained)				
_cons	.7774129	.0906282	8.58	0.000	.5997848	.955041
var(M1[idcode])	.080247	.0073188			.0671113	.0959537
var(e.ln_w~e)	.078449	.0028627			.0730342	.0842653

Notes:

1. The `ln_wage <- M1[idcode]` coefficient is constrained to be 1. Such constraints are automatically supplied by `gsem` to identify the latent variable. Our model is

$$\begin{aligned}\texttt{ln_wage} &= \cdots + \beta_3 \times \texttt{M1[idcode]} + \texttt{_cons}\\ &= \cdots + 1 \times \texttt{M1[idcode]} + 0.7774\end{aligned}$$

   Thus `M1[idcode]` is being used as a random intercept.

   Remember that the square bracketed `[idcode]` means that `M1` is constant within `idcode` and varies only across `idcode`.

2. The variance of our random intercept is estimated to be 0.0802, which is greater than the estimated error variance of 0.0784.

3. Although it is obvious in this case that the latent variable (random intercept) has sufficient variance that it cannot be ignored, we can test whether the variance is large enough that we could not ignore it. The test will be up against a boundary (variances cannot be less than 0), and so the test will be conservative. To perform the test, we would type

```
. gsem (ln_wage <- i.union grade)              // fit model w/o random intercept
. estimates store without
. gsem (ln_wage <- i.union grade M1[idcode])   // fit model w/ random intercept
. estimates store with
. lrtest without with                          // order of arguments unimportant
```

Random-intercept model, within-and-between formulation

The other way to write the random-intercept model is

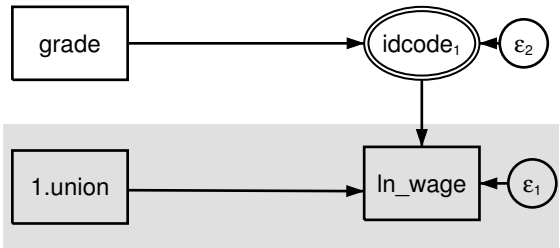


Figure 2

Do not read `grade` pointing to double-ringed `idcode` as `grade` being a predictor of `idcode`. That would make no sense. Double rings indicate a latent variable, and `grade` is a predictor of a latent variable. In particular, the subscript 1 on `idcode` indicates that the latent variable is named `M1`. Thus `grade` is a predictor of `M1`. The `idcode` inside the double rings says that `M1` is constant within `idcode`. Thus `grade`, which itself does not vary within `idcode`, is a predictor of `M1`, which does not vary within `idcode`; said more elegantly, `grade` is a predictor of `M1` at the subject level.

It is logically required that `grade` vary at the same or higher level as `M1[idcode]`, and `gsem` will check that requirement for you.

In this model, `M1[idcode]` contains both the random intercept and the grade effect. There is now an equation for `M1[idcode]`, with an error associated with it, and it will be the variance of the error term that will reflect the variance of the random intercept.

To fit this within-and-between formulation of our model, we type

```
. gsem (ln_wage <- i.union M1[idcode]) (M1[idcode] <- grade)
Fitting fixed-effects model:
Iteration 0:   log likelihood =  -1091.655
Iteration 1:   log likelihood =  -1091.655
Refining starting values:
Grid node 0:   log likelihood = -886.50236
Fitting full model:
Iteration 0:   log likelihood = -886.50236  (not concave)
Iteration 1:   log likelihood =  -683.7337
Iteration 2:   log likelihood = -630.91327
Iteration 3:   log likelihood = -608.14536
Iteration 4:   log likelihood = -607.49376
Iteration 5:   log likelihood = -607.49233
Iteration 6:   log likelihood = -607.49233
Generalized structural equation model           Number of obs     =      1,904
Response       : ln_wage
Family         : Gaussian
Link           : identity
Log likelihood = -607.49233
 ( 1)  [ln_wage]M1[idcode] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ln_wage						
1.union	.1637408	.0227254	7.21	0.000	.1191998	.2082818
M1[idcode]	1	(constrained)				
_cons	.7774129	.0906282	8.58	0.000	.5997848	.955041
M1[idcode]						
grade	.0767919	.0067923	11.31	0.000	.0634791	.0901046
var(e.M1[idcode])	.080247	.0073188			.0671113	.0959537
var(e.ln_w~e)	.078449	.0028627			.0730342	.0842653

Notes:

1. Results are identical to what we previously obtained.
2. The within-and-between formulation is equivalent to the single-equation formulation if there are no missing values in the data.
3. In this simple model, the two formulations are also equivalent even in the presence of missing values.
4. If `M1[idcode]` were also being used to predict another endogenous variable, then missing values in `grade` would only cause the equation for the other endogenous variable to have to omit those observations in the within-and-between formulation.

Random-slope model, single-equation formulation

Let us now turn to random slopes. Because it is generally—not always—a good idea to include random intercepts with random slopes, we are going to include both. In addition, we are going to complicate our model by adding an interaction term between union membership and grade so we can make another point that we will not explain until the next section.

The model we wish to fit is

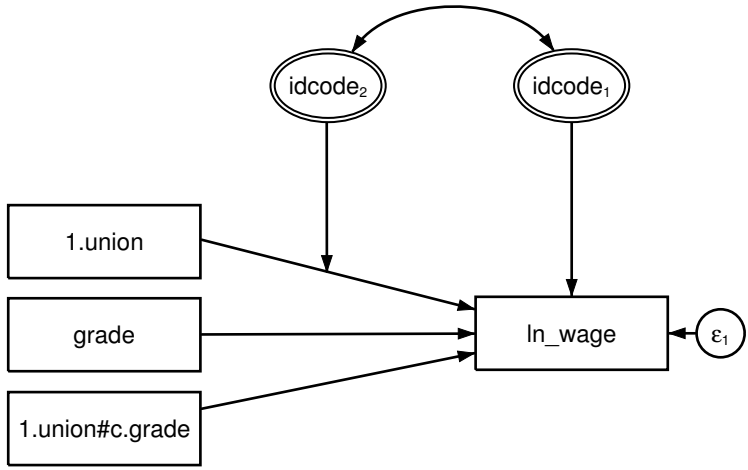


Figure 3. Demonstrated formulation 1

Ignore the interaction between union membership and grade. We could have omitted it from this example, but we want to show later the strikingly different way that interaction can be written in the within-and-between formulation.

In this model, we have double-ringed `idcode`, just as we did in a previous section introducing a random intercept into the model. This time, however, we also have a second double-ringed `idcode` adding a random slope. When a path from a latent variable points to another path, it is specifying a random slope. When specifying a random slope for a variable, you still include the variable in the usual, fixed-slope way, and then you add the extra component.

The result of the extra component will be to add an interaction into the model. `1.union` will affect `ln_wage`, just as we have drawn, and the extra component will add `1.union` × `M2[idcode]`. That is, the effect of `1.union` in our model will be

$$\begin{aligned}\texttt{ln_wage} &= \cdots + \beta_3 \times \texttt{1.union} + \beta_4 \times \texttt{1.union} \times \texttt{M2[idcode]} + \cdots \\ &= \cdots + \beta_3 \times \texttt{1.union} + \beta_4 \times \texttt{M2[idcode]} \times \texttt{1.union} + \cdots \\ &= \cdots + (\beta_3 + \beta_4 \times \texttt{M2[idcode]}) \times \texttt{1.union} + \cdots\end{aligned}$$

The fixed-plus-random, total slope is $\beta_3 + \beta_4 \times$ `M2[idcode]`. Latent variable `M2` has an arbitrary scale, and therefore coefficient β_4 cannot be identified. So constraining $\beta_4 = 1$ results in the total slope being $\beta_3 +$ `M2[idcode]`, which is exactly how you would expect a fixed-plus-random slope to look. In other words, drawing the path from $\texttt{idcode}_2$ to the path with coefficient β_3 indicates that `M2[idcode]` is added to β_3.

To fit this model in the command language, we type

```
. gsem (ln_wage <- i.union grade i.union#c.grade M1[idcode] 1.union#M2[idcode])
Fitting fixed-effects model:
Iteration 0:   log likelihood = -925.06629
Iteration 1:   log likelihood = -925.06629
Refining starting values:
Grid node 0:   log likelihood = -869.92256
Fitting full model:
Iteration 0:   log likelihood = -869.92256  (not concave)
Iteration 1:   log likelihood = -727.15806  (not concave)
Iteration 2:   log likelihood = -711.74718  (not concave)
Iteration 3:   log likelihood = -684.33867  (not concave)
Iteration 4:   log likelihood = -665.94123  (not concave)
Iteration 5:   log likelihood = -610.14526
Iteration 6:   log likelihood = -589.89989
Iteration 7:   log likelihood = -582.24119
Iteration 8:   log likelihood =   -581.298
Iteration 9:   log likelihood = -581.29004
Iteration 10:  log likelihood = -581.29003
Generalized structural equation model           Number of obs     =      1,904
Response       : ln_wage
Family         : Gaussian
Link           : identity
Log likelihood = -581.29003
 ( 1)  [ln_wage]M1[idcode] = 1
 ( 2)  [ln_wage]1.union#M2[idcode] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ln_wage						
1.union	.1199049	.1508189	0.80	0.427	-.1756946	.4155045
grade	.0757883	.0081803	9.26	0.000	.0597552	.0918215
union# c.grade						
1	.0019983	.0113534	0.18	0.860	-.020254	.0242506
M1[idcode]	1	(constrained)				
union# M2[idcode]						
1	1	(constrained)				
_cons	.7873884	.1086476	7.25	0.000	.574443	1.000334
var(M1[idcode])	.0927931	.0088245			.0770136	.1118056
var(M2[idcode])	.0823065	.018622			.052826	.1282392
cov(M1[idcode], M2[idcode])	-.0549821	.0116103	-4.74	0.000	-.077738	-.0322263
var(e.ln_w~e)	.0720873	.0027135			.0669603	.0776068

Notes:

1. `M1[idcode]` is the random intercept. The coefficient on it is constrained to be 1, just as previously and just as we would expect.
2. The coefficient on `1.union` is the fixed part of the slope of `union`.
3. `M2[idcode]` is the random part of the slope of `union`.

   The coefficient on `1.union#M2[idcode]` is constrained to be 1, just as we would expect. `1.union#M2[idcode]` is the way Stata writes `1.union` × `M2[idcode]`.
4. There is an unexpected term in the output, `0.union#M2[idcode]`, shown with coefficient 0. The first thing to remember about unexpected terms is that they are irrelevant if their coefficients are 0. `gsem` reports the coefficient as being `0 (omitted)`, which is `gsem`'s way of saying, "Here is a line that I did not even include in the model." There are a lot of terms `gsem` could tell us about that were not included in the model, so why did `gsem` feel obligated to tell us about this term? The term has to do with how Stata tracks base levels of factor variables.

   There is a setting—`set showbaselevels off`—that will prevent lines like that from being displayed. There is also a setting—`set showbaselevels all`—that will show even more of them! The default is `set showbaselevels on`.
5. We specified the interaction as `1.union#M2[idcode]` rather than `i.union#M2[idcode]`. Even so, using `#.union#M2[idcode]` or `i.union#M2[idcode]` makes no difference because `union` takes on two values. If `union` took on three values, however, think about how we would diagram the model. We would have two latent variables, and we would want `1.union#M2[idcode]` and `2.union#M3[idcode]`. If `union` took on three or more values, typing `i.union#M2[idcode]` simply would not produce the desired result.

Random-slope model, within-and-between formulation

To properly demonstrate the within-and-between formulation, we need a more complicated model; we made one in the previous section when we included `1.union#c.grade` and then told you to ignore it.

Pretend that we had omitted `1.union#c.grade` from the random-slope, single-equation formulation model. It would have looked like this:

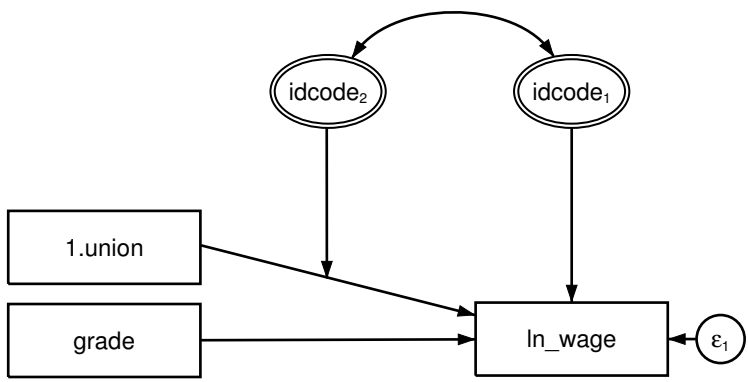


Figure 4. Simple formulation 1

Let's call this model Simple formulation 1 as compared with the model that we fit in the previous section, which we will call Demonstrated formulation 1. We could have fit Simple formulation 1 by typing

```
. gsem (ln_wage <- i.union grade M1[idcode] 1.union#M2[idcode])
```

The corresponding within-and-between model, which we will call Simple formulation 2, would look something like this:

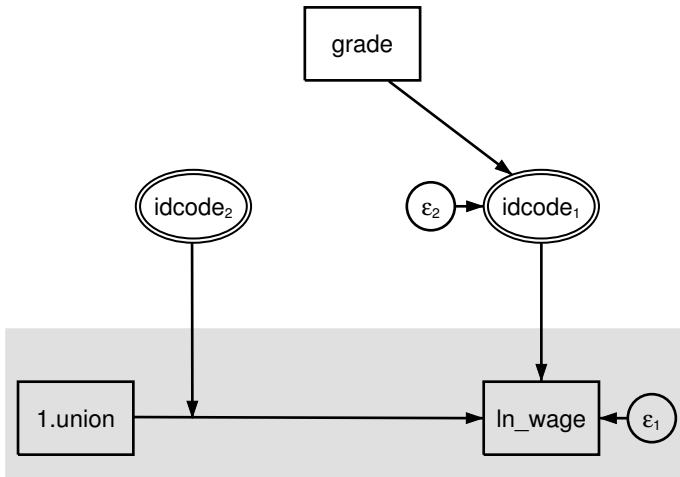


Figure 5. Simple formulation 2

We say that it would look something like the above because Simple formulations 1 and 2 are not identical models. In particular, there is nothing in Simple formulation 2 that corresponds to the correlation between `M1[idcode]` and `M2[idcode]` in Simple formulation 1. Other than that, the models are identical. In Simple formulation 1, `grade` directly affects `ln_wage`. In Simple formulation 2, `grade` affects `M1[idcode]` and `M1[idcode]` affects `ln_wage`. The results are the same either way. Many researchers prefer Simple formulation 2 because it visually separates the within (subject) and between (subject) levels.

But as we said, the two formulations are not identical. They would be identical if we added a curved covariance path between ϵ_2 and `idcode`$_2$, but Stata will not allow us to do that.

On the other hand, if you wanted to fit the simple model without the correlation, you could do it in either formulation. For Simple formulation 1, you would type

```
. gsem (ln_wage <- i.union grade M1[idcode] 1.union#M2[idcode]), ///
                                                  cov(M1[idcode]*M2[idcode]@0)
```

For Simple formulation 2, you would type

```
. gsem (ln_wage <- i.union M1[idcode] 1.union#M2[idcode]) (M1[idcode] <- grade)
```

Either way, results would be identical.

In more complicated models, you can correlate the random intercepts and slopes even in a second formulation. Consider Demonstrated formulation 2:

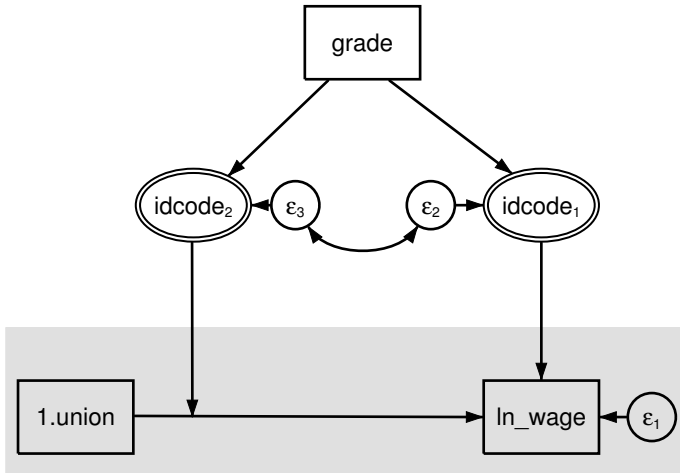


Figure 6. Demonstrated formulation 2

In this model, we allow correlation between ϵ_3 and ϵ_2—which Stata does allow—and that is the same as allowing correlation between `M1[idcode]` and `M2[idcode]` in the formulation-1 model. The general rule is that formulation 2 can be used to correlate the random intercepts and random slopes when both `M1` and `M2` are (latent) exogenous or (latent) endogenous.

Demonstrated formulation 2 corresponds to Demonstrated formulation 1. In Demonstrated formulation 1, we included `ln_wage <- i.union#c.grade`. Do you see how that term is implied by the path diagram above? Variable `grade` affects `M2[idcode]`, which in turn affects the slope of `1.union`. Affecting the slope is the same as an interaction.

And now you know why we complicated Demonstrated formulation 1. We did that so that we can show you Demonstrated formulation 2:

```
. gsem (ln_wage  <- i.union M1[idcode] 1.union#M2[idcode])
>      (M1[idcode] M2[idcode] <- grade), cov(e.M1[idcode]*e.M2[idcode])
Fitting fixed-effects model:
Iteration 0:   log likelihood =  -1091.655
Iteration 1:   log likelihood =  -1091.655
Refining starting values:
Grid node 0:   log likelihood = -982.18261
Fitting full model:
Iteration 0:   log likelihood = -982.18261  (not concave)
Iteration 1:   log likelihood = -690.60714  (not concave)
Iteration 2:   log likelihood = -677.13819  (not concave)
Iteration 3:   log likelihood = -661.26658
Iteration 4:   log likelihood = -615.66049
Iteration 5:   log likelihood = -585.69402
Iteration 6:   log likelihood = -581.34158
Iteration 7:   log likelihood = -581.29005
Iteration 8:   log likelihood = -581.29003
```

```
Generalized structural equation model           Number of obs     =      1,904
Response       : ln_wage
Family         : Gaussian
Link           : identity
Log likelihood = -581.29003
 ( 1)  [ln_wage]M1[idcode] = 1
 ( 2)  [ln_wage]1.union#M2[idcode] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ln_wage						
1.union	.119905	.1508188	0.80	0.427	-.1756945	.4155045
M1[idcode]	1	(constrained)				
union# M2[idcode]						
1	1	(constrained)				
_cons	.7873884	.1086476	7.25	0.000	.5744431	1.000334
M1[idcode]						
grade	.0757883	.0081803	9.26	0.000	.0597552	.0918215
M2[idcode]						
grade	.0019983	.0113534	0.18	0.860	-.020254	.0242506
var(e.M1[idcode])	.092793	.0088245			.0770136	.1118055
var(e.M2[idcode])	.0823065	.018622			.052826	.1282392
cov(e.M1[idcode], e.M2[idcode])	-.0549822	.0116103	-4.74	0.000	-.077738	-.0322263
var(e.ln_w~e)	.0720873	.0027135			.0669603	.0776068

Note:

1. Results of the above are identical to the results of Demonstrated formulation 1.

Fitting the random-intercept model with the Builder

Use the diagram in *Random-intercept model, single-equation formulation* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_nlsy
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the endogenous variable.

   a. Select the Add observed variable tool, , and then click in the diagram about one-third of the way in from the right and one-third of the way up from the bottom. After adding it, you can click inside the rectangle and move the variable if you wish.

   b. In the Contextual Toolbar, select `ln_wage` with the *Variable* control.

5. Create the observed exogenous variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-third of the way in from the left and one-third of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control and select `grade`;

   c. type `1.union` in the *Variables* control after `grade` (typing `1.union` rather than using the button to create `i.union` prevents the rectangle corresponding to the base category for this binary variable from being created);

   d. select `Vertical` in the *Orientation* control;

   e. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

6. Create the multilevel latent variable corresponding to the random intercept.

   a. Select the Add multilevel latent variable tool, , and click above the rectangle for `grade`.

   b. In the Contextual toolbar, click on the button.

   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `idcode` > *Observations* in the next control.

   d. Specify `M1` as the *Base name*.

   e. Click on **OK**.

7. Create the paths from the exogenous variables to `ln_wage`.

   a. Select the Add path tool, .

   b. Click in the right side of the `1.union` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `ln_wage` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the tool, draw paths from the right side of the `grade` rectangle to the left side of the `ln_wage` rectangle and from the right side of the $\texttt{idcode}_1$ double oval to the left side of the `ln_wage` rectangle.

8. Clean up the location of the paths.

   If you do not like where the paths have been connected to the rectangles or oval, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle or oval and drag the endpoint.

9. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_rint
```

Fitting the random-slope model with the Builder

Use the diagram in *Random-slope model, single-equation formulation* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_nlsy
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Increase the width of the observed variable rectangles to accommodate the length of the name of the interaction term.

   From the SEM Builder menu, select **Settings > Variables > All observed...**.

   In the resulting dialog box, change the first size to `1` and click on **OK**.

5. Create the endogenous variable.

   a. Select the Add observed variable tool, , and then click in the diagram about one-third of the way in from the right and one-third of the way up from the bottom. After adding it, you can click inside the rectangle and move the variable if you wish.

   b. In the Contextual Toolbar, select `ln_wage` with the *Variable* control.

6. Create the observed exogenous variables.

   Select the Add observed variables Set tool, , and then click in the diagram about one-third of the way in from the left and one-third of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. type `1.union` in the *Variables* control (typing `1.union` rather than using the button to create `i.union` prevents the rectangle corresponding to the base category for this binary variable from being created);

   c. use the *Variables* control and select `grade`;

   d. type `1.union#c.grade` in the *Variables* control after `grade`;

   e. select `Vertical` in the *Orientation* control;

   f. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

6. Create the multilevel latent variable corresponding to the random intercept.
   a. Select the Add multilevel latent variable tool, , and click above the rectangle for `ln_wage`.
   b. In the Contextual Toolbar, click on the "" button.
   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `idcode` > *Observations* in the next control.
   d. Specify `M1` as the *Base name*.
   e. Click on **OK**.
7. Create the paths from the exogenous variables to `ln_wage`.
   a. Select the Add path tool, .
   b. Click in the right side of the `1.union` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `ln_wage` rectangle (it will highlight when you can release to connect the path).
   c. Continuing with the tool, draw paths from the right sides of the `grade` and `1.union#c.grade` rectangles to the left side of the `ln_wage` rectangle and from the bottom of the $\texttt{idcode}_1$ double oval to the top of the `ln_wage` rectangle.
8. Create the random slope.
   a. Select the Add multilevel latent variable tool, , and click above the path from `1.union` to `ln_wage`.
   b. In the Contextual Toolbar, click on the "" button.
   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `idcode` > *Observations* in the next control.
   d. Specify `M2` as the *Base name*.
   e. Click on **OK**.
   f. Select the Add path tool, .
   g. Click in the bottom of the $\texttt{idcode}_2$ double oval, and drag a path to the path between `1.union` and `ln_wage`.
9. Create the covariance between the random slope and random intercept.
   a. Select the Add covariance tool, .
   b. Click in the top-right quadrant of the $\texttt{idcode}_2$ double oval, and drag a covariance to the top left of the $\texttt{idcode}_1$ double oval.
10. Clean up paths and covariance.

    If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint. Similarly, you can change where the covariance connects to the latent variables by clicking on the covariance and dragging the endpoint. You can also change the bow of the covariance by clicking on the covariance and dragging the control point that extends from one end of the selected covariance.

11. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_rslope
```

Reference

Center for Human Resource Research. 1989. *National Longitudinal Survey of Labor Market Experience, Young Women 14–24 years of age in 1968*. Columbus, OH: Ohio State University Press.

Also see

[SEM] **Example 39g** — Three-level model (multilevel, generalized response)

[SEM] **Example 42g** — One- and two-level mediation models (multilevel)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 39g — Three-level model (multilevel, generalized response)

Description Remarks and examples References Also see

Description

To demonstrate three-level models, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_melanoma
(Skin cancer (melanoma) data)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_melanoma.dta
  obs:           354                          Skin cancer (melanoma) data
 vars:             6                          25 Mar 2018 15:28
                                               (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
nation          byte    %12.0g     nation     Nation ID
region          byte    %9.0g                 Region ID: EEC level-I areas
county          int     %9.0g                 County ID: EEC level-II/level-III
                                                areas
deaths          int     %9.0g                 No. deaths during 1971-1980
expected        float   %9.0g                 No. expected deaths
uv              float   %9.0g                 UV dose, mean-centered
-------------------------------------------------------------------------------
Sorted by: nation  region  county
. notes
_dta:
  1.  Smans, M., C. S. Muir, and P. Boyle. 1992. _Atlas of Cancer Mortality in
      the European Economic Community_.  Lyon, France: IARC Scientific
      Publications
  2.  Data on 7 nations, 3-95 regions w/i nation, 1-13 counties w/i region.
  3.  Variable deaths is # of deaths among males due to malignant melanoma,
      1971-1980.
  4.  Variable expected contains # of expected male deaths based on crude rates
      for the combined counties.
```

Rabe-Hesketh and Skrondal (2012, exercise 13.7) describe data from the *Atlas of Cancer Mortality in the European Economic Community* (EEC) (Smans, Mair, and Boyle 1993). The data were analyzed in Langford, Bentham, and McDonald (1998) and record the number of deaths among males due to malignant melanoma during 1971–1980.

Data are stored in the long form. Observations are counties within regions within nation. These data and some of the models fit below are also demonstrated in [ME] **menbreg**.

See *Structural models 4: Count models* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

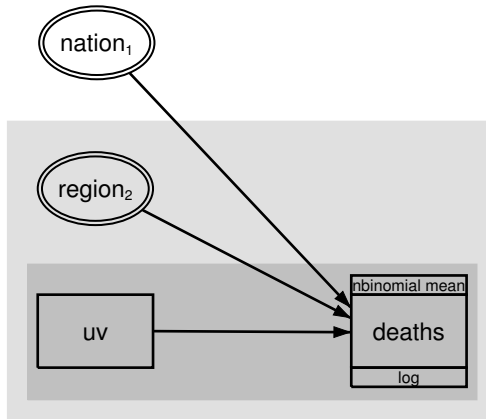
Remarks and examples

Remarks are presented under the following headings:

Three-level negative binomial model
Three-level Poisson model
Testing for overdispersion
Fitting the models with the Builder

Three-level negative binomial model

The model we wish to fit is



Deaths due to malignant melanoma at the county level are modeled as being affected by ultraviolet exposure with random region and nation effects.

To fit this model, we type

```
. gsem (deaths <- uv M1[nation] M2[nation>region]), nbreg exposure(expected)
Fitting fixed-effects model:
Iteration 0:   log likelihood =  -1361.855
Iteration 1:   log likelihood = -1230.0211
Iteration 2:   log likelihood =  -1211.049
Iteration 3:   log likelihood = -1202.5641
Iteration 4:   log likelihood = -1202.5329
Iteration 5:   log likelihood = -1202.5329
Refining starting values:
Grid node 0:   log likelihood = -1209.6951
Fitting full model:
Iteration 0:   log likelihood = -1209.6951  (not concave)
Iteration 1:   log likelihood = -1195.0761  (not concave)
Iteration 2:   log likelihood = -1189.7235  (not concave)
Iteration 3:   log likelihood =   -1167.58  (not concave)
Iteration 4:   log likelihood = -1145.4325  (not concave)
Iteration 5:   log likelihood = -1138.4471
Iteration 6:   log likelihood = -1088.3882
Iteration 7:   log likelihood = -1086.7992
Iteration 8:   log likelihood = -1086.4085
Iteration 9:   log likelihood = -1086.3903
Iteration 10:  log likelihood = -1086.3902
Iteration 11:  log likelihood = -1086.3902
Generalized structural equation model           Number of obs     =        354
Response       : deaths
Family         : nbinomial
Dispersion     : mean
Link           : log
Log likelihood = -1086.3902
 ( 1)  [deaths]M1[nation] = 1
 ( 2)  [deaths]M2[nation>region] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
deaths						
uv	-.0335933	.0113725	-2.95	0.003	-.055883	-.0113035
M1[nation]	1	(constrained)				
M2[nation>region]	1	(constrained)				
_cons	-.0790606	.1295931	-0.61	0.542	-.3330583	.1749372
ln(expected)	1	(exposure)				
/deaths						
lnalpha	-4.182603	.3415036			-4.851937	-3.513268
var(M1[nation])	.1283614	.0678971			.0455187	.3619758
var(M2[nation>region])	.0401818	.0104855			.0240938	.067012

Notes:

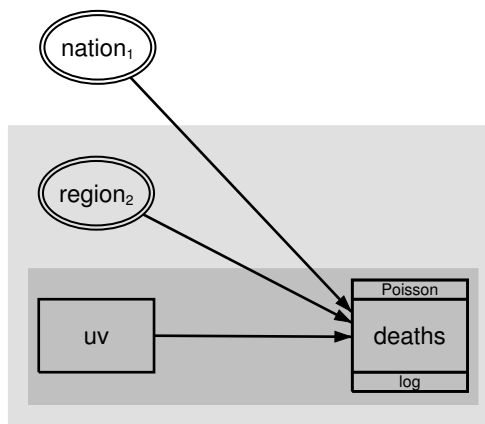
1. This is a three-level model of counties nested within region nested within nation, so we specified the latent variables as `M1[nation] M2[nation>region]`. Actually, we did the same thing in the diagram when we used the SEM Builder to define the latent variables, but the nesting information does not show in the double rings.
2. We fit this model by using negative binomial regression, also known as a mean-dispersion model. In the command, we typed `nbreg`, which is shorthand for `family(nbinomial mean) link(log)`.
3. A negative binomial distribution can be regarded as a gamma mixture of Poisson random variables, where said gamma distribution has mean 1 and variance α. The estimated $\ln(\alpha)$ is -4.183, which is small; α is estimated as 0.0153.
4. Zero does not mean lack of overdispersion, because we are including random effects that also allow for extra dispersion. For a discussion on these issues, see [ME] **menbreg**.
5. Notice that we specified `exposure(expected)`, where variable `expected` contains the expected number of deaths based on crude rates.

   The `exposure()` option is allowed with Poisson and negative binomial models. If we specify `exposure(`*varname*`)`, we are usually saying that each observation's time at risk is recorded in variable *varname*. When we omit the option, we are saying that each observation has the same time at risk. Obviously, if one observation had twice the time at risk of another observation, but was otherwise identical, we would expect twice the number of events in the first observation.

   In this case, however, we are using `exposure()` differently. We have a variable called `expected` containing the expected number of deaths from crude rates, and we are claiming `exposure(expected)`. What this is doing is saying that in two otherwise identical observations, if the number of expected deaths differed, we would expect the number of deaths due to melanoma to differ, too, and by the same proportion. See [SEM] **gsem family-and-link options**.

Three-level Poisson model

The same model, fit with Poisson, is



To fit the model in the command language, we type

```
. gsem (deaths <- uv  M1[nation] M2[nation>region]), poisson exposure(expected)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2136.5847
Iteration 1:   log likelihood = -1723.8955
Iteration 2:   log likelihood = -1723.7727
Iteration 3:   log likelihood = -1723.7727
Refining starting values:
Grid node 0:   log likelihood = -1166.6536
Refining starting values (unscaled likelihoods):
Grid node 0:   log likelihood = -1166.6536
Fitting full model:
Iteration 0:   log likelihood = -1166.6536  (not concave)
Iteration 1:   log likelihood = -1152.2741  (not concave)
Iteration 2:   log likelihood = -1146.3094  (not concave)
Iteration 3:   log likelihood = -1119.8479  (not concave)
Iteration 4:   log likelihood = -1108.0129  (not concave)
Iteration 5:   log likelihood = -1098.8067
Iteration 6:   log likelihood = -1095.7563
Iteration 7:   log likelihood = -1095.3164
Iteration 8:   log likelihood =   -1095.31
Iteration 9:   log likelihood =   -1095.31
Generalized structural equation model           Number of obs     =        354
Response       : deaths
Family         : Poisson
Link           : log
Log likelihood =   -1095.31
 ( 1)  [deaths]M1[nation] = 1
 ( 2)  [deaths]M2[nation>region] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
deaths						
uv	-.0282041	.0113998	-2.47	0.013	-.0505473	-.0058608
M1[nation]	1	(constrained)				
M2[nation>region]	1	(constrained)				
_cons	-.0639672	.1335515	-0.48	0.632	-.3257234	.197789
ln(expected)	1	(exposure)				
var(M1[nation])	.1371732	.0723303			.048802	.3855676
var(M2[nation>region])	.0483483	.0109079			.0310699	.0752353

Testing for overdispersion

The negative binomial model allows for overdispersion, or in a multilevel framework, allows for conditional overdispersion. The Poisson model has no overdispersion, or in a multilevel model, no overdispersion beyond that predicted by the latent variables. We can test whether there is dispersion beyond what Poisson would predict:

```
. gsem (deaths <- uv M1[nation] M2[nation>region]), nbreg exposure(expected)
  (output omitted)
. estimates store nbreg
. gsem (deaths <- uv M1[nation] M2[nation>region]), poisson exposure(expected)
  (output omitted)
. estimates store poisson
. lrtest nbreg poisson
Likelihood-ratio test                                 LR chi2(1)  =     17.84
(Assumption: poisson nested in nbreg)                 Prob > chi2 =    0.0000
```

We can reject at any reasonable level that the Poisson model adequately accounts for the dispersion in these data. Be aware that this test is conservative, because we are testing whether a variance goes to 0. `lrtest` usually issues a warning in such cases, but `lrtest` does not know that the relationship between negative binomial regression and Poisson regression involves a variance going to 0.

Fitting the models with the Builder

Use the diagram in *Three-level negative binomial model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_melanoma
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the generalized response variable.

   a. Select the Add generalized response variable tool, .

   b. Click in the diagram about one-third of the way in from the right and one-fourth of the way up from the bottom.

   c. In the Contextual Toolbar, select `Nbinomial mean, Log` in the *Family/Link* control.

   d. In the Contextual Toolbar, select `deaths` in the *Variable* control.

5. Create the observed exogenous variable.

   a. Select the Add observed variable tool, , and then click in the diagram about one-third of the way in from the right and one-fourth of the way up from the bottom.

   b. In the Contextual Toolbar, select `uv` with the *Variable* control.

6. Create the level-three latent variable.

   a. Select the Add multilevel latent variable tool, , and click above the rectangle for `uv` about one-fourth of the way down from the top.

   b. In the Contextual Toolbar, click on the "" button.

   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `nation` > *Observations* in the next line.

d. Specify `M1` as the *Base name*.

e. Click on **OK**.

7. Create the level-two latent variable.

a. Select the Add multilevel latent variable tool, , and click between the rectangle for `uv` and the double oval for $\texttt{nation}_1$.

b. In the Contextual Toolbar, click on the "" button.

c. Select the nesting level and nesting variable by selecting `3` from the *Nesting depth* control and selecting `nation` > `region` > *Observations* in the next control.

d. Specify `M2` as the *Base name*.

e. Click on **OK**.

8. Create the paths from the exogenous variables to `deaths`.

a. Select the Add path tool, .

b. Click in the right side of the `uv` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `deaths` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the tool, draw paths from the right side of the double ovals for $\texttt{nation}_1$ and $\texttt{region}_2$ to the left side of the `deaths` rectangle.

9. Specify the level of exposure.

Use the Select tool, , and double-click in the `deaths` rectangle. In the resulting dialog box, select `expected` in the *Exposure* control, and click on **OK**.

10. Clean up the location of the paths.

If you do not like where the paths have been connected to the rectangles or oval, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle or oval and drag the endpoint.

11. Estimate.

Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_3lev
```

References

Langford, I. H., G. Bentham, and A. McDonald. 1998. Multi-level modelling of geographically aggregated health data: A case study on malignant melanoma mortality and UV exposure in the European community. *Statistics in Medicine* 17: 41–57.

Rabe-Hesketh, S., and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*. 3rd ed. College Station, TX: Stata Press.

Smans, M., C. S. Mair, and P. Boyle. 1993. *Atlas of Cancer Mortality in the European Economic Community*. Lyon, France: IARC Scientific Publications.

Also see

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Example 34g** — Combined models (generalized responses)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 40g — Crossed models (multilevel)

Description Remarks and examples Reference Also see

Description

To illustrate crossed models, we use

```
. use https://www.stata-press.com/data/r16/gsem_fifeschool
(School data from Fife, Scotland)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_fifeschool.dta
  obs:         3,435                          School data from Fife, Scotland
 vars:             5                          25 Mar 2018 16:17
                                              (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
pid             int     %9.0g                 Primary school ID
sid             byte    %9.0g                 Secondary school ID
attain          byte    %9.0g                 Attainment score at age 16
vrq             int     %9.0g                 Verbal-reasoning score from final
                                                year of primary school
sex             byte    %9.0g                 1: female; 0: male
-------------------------------------------------------------------------------
Sorted by: pid  sid
. notes
_dta:
  1.  Paterson, L. 1991. "Socio-economic status and education attainment:  A
      multidimensional and multilevel study" in _Evaluation and Research in
      Education_ 5: 97-121.
  2.  Each observation is a different student. Each student attended a primary
      school (pid) and a secondary school (sid).
  3.  pid and sid are crossed, not nested.  All combinations are possible.
```

Rabe-Hesketh and Skrondal (2012, 443–460) give an introduction to crossed-effects models and provide other examples of crossed-effects models by using the school data from Fife, Scotland.

See *Structural models 1: Linear regression* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

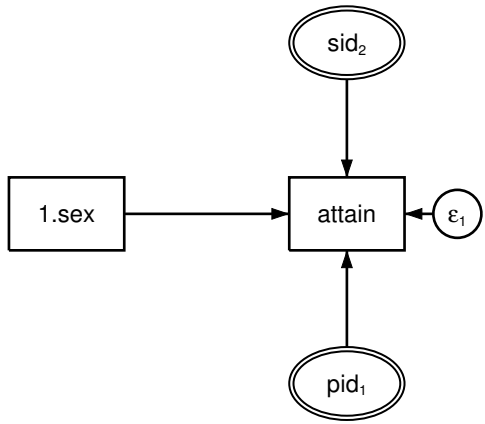
Remarks and examples

Remarks are presented under the following headings:

The crossed model
Fitting the model with the Builder

The crossed model

In these data, 3,435 students attended 148 different primary schools and 19 different secondary schools. The model we wish to fit is



We include latent (random) effects for primary and secondary school because we think that school identities may have an effect.

The command syntax to fit this model is

```
. gsem (attain <- i.sex M1[pid] M2[sid])
note: crossed random-effects model specified; option intmethod(laplace)
implied

Fitting fixed-effects model:

Iteration 0:   log likelihood = -8701.1372
Iteration 1:   log likelihood = -8701.1372

Refining starting values:

Grid node 0:   log likelihood = -8587.4875

Fitting full model:

Iteration 0:   log likelihood = -8587.4875
Iteration 1:   log likelihood = -8573.5619
Iteration 2:   log likelihood = -8563.0123
Iteration 3:   log likelihood = -8561.7426
Iteration 4:   log likelihood = -8561.7354
Iteration 5:   log likelihood = -8561.7354

Generalized structural equation model           Number of obs     =      3,435
Response       : attain
Family         : Gaussian
Link           : identity
Log likelihood = -8561.7354

 ( 1)  [attain]M2[sid] = 1
 ( 2)  [attain]M1[pid] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
attain						
1.sex	.4986152	.0982634	5.07	0.000	.3060224	.6912079
M1[pid]	1	(constrained)				
M2[sid]	1	(constrained)				
_cons	5.257372	.1813785	28.99	0.000	4.901876	5.612867
var(M1[pid])	1.104316	.2022595			.7712452	1.581226
var(M2[sid])	.3457089	.160868			.1388746	.8605941
var(e.attain)	8.053437	.1990023			7.672694	8.453074

Notes:

1. These data are not nested, but the diagram above would look the same even if they were. The fact that primary and secondary schools are crossed and not nested is, however, specified when we enter the model into the SEM Builder and is implicit in the command syntax.

2. We typed `attain <- i.sex M1[pid] M2[sid]`. We would have typed `attain <- i.sex M1[pid] M2[sid<pid]` had secondary school been nested within primary school.

3. `gsem` produced the following note when it began estimation: "crossed random effects detected; option intmethod(laplace) assumed". `gsem` provides four integration methods. The default is `mvaghermite`, which stands for mean–variance adaptive Gauss–Hermite quadrature. The others are `mcaghermite` (mode-curvature adaptive Gauss–Hermite quadrature); `ghermite` (nonadaptive Gauss–Hermite quadrature); and `laplace` (Laplacian approximation).

   In general, the adaptive methods `mvaghermite` and `mcaghermite` are considered superior in terms of accuracy to the nonadaptive method `ghermite`, which is considered superior to the approximation method `laplace`. They also take longer.

Fitting crossed models can be difficult. You may specify `intmethod()` with one of the superior methods, but be aware, convergence may not be achieved in finite time.

Fitting the model with the Builder

Use the diagram in *The crossed model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_fifeschool
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the endogenous variable.

   a. Select the Add observed variable tool, □, and then click in the diagram about one-third of the way in from the right and halfway down from the top. After adding it, you can click inside the rectangle and move the variable if you wish.

   b. In the Contextual Toolbar, select `attain` with the *Variable* control.

5. Create the observed exogenous variable.

   a. Select the Add observed variable tool, □, and then click in the diagram about one-third of the way in from the left and halfway down from the top.

   b. In the Contextual Toolbar, type `1.sex` in the *Variable* control (typing `1.sex` rather than using the "" button to create `i.sex` prevents the rectangle corresponding to the base category for this binary variable from being created);

6. Create the `pid`-level latent variable.

   a. Select the Add multilevel latent variable tool, ◎, and click below the rectangle for `attain`.

   b. In the Contextual Toolbar, click on the "" button.

   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `pid` > *Observations* in the next control.

   d. Specify `M1` as the *Base name*.

   e. Click on **OK**.

7. Create the `sid`-level latent variable.

   a. Select the Add multilevel latent variable tool, ◎, and click above the rectangle for `attain`.

   b. In the Contextual Toolbar, click on the "" button.

   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `sid` > *Observations* in the next control.

   d. Specify `M2` as the *Base name*.

   e. Click on **OK**.

8. Create the paths from the exogenous variables to `attain`.

   a. Select the Add path tool, →.

   b. Click in the right side of the `1.sex` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `attain` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the → tool, draw paths from bottom of the $\texttt{sid}_2$ double oval to the top of the `attain` rectangle and from the top of the $\texttt{pid}_1$ double oval to the bottom of the `attain` rectangle.

9. Clean up the location of the paths.

   If you do not like where the paths have been connected to the rectangles or oval, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle or oval and drag the endpoint.

10. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_cross
```

Reference

Rabe-Hesketh, S., and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*. 3rd ed. College Station, TX: Stata Press.

Also see

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 41g — Two-level multinomial logistic regression (multilevel)

Description Remarks and examples References Also see

Description

We demonstrate two-level multinomial logistic regression with random effects by using the following data:

```
. use https://www.stata-press.com/data/r16/gsem_lineup
(Fictional suspect identification data)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_lineup.dta
  obs:         6,535                          Fictional suspect
                                                identification data
 vars:             6                          29 Mar 2018 10:35
                                               (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
suspect         float   %9.0g                 suspect id
suswhite        float   %9.0g                 suspect is white
violent         float   %9.0g                 violent crime
location        float   %14.0g     loc        lineup location
witmale         float   %9.0g                 witness is male
chosen          float   %9.0g      choice     indvidual identified in linup by
                                                witness
-------------------------------------------------------------------------------
Sorted by: suspect
. notes
_dta:
  1.  Fictional data inspired by Wright, D.B and Sparks, A.T., 1994, "Using
      multilevel multinomial regression to analyse line-up data", _Multilevel
      Modeling Newsletter_, Vol. 6, No. 1
  2.  Data contain repeated values of variable suspect.  Each suspect is viewed
      by multiple witnesses and each witness (1) declines to identify a
      suspect, (2) chooses a foil, or (3) chooses the suspect.
. tabulate location
       lineup
     location |      Freq.     Percent        Cum.
--------------+-----------------------------------
police_station |      2,228       34.09       34.09
       suite_1 |      1,845       28.23       62.33
       suite_2 |      2,462       37.67      100.00
--------------+-----------------------------------
         Total |      6,535      100.00
```

```
. tabulate chosen
  indvidual
 identified
in linup by
    witness |      Freq.     Percent        Cum.
------------+-----------------------------------
       none |      2,811       43.01       43.01
       foil |      1,369       20.95       63.96
    suspect |      2,355       36.04      100.00
------------+-----------------------------------
      Total |      6,535      100.00
```

In what follows, we re-create results similar to those of Wright and Sparks (1994), but we use fictional data. These data resemble the real data used by the authors in proportion of observations having each level of the outcome variable `chosen`, and the data produce results similar to those presented by the authors.

See *Structural models 6: Multinomial logistic regression* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

For additional discussion of fitting multilevel multinomial logistic regression models, see Skrondal and Rabe-Hesketh (2003).

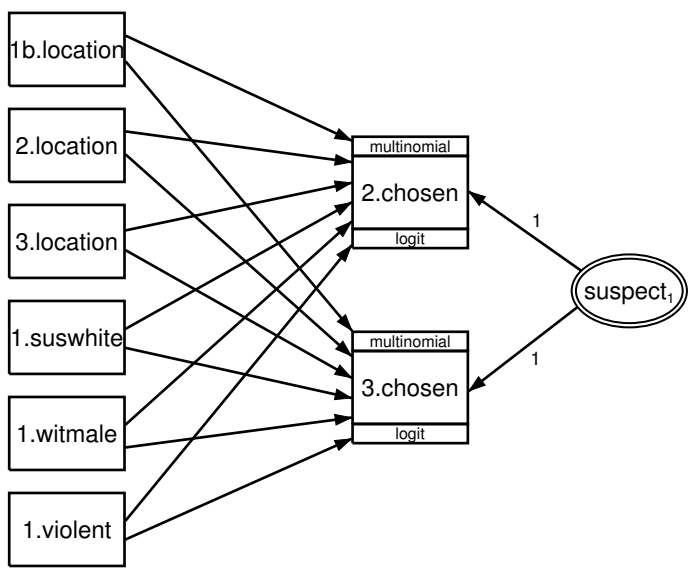
Remarks and examples

Remarks are presented under the following headings:

Two-level multinomial logistic model with shared random effects
Two-level multinomial logistic model with separate but correlated random effects
Fitting the model with the Builder

Two-level multinomial logistic model with shared random effects

We wish to fit the following model:



This model concerns who is chosen in a police lineup. The response variables are `1.chosen`, `2.chosen`, and `3.chosen`, meaning `chosen` = 1 (code for not chosen), `chosen` = 2 (code for foil chosen), and `chosen` = 3 (code for suspect chosen). A foil is a stand-in who could not possibly be guilty of the crime.

We say the response variables are `1.chosen`, `2.chosen`, and `3.chosen`, but `1.chosen` does not even appear in the diagram. By its omission, we are specifying that `chosen` = 1 be treated as the base mlogit category. There are other ways we could have drawn this; see [SEM] **Example 37g**.

In these data, each suspect was viewed by multiple witnesses. In the model, we include a random effect at the suspect level, and we constrain the effect to be equal for chosen values 2 and 3 (selecting the foil or the suspect).

We can fit this model with command syntax by typing

```
. gsem (i.chosen <- i.location i.suswhite i.witmale i.violent M1[suspect]@1),
> mlogit
Fitting fixed-effects model:
Iteration 0:   log likelihood = -6914.9098
Iteration 1:   log likelihood = -6696.7136
Iteration 2:   log likelihood = -6694.0006
Iteration 3:   log likelihood = -6693.9974
Iteration 4:   log likelihood = -6693.9974
Refining starting values:
Grid node 0:   log likelihood = -6705.0919
Fitting full model:
Iteration 0:   log likelihood = -6705.0919  (not concave)
Iteration 1:   log likelihood = -6654.5724
Iteration 2:   log likelihood = -6653.5717
Iteration 3:   log likelihood = -6653.5671
Iteration 4:   log likelihood = -6653.5671
Generalized structural equation model           Number of obs     =      6,535
Response       : chosen
Base outcome   : 1
Family         : multinomial
Link           : logit
Log likelihood = -6653.5671
 ( 1)  [2.chosen]M1[suspect] = 1
 ( 2)  [3.chosen]M1[suspect] = 1
```

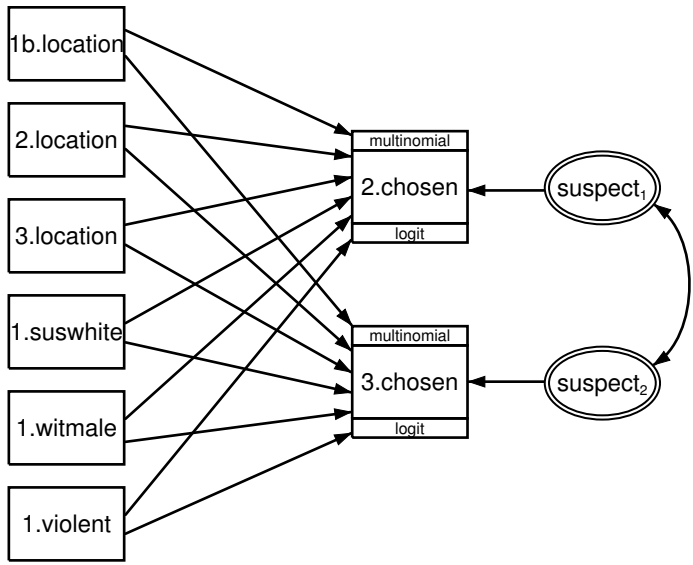
	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.chosen	(base outcome)					
2.chosen						
location						
suite_1	.3867066	.1027161	3.76	0.000	.1853868	.5880264
suite_2	.4915675	.0980312	5.01	0.000	.2994299	.6837051
1.suswhite	-.0275501	.0751664	-0.37	0.714	-.1748736	.1197734
1.witmale	-.0001844	.0680803	-0.00	0.998	-.1336193	.1332505
1.violent	.0356477	.0773658	0.46	0.645	-.1159864	.1872819
M1[suspect]	1	(constrained)				
_cons	-1.002334	.099323	-10.09	0.000	-1.197003	-.8076643
3.chosen						
location						
suite_1	-.2832042	.0936358	-3.02	0.002	-.4667271	-.0996814
suite_2	.1391796	.0863473	1.61	0.107	-.0300581	.3084172
1.suswhite	-.2397561	.0643075	-3.73	0.000	-.3657965	-.1137158
1.witmale	.1419285	.059316	2.39	0.017	.0256712	.2581857
1.violent	-1.376579	.0885126	-15.55	0.000	-1.55006	-1.203097
M1[suspect]	1	(constrained)				
_cons	.1781047	.0833393	2.14	0.033	.0147627	.3414468
var(M1[suspect])	.2538014	.0427302			.1824673	.3530228

Notes:

1. We show the interpretation of mlogit coefficients in [SEM] **Example 37g**.
2. The estimated variance of the random effect is 0.2538, implying a standard deviation of 0.5038. Thus a 1-standard-deviation change in the random effect amounts to a $\exp(0.5038) = 1.655$ change in the relative-risk ratio. The effect is both practically significant and, from the output, statistically significant.
3. This is not the model fit by Wright and Sparks (1994). Those authors did not constrain the random effect to be the same for `chosen` equal to 2 and 3. They included separate but correlated random effects, and then took that even a step further.

Two-level multinomial logistic model with separate but correlated random effects

The model we wish to fit is



This is one of the models fit by Wright and Sparks (1994), although remember that we are using fictional data.

We can fit this model with command syntax by typing

```
. gsem (2.chosen <- i.location i.suswhite i.witmale i.violent M1[suspect])  ///
>      (3.chosen <- i.location i.suswhite i.witmale i.violent M2[suspect]), ///
>      mlogit
```

We did not even mention the assumed covariance between the random effects because latent exogenous variables are assumed to be correlated in the command language. Even so, we can specify the `cov()` option if we wish, and we might do so for emphasis or because we are unsure whether the parameter would be included.

```
. gsem (2.chosen <- i.location i.suswhite i.witmale i.violent M1[suspect])
>      (3.chosen <- i.location i.suswhite i.witmale i.violent M2[suspect]),
>      cov(M1[suspect]*M2[suspect]) mlogit

Fitting fixed-effects model:

Iteration 0:   log likelihood = -6914.9098
Iteration 1:   log likelihood = -6696.7136
Iteration 2:   log likelihood = -6694.0006
Iteration 3:   log likelihood = -6693.9974
Iteration 4:   log likelihood = -6693.9974

Refining starting values:

Grid node 0:   log likelihood = -6793.4228

Fitting full model:

Iteration 0:   log likelihood = -6793.4228  (not concave)
Iteration 1:   log likelihood = -6717.7507  (not concave)
Iteration 2:   log likelihood = -6684.6592
Iteration 3:   log likelihood = -6660.1404
Iteration 4:   log likelihood = -6652.1368
Iteration 5:   log likelihood = -6651.7841
Iteration 6:   log likelihood = -6651.7819
Iteration 7:   log likelihood = -6651.7819

Generalized structural equation model           Number of obs     =      6,535
Response       : chosen
Base outcome   : 1
Family         : multinomial
Link           : logit
Log likelihood = -6651.7819

 ( 1)  [2.chosen]M1[suspect] = 1
 ( 2)  [3.chosen]M2[suspect] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.chosen	(base outcome)					
2.chosen						
location						
suite_1	.3881676	.1004754	3.86	0.000	.1912394	.5850958
suite_2	.48938	.0960311	5.10	0.000	.3011625	.6775974
1.suswhite	-.0260152	.0749378	-0.35	0.728	-.1728906	.1208602
1.witmale	-.0007652	.0679187	-0.01	0.991	-.1338833	.132353
1.violent	.0369381	.0771594	0.48	0.632	-.1142915	.1881677
M1[suspect]	1	(constrained)				
_cons	-1.000382	.0992546	-10.08	0.000	-1.194918	-.8058469
3.chosen						
location						
suite_1	-.2904225	.0968578	-3.00	0.003	-.4802604	-.1005847
suite_2	.1364246	.089282	1.53	0.127	-.0385649	.3114142
1.suswhite	-.2437654	.0647275	-3.77	0.000	-.370629	-.1169018
1.witmale	.139826	.0596884	2.34	0.019	.0228389	.256813
1.violent	-1.388013	.0891863	-15.56	0.000	-1.562815	-1.213212
M2[suspect]	1	(constrained)				
_cons	.1750622	.0851614	2.06	0.040	.008149	.3419754

var(M1[suspect])	.2168248	.0549321			.131965	.3562533
var(M2[suspect])	.2978104	.0527634			.2104416	.421452
cov(M1[suspect], M2[suspect])	.2329749	.0438721	5.31	0.000	.1469872	.3189627

Notes:

1. The estimated variances of the two random effects are 0.2168 and 0.2978, which as explained in the second note of above example, are both practically and statistically significant.
2. The covariance is estimated to be 0.2300. Therefore, $0.2300/\sqrt{0.2168 \times 0.2978} = 0.9052$ is the estimated correlation.
3. Wright and Sparks (1994) were interested in whether the location of the lineup mattered. They found that it did, and that foils were more likely to be chosen at lineups outside of the police station (at the two "specialist" suites). They speculated the cause might be that the police at the station strongly warn witnesses against misidentification, or possibly because the specialist suites had better foils.

Fitting the model with the Builder

Use the first diagram in *Two-level multinomial logistic model with shared random effects* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_lineup
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the independent variables.

   Select the Add observed variables set tool, ▫▫▫, and then click near the bottom of the diagram about one-third of the way in from the left.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. include the levels of the factor variable `location` by clicking on the ▫▫ button next to the *Variables* control. In the resulting dialog box, select the *Factor variable* radio button, select `Main effect` in the *Specification* control, and select `location` in the *Variables* control for *Variable 1*. Click on **Add to varlist**, and then click on **OK**;

   c. type `1.suswhite 1.witmale 1.violent` in the *Variables* control after `i.location` (typing `1.`*varname* rather than using the ▫▫ button to create them as `i.`*varname* factor variables prevents rectangles corresponding to the base categories for these binary variables from being created);

d. select `Vertical` in the *Orientation* control;

e. click on **OK**.

If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the rectangles for the possible outcomes of the multinomial endogenous variable.

   Select the Add observed variables set tool, , and then click in the diagram about one-third of the way in from the right and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. check *Make variables generalized responses*;

   c. select `Multinomial, Logit` in the *Family/Link* control;

   d. select `chosen` in the *Variable* control;

   e. under *Levels*, remove `1b` to prevent the rectangle corresponding to the base category from being created;

   f. select `Vertical` in the *Orientation* control;

   g. select the **Distances** tab;

   h. select `.5` (inch) from the from the *Distance between variables* control;

   i. click on **OK**.

6. Create the paths from the independent variables to the rectangles for outcomes `chosen` = 2 and `chosen` = 3.

   a. Select the Add path tool, .

   b. Click in the right side of the `1b.location` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `2.chosen` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the tool, click in the right side of each independent variable and drag a path to both the `2.chosen` and `3.chosen` rectangles.

7. Create the suspect-level latent variable.

   a. Select the Add multilevel latent variable tool, , and click near the right side of the diagram, vertically centered between `2.chosen` and `3.chosen`.

   b. In the Contextual Toolbar, click on the button.

   c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `suspect` > *Observations* in the next control.

   d. Specify `M1` as the *Base name*.

   e. Click on **OK**.

8. Create the paths from the multilevel latent variable to the rectangles for outcomes `chosen` = 2 and `chosen` = 3.

   a. Select the Add path tool, .

   b. Click in the upper-left quadrant of the $\texttt{suspect}_1$ double oval, and drag a path to the right side of the `2.chosen` rectangle.

c. Continuing with the tool, click in the lower-left quadrant of the suspect$_1$ double oval, and drag a path to the right side of the 3.chosen rectangle.

9. Place constraints on path coefficients from the multilevel latent variable.

   Use the Select tool, , to select the path from the suspect$_1$ double oval to the 2.chosen rectangle. Type 1 in the box in the Contextual Toolbar and press *Enter*. Repeat this process to constrain the coefficient on the path from the suspect$_1$ double oval to the 3.chosen rectangle to 1.

10. Clean up the location of the paths.

    If you do not like where the paths have been connected to the rectangles, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.

11. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

12. If you wish to fit the model described in *Two-level multinomial logistic model with separate but correlated random effects*, use the Select tool to select the path from the suspect$_1$ double oval to the 3.chosen rectangle in the diagram created above. Select **Object > Delete** from the SEM Builder menu.

    Using the Select tool, select the suspect$_1$ double oval and move it up so that it is parallel with the rectangle for 2.chosen.

13. Create the multilevel latent variable corresponding to the random effects of suspect in the 3.chosen equation.

    a. Select the Add multilevel latent variable tool, , and click near the right side of the diagram, next to the 3.chosen rectangle.

    b. In the Contextual Toolbar, click on the button.

    c. Select the nesting level and nesting variable by selecting 2 from the *Nesting depth* control and selecting suspect > *Observations* in the next control.

    d. Specify M2 as the *Base name*.

    e. Click on **OK**.

14. Draw a path from the newly added suspect-level latent variable to 3.chosen.

    Select the Add path tool, click in the left of the suspect$_2$ double oval, and drag a path to the right side of the 3.chosen rectangle.

15. Create the covariance between the random effects.

    a. Select the Add covariance tool, .

    b. Click in the bottom-right quadrant of the suspect$_1$ double oval, and drag a covariance to the top right of the suspect$_2$ double oval.

16. Clean up paths and covariance.

    If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint. Similarly, you can change where the covariance connects to the latent variables by clicking on the covariance and dragging the endpoint. You can also change the bow of the covariance by

clicking on the covariance and dragging the control point that extends from one end of the selected covariance.

17. Estimate again.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram for the first model in the Builder by typing

```
. webgetsem gsem_mlmlogit1
```

You can open a completed diagram for the second model in the Builder by typing

```
. webgetsem gsem_mlmlogit2
```

References

Skrondal, A., and S. Rabe-Hesketh. 2003. Multilevel logistic regression for polytomous data and rankings. *Psychometrika* 68: 267–287.

Wright, D. B., and A. T. Sparks. 1994. Using multilevel multinomial regression to analyse line-up data. *Multilevel Modelling Newsletter* 6: 8–10.

Also see

[SEM] **Example 37g** — Multinomial logistic regression

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 42g — One- and two-level mediation models (multilevel)

Description Remarks and examples References Also see

Description

To demonstrate linear mediation models, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_multmed
(Fictional job-performance data)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
branch	1,500	38	21.65593	1	75
support	1,500	.0084667	.5058316	-1.6	1.8
satis	1,500	.0212	.6087235	-1.6	2
perform	1,500	5.005317	.8949845	2.35022	8.084294

```
. notes
_dta:
  1.  Fictional data on job performance, job satisfaction, and perceived
      support from managers for 1,500 sales employees of a large department
      store in 75 locations.
  2.  Variable support is average of Likert-scale questions, each question
      scored from -2 to 2.
  3.  Variable satis is average of Likert-scale questions, each question scored
      from -2 to 2.
  4.  Variable perform is job performance measured on continuous scale.
```

See *Structural models 1: Linear regression* and *Multilevel mixed-effects models* in [SEM] **Intro 5** for background.

Remarks and examples

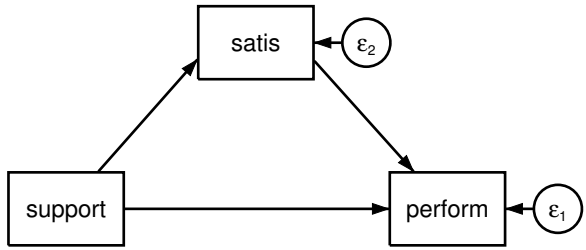
Remarks are presented under the following headings:

One-level model with sem
One-level model with gsem
Two-level model with gsem
Fitting the models with the Builder

One-level model with sem

You can fit single-level mediation models with `sem` or `gsem`. You will be better off using `sem` because then you can use `estat teffects` afterward to compute indirect and total effects.

The model we wish to fit is the simplest form of a mediation model, namely,



We are interested in the effect of managerial support on job performance, but we suspect a portion of the effect might be mediated through job satisfaction. In traditional mediation analysis, the model would be fit by a series of linear regression models as described in Baron and Kenny (1986). That approach is sufficient because the errors are not correlated. The advantage of using structural equation modeling is that you can fit a single model and estimate the indirect and total effects, and you can embed the simple mediation model in a larger model and even use latent variables to measure any piece of the mediation model.

To fit this model with the command syntax, we type

```
. sem (perform <- satis support) (satis <- support)
Endogenous variables
Observed:  perform satis
Exogenous variables
Observed:  support
Fitting target model:
Iteration 0:   log likelihood = -3779.9224
Iteration 1:   log likelihood = -3779.9224
Structural equation model                       Number of obs     =      1,500
Estimation method  = ml
Log likelihood     = -3779.9224
```

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
perform						
satis	.8984401	.0251903	35.67	0.000	.849068	.9478123
support	.6161077	.0303143	20.32	0.000	.5566927	.6755227
_cons	4.981054	.0150589	330.77	0.000	4.951539	5.010569
satis						
support	.2288945	.0305047	7.50	0.000	.1691064	.2886826
_cons	.019262	.0154273	1.25	0.212	-.0109749	.0494989
var(e.perf~m)	.3397087	.0124044			.3162461	.364912
var(e.satis)	.3569007	.0130322			.3322507	.3833795

```
LR test of model vs. saturated: chi2(0)   =      0.00, Prob > chi2 =      .
```

Notes:

1. The direct effect of managerial support on job performance is measured by `perform <- support` and is estimated to be 0.6161. The effect is small albeit highly statistically significant. The standard deviations of performance and support are 0.89 and 0.51. A one standard deviation increase in support improves performance by a third of a standard deviation.
2. The direct effect of job satisfaction on job performance is measured by `perform <- satis` and is estimated to be 0.8984. That also is a moderate effect, practically speaking, and is highly statistically significant.
3. The effect of managerial support on job satisfaction is measured by `satis <- support` and is practically small but statistically significant.
4. What is the total effect of managerial support on performance? It is the direct effect (0.6161) plus the indirect effect of support on satisfaction on performance ($0.2289 \times 0.8984 = 0.2056$), meaning the total effect is 0.8217. It would be desirable to put a standard error on that, but that's more work.

We can use `estat teffects` after estimation to obtain the total effect and its standard error:

```
. estat teffects
```

Direct effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
perform						
satis	.8984401	.0251903	35.67	0.000	.849068	.9478123
support	.6161077	.0303143	20.32	0.000	.5566927	.6755227
satis						
support	.2288945	.0305047	7.50	0.000	.1691064	.2886826

Indirect effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
perform						
satis	0	(no path)				
support	.205648	.0280066	7.34	0.000	.150756	.26054
satis						
support	0	(no path)				

Total effects

	OIM Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Structural						
perform						
satis	.8984401	.0251903	35.67	0.000	.849068	.9478123
support	.8217557	.0404579	20.31	0.000	.7424597	.9010516
satis						
support	.2288945	.0305047	7.50	0.000	.1691064	.2886826

One-level model with gsem

We can fit the same model with `gsem`. The command is the same except that we substitute `gsem` for `sem`, and results are identical:

```
. gsem (perform <- satis support) (satis <- support)
Iteration 0:   log likelihood = -2674.3421
Iteration 1:   log likelihood = -2674.3421  (backed up)
Generalized structural equation model           Number of obs     =      1,500
Response       : perform
Family         : Gaussian
Link           : identity
Response       : satis
Family         : Gaussian
Link           : identity
Log likelihood = -2674.3421
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
perform						
satis	.8984401	.0251903	35.67	0.000	.849068	.9478123
support	.6161077	.0303143	20.32	0.000	.5566927	.6755227
_cons	4.981054	.0150589	330.77	0.000	4.951539	5.010569
satis						
support	.2288945	.0305047	7.50	0.000	.1691064	.2886826
_cons	.019262	.0154273	1.25	0.212	-.0109749	.0494989
var(e.perf~m)	.3397087	.0124044			.3162461	.364912
var(e.satis)	.3569007	.0130322			.3322507	.3833795

After `gsem`, however, we cannot use `estat teffects`:

```
. estat teffects
estat teffects not valid
r(321);
```

We can, however, calculate the indirect and total effects for ourselves and obtain the standard error by using `nlcom`. Referring back to note 4 of the previous section, the formula for the indirect effect and total effects are

$$\text{indirect effect} = \beta_1\beta_4$$

$$\text{total effect} = \beta_2 + \beta_1\beta_4$$

where

β_1 = path coefficient for `perform <- satis`

β_4 = path coefficient for `satis <- support`

β_2 = path coefficient for `perform <- support`

It turns out that we can access the coefficients by typing

β_1 = `_b[perform:satis]`

β_4 = `_b[satis:support]`

β_2 = `_b[perform:support]`

which is most easily revealed by typing

```
. gsem, coeflegend
(output omitted)
```

Thus we can obtain the indirect effect by typing

```
. nlcom _b[perform:satis]*_b[satis:support]
       _nl_1:  _b[perform:satis]*_b[satis:support]
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_nl_1	.205648	.0280066	7.34	0.000	.150756	.26054

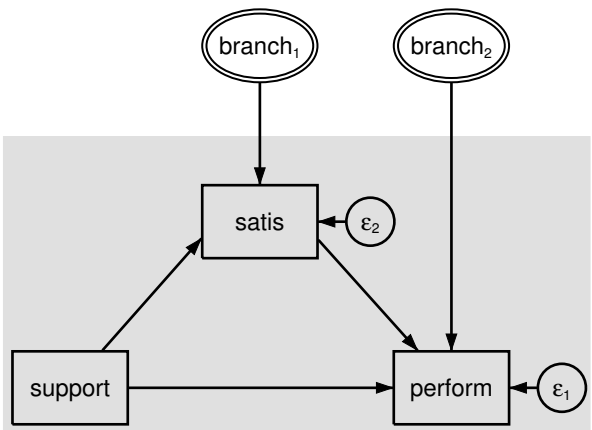
and we can obtain the total effect by typing

```
. nlcom _b[perform:support]+_b[perform:satis]*_b[satis:support]
       _nl_1:  _b[perform:support]+_b[perform:satis]*_b[satis:support]
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_nl_1	.8217557	.0404579	20.31	0.000	.7424597	.9010516

Two-level model with gsem

It may be easier to use `sem` rather than `gsem` for fitting single-level models, but if you want to fit multilevel models, you must use `gsem`. A variation on the model we just fit is



In this model, we include a random intercept in each equation at the branch (individual store) level. The model above is one of many variations on two-level mediation models; see Krull and MacKinnon (2001) for an introduction to multilevel mediation models, and see Preacher, Zyphur, and Zhang (2010) for a discussion of fitting these models with structural equation modeling.

To fit this model with the command syntax, we type

```
. gsem (perform <- satis support M1[branch]) (satis <- support M2[branch]),
> cov(M1[branch]*M2[branch]@0)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2674.3421
Iteration 1:   log likelihood = -2674.3421
Refining starting values:
Grid node 0:   log likelihood = -2132.1613
Fitting full model:
Iteration 0:   log likelihood = -2132.1613  (not concave)
Iteration 1:   log likelihood = -1801.3155
Iteration 2:   log likelihood = -1769.6421
Iteration 3:   log likelihood = -1705.1282
Iteration 4:   log likelihood =  -1703.746
Iteration 5:   log likelihood = -1703.7141
Iteration 6:   log likelihood =  -1703.714
Generalized structural equation model           Number of obs     =      1,500
Response       : perform
Family         : Gaussian
Link           : identity
Response       : satis
Family         : Gaussian
Link           : identity
Log likelihood =  -1703.714
 ( 1)  [perform]M1[branch] = 1
 ( 2)  [satis]M2[branch] = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
perform						
satis	.604264	.0336398	17.96	0.000	.5383313	.6701968
support	.6981525	.0250432	27.88	0.000	.6490687	.7472364
M1[branch]	1	(constrained)				
_cons	4.986596	.0489465	101.88	0.000	4.890663	5.082529
satis						
support	.2692633	.0179649	14.99	0.000	.2340528	.3044739
M2[branch]	1	(constrained)				
_cons	.0189202	.0570868	0.33	0.740	-.0929678	.1308083
var(M1[branch])	.1695962	.0302866			.119511	.2406713
var(M2[branch])	.2384738	.0399154			.1717781	.3310652
var(e.perform)	.201053	.0075451			.1867957	.2163985
var(e.satis)	.1188436	.0044523			.1104299	.1278983

Notes:

1. In *One-level model with sem* above, we measured the direct effects on job performance of job satisfaction and managerial support as 0.8984 and 0.6161. Now the direct effects are 0.6043 and 0.6982.

2. We can calculate the indirect and total effects just as we did in the previous section, which we will do below. We mentioned earlier that there are other variations of two-level mediation models, and how you calculate total effects depends on the model chosen.

   In this case, the indirect effect is

```
. nlcom _b[perform:satis]*_b[satis:support]
       _nl_1:  _b[perform:satis]*_b[satis:support]
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_nl_1	.1627062	.0141382	11.51	0.000	.1349958	.1904165

   and the total effect is

```
. nlcom _b[perform:support]+_b[perform:satis]*_b[satis:support]
       _nl_1:  _b[perform:support]+_b[perform:satis]*_b[satis:support]
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_nl_1	.8608587	.0257501	33.43	0.000	.8103894	.911328

Fitting the models with the Builder

Use the diagram in *One-level model with sem* above for reference.

1. Open the dataset.

   In the Command window, type

```
. use https://www.stata-press.com/data/r16/gsem_multmed
```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create a regression component for the `perform` outcome.

   Select the Add regression component tool, , and then click in the center of the diagram.

   In the resulting dialog box,

   a. select `perform` in the *Dependent variable* control;

   b. select `support` with the *Independent variables* control;

   c. select `Left` in the *Independent variables' direction* control;

   d. click on **OK**.

   e. Use the Select tool, , to select only the `perform` rectangle, and drag it to the right to increase the distance between the rectangles. (You can hold the *Shift* key while dragging to ensure that the movement is directly to the right.)

4. Create the mediating variable.

   a. Select the Add observed variable tool, , and then click in the diagram above the path from `support` to `perform`.

   b. In the Contextual Toolbar, select `satis` with the *Variable* control.

5. Create the paths to and from the mediating variable.
   a. Select the Add path tool, .
   b. Click in the upper right of the `support` rectangle (it will highlight when you hover over it), and drag a path to the lower left of the `satis` rectangle (it will highlight when you can release to connect the path).
   c. Continuing with the tool, draw a path from the lower right of the `satis` rectangle to the upper left of the `perform` rectangle.
6. Clean up the direction of the error term.

   We want the error for each of the endogenous variables to be to the right of the rectangle. The error for `satis` may have been created in another direction. If so,
   a. choose the Select tool, ;
   b. click in the `satis` rectangle;
   c. click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is to the right of the rectangle.
7. Clean up the location of the paths.

   If you do not like where the paths have been connected to the rectangles, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.
8. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.
9. To fit the model in *Two-level model with gsem*, continue with the previous diagram, and put the builder in `gsem` mode by clicking on the button.
10. Create the multilevel latent variable corresponding to the random intercept for `satis`.
    a. Select the Add multilevel latent variable tool, , and click above the rectangle for `satis`.
    b. In the Contextual Toolbar, click on the button.
    c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `branch` > *Observations* in the next control.
    d. Specify `M1` as the *Base name*.
    e. Click on **OK**.
11. Create the multilevel latent variable corresponding to the random intercept for `perform`.
    a. Select the Add multilevel latent variable tool, , and click above the rectangle for `satis` and to the right of the $\texttt{branch}_1$ double oval.
    b. In the Contextual Toolbar, click on the button.
    c. Select the nesting level and nesting variable by selecting `2` from the *Nesting depth* control and selecting `branch` > *Observations* in the next control.
    d. Specify `M2` as the *Base name*.
    e. Click on **OK**.

12. Draw paths from the multilevel latent variables to their corresponding endogenous variables.

    a. Select the Add path tool, →.

    b. Click in the bottom of the `branch`$_1$ double oval, and drag a path to the top of the `satis` rectangle.

    c. Continuing with the → tool, click in the bottom of the `branch`$_2$ double oval, and drag a path to the top of the `perform` rectangle.

13. Estimate again.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram for the first model in the Builder by typing

```
. webgetsem sem_med
```

You can open a completed diagram for the second model in the Builder by typing

```
. webgetsem gsem_mlmed
```

References

Baron, R. M., and D. A. Kenny. 1986. The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology* 51: 1173–1182.

Krull, J. L., and D. P. MacKinnon. 2001. Multilevel modeling of individual and group level mediated effects. *Multivariate Behavorial Research* 36: 249–277.

Preacher, K. J., M. J. Zyphur, and Z. Zhang. 2010. A general multilevel SEM framework for assessing multilevel mediation. *Psychological Methods* 15: 209–233.

Also see

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 43g — Tobit regression

Description Remarks and examples Also see

Description

Tobit regression is demonstrated using `auto.dta`:

```
. use https://www.stata-press.com/data/r16/auto
(1978 Automobile Data)
. generate wgt = weight/1000
```

See *Structural models 1: Linear regression* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

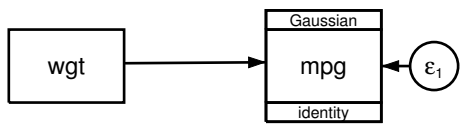
Fitting tobit regression models
Fitting the model with the Builder

Fitting tobit regression models

The first example in [R] **tobit** is

```
. tobit mpg wgt, ll(17)
```

This model corresponds to



Censoring information does not appear in the path diagram by default. It can be added to the path diagram by customizing the appearance of `mpg` in the Builder. The Builder reports the censoring information for `mpg` in the Details pane.

To fit this model with gsem, we type

```
. gsem mpg <- wgt, family(gaussian, lcensored(17))
Refining starting values:
Grid node 0:   log likelihood = -170.32555
Fitting full model:
Iteration 0:   log likelihood = -170.32555
Iteration 1:   log likelihood = -164.66209
Iteration 2:   log likelihood = -164.25471
Iteration 3:   log likelihood = -164.25438
Iteration 4:   log likelihood = -164.25438
Generalized structural equation model           Number of obs     =         74
Response       : mpg                               Uncensored     =         56
Lower limit    : 17                                Left-censored  =         18
Family         : Gaussian                          Right-censored =          0
Link           : identity
Log likelihood = -164.25438
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mpg						
wgt	-6.87305	.700257	-9.82	0.000	-8.245529	-5.500572
_cons	41.49856	2.058384	20.16	0.000	37.4642	45.53291
var(e.mpg)	14.78942	2.817609			10.18085	21.48414

Notes:

1. The reported coefficients and standard errors (SEs) match those reported by tobit.
2. gsem reports the point estimate of e.mpg as 14.78942. This is an estimate of σ^2, the error variance. tobit reports an estimated σ as 3.845701. And $\sqrt{14.78942} = 3.8457$.

Fitting the model with the Builder

Use the diagram in *Fitting tobit regression models* above for reference.

1. Open the dataset and create the rescaled weight variable.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/auto
   . generate wgt = weight/1000
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in gsem mode by clicking on the GSEM button.

4. Create the independent variable.

   a. Select the Add observed variable tool, □.

   b. Click in the diagram about one-fourth of the way in from the left and half of the way up from the bottom.

   c. In the Contextual Toolbar, use the *Variable* control to select the variable wgt.

5. Create the tobit response.

   a. Select the Add generalized response variable tool, .

   b. Click about one-third of the way in from the right side of the diagram, to the right of the `wgt` rectangle.

   c. In the Contextual Toolbar, select `Gaussian, Identity` in the *Family/Link* control (it may already be selected).

   d. In the Contextual Toolbar, use the *Variable* control to select the variable `mpg`.

   e. In the Contextual Toolbar, click on the **Properties...** button.

   f. In the resulting *Variable properties* dialog box, click on the **Censoring...** button in the **Variable** tab.

   g. In the resulting *Censoring* dialog box, select the *Left censored* radio button. In the resulting *Left censoring* box below, select the *Constant* radio button (it may already be selected), and type `17` in the *Constant* control.

   h. Click on **OK** in the *Censoring* dialog box, and then click on **OK** in the *Variable properties* dialog box. The Details pane will now show the censoring information for `mpg`.

6. Create a path from the independent variable to the dependent variable.

   a. Select the Add path tool, .

   b. Click in the right side of the `wgt` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `mpg` rectangle (it will highlight when you can release to connect the path).

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_tobit
```

Also see

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Example 44g** — Interval regression

[SEM] **Example 45g** — Heckman selection model

[SEM] **Example 46g** — Endogenous treatment-effects model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 44g — Interval regression

Description Remarks and examples Also see

Description

Interval regression is demonstrated using `intregxmpl.dta`:

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

See *Structural models 1: Linear regression* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

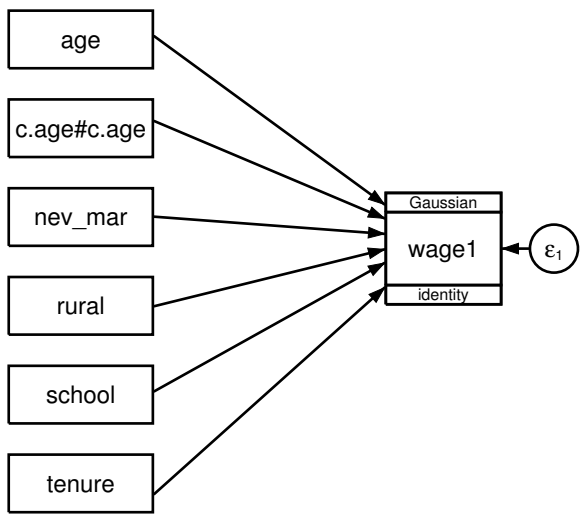
Fitting interval regression models
Fitting the model with the Builder

Fitting interval regression models

The first example in [R] **intreg** is

```
. intreg wage1 wage2 age c.age#c.age nev_mar rural school tenure
```

This model corresponds to



Interval measure information does not appear in the path diagram by default. It can be added to the path diagram by customizing the appearance of `wage1` in the Builder. The Builder reports the interval measure information for `wage1` in the Details pane.

To fit this model with gsem, we type

```
. gsem wage1 <- age c.age#c.age nev_mar rural school tenure,
> family(gaussian, udepvar(wage2))
Refining starting values:
Grid node 0:   log likelihood = -856.59446
Fitting full model:
Iteration 0:   log likelihood = -856.59446
Iteration 1:   log likelihood = -856.33321
Iteration 2:   log likelihood = -856.33293
Iteration 3:   log likelihood = -856.33293
Generalized structural equation model           Number of obs     =        488
Lower response : wage1                             Uncensored     =          0
Upper response : wage2                             Left-censored  =         14
Family         : Gaussian                          Right-censored =          6
Link           : identity                          Interval-cens. =        468
Log likelihood = -856.33293
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
wage1						
age	.7914438	.4433604	1.79	0.074	-.0775265	1.660414
c.age#c.age	-.0132624	.0073028	-1.82	0.069	-.0275757	.0010509
nev_mar	-.2075022	.8119581	-0.26	0.798	-1.798911	1.383906
rural	-3.043044	.7757324	-3.92	0.000	-4.563452	-1.522637
school	1.334721	.1357873	9.83	0.000	1.068583	1.600859
tenure	.8000664	.1045077	7.66	0.000	.5952351	1.004898
_cons	-12.70238	6.367117	-1.99	0.046	-25.1817	-.2230584
var(e.wage1)	53.28454	3.693076			46.51635	61.0375

Notes:

1. Just like intreg, gsem requires two dependent variables for fitting interval regression models. The udepvar() suboption in family(gaussian) allows you to specify the dependent variable containing the upper-limit values for the interval regression. Consequently, the dependent variable participating in the path specification necessarily contains the lower-limit values.
2. Reported coefficients match those reported by intreg.
3. Reported standard errors (SEs) match those reported by intreg.
4. gsem reports the point estimate of e.wage1 as 53.28454. This is an estimate of σ^2, the error variance. intreg reports an estimated σ as 7.299626. And $\sqrt{53.28454} = 7.299626$.

Fitting the model with the Builder

Use the diagram in *Fitting interval regression models* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/intregxmpl
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the independent variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-fourth of the way in from the left and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. type the following variable names in the *Variables* control: `age`, `c.age#c.age`, `nev_mar`, `rural`, `school`, and `tenure`;

   c. select `Vertical` in the *Orientation* control; and

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the interval responses.

   a. Select the Add generalized response variable tool, .

   b. Click about one-third of the way in from the right side of the diagram, to the right of the `nev_mar` rectangle.

   c. In the Contextual Toolbar, select `Gaussian, Identity` in the *Family/Link* control (it may already be selected).

   d. In the Contextual Toolbar, use the *Variable* control to select the variable `wage1`.

   e. In the Contextual Toolbar, click on the **Properties...** button.

   f. In the resulting *Variable properties* dialog box, click on the **Censoring...** button in the **Variable** tab.

   g. In the resulting *Censoring* dialog box, select the *Interval measured, depvar is lower boundary* radio button. In the resulting *Interval measured* box below, use the *Upper bound* control to select the variable `wage2`.

   h. Click on **OK** in the *Censoring* dialog box, and then click on **OK** in the *Variable properties* dialog box. The Details pane will now show that `wage1` is the lower bound and `wage2` is the upper bound of our interval measure.

6. Create paths from the independent variables to the dependent variable.

   a. Select the Add path tool, .

   b. Click in the right side of the `age` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `wage1` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the → tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable:

```
c.age#c.age -> wage1
nev_mar -> wage1
rural -> wage1
school -> wage1
tenure -> wage1
```

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_intreg
```

Also see

[SEM] **Example 38g** — Random-intercept and random-slope models (multilevel)

[SEM] **Example 43g** — Tobit regression

[SEM] **Example 45g** — Heckman selection model

[SEM] **Example 46g** — Endogenous treatment-effects model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 45g — Heckman selection model

Description Remarks and examples References Also see

Description

To demonstrate selection models, we will use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_womenwk
(Fictional data on women and work)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age	2,000	36.208	8.28656	20	59
educ	2,000	13.084	3.045912	10	20
married	2,000	.6705	.4701492	0	1
children	2,000	1.6445	1.398963	0	5
wage	1,343	23.69217	6.305374	5.88497	45.80979

```
. notes
_dta:
  1.  Fictional data on 2,000 women, 1,343 of whom work.
  2.  age ....... age in years
  3.  educ ...... years of schooling
  4.  married ... 1 if married spouse present
  5.  children .. # of children under 12 years
  6.  wage ...... hourly wage (missing if not working)
```

See *Structural models 8: Dependencies between response variables* and *Structural models 9: Unobserved inputs, outputs, or both* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

The Heckman selection model as an SEM
Fitting the Heckman selection model as an SEM
Transforming results and obtaining rho
Fitting the model with the Builder

The Heckman selection model as an SEM

We demonstrate below how `gsem` can be used to fit the Heckman selection model (Gronau 1974; Lewis 1974; Heckman 1976) and produce results comparable to those of Stata's dedicated `heckman` command; see [R] **heckman**.

Our purpose is not to promote `gsem` as an alternative to `heckman`. We have two other purposes.

One is to show that `gsem` can be used to generalize the Heckman selection model to response functions other than linear and, in addition or separately, to include multilevel effects when such effects are present.

The other is to show how Heckman selection models can be included in more complicated SEMs.

For those unfamiliar with this model, it deals with a continuous outcome that is observed only when another equation determines that the observation is selected, and the errors of the two equations are allowed to be correlated. Subjects often choose to participate in an event or medical trial or even the labor market, and thus the outcome of interest might be correlated with the decision to participate. Heckman won a Nobel Prize for this work.

The model is sometimes cast in terms of female labor supply, but it obviously has broader application. Nevertheless, we will consider a female labor-supply example.

Women are offered employment at a wage of w,

$$w_i = \mathbf{X}_i\boldsymbol{\beta} + \epsilon_i$$

Not all women choose to work, and w is observed only for those women who do work. Women choose to work if

$$\mathbf{Z}_i\boldsymbol{\gamma} + \xi_i > 0$$

where

$$\begin{aligned} \epsilon_i &\sim N(0, \sigma^2) \\ \xi_i &\sim N(0, 1) \\ \text{corr}(\epsilon, \xi) &= \rho \end{aligned}$$

More generally, we can think of this model as applying to any continuously measured outcome w_i, which is observed only if $\mathbf{Z}_i\boldsymbol{\gamma} + \xi_i > 0$. The important feature of the model is that the errors ξ_i of the selection equation and the errors ϵ_i of the observed-data equation are allowed to be correlated.

The Heckman selection model can be recast as a two-equation SEM—one linear regression (for the continuous outcome) and the other censored regression (for selection)—and with a latent variable L_i added to both equations. The latent variable is constrained to have variance 1 and to have coefficient 1 in the selection equation, leaving only the coefficient in the continuous-outcome equation to be estimated. For identification, the variance from the censored regression will be constrained to be equal to that of the linear regression. The results of doing this are the following:

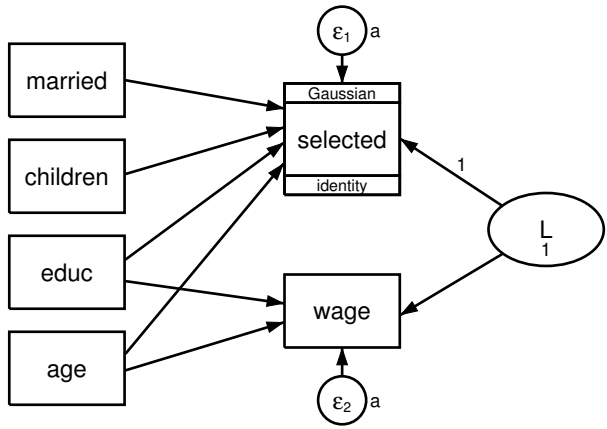
1. Latent variable L_i becomes the vehicle for carrying the correlation between the two equations.
2. All the parameters given above, namely, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, σ^2, and ρ, can be recovered from the SEM estimates.
3. If we call the estimated parameters in the SEM formulation $\boldsymbol{\beta}^*$, $\boldsymbol{\gamma}^*$, and ${\sigma^2}^*$, and let κ denote the coefficient on L_i in the continuous-outcome equation, then

$$\begin{aligned} \boldsymbol{\beta} &= \boldsymbol{\beta}^* \\ \boldsymbol{\gamma} &= \boldsymbol{\gamma}^*/\sqrt{{\sigma^2}^* + 1} \\ \sigma^2 &= {\sigma^2}^* + \kappa^2 \\ \rho &= \kappa/\sqrt{({\sigma^2}^* + \kappa^2)({\sigma^2}^* + 1)} \end{aligned}$$

This parameterization places no restriction on the range or sign of ρ. See Skrondal and Rabe-Hesketh (2004, 107–108).

Fitting the Heckman selection model as an SEM

We wish to fit the following Heckman selection model:



What makes this a Heckman selection model is

1. the inclusion of latent variable `L` in both the continuous-outcome (wage) equation and the censored-outcome selection equation;
2. constraining the `selected <- L` path coefficient to be 1;
3. constraining the variance of `L` to be 1; and
4. constraining the error variances to be equal.

Before we can fit this model, we need to create new variables `selected` and `notselected`. `selected` will equal 0 if the woman works (`wage` is not missing) and missing otherwise. `notselected` is the complement of `selected`: it equals 0 if the woman does not work (`wage` is missing) and missing otherwise. `selected` and `notselected` will be used as the dependent variables in the censored regression, providing the equivalent of a scaled probit regression.

```
. generate selected = 0 if wage < .
(657 missing values generated)
. generate notselected = 0 if wage >= .
(1,343 missing values generated)
. tabulate selected notselected, missing
```

selected	notselected 0	.	Total
0	0	1,343	1,343
.	657	0	657
Total	657	1,343	2,000

Old-time Stata users may be worried that because `wage` is missing in so many observations, namely, all those corresponding to nonworking women, there must be something special we need to do so that `gsem` uses all the data. There is nothing special we need to do. `gsem` counts missing values on an equation-by-equation basis, so it will use all the data for the censored regression part of the model while simultaneously using only the working-woman subsample for the continuous-outcome (wage) part of the model. We use all the data for the censored regression because `gsem` understands the meaning of missing values in the censored dependent variables so long as one of them is nonmissing.

To fit this model in command syntax, we type

```
. gsem (wage <- educ age  L)
>  (selected <- married children educ age L@1,
>   family(gaussian, udepvar(notselected))), var(L@1 e.wage@a e.selected@a)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -5752.6506
Iteration 1:   log likelihood = -5260.9961
Iteration 2:   log likelihood = -5209.2571
Iteration 3:   log likelihood = -5208.9039
Iteration 4:   log likelihood = -5208.9038
Refining starting values:
Grid node 0:   log likelihood = -5208.7006
Fitting full model:
Iteration 0:   log likelihood = -5208.5322  (not concave)
Iteration 1:   log likelihood = -5208.0269
Iteration 2:   log likelihood =  -5202.872  (not concave)
Iteration 3:   log likelihood = -5202.0258
Iteration 4:   log likelihood = -5198.6178
Iteration 5:   log likelihood = -5193.0576
Iteration 6:   log likelihood = -5191.8655
Iteration 7:   log likelihood = -5178.5058
Iteration 8:   log likelihood = -5178.3095
Iteration 9:   log likelihood = -5178.3046
Iteration 10:  log likelihood = -5178.3046
Generalized structural equation model           Number of obs     =      2,000
Response       : wage                           Number of obs     =      1,343
Family         : Gaussian
Link           : identity
Lower response : selected                       Number of obs     =      2,000
Upper response : notselected                       Uncensored     =          0
Family         : Gaussian                          Left-censored  =        657
Link           : identity                          Right-censored =      1,343
                                                   Interval-cens. =          0
Log likelihood = -5178.3046
 ( 1)  [selected]L = 1
 ( 2)  - [/]var(e.wage) + [/]var(e.selected) = 0
 ( 3)  [/]var(L) = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
wage						
educ	.9899512	.0532552	18.59	0.000	.8855729	1.094329
age	.2131282	.020602	10.35	0.000	.172749	.2535074
L	5.923736	.1846818	32.08	0.000	5.561767	6.285706
_cons	.4859114	1.076865	0.45	0.652	-1.624705	2.596528
selected						
married	.6242746	.1054319	5.92	0.000	.4176319	.8309173
children	.6152095	.0652002	9.44	0.000	.4874196	.7429995
educ	.0781542	.0162868	4.80	0.000	.0462327	.1100757
age	.0511983	.006637	7.71	0.000	.0381901	.0642066
L	1	(constrained)				
_cons	-3.493217	.3730379	-9.36	0.000	-4.224357	-2.762076
var(L)	1	(constrained)				
var(e.wage)	.9664635	.2689653			.560141	1.66753
var(e.sele~d)	.9664635	.2689653			.560141	1.66753

Notes:

1. Some of the estimated coefficients and parameters above will match those reported by the `heckman` command and others will not. The above parameters are in the transformed structural equation modeling metric. That metric can be transformed back to the Heckman metric and results will match. The relationship to the Heckman metric is

$$\begin{aligned}\boldsymbol{\beta} &= \boldsymbol{\beta}^*\\ \boldsymbol{\gamma} &= \boldsymbol{\gamma}^*/\sqrt{\sigma^{2^*}+1}\\ \sigma^2 &= \sigma^{2^*}+\kappa^2\\ \rho &= \kappa/\sqrt{(\sigma^{2^*}+\kappa^2)(\sigma^{2^*}+1)}\end{aligned}$$

2. $\boldsymbol{\beta}$ refers to the coefficients on the continuous-outcome (wage) equation. We can read those coefficients directly, without transformation except that we ignore the `wage <- L` path:

$$\texttt{wage} = 0.9900\,\texttt{educ} + 0.2131\,\texttt{age} + 0.4859$$

3. $\boldsymbol{\gamma}$ refers to the selection equation, and because $\boldsymbol{\gamma} = \boldsymbol{\gamma}^*/\sqrt{\sigma^{2^*}+1}$, we must divide the reported coefficients by the square root of $\sigma^{2^*}+1$. What has happened here is that the scaled probit has variance $\sigma^{2^*}+1$, and we are merely transforming back to the standard probit model, which has variance 1. The results are

$$\Pr(\texttt{selected}=0) = \Phi(0.4452\,\texttt{married} + 0.4387\,\texttt{children} + 0.0557\,\texttt{educ} + 0.0365\,\texttt{age} - 2.4910)$$

4. To calculate ρ, we first calculate $\sigma^2 = \sigma^{2^*}+\kappa^2$ and then calculate $\rho = \kappa/\sqrt{\sigma^2(\sigma^{2^*}+1)}$:

$$\begin{aligned}\sigma^2 &= 0.9664 + 5.9237^2 = 36.0571\\ \rho &= 5.9237/\sqrt{\sigma^2(.9664+1)} = 0.7035\end{aligned}$$

5. These transformed results match the results that would have been reported had we typed

```
. heckman wage educ age, select(married children educ age)
```
(*output omitted*)

6. There is an easier way to obtain the transformed results than by hand, and the easier way provides standard errors. That is the subject of the next section.

Transforming results and obtaining rho

We can use Stata's `nlcom` command to perform the transformations we made by hand above, and we can obtain standard errors.

Let's start by obtaining σ^2 and ρ. To remind you, the formulas are

$$\begin{aligned}\sigma^2 &= \sigma^{2^*}+\kappa^2\\ \rho &= \kappa/\sqrt{\sigma^2(\sigma^{2^*}+1)}\end{aligned}$$

We must describe these two formulas in a way that `nlcom` can understand. The Stata notation for parameters σ^{2^*} and κ fit by `gsem` is

$$\begin{aligned}\sigma^{2^*}&: \quad \texttt{_b[/var(e.wage)]}\\ \kappa&: \quad \texttt{_b[wage:L]}\end{aligned}$$

We cannot remember that notation; however, we can type `gsem, coeflegend` to be reminded. We now have all that we need to obtain the estimates of σ^2 and ρ. Because `heckman` reports σ rather than σ^2, we will tell `nlcom` to report the `sqrt(`σ^2`)`:

```
. nlcom (sigma: sqrt(_b[/var(e.wage)] +_b[wage:L]^2))
>       (rho: _b[wage:L]/(sqrt((_b[/var(e.wage)]+1)*(_b[/var(e.wage)]
>  + _b[wage:L]^2))))
       sigma:  sqrt(_b[/var(e.wage)] +_b[wage:L]^2)
         rho:  _b[wage:L]/(sqrt((_b[/var(e.wage)]+1)*(_b[/var(e.wage)]
> + _b[wage:L]^2)))
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sigma	6.004758	.1656471	36.25	0.000	5.680095	6.32942
rho	.703489	.0511861	13.74	0.000	.603166	.8038119

The output above nearly matches what `heckman` reports. `heckman` does not report the test statistics and p-values for these two parameters. In addition, the confidence interval that `heckman` reports for ρ will differ slightly from the above and is better. `heckman` uses a method that will not allow ρ to be outside of -1 and 1, whereas `nlcom` is simply producing a confidence interval for the calculation we requested and in absence of the knowledge that the calculation corresponds to a correlation coefficient. The same applies to the confidence interval for σ, where the bounds are 0 and infinity.

To obtain the coefficients and standard errors for the selection equation, we type

```
. nlcom (married:  _b[selected:married]/sqrt(_b[/var(e.wage)]+1))
>       (children: _b[selected:children]/sqrt(_b[/var(e.wage)]+1))
>       (educ: _b[selected:educ]/sqrt(_b[/var(e.wage)]+1))
>       (age: _b[selected:age]/sqrt(_b[/var(e.wage)]+1))
     married:  _b[selected:married]/sqrt(_b[/var(e.wage)]+1)
    children:  _b[selected:children]/sqrt(_b[/var(e.wage)]+1)
        educ:  _b[selected:educ]/sqrt(_b[/var(e.wage)]+1)
         age:  _b[selected:age]/sqrt(_b[/var(e.wage)]+1)
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
married	.445177	.0673953	6.61	0.000	.3130847	.5772693
children	.4387126	.0277788	15.79	0.000	.3842671	.4931581
educ	.0557326	.0107348	5.19	0.000	.0346927	.0767725
age	.0365101	.0041534	8.79	0.000	.0283696	.0446505

The above output matches what `heckman` reports.

Fitting the model with the Builder

Use the diagram in *Fitting the Heckman selection model as an SEM* above for reference.

1. Open the dataset and create the selection variable.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_womenwk
   . generate selected = 0 if wage < .
   . generate notselected = 0 if wage >= .
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.
4. Create the independent variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-fourth of the way in from the left and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. use the *Variables* control to select the variables `married`, `children`, `educ`, and `age` in this order;

   c. select `Vertical` in the *Orientation* control;

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.
5. Create the generalized response for selection.

   a. Select the Add generalized response variable tool, .

   b. Click about one-third of the way in from the right side of the diagram, to the right of the `married` rectangle.

   c. In the Contextual Toolbar, select `Gaussian, Identity` in the *Family/Link* control (it may already be selected).

   d. In the Contextual Toolbar, select `selected` in the *Variable* control.

   e. In the Contextual Toolbar, click on the **Properties...** button.

   f. In the resulting *Variable properties* dialog box, click on the **Censoring...** button in the **Variable** tab.

   g. In the resulting *Censoring* dialog box, select the *Interval measured, depvar is lower boundary* radio button. In the resulting *Interval measured* box below, use the *Upper bound* control to select the variable `notselected`.

   h. Click on **OK** in the *Censoring* dialog box, and then click on **OK** in the *Variable properties* dialog box. The Details pane will now show `selected` as the lower bound and `notselected` as the upper bound of our interval measure.
6. Create the endogenous `wage` variable.

   a. Select the Add observed variable tool, , and then click about one-third of the way in from the right side of the diagram, to the right of the `age` rectangle.

   b. In the Contextual Toolbar, select `wage` with the *Variable* control.
7. Create paths from the independent variables to the dependent variables.

   a. Select the Add path tool, .

   b. Click in the right side of the `married` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `selected` rectangle (it will highlight when you can release to connect the path).

   c. Continuing with the tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable:

```
children -> selected
educ -> selected
age -> selected
educ -> wage
age -> wage
```

8. Clean up the direction of the error terms.

   We want the error for `selected` to be above the rectangle and the error for `wage` to be below the rectangle, but it is likely they have been created in other directions.

   a. Choose the Select tool, .

   b. Click in the `selected` rectangle.

   c. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is above the rectangle.

   d. Click in the `wage` rectangle.

   e. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is below the rectangle.

9. Create the latent variable.

   a. Select the Add latent variable tool, , and then click at the far right of the diagram and vertically centered between the `selected` and `wage` variables.

   b. In the Contextual Toolbar, type `L` in the *Name* control and press *Enter*.

10. Draw paths from the latent variable to each endogenous variable.

    a. Select the Add path tool, .

    b. Click in the upper left quadrant of the `L` oval, and drag a path to the right side of the `selected` rectangle.

    c. Continuing with the tool, create another path by clicking first in the lower-left quadrant of the `L` oval and dragging a path to the right side of the `wage` rectangle.

11. Place constraints on the variances and on the path from `L` to `selected`.

    a. Choose the Select tool, .

    b. Click on the `L` oval. In the Contextual Toolbar, type `1` in the σ^2 box and press *Enter*.

    c. Click on the error oval attached to the `wage` rectangle. In the Contextual Toolbar, type `a` in the σ^2 box and press *Enter*.

    d. Click on the error oval attached to the `selected` rectangle. In the Contextual Toolbar, type `a` in the σ^2 box and press *Enter*.

    e. Click on the path from `L` to `selected`. In the Contextual Toolbar, type `1` in the β box and press *Enter*.

12. Clean up the location of the paths.

    If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a rectangle and drag the endpoint.

13. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_select
```

References

Gronau, R. 1974. Wage comparisons: A selectivity bias. *Journal of Political Economy* 82: 1119–1143.

Heckman, J. 1976. The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models. *Annals of Economic and Social Measurement* 5: 475–492.

Lewis, H. G. 1974. Comments on selectivity biases in wage comparisons. *Journal of Political Economy* 82: 1145–1155.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Also see

[SEM] **Example 34g** — Combined models (generalized responses)

[SEM] **Example 46g** — Endogenous treatment-effects model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 46g — Endogenous treatment-effects model

Description Remarks and examples References Also see

Description

To illustrate the treatment-effects model, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_union3
(NLSY 1972)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_union3.dta
  obs:         1,693                          NLSY 1972
 vars:            24                          29 Mar 2018 11:39
                                              (_dta has notes)
```

variable name	storage type	display format	value label	variable label
idcode	int	%8.0g		NLS ID
year	int	%8.0g		interview year
birth_yr	byte	%8.0g		birth year
age	byte	%8.0g		age in current year
race	byte	%8.0g	racelbl	race
msp	byte	%8.0g		1 if married, spouse present
nev_mar	byte	%8.0g		1 if never married
grade	byte	%8.0g		current grade completed
collgrad	byte	%8.0g		1 if college graduate
not_smsa	byte	%8.0g		1 if not SMSA
c_city	byte	%8.0g		1 if central city
south	byte	%8.0g		1 if south
ind_code	byte	%8.0g		industry of employment
occ_code	byte	%8.0g		occupation
union	byte	%8.0g		1 if union
wks_ue	byte	%8.0g		weeks unemployed last year
ttl_exp	float	%9.0g		total work experience
tenure	float	%9.0g		job tenure, in years
hours	int	%8.0g		usual hours worked
wks_work	int	%8.0g		weeks worked last year
ln_wage	float	%9.0g		ln(wage/GNP deflator)
wage	double	%10.0g		real wage
black	float	%9.0g		race black
smsa	byte	%8.0g		1 if SMSA

```
Sorted by: idcode
. notes
_dta:
  1.  Data from National Longitudinal Survey of Young Women 14-27 years of age
      (NLSY) in 1968, Center for Human Resource Research, Ohio State
      University, first released in 1989.
  2.  This data was subsetted for purposes of demonstration. Among other
      things, it contains data only for 1972.
```

See *Structural models 8: Dependencies between response variables* and *Structural models 9: Unobserved inputs, outputs, or both* in [SEM] **Intro 5** for background.

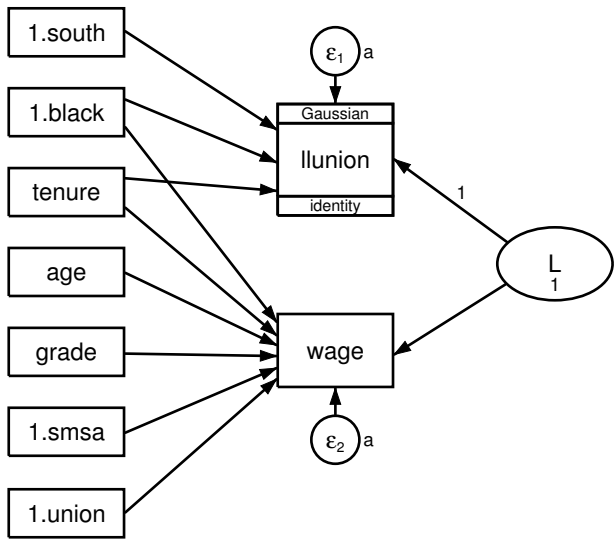
Remarks and examples

Remarks are presented under the following headings:

Fitting the treatment-effects model
Fitting the model with the Builder

Fitting the treatment-effects model

We wish to fit the following model:



We wish to estimate the "treatment effect" of being a union member. That is, we speculate that union membership has an effect on wages, and we want to measure that effect. The problem would be easy if we had data on the same workers from two different but nearly identical universes, one in which the workers were not union members and another in which they were.

The model above is similar to the Heckman selection model we fit in [SEM] **Example 45g**. The differences are that the continuous variable (`wage`) is observed in all cases and that we have a path from the treatment indicator (previously selection, now treatment) to the continuous variable. Just as with the Heckman selection model, we allow for correlation by introducing a latent variable with model identification constraints.

Before we can fit this model, we need to create new variables `llunion` and `ulunion`. `llunion` will equal 0 if `union` is 1 and missing otherwise. `ulunion` is the complement of `llunion`: it equals 0 if `union` is 0 and missing otherwise. `llunion` and `ulunion` will be used as the dependent variables in the treatment equation, providing the equivalent of a scaled probit regression.

```
. generate llunion = 0 if union == 1
(1,433 missing values generated)
. generate ulunion = 0 if union == 0
(709 missing values generated)
```

We can now fit this model with command syntax by typing

```
. gsem (wage <- age grade i.smsa i.black tenure 1.union L)
>      (llunion <- i.black tenure i.south L@1,
>       family(gaussian, udepvar(ulunion))),
>      var(L@1 e.wage@a e.llunion@a)
Fitting fixed-effects model:

Iteration 0:   log likelihood = -3376.1731
Iteration 1:   log likelihood = -3096.7893
Iteration 2:   log likelihood = -3061.7875
Iteration 3:   log likelihood = -3061.4973
Iteration 4:   log likelihood =  -3061.497
Iteration 5:   log likelihood =  -3061.497

Refining starting values:

Grid node 0:   log likelihood = -3064.9724

Fitting full model:

Iteration 0:   log likelihood = -3064.9427
Iteration 1:   log likelihood = -3060.3304
Iteration 2:   log likelihood = -3055.0099
Iteration 3:   log likelihood = -3096.6772
Iteration 4:   log likelihood = -3062.6735
Iteration 5:   log likelihood = -3054.2853
Iteration 6:   log likelihood =  -3051.673
Iteration 7:   log likelihood = -3051.5758
Iteration 8:   log likelihood =  -3051.575
Iteration 9:   log likelihood =  -3051.575

Generalized structural equation model           Number of obs     =      1,210

Response       : wage
Family         : Gaussian
Link           : identity

                                                Censoring of obs:
Lower response : llunion                          Uncensored     =          0
Upper response : ulunion                          Left-censored  =        957
Family         : Gaussian                         Right-censored =        253
Link           : identity                         Interval-cens. =          0

Log likelihood =  -3051.575

 ( 1)  [llunion]L = 1
 ( 2)  - [/]var(e.wage) + [/]var(e.llunion) = 0
 ( 3)  [/]var(L) = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
wage						
age	.1487409	.0193291	7.70	0.000	.1108566	.1866252
grade	.4205658	.0293577	14.33	0.000	.3630258	.4781057
1.smsa	.9117044	.1249041	7.30	0.000	.6668969	1.156512
1.black	-.7882472	.1367077	-5.77	0.000	-1.056189	-.520305
tenure	.1524015	.0369595	4.12	0.000	.0799621	.2248408
1.union	2.945816	.2749549	10.71	0.000	2.406914	3.484718
L	-1.706795	.1288024	-13.25	0.000	-1.959243	-1.454347
_cons	-4.351572	.5283952	-8.24	0.000	-5.387207	-3.315936
llunion						
1.black	.6704049	.148057	4.53	0.000	.3802185	.9605913
tenure	.1282024	.0357986	3.58	0.000	.0580384	.1983664
1.south	-.8542673	.136439	-6.26	0.000	-1.121683	-.5868518
L	1	(constrained)				
_cons	-1.302676	.1407538	-9.25	0.000	-1.578548	-1.026804

var(L)	1	(constrained)		
var(e.wage)	1.163821	.2433321	.7725324	1.753298
var(e.llun~n)	1.163821	.2433321	.7725324	1.753298

Notes:

1. The treatment effect is measured by the coefficient on the path *treatment_variable*`->`*continuous_variable* or, in our case, `1.union->wage`. It is estimated to be 2.9458, which is practically large and statistically significant.

2. The interpretation formulas are the same as for the Heckman selection model in [SEM] **Example 45g**, namely,

$$\boldsymbol{\beta} = \boldsymbol{\beta}^*$$
$$\boldsymbol{\gamma} = \boldsymbol{\gamma}^*/\sqrt{\sigma^{2*}+1}$$
$$\sigma^2 = \sigma^{2*} + \kappa^2$$
$$\rho = \kappa/\sqrt{(\sigma^{2*}+\kappa^2)(\sigma^{2*}+1)}$$

   To remind you, $\boldsymbol{\beta}$ are the coefficients in the continuous-outcome equation, $\boldsymbol{\gamma}$ are the coefficients in the treatment equation, σ^2 is the variance of the error in the continuous-outcome equation, and ρ is the correlation between the errors in the treatment and continuous-outcome equations.

3. In the output above, σ^{2*} (`var(e.wage)`) is 1.1638 and κ (the path coefficient on `wage<-L`) is -1.7068. In [SEM] **Example 45g**, we calculated ρ by hand and then showed how to use the software to obtain the value and its standard error. This time, we will go right to the software.

   After obtaining symbolic names by typing `gsem, coeflegend`, we type the following to obtain ρ:

```
. nlcom (rho: _b[wage:L]/(sqrt(_b[/var(e.wage)] + 1)*sqrt(_b[/var(e.wage)] +
> _b[wage:L]^2)))
        rho:  _b[wage:L]/(sqrt(_b[/var(e.wage)] + 1)*sqrt(_b[/var(e.wage)] + _
> b[wage:L]^2))
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
rho	-.574648	.060969	-9.43	0.000	-.6941451	-.4551509

4. We can obtain the untransformed treatment coefficients just as we did in [SEM] **Example 45g**.

5. Just as with the Heckman selection model, with `gsem`, the treatment-effects model can be applied to generalized outcomes and include multilevel effects. See Skrondal and Rabe-Hesketh (2004, chap. 14.5) for an example with a Poisson response function.

Fitting the model with the Builder

Use the diagram in *Fitting the treatment-effects model* above for reference.

1. Open the dataset.

   In the Command window, type

```
. use https://www.stata-press.com/data/r16/gsem_union3
. generate llunion = 0 if union == 1
. generate ulunion = 0 if union == 0
```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the independent variables.

   Select the Add observed variables set tool, , and then click in the diagram about one-fourth of the way in from the left and one-fourth of the way up from the bottom.

   In the resulting dialog box,

   a. select the *Select variables* radio button (it may already be selected);

   b. type `1.south`, `1.black`, `tenure`, `age`, `grade`, `1.smsa`, and `1.union` in the *Variables* control (typing `1.`*varname* rather than using the button to create `i.`*varname* prevents the rectangle corresponding to the base category for these binary variables from being created);

   c. select `Vertical` in the *Orientation* control; and

   d. click on **OK**.

   If you wish, move the set of variables by clicking on any variable and dragging it.

5. Create the generalized response for `llunion`.

   a. Select the Add generalized response variable tool, .

   b. Click about one-third of the way in from the right side of the diagram, to the right of the `1.black` rectangle.

   c. In the Contextual Toolbar, select `Gaussian, Identity` in the *Family/Link* control (it may already be selected).

   d. In the Contextual Toolbar, select `llunion` in the *Variable* control.

   e. In the Contextual Toolbar, click on the **Properties...** button.

   f. In the resulting *Variable properties* dialog box, click on the **Censoring...** button in the **Variable** tab.

   g. In the resulting *Censoring* dialog box, select the *Interval measured, depvar is lower boundary* radio button. In the resulting *Interval measured* box below, use the *Upper bound* control to select the variable `ulunion`.

   h. Click on **OK** in the *Censoring* dialog box, and then click on **OK** in the *Variable properties* dialog box. The Details pane will now show `llunion` as the lower bound and `ulunion` as the upper bound for our interval measure.

6. Create the endogenous `wage` variable.

   a. Select the Add observed variable tool, , and then click about one-third of the way in from the right side of the diagram, to the right of the `grade` rectangle.

   b. In the Contextual Toolbar, select `wage` with the *Variable* control.

7. Create paths from the independent variables to the dependent variables.

   a. Select the Add path tool, .

   b. Click in the right side of the `1.south` rectangle (it will highlight when you hover over it), and drag a path to the left side of the `llunion` rectangle (it will highlight when you can release to connect the path).

c. Continuing with the → tool, create the following paths by clicking first in the right side of the rectangle for the independent variable and dragging it to the left side of the rectangle for the dependent variable:

```
1.black -> llunion
tenure -> llunion
1.black -> wage
tenure -> wage
age -> wage
grade -> wage
1.smsa -> wage
1.union -> wage
```

8. Clean up the direction of the error terms.

We want the error for `llunion` to be above the rectangle and the error for `wage` to be below the rectangle, but it is likely they have been created in other directions.

a. Choose the Select tool, .

b. Click in the `llunion` rectangle.

c. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is above the rectangle.

d. Click in the `wage` rectangle.

e. Click on one of the **Error rotation** buttons, , in the Contextual Toolbar until the error is below the rectangle.

9. Create the latent variable.

a. Select the Add latent variable tool, , and then click at the far right of the diagram and vertically centered between the `llunion` and `wage` variables.

b. In the Contextual Toolbar, type `L` in the *Name* control and press *Enter*.

10. Draw paths from the latent variable to each endogenous variable.

a. Select the Add path tool, →.

b. Click in the upper-left quadrant of the `L` oval, and drag a path to the right side of the `llunion` rectangle.

c. Continuing with the → tool, create another path by clicking first in the lower-left quadrant of the `L` oval and dragging a path to the right side of the `wage` rectangle.

11. Place constraints on the variances and on the path from `L` to `llunion`.

a. Choose the Select tool, .

b. Click on the `L` oval. In the Contextual Toolbar, type `1` in the σ^2 box and press *Enter*.

c. Click on the error oval attached to the `wage` rectangle. In the Contextual Toolbar, type `a` in the σ^2 box and press *Enter*.

d. Click on the error oval attached to the `llunion` rectangle. In the Contextual Toolbar, type `a` in the σ^2 box and press *Enter*.

e. Click on the path from `L` to `llunion`. In the Contextual Toolbar, type `1` in the β box and press *Enter*.

12. Clean up the location of the paths.

    If you do not like where a path has been connected to its variables, use the Select tool, , to click on the path, and then simply click on where it connects to a variable and drag the endpoint.

13. Estimate.

    Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_treat
```

References

Center for Human Resource Research. 1989. *National Longitudinal Survey of Labor Market Experience, Young Women 14–24 years of age in 1968*. Columbus, OH: Ohio State University Press.

Drukker, D. M. 2016. A generalized regression-adjustment estimator for average treatment effects from panel data. *Stata Journal* 16: 826–836.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

Also see

[SEM] **Example 34g** — Combined models (generalized responses)

[SEM] **Example 45g** — Heckman selection model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 47g — Exponential survival model

Description Remarks and examples Also see

Description

In this example, we demonstrate how to fit parametric survival models with `gsem`. Specifically, in this example, we fit an exponential model, but much of the discussion applies to Weibull, gamma, loglogistic, and lognormal models as well.

```
. use https://www.stata-press.com/data/r16/gsem_kva
(Generator experiment)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_kva.dta
  obs:            12                          Generator experiment
 vars:             3                          23 Jan 2019 21:41
                                              (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
failtime        int     %9.0g                 Time until failure (hrs.)
load            byte    %9.0g                 Overload (kVA)
bearings        byte    %9.0g                 Has new bearings
------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Artificial experimental data on two types of bearings in emergency
      generators.
  2.  The purpose is to compare the ability to withstand overloads of new-style
      bearings to old-style bearings.
  3.  All generators are run until failure.
  4.  Experiments were performed under overloads of 20, 25, 30, 35, and 40 kVA.
```

See *Structural models 7: Survival models* in [SEM] **Intro 5** for background.

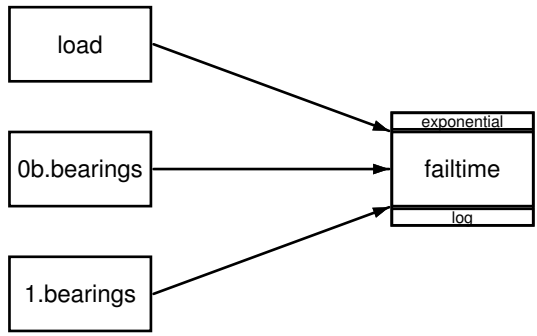
Remarks and examples

Remarks are presented under the following headings:

Fitting the exponential model
Obtaining hazard ratios
Fitting the model with the Builder

Fitting the exponential model

We wish to fit the following model:



That is, we wish to fit an exponential model in which the failure time of bearings (`failtime`) depends on the amount of overload (`load`) and whether the bearings are the new style (`bearings` = 1) or the old style (`bearings` = 0).

To fit this model, we use the `gsem` command with the `exponential` option.

```
. gsem (failtime <- load i.bearings), exponential
Iteration 0:   log likelihood =  -84108.99
Iteration 1:   log likelihood = -67.363768
Iteration 2:   log likelihood = -66.855013
Iteration 3:   log likelihood = -62.300033
Iteration 4:   log likelihood = -62.227842
Iteration 5:   log likelihood = -62.227568
Iteration 6:   log likelihood = -62.227568
Generalized structural equation model           Number of obs     =         12
Response       : failtime                       No. of failures   =         12
Family         : exponential                    Time at risk      =        896
Form           : proportional hazards
Link           : log
Log likelihood = -62.227568
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
failtime						
load	.0611166	.0354318	1.72	0.085	-.0083284	.1305616
1.bearings	-.2194825	.5773503	-0.38	0.704	-1.351068	.9121033
_cons	-5.756595	1.056403	-5.45	0.000	-7.827106	-3.686084

Notes:

1. By default, exponential models are fit in the proportional-hazards metric. The `exponential` option can be replaced with `family(exponential, aft)` if you want to fit the model in the accelerated failure-time metric.
2. `gsem` reports coefficients. These coefficients match those reported by the equivalent `streg` command,

   ```
   . stset failtime
   . streg load bearings, distribution(exponential) nohr
   ```

3. Without the `nohr` option, `streg` reports hazard ratios, which are the exponentiated coefficients.

Obtaining hazard ratios

The `estat eform` command reports exponentiated coefficients for models fit with `gsem`, so we can obtain hazard ratios as follows.

```
. estat eform
```

	exp(b)	Std. Err.	z	P>|z|	[95% Conf.	Interval]
failtime						
load	1.063023	.0376648	1.72	0.085	.9917062	1.139468
1.bearings	.8029342	.4635743	-0.38	0.704	.2589635	2.489553
_cons	.0031619	.0033402	-5.45	0.000	.0003988	.02507

The hazard ratio of 0.80 for `bearings` = 1 indicates that the predicted hazard of failure for the new style of bearing is 80% of the hazard for a bearing of the old type, provided that they have the same loading.

Fitting the model with the Builder

Use the diagram in *Fitting the exponential model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_kva
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the exponential regression component for `failtime`.

   Select the Add regression component tool, , and then click in the diagram about one-third of the way in from the left and halfway down.

   In the resulting dialog box,

   a. select `failtime` in the *Dependent variable* control;

   b. check *Make response generalized*;

   c. select `Exponential, Log` in the *Family/Link* control;

   d. select the *Select variables* radio button (it may already be selected);

   e. use the *Independent variables* control to select the variable `load`;

   f. include the levels of the factor variable `bearings` by clicking on the "" button next to the *Independent variables* control. In the resulting dialog box, select the *Factor variable* radio button, select `Main effect` in the *Specification* control, and select `bearings` in the *Variables* control for *Variable 1*. Click on **Add to varlist**, and then click on **OK**;

   g. select `Left` in the *Independent variables' direction* control;

   h. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Clean up.

   The box for `failtime` is created closer to the independent variables than it is in the example diagram. Use the Select tool, , and click on the box for `failtime`. Drag it to the right to allow more space for results along the paths.

6. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_exp
```

Also see

[SEM] **Example 48g** — Loglogistic survival model with censored and truncated data

[SEM] **Example 49g** — Multiple-group Weibull survival model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat eform** — Display exponentiated coefficients

Title

Example 48g — Loglogistic survival model with censored and truncated data

Description Remarks and examples Reference Also see

Description

In this example, we demonstrate how to fit a survival model to data that are both left-truncated and right-censored.

```
. use https://www.stata-press.com/data/r16/gsem_diet
(Diet data with dates)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_diet.dta
  obs:           337                          Diet data with dates
 vars:            11                          16 Jan 2019 11:24
                                              (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
id              int     %9.0g                 Subject identity number
fail            byte    %8.0g                 Outcome (CHD = 1 3 13)
job             byte    %8.0g                 Occupation
month           byte    %8.0g                 month of survey
energy          float   %9.0g                 Total energy (1000kcals/day)
height          float   %9.0g                 Height (cm)
weight          float   %9.0g                 Weight (kg)
hienergy        byte    %9.0g                 Indicator for high energy
doe             int     %td                   Date of entry
dox             int     %td                   Date of exit
dob             int     %td                   Date of birth
------------------------------------------------------------------------------
Sorted by: id
. notes
_dta:
  1.  Data from J.N. Morris, J. W. Marr, and D. G. Clayton, 19 Nov 1977, "Diet
      and heart: A postscript", _British Medical Journal_, vol. 2, no. 6098,
      1307-1314.
```

See *Structural models 7: Survival models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

Censoring and truncation
Using stset to declare survival characteristics
Fitting the loglogistic model
Fitting the model with the Builder

Censoring and truncation

Survival datasets often include observations that are right-censored or left-truncated or both. When fitting survival models with `family(exponential)`, `family(gamma)`, `family(loglogistic)`, `family(lognormal)`, and `family(weibull)`, we can include the suboption `ltruncated()`, which specifies a left-truncation point, and the suboption `failure()`, which indicates whether an observation records a failure or whether it was censored. For instance,

```
. gsem failtime ..., failure(weibull, failure(failed) ltruncated(enter))
```

fits a Weibull model for time to failure (`failtime`) where a failure is observed for observations with `failed` = 1 and observations with `failed` = 0 are censored. In addition, observations are left-truncated at the time specified in `enter`.

Using stset to declare survival characteristics

If you are familiar with Stata's other commands for survival analysis, such as `streg`, you may have been surprised to see that we did not need to `stset` our data to specify the failure time, censoring, and the truncation variables before using the `gsem` command above. Most survival analysis commands rely on `stset` to record information on censoring and truncation.

`stset` can also be used to transform time in an analysis-time metric. Analysis time is the time a subject is at risk. In this metric, a time of 0 is the time when the subject becomes at risk. `gsem` assumes that the dependent variable is already recorded in analysis time. If you have data in another scale, such as calendar time, you will need to transform your variables.

Although not required by `gsem`, `stset` provides a convenient way to transform data into analysis time. You can also specify truncation and censoring variables just as you would before fitting survival models with other commands. See [ST] **stset** for details on declaring survival data using this command.

In the dataset described above, `dox` records the date an individual is diagnosed with coronary heart disease, cancer, or another disease of interest. `fail` has a nonzero code for individuals diagnosed with a disease and a zero for individuals who were censored. `dob` records date of birth, and `doe` is the date of entry to the study. We could transform the data into analysis time using `stset` as follows:

```
. stset dox, failure(fail) origin(time dob) enter(time doe) id(id)
```

This syntax gives us analysis time in days. Instead, we want to express the analysis time in years, so we type

```
. stset dox, failure(fail) origin(time dob) enter(time doe) id(id) scale(365.25)
                id:  id
     failure event:  fail != 0 & fail < .
obs. time interval:  (dox[_n-1], dox]
 enter on or after:  time doe
 exit on or before:  failure
    t for analysis:  (time-origin)/365.25
            origin:  time dob
--------------------------------------------------------------------------------
       337  total observations
         0  exclusions
--------------------------------------------------------------------------------
       337  observations remaining, representing
       337  subjects
        80  failures in single-failure-per-subject data
 4,603.669  total analysis time at risk and under observation
                                             at risk from t =         0
                                  earliest observed entry t =  30.07529
                                       last observed exit t =  69.99863
```

The `stset` command generates the variables `_t0`, `_t`, `_d`, and `_st`.

Variable `_st` is equal to 1 unless there is a problem in the settings (for example, somebody dies before being born), in which case it is equal to 0.

Variable `_t0` indicates when the individuals enter the study, in the analysis-time scale. Variable `_t` indicates when the individual failed or was censored, also in analysis-time scale. Variable `_d` is the failure indicator.

For example,

```
. list dob doe dox fail _t0 _t _d _st if id == 1
```

```
1.
```

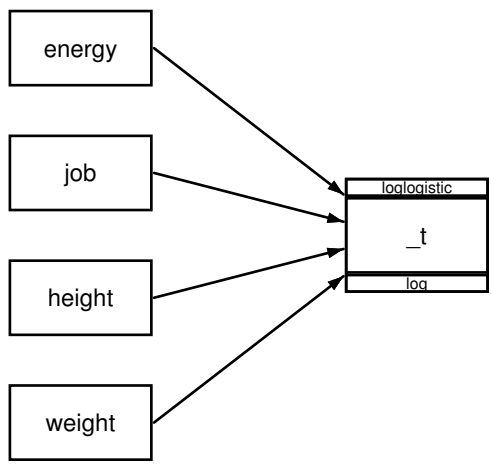
dob	doe	dox	fail	_t0	_t	_d
04jan1915	16aug1964	01dec1976	0	49.615332	61.908282	0

_st
1

This participant was born in 1915, entered the study in 1964, and was censored in 1976. In analysis time, this is expressed as follows: The person entered the study at age 49.6 and was censored at age 61.9.

Fitting the loglogistic model

We can use the variables created by `stset` to fit the model



We specify these variables directly in our gsem command.

```
. gsem (_t <- energy job height weight,
> family(loglogistic, failure(_d) ltruncated(_t0)))
```

(*iteration log omitted*)

```
Generalized structural equation model           Number of obs     =        332
Response       : _t                             No. of failures   =         78
Family         : loglogistic                    Time at risk      =  4533.5715
Form           : accelerated failure-time
Link           : log
Log likelihood = -378.82795
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_t						
energy	.0694548	.0583516	1.19	0.234	-.0449123	.1838219
job	.0102962	.0297246	0.35	0.729	-.0479629	.0685552
height	.0107453	.0047026	2.28	0.022	.0015283	.0199623
weight	.0004624	.0025579	0.18	0.857	-.004551	.0054758
_cons	2.210313	.7830543	2.82	0.005	.6755544	3.745071
/_t						
logs	-1.818008	.1701509			-2.151498	-1.484518

This model is fit in the accelerated failure-time metric, and gsem reports coefficients. We can use estat eform to obtain exponentiated coefficients, which are interpreted as time ratios.

```
. estat eform
```

	exp(b)	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_t						
energy	1.071924	.0625485	1.19	0.234	.9560813	1.201802
job	1.010349	.0300322	0.35	0.729	.9531691	1.07096
height	1.010803	.0047534	2.28	0.022	1.00153	1.020163
weight	1.000463	.0025591	0.18	0.857	.9954593	1.005491
_cons	9.118567	7.140333	2.82	0.005	1.965122	42.31201

Each of the time ratios is just above 1, so an increase in any of the covariates would slightly increase the expected time to failure (for example, having a job increases the expected time until developing a disease by 1.01). However, only the time ratio for height is significantly different from 1.

Fitting the model with the Builder

Use the diagram in *Fitting the loglogistic model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_diet
   . stset dox, failure(fail) origin(time dob) enter(time doe) id(id) scale(365.25)
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.
4. Create the loglogistic regression component for `_t`.

   Select the Add regression component tool, , and then click in the diagram about one-third of the way in from the left and halfway down.

   In the resulting dialog box,

   a. select `_t` in the *Dependent variable* control;
   b. check *Make response generalized*;
   c. select `Loglogistic, Log` in the *Family/Link* control;
   d. select the *Select variables* radio button (it may already be selected);
   e. use the *Independent variables* control to select the variables `energy`, `job`, `height`, and `weight`;
   f. select `Left` in the *Independent variables' direction* control;
   g. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.
5. Specify censoring and truncation variables.

   a. Choose the Select tool, .
   b. Click on the box for `_t`.
   c. In the Contextual Toolbar, click on the **Properties...** button.
   d. In the resulting *Variable properties* dialog box, click on the **Failure and truncation...** button in the **Variable** tab.
   e. In the resulting *Failure and truncation* dialog box, select `_d` in the *Failure variable* control. Check the *Survival time is left-truncated* box. Select the *Variable containing truncation values* radio button, and select `_t0` in the *Variable* control. Click on **OK**.
   f. Click on **OK** in the *Variables properties* dialog box.
6. Clean up.

   The box for `_t` is created closer to the independent variables than it is in the example diagram. Use the Select tool, , and click on the box for `_t`. Drag it to the right to allow more space for results along the paths.
7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *GSEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_llog
```

Reference

Morris, J. N., J. W. Marr, and D. G. Clayton. 1977. Diet and heart: A postscript. *British Medical Journal* 19: 1307–1314.

Also see

[SEM] **Example 47g** — Exponential survival model

[SEM] **Example 49g** — Multiple-group Weibull survival model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat eform** — Display exponentiated coefficients

Title

Example 49g — Multiple-group Weibull survival model

Description Remarks and examples Also see

Description

Below we demonstrate `gsem`'s `group()` option, which allows us to fit models in which coefficients, intercepts, and other types of parameters differ across groups of the data. We will fit a Weibull model to the following survival data

```
. use https://www.stata-press.com/data/r16/gsem_cancer
(Patient Survival in Drug Trial)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_cancer.dta
  obs:            48                          Patient Survival in Drug Trial
 vars:             4                          16 Jan 2019 15:41
                                              (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
studytime       byte    %8.0g                 Months to death or end of exp.
died            byte    %8.0g                 1 if patient died
drug            byte    %8.0g                 Drug type (1=placebo)
age             byte    %8.0g                 Patient's age at start of exp.
-------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Artificial data on cancer patient survival.
  2.  Drug 1 is a placebo.  Drugs 2 and 3 are alternative treatments.
```

See [SEM] **Example 47g** and [SEM] **Example 48g** for how to fit survival models using `gsem`. In this example, we focus on fitting multiple-group models using `gsem`'s `group()` and `ginvariant()` options. See [SEM] **Intro 6** for background on these options.

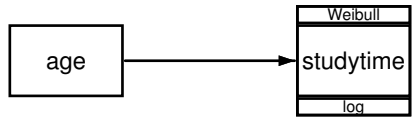
Remarks and examples

Remarks are presented under the following headings:

Fitting the multiple-group model
Fitting the model with the Builder

Fitting the multiple-group model

We want to fit the model



If we do not allow for group differences, we can fit this model by typing

```
. gsem (studytime <- age, family(weibull, failure(died)))
```

However, this study used one placebo and two drugs. We want to allow the intercept and auxiliary parameter to vary across the levels of `drug` but constrain the coefficient on `age` to be equal across the levels of `drug`. In other words, we want to fit a stratified Weibull model. We add the `group(drug)` and `ginvariant(coef)` options to our `gsem` command to specify that `drug` is the group identifier and that coefficients should not vary across groups.

```
. gsem (studytime <- age, family(weibull, failure(died))),
> group(drug) ginvariant(coef)
```

(*iteration log omitted*)

```
Generalized structural equation model           Number of obs     =         48
Grouping variable  = drug                       Number of groups  =          3
Log likelihood     = -109.28976

 ( 1)  [studytime]1bn.drug#c.age - [studytime]3.drug#c.age = 0
 ( 2)  [studytime]2.drug#c.age - [studytime]3.drug#c.age = 0

Group          : 1                              Number of obs     =         20
Response       : studytime                      No. of failures   =         19
Family         : Weibull                        Time at risk      =        180
Form           : proportional hazards
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
studytime						
age	.1212332	.0367538	3.30	0.001	.049197	.1932694
_cons	-10.36921	2.341022	-4.43	0.000	-14.95753	-5.780896
/studytime						
ln_p	.4541282	.1715663			.1178645	.7903919

```
Group          : 2                              Number of obs     =         14
Response       : studytime                      No. of failures   =          6
Family         : Weibull                        Time at risk      =        209
Form           : proportional hazards
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
studytime						
age	.1212332	.0367538	3.30	0.001	.049197	.1932694
_cons	-14.93039	3.445179	-4.33	0.000	-21.68282	-8.177965
/studytime						
ln_p	.9413477	.2943728			.3643876	1.518308

```
Group          : 3                              Number of obs     =         14
Response       : studytime                      No. of failures   =          6
Family         : Weibull                        Time at risk      =        355
Form           : proportional hazards
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
studytime						
age	.1212332	.0367538	3.30	0.001	.049197	.1932694
_cons	-14.08495	3.242463	-4.34	0.000	-20.44006	-7.72984
/studytime						
ln_p	.6735495	.369625			-.0509022	1.398001

Notes:

1. In [SEM] **Intro 6**, we wrote that `gsem` classifies each parameter into one of seven classes, which are the following:

Class description	Class name
1. intercepts and cutpoints	`cons`
2. fixed coefficients	`coef`
3. latent variable coefficients	`loading`
4. covariances of errors	`errvar`
5. scaling parameters	`scale`
6. means of exogenous variables	`means`
7. covariances of exogenous latent variables	`covex`
8. all the above	`all`
9. none of the above	`none`

   By default, classes 1, 2, and 3 are constrained to be equal, and the others are allowed to vary.

2. In the output, we have a separate table of parameter estimates for each level of `drug`. The coefficient on age is 0.12 in all groups, but, as expected, the other parameters differ across groups.

We can replay the results with the `byparm` option to request that results be sorted by parameter rather than by groups. This output makes it easy to compare estimates across the groups. Alternatively, we could have added the `byparm` option when we fit the model.

```
. gsem, byparm
Generalized structural equation model          Number of obs     =         48
Grouping variable  = drug                      Number of groups  =          3

Group          : 1                              Number of obs     =         20
Response       : studytime                      No. of failures   =         19
Family         : Weibull                        Time at risk      =        180
Form           : proportional hazards
Link           : log

Group          : 2                              Number of obs     =         14
Response       : studytime                      No. of failures   =          6
Family         : Weibull                        Time at risk      =        209
Form           : proportional hazards
Link           : log
```

```
Group          : 3                              Number of obs     =         14
Response       : studytime                      No. of failures   =          6
Family         : Weibull                        Time at risk      =        355
Form           : proportional hazards
Link           : log

Log likelihood     = -109.28976

 ( 1)  [studytime]1bn.drug#c.age - [studytime]3.drug#c.age = 0
 ( 2)  [studytime]2.drug#c.age - [studytime]3.drug#c.age = 0
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
studytime						
age						
1	.1212332	.0367538	3.30	0.001	.049197	.1932694
2	.1212332	.0367538	3.30	0.001	.049197	.1932694
3	.1212332	.0367538	3.30	0.001	.049197	.1932694
_cons						
1	-10.36921	2.341022	-4.43	0.000	-14.95753	-5.780896
2	-14.93039	3.445179	-4.33	0.000	-21.68282	-8.177965
3	-14.08495	3.242463	-4.34	0.000	-20.44006	-7.72984
/studytime						
ln_p						
1	.4541282	.1715663			.1178645	.7903919
2	.9413477	.2943728			.3643876	1.518308
3	.6735495	.369625			-.0509022	1.398001

The estimated intercepts, labeled `_cons` in the first section of the table, do not vary much across groups. We can perform a Wald test of whether these coefficients are the same using the `test` command. If we replay the results by typing `gsem, coeflegend`, we find that we can refer to the three intercepts as `_b[studytime:1.drug]`, `_b[studytime:2.drug]`, and `_b[studytime:3.drug]`. Therefore, our test command is

```
. test _b[studytime:1.drug]=_b[studytime:2.drug]=_b[studytime:3.drug]
 ( 1)  [studytime]1bn.drug - [studytime]2.drug = 0
 ( 2)  [studytime]1bn.drug - [studytime]3.drug = 0

           chi2(  2) =    5.49
         Prob > chi2 =    0.0641
```

Using a 5% significance level, we fail to reject the null hypothesis that the intercepts are equal. If we wanted to refit the model based on these findings, we could include `cons` in the `ginvariant()` option as follows and constrain both the coefficients and the intercepts across groups.

```
. gsem (studytime <- age, family(weibull, failure(died))),  ///
  group(drug) ginvariant(coef cons)
```

Fitting the model with the Builder

Use the diagram in *Fitting the multiple-group model* above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use https://www.stata-press.com/data/r16/gsem_cancer
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Put the Builder in `gsem` mode by clicking on the GSEM button.

4. Create the Weibull regression component for `studytime`.

   Select the Add regression component tool, , and then click in the diagram about one-third of the way in from the left and halfway down.

   In the resulting dialog box,

   a. select `studytime` in the *Dependent variable* control;

   b. check *Make response generalized*;

   c. select `Weibull, Log` in the *Family/Link* control;

   d. select the *Select variables* radio button (it may already be selected);

   e. use the *Independent variables* control to select the variable `age`;

   f. select `Left` in the *Independent variables' direction* control;

   g. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

5. Specify the censoring variable.

   a. Choose the Select tool, .

   b. Click on the box for `studytime`.

   c. In the Contextual Toolbar, click on the **Properties...** button.

   d. In the resulting *Variable properties* dialog box, click on the **Failure and truncation...** button in the **Variable** tab.

   e. In the resulting *Failure and truncation* dialog box, select `died` in the *Failure variable* control. Click on **OK**.

   f. Click on **OK** in the *Variables properties* dialog box.

6. Clean up.

   The box for `_t` is created closer to the independent variables than it is in the example diagram. Use the Select tool, , and click on the box for `_t`. Drag it to the right to allow more space for results along the paths.

7. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar.

   In the resulting dialog box, do the following:

   a. Select the **Group** tab.

   b. Select the *Group analysis* radio button. Select the variable `drug` in the *Group variable* control.

   c. Select *Fixed coefficients* in the *Parameters that are equal across groups* control.

   d. Click on **OK**.

e. In the Standard Toolbar, use the *Group* control to toggle between results for group 1 and group 2.

You can open a completed diagram in the Builder by typing

```
. webgetsem gsem_grp
```

Also see

[SEM] **Example 47g** — Exponential survival model

[SEM] **Example 48g** — Loglogistic survival model with censored and truncated data

[SEM] **Intro 5** — Tour of models

[SEM] **Intro 6** — Comparing groups

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 50g — Latent class model

Description Remarks and examples References Also see

Description

To demonstrate latent class models, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_lca1
(Latent class analysis)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_lca1.dta
  obs:           216                          Latent class analysis
 vars:             4                          17 Jan 2019 12:52
                                               (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
accident        byte    %9.0g                 would testify against friend in
                                                accident case
play            byte    %9.0g                 would give negative review of
                                                friend's play
insurance       byte    %9.0g                 would disclose health concerns to
                                                friend's insurance company
stock           byte    %9.0g                 would keep company secret from
                                                friend
-------------------------------------------------------------------------------
Sorted by: accident  play  insurance  stock
. notes in 1/4
_dta:
  1.  Data from Samuel A. Stouffer and Jackson Toby, March 1951, "Role conflict
      and personality", _The American Journal of Sociology_, vol. 56 no. 5,
      395-406.
  2.  Variables represent responses of students from Harvard and Radcliffe who
      were asked how they would respond to four situations. Respondents
      selected either a particularistic response (based on obligations to a
      friend) or universalistic response (based on obligations to society).
  3.  Each variable is coded with 0 indicating a particularistic response and 1
      indicating a universalistic response.
  4.  For a full description of the questions, type "notes in 5/8".
```

See *Latent class models* in [SEM] **Intro 5** for background.

Remarks and examples

A latent class model is characterized by having a categorical (rather than continuous) latent variable. The levels of the categorical latent variable represent groups in the population and are called classes. We are interested in identifying and understanding these classes.

Goodman (2002) fits a variety of latent class models to the dataset described above with a focus on understanding how groups of individuals differ in response to situations that require making a decision between helping a friend (a particularistic choice) and doing what is right for society (a universalistic choice). Individuals were asked how they would respond in four such situations, and their responses were recorded in the variables `accident`, `play`, `insurance`, and `stock`. These variables are coded such that 1 is a universalistic response and 0 is a particularistic response.

To fit a latent class model, we must specify the number of classes in the latent variable. In the basic form of the latent class model demonstrated here, we have one categorical latent variable with two classes. The parameters in the model, namely, the intercepts in logistic regression models for the four observed variables, are allowed to vary across the classes.

More specifically, the model that we will fit estimates an intercept, α, for each observed variable for the first class,

$$\Pr(\texttt{accident} = 1 \,|\, C = 1) = \frac{\exp(\alpha_{11})}{1 + \exp(\alpha_{11})}$$

$$\Pr(\texttt{play} = 1 \,|\, C = 1) = \frac{\exp(\alpha_{21})}{1 + \exp(\alpha_{21})}$$

$$\Pr(\texttt{insurance} = 1 \,|\, C = 1) = \frac{\exp(\alpha_{31})}{1 + \exp(\alpha_{31})}$$

$$\Pr(\texttt{stock} = 1 \,|\, C = 1) = \frac{\exp(\alpha_{41})}{1 + \exp(\alpha_{41})}$$

and a corresponding intercept for the second class,

$$\Pr(\texttt{accident} = 1 \,|\, C = 2) = \frac{\exp(\alpha_{12})}{1 + \exp(\alpha_{12})}$$

$$\Pr(\texttt{play} = 1 \,|\, C = 2) = \frac{\exp(\alpha_{22})}{1 + \exp(\alpha_{22})}$$

$$\Pr(\texttt{insurance} = 1 \,|\, C = 2) = \frac{\exp(\alpha_{32})}{1 + \exp(\alpha_{32})}$$

$$\Pr(\texttt{stock} = 1 \,|\, C = 2) = \frac{\exp(\alpha_{42})}{1 + \exp(\alpha_{42})}$$

We also estimate the probability of being in each class using multinomial logistic regression,

$$\Pr(C = 1) = \frac{e^{\gamma_1}}{e^{\gamma_1} + e^{\gamma_2}}$$

$$\Pr(C = 2) = \frac{e^{\gamma_2}}{e^{\gamma_1} + e^{\gamma_2}}$$

where γ_1 and γ_2 are intercepts in the multinomial logit model. By default, the first class will be treated as the base, so $\gamma_1 = 0$.

To fit this model, we type

```
. gsem (accident play insurance stock <- ), logit lclass(C 2)
```

No variables are listed on the right side of the arrow because we are fitting intercept-only models for each observed variable. `logit` specifies that we are fitting logistic regression models for all four variables. The `lclass(C 2)` option specifies that the name of our categorical latent variable is `C` and that it has two latent classes.

The result of typing our estimation command is

```
. gsem (accident play insurance stock <- ), logit lclass(C 2)
Fitting class model:
Iteration 0:   (class) log likelihood = -149.71979
Iteration 1:   (class) log likelihood = -149.71979
Fitting outcome model:
Iteration 0:   (outcome) log likelihood = -403.97142
Iteration 1:   (outcome) log likelihood = -398.15909
Iteration 2:   (outcome) log likelihood = -397.81953
Iteration 3:   (outcome) log likelihood =  -397.8164
Iteration 4:   (outcome) log likelihood =  -397.8164
Refining starting values:
Iteration 0:   (EM) log likelihood = -570.24204
Iteration 1:   (EM) log likelihood = -576.20485
Iteration 2:   (EM) log likelihood = -577.41464
Iteration 3:   (EM) log likelihood = -576.88554
Iteration 4:   (EM) log likelihood = -575.59242
Iteration 5:   (EM) log likelihood = -573.90567
Iteration 6:   (EM) log likelihood = -571.99868
Iteration 7:   (EM) log likelihood = -569.97482
Iteration 8:   (EM) log likelihood = -567.90955
Iteration 9:   (EM) log likelihood = -565.86392
Iteration 10:  (EM) log likelihood = -563.88815
Iteration 11:  (EM) log likelihood = -562.02165
Iteration 12:  (EM) log likelihood = -560.29231
Iteration 13:  (EM) log likelihood = -558.71641
Iteration 14:  (EM) log likelihood = -557.29974
Iteration 15:  (EM) log likelihood = -556.03949
Iteration 16:  (EM) log likelihood = -554.92679
Iteration 17:  (EM) log likelihood = -553.94914
Iteration 18:  (EM) log likelihood = -553.09241
Iteration 19:  (EM) log likelihood = -552.34233
Iteration 20:  (EM) log likelihood = -551.68539
Note: EM algorithm reached maximum iterations.
Fitting full model:
Iteration 0:   log likelihood = -504.62913
Iteration 1:   log likelihood = -504.47255
Iteration 2:   log likelihood = -504.46773
Iteration 3:   log likelihood = -504.46767
Iteration 4:   log likelihood = -504.46767
Generalized structural equation model           Number of obs     =        216
Log likelihood = -504.46767
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C	(base outcome)					
2.C						
_cons	-.9482041	.2886333	-3.29	0.001	-1.513915	-.3824933

```
Class          : 1
Response       : accident
Family         : Bernoulli
Link           : logit
Response       : play
Family         : Bernoulli
Link           : logit
Response       : insurance
Family         : Bernoulli
Link           : logit
Response       : stock
Family         : Bernoulli
Link           : logit
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
accident						
_cons	.9128742	.1974695	4.62	0.000	.5258411	1.299907
play						
_cons	-.7099072	.2249096	-3.16	0.002	-1.150722	-.2690926
insurance						
_cons	-.6014307	.2123096	-2.83	0.005	-1.01755	-.1853115
stock						
_cons	-1.880142	.3337665	-5.63	0.000	-2.534312	-1.225972

```
Class          : 2
Response       : accident
Family         : Bernoulli
Link           : logit
Response       : play
Family         : Bernoulli
Link           : logit
Response       : insurance
Family         : Bernoulli
Link           : logit
Response       : stock
Family         : Bernoulli
Link           : logit
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
accident						
_cons	4.983017	3.745987	1.33	0.183	-2.358982	12.32502
play						
_cons	2.747366	1.165853	2.36	0.018	.4623372	5.032395
insurance						
_cons	2.534582	.9644841	2.63	0.009	.6442279	4.424936
stock						
_cons	1.203416	.5361735	2.24	0.025	.1525356	2.254297

Notes:

1. The output shows four iteration logs. The first three are for models that are fit to obtain good starting values. Starting values are challenging for latent class models, and `gsem` provides a variety of options for specifying and computing starting values. See [SEM] **gsem estimation options** and [SEM] **Intro 12** for more information on these options.
2. The first table in the output provides the estimated coefficients in the multinomial logit model for `C`.
3. The next two tables are the results for the logistic regression models for the first and second classes.

To better understand this model, let's examine how the probabilities of giving a universalistic response differ across classes. The `estat lcmean` command reports class-specific marginal means for each variable. Because we are using logistic regression, these means are actually the predicted probabilities.

```
. estat lcmean
Latent class marginal means                     Number of obs     =        216
```

	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
1				
accident	.7135879	.0403588	.6285126	.7858194
play	.3296193	.0496984	.2403573	.4331299
insurance	.3540164	.0485528	.2655049	.4538042
stock	.1323726	.0383331	.0734875	.2268872
2				
accident	.9931933	.0253243	.0863544	.9999956
play	.9397644	.0659957	.6135685	.9935191
insurance	.9265309	.0656538	.6557086	.9881667
stock	.769132	.0952072	.5380601	.9050206

The first section of this table reports the probabilities for class 1. In this class, the probability of giving a universalistic response to the first question—the question that concerns testifying against a friend who was involved in an accident—is 0.714. The probability of giving a universalistic response to the last question—the question about warning a friend about a stock price that is about to fall—is 0.132.

The second section of the table reports the corresponding probabilities for class 2. We find that the probability of giving a universalistic response to each question is higher in class 2 than in class 1. Class 2 appears to represent a more universalistically inclined group.

We can estimate probabilities of being in each class using `estat lcprob`.

```
. estat lcprob
Latent class marginal probabilities             Number of obs     =        216
```

	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
C				
1	.7207539	.0580926	.5944743	.8196407
2	.2792461	.0580926	.1803593	.4055257

This indicates that 72% of individuals are expected to be in the less universalistic class and 28% are expected to be in the more universalistic class.

We can use the predictions of the posterior probability of class membership to evaluate an individual's probability of being in each class.

```
. predict classpost*, classposteriorpr
. list in 1, abbrev(10)
```

	accident	play	insurance	stock	classpost1	classpost2
1.	0	0	0	0	.999975	.000025

For the first individual in our dataset, who responded with a particularistic answer to all four questions, the probability of being in class 1, the less universalistic class, is almost 1.

We can determine the expected class for each individual based on whether the posterior probability is greater than 0.5.

```
. generate expclass = 1 + (classpost2>0.5)
. tabulate expclass
```

expclass	Freq.	Percent	Cum.
1	145	67.13	67.13
2	71	32.87	100.00
Total	216	100.00	

In our dataset, 145 individuals are expected to be in class 1 and 71 individuals are expected to be in class 2.

References

Goodman, L. A. 2002. Latent class analysis: The empirical study of latent types, latent variables, and latent structures. In *Applied Latent Class Analysis*, ed. J. A. Hagenaars and A. L. McCutcheon, 3–55. Cambridge: Cambridge University Press.

Stouffer, S. A., and J. Toby. 1951. Role conflict and personality. *American Journal of Sociology* 56: 395–406.

Also see

[SEM] **Example 51g** — Latent class goodness-of-fit statistics

[SEM] **Example 52g** — Latent profile model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat lcmean** — Latent class marginal means

[SEM] **estat lcprob** — Latent class marginal probabilities

Title

Example 51g — Latent class goodness-of-fit statistics

Description Remarks and examples Reference Also see

Description

Here we demonstrate how to obtain goodness-of-fit statistics for latent class models.

We continue with [SEM] **Example 50g**, where we fit a two-class model:

```
. use https://www.stata-press.com/data/r16/gsem_lca1
. gsem (accident play insurance stock <- ), logit lclass(C 2)
```

See *Latent class models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

Likelihood-ratio (G^2) test
Comparing models

Likelihood-ratio (G^2) test

For standard latent class models with observed variables that are all categorical, one way to evaluate model fit is to compare the model we have just fit with a saturated model. We can use the `estat lcgof` command to perform a likelihood-ratio test of whether our model fits as well as the saturated model. The corresponding likelihood-ratio statistic is sometimes referred to as G^2 in latent class analysis literature.

```
. estat lcgof
```

Fit statistic	Value	Description
Likelihood ratio		
chi2_ms(6)	2.720	model vs. saturated
p > chi2	0.843	
Information criteria		
AIC	1026.935	Akaike's information criterion
BIC	1057.313	Bayesian information criterion

We fail to reject the null hypothesis that our model fits as well as the saturated model.

`estat lcgof` also reports Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (BIC). These are useful for comparing models but not useful for determining goodness-of-fit for a single model.

Comparing models

In latent class analysis, we often compare models that have different numbers of classes. Following Goodman (2002), we compare models that allow for one, two, and three latent classes. We have already fit the two-class model using the `gsem` command above. Before we move on, we will store the results of this model.

```
. estimates store twoclass
```

Next, we fit the one-class model, store the results, and perform the likelihood-ratio test comparing it with the saturated model.

```
. quietly gsem (accident play insurance stock <- ), logit lclass(C 1)
. estimates store oneclass
. estat lcgof
```

Fit statistic	Value	Description
Likelihood ratio		
chi2_ms(11)	81.084	model vs. saturated
p > chi2	0.000	
Information criteria		
AIC	1095.300	Akaike's information criterion
BIC	1108.801	Bayesian information criterion

We reject the null hypothesis in this case. The one-class model does not fit well.

We also fit the three-class model.

```
. quietly gsem (accident play insurance stock <- ), logit lclass(C 3)
. estimates store threeclass
. estat lcgof
```

Fit statistic	Value	Description
Likelihood ratio		
chi2_ms(1)	0.387	model vs. saturated
p > chi2	0.534	
Information criteria		
AIC	1034.602	Akaike's information criterion
BIC	1081.856	Bayesian information criterion

Based on this test, the three-class model, like the two-class model, does not fit worse than the saturated model.

We will compare our three models using AIC and BIC. Smaller values of AIC and BIC are better. We could look back at the AIC and BIC values reported by our three `estat lcgof` commands, but we will instead create a table that reports these information criteria for all three models using `estimates stats`.

```
. estimates stats oneclass twoclass threeclass
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
oneclass	216	.	-543.6498	4	1095.3	1108.801
twoclass	216	.	-504.4677	9	1026.935	1057.313
threeclass	216	.	-503.3011	14	1034.602	1081.856

Note: BIC uses N = number of observations. See **[R] BIC note**.

The two-class model has both the smallest AIC and the smallest BIC.

Reference

Goodman, L. A. 2002. Latent class analysis: The empirical study of latent types, latent variables, and latent structures. In *Applied Latent Class Analysis*, ed. J. A. Hagenaars and A. L. McCutcheon, 3–55. Cambridge: Cambridge University Press.

Also see

[SEM] **Example 50g** — Latent class model

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat lcgof** — Latent class goodness-of-fit statistics

Title

Example 52g — Latent profile model

Description Remarks and examples References Also see

Description

To demonstrate latent profile models, we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_lca2
(Latent profile analysis)
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_lca2.dta
  obs:           145                          Latent profile analysis
 vars:             7                          18 Jan 2019 12:39
                                               (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
patient         int     %9.0g                 Patient ID
relwgt          float   %9.0g                 Relative weight
fglucose        int     %9.0g                 Fasting plasma glucose
glucose         float   %9.0g                 Glucose area (mg/10mL/hr)
insulin         float   %9.0g                 Insulin area (mIU/10mL/hr)
sspg            float   %9.0g                 Steady-state plasma glucose
cclass          byte    %17.0g     class      Clinical classification
-------------------------------------------------------------------------------
Sorted by:
. notes
_dta:
  1.  Data originally analyzed in G. M. Reaven and R. G. Miller, 1979, "An
      attempt to define the nature of chemical diabetes using a
      multidimensional analysis", _Diabetologia_., vol. 16, 17-24.
  2.  Data made publicly available in D. F. Andrews and A. M. Herzberg, _Data:
      A Collection of Problems from Many Fields for the Student and Research
      Worker_, New York: Springer.
  3.  Data includes variables related to diabetes for 145 non-obese adults.
```

See *Latent class models* in [SEM] **Intro 5** for background.

Remarks and examples

Remarks are presented under the following headings:

Fitting the two-class model
Comparing models
Fitting the three-class model with covariances

Fitting the two-class model

In this manual, when we talk about latent class analysis, we are referring to an analysis that involves fitting models with categorical latent variables. Sometimes, these models are given more specific names. In [SEM] **Example 50g**, we fit a latent class model with a categorical latent variable and categorical observed variables. This is a typical latent class model. However, models with categorical latent variables are not limited to having categorical observed variables. A latent class model that instead has continuous observed variables is often referred to as a latent profile model.

Masyn (2013) uses the data described above to fit a series of latent profile models, each having one categorical latent variable and three observed variables, `glucose`, `insulin`, and `sspg`. The goal is to determine categories of diabetes based on these three variables. We begin by fitting a model in which the latent variable, `C`, has two classes. We fit a linear regression model for each observed variable where the intercept, α_{jc}, is allowed to vary across the classes of the latent variable. Because we are using linear regression, we also estimate the variances of the error terms `e.glucose`, `e.insulin`, and `e.sspg`.

More specifically, for class 1 we fit

$$\texttt{glucose} = \alpha_{11} + \texttt{e.glucose}$$
$$\texttt{insulin} = \alpha_{21} + \texttt{e.insulin}$$
$$\texttt{sspg} = \alpha_{31} + \texttt{e.sspg}$$

and for class 2 we fit

$$\texttt{glucose} = \alpha_{12} + \texttt{e.glucose}$$
$$\texttt{insulin} = \alpha_{22} + \texttt{e.insulin}$$
$$\texttt{sspg} = \alpha_{32} + \texttt{e.sspg}$$

We also estimate the probability of being in each class using multinomial logistic regression,

$$\Pr(C = 1) = \frac{e^{\gamma_1}}{e^{\gamma_1} + e^{\gamma_2}}$$
$$\Pr(C = 2) = \frac{e^{\gamma_2}}{e^{\gamma_1} + e^{\gamma_2}}$$

where γ_1 and γ_2 are intercepts in the multinomial logit model. By default, the first class will be treated as the base, so $\gamma_1 = 0$.

We will assume that the errors are uncorrelated, which is the default, and that the variances do not differ across classes, also the default.

```
. gsem (glucose insulin sspg <- _cons), lclass(C 2)
```

(*iteration log omitted*)

```
Generalized structural equation model           Number of obs     =        145
Log likelihood = -1702.5542
 ( 1)  [/]var(e.glucose)#1bn.C - [/]var(e.glucose)#2.C = 0
 ( 2)  [/]var(e.insulin)#1bn.C - [/]var(e.insulin)#2.C = 0
 ( 3)  [/]var(e.sspg)#1bn.C - [/]var(e.sspg)#2.C = 0
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C	(base outcome)					
2.C						
_cons	-1.541025	.2205682	-6.99	0.000	-1.973331	-1.10872

```
Class          : 1
Response       : glucose
Family         : Gaussian
Link           : identity
Response       : insulin
Family         : Gaussian
Link           : identity
Response       : sspg
Family         : Gaussian
Link           : identity
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
glucose						
_cons	41.22237	1.298051	31.76	0.000	38.67824	43.7665
insulin						
_cons	20.98005	1.000974	20.96	0.000	19.01817	22.94192
sspg						
_cons	14.96579	.6868081	21.79	0.000	13.61967	16.31191
var(e.gluc~e)	191.5596	23.83815			150.0992	244.4723
var(e.insu~n)	119.0542	14.00336			94.54204	149.9217
var(e.sspg)	55.91283	6.713667			44.18801	70.7487

```
Class          : 2

Response       : glucose
Family         : Gaussian
Link           : identity

Response       : insulin
Family         : Gaussian
Link           : identity

Response       : sspg
Family         : Gaussian
Link           : identity
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
glucose						
_cons	115.7123	2.849914	40.60	0.000	110.1266	121.2981
insulin						
_cons	7.553144	2.160949	3.50	0.000	3.317761	11.78853
sspg						
_cons	34.5529	1.53117	22.57	0.000	31.55187	37.55394
var(e.gluc~e)	191.5596	23.83815			150.0992	244.4723
var(e.insu~n)	119.0542	14.00336			94.54204	149.9217
var(e.sspg)	55.91283	6.713667			44.18801	70.7487

```
. estimates store c2inv
```

Notes:

1. The first table in the output provides the estimated coefficients in the multinomial logit model for `C`.
2. The next two tables are the results for the linear regression models for the first and second classes.

Comparing models

Before we interpret any results, we will fit and compare other models. We modify our command above to specify that `C` has three, four, and then five latent classes, and we store the results of those models by typing

```
. gsem (glucose insulin sspg <- _cons), lclass(C 3)
. estimates store c3inv
. gsem (glucose insulin sspg <- _cons), lclass(C 4) ///
  startvalues(randomid, draws(5) seed(15)) emopts(iter(20))
. estimates store c4inv
. gsem (glucose insulin sspg <- _cons), lclass(C 5) ///
  startvalues(randomid, draws(5) seed(15)) emopts(iter(20))
. estimates store c5inv
```

For the models with four and five latent classes, we added the `startvalues(randomid, draws(5) seed(15))` option to request that starting values be computed using random class assignments. In this option, `draws(5)` specifies that five random draws be taken and that the one with the best log likelihood after the EM iterations be selected. The `emopts(iter(20))` option says that 20 EM iterations are used for each random draw. We also set the seed for reproducible results. We could have used the same options in the models with two classes and three classes. Difficulty finding good starting values is fairly common when fitting latent class models, so `gsem` provides a variety

of options for obtaining starting values. See [SEM] **Intro 12** and [SEM] **gsem estimation options** for more information on starting values.

We can compare the four models fit above using Akaike's information criterion (AIC) and Schwarz's Bayesian information criterion (BIC).

```
. estimates stats c2inv c3inv c4inv c5inv
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
c2inv	145	.	-1702.554	10	3425.108	3454.876
c3inv	145	.	-1653.238	14	3334.476	3376.15
c4inv	145	.	-1626.828	18	3289.656	3343.237
c5inv	145	.	-1578.207	22	3200.414	3265.902

```
Note: BIC uses N = number of observations. See [R] BIC note.
```

The model with five latent classes has the smallest values of both AIC and BIC and would be considered the best based on these information criteria.

Fitting the three-class model with covariances

Masyn's final model was a three-class model that allowed for covariances among the error terms and that estimated all parameters separately across classes. To estimate the covariances, we add the `covstructure(e._OEn, unstructured)` option. See [SEM] **sem and gsem option covstructure()** for details on this option. To allow all parameters to vary across classes, we add the `lcinvariant(none)` option. Here `none` specifies that no parameters are constrained to be equal across classes.

```
. gsem (glucose insulin sspg <- _cons), lclass(C 3) lcinvariant(none)
> covstructure(e._OEn, unstructured)
```

(*iteration log omitted*)

```
Generalized structural equation model           Number of obs     =        145
Log likelihood = -1536.6409
```

		Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C		(base outcome)					
2.C	_cons	-.8853513	.2386536	-3.71	0.000	-1.353104	-.4175988
3.C	_cons	-.612664	.2260018	-2.71	0.007	-1.055619	-.1697085

```
Class          : 1

Response       : glucose
Family         : Gaussian
Link           : identity

Response       : insulin
Family         : Gaussian
Link           : identity

Response       : sspg
Family         : Gaussian
Link           : identity
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
glucose						
_cons	35.68584	.5741752	62.15	0.000	34.56048	36.81121
insulin						
_cons	16.58066	.6204724	26.72	0.000	15.36456	17.79677
sspg						
_cons	10.49755	.5833606	17.99	0.000	9.354183	11.64091
var(e.gluc~e)	19.30952	3.932547			12.9544	28.78233
var(e.insu~n)	26.7354	4.494093			19.23108	37.16804
var(e.sspg)	18.71079	3.970509			12.34422	28.36094
cov(e.gluc~e, e.insulin)	3.456027	2.942391	1.17	0.240	-2.310954	9.223008
cov(e.gluc~e, e.sspg)	5.474303	2.811729	1.95	0.052	-.0365846	10.98519
cov(e.insu~n, e.sspg)	7.995803	3.020304	2.65	0.008	2.076115	13.91549

```
Class          : 2

Response       : glucose
Family         : Gaussian
Link           : identity

Response       : insulin
Family         : Gaussian
Link           : identity

Response       : sspg
Family         : Gaussian
Link           : identity
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
glucose						
_cons	47.66176	1.492718	31.93	0.000	44.73609	50.58744
insulin						
_cons	34.35203	3.00337	11.44	0.000	28.46554	40.23853
sspg						
_cons	24.414	.7395383	33.01	0.000	22.96453	25.86347
var(e.gluc~e)	53.21326	15.56547			29.99396	94.40735
var(e.insu~n)	228.6332	59.03553			137.832	379.2526
var(e.sspg)	13.75515	3.838523			7.960284	23.76853
cov(e.gluc~e, e.insulin)	40.02875	23.12762	1.73	0.083	-5.300552	85.35805
cov(e.gluc~e, e.sspg)	.7294854	5.48065	0.13	0.894	-10.01239	11.47136
cov(e.insu~n, e.sspg)	-5.743169	11.4943	-0.50	0.617	-28.27158	16.78524

```
Class          : 3

Response       : glucose
Family         : Gaussian
Link           : identity

Response       : insulin
Family         : Gaussian
Link           : identity

Response       : sspg
Family         : Gaussian
Link           : identity
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
glucose						
_cons	93.92473	6.985336	13.45	0.000	80.23372	107.6157
insulin						
_cons	10.37614	1.123135	9.24	0.000	8.174836	12.57744
sspg						
_cons	28.4787	1.94975	14.61	0.000	24.65726	32.30013
var(e.gluc~e)	1279.011	312.6774			792.1048	2065.218
var(e.insu~n)	36.38521	9.26287			22.09163	59.92692
var(e.sspg)	113.3239	27.67628			70.21642	182.8961
cov(e.gluc~e, e.insulin)	-163.4383	47.637	-3.43	0.001	-256.8051	-70.07153
cov(e.gluc~e, e.sspg)	276.9206	81.60543	3.39	0.001	116.9769	436.8643
cov(e.insu~n, e.sspg)	-25.4313	11.66564	-2.18	0.029	-48.29554	-2.567057

Because we do not have any predictors in our regression models, the intercepts can be interpreted as the predicted class-specific means of the corresponding variables. In class 1, `glucose` has an estimated mean of 35.69, `insulin` has an estimated mean of 16.58, and `sspg` has an estimated mean of 10.50. Also because we have no predictors, the estimated variances and covariances of the error terms are simply class-specific estimates of the variances and covariances of the variables. In class 1, the estimated variance of `glucose` is 19.31, the estimated covariance of `glucose` and `insulin` is 3.46. The remaining coefficients can be interpreted in a similar manner.

We can determine expected classification for each individual in the dataset based on the predicted posterior class probabilities.

```
. predict cpost*, classposteriorpr

. egen max = rowmax(cpost*)

. generate predclass = 1 if cpost1==max
(69 missing values generated)

. replace predclass = 2 if cpost2==max
(32 real changes made)

. replace predclass = 3 if cpost3==max
(37 real changes made)
```

```
. tabulate cclass predclass, col
```

Key
frequency *column percentage*

Clinical classification	predclass 1	2	3	Total
overt diabetic	0 0.00	2 6.25	31 83.78	33 22.76
chemical diabetic	7 9.21	23 71.88	6 16.22	36 24.83
normal	69 90.79	7 21.88	0 0.00	76 52.41
Total	76 100.00	32 100.00	37 100.00	145 100.00

When we compare the predicted classes (`predclass`) with the assigned clinical classifications (`cclass`) given to these individuals, we see that 91% of the individuals predicted to be in class 1 were given a clinical classification of normal. Of those predicted to be in class 2, 72% were assigned a clinical classification of chemical diabetic. Finally, 84% of those predicted to be in class 3 had a clinical classification of overt diabetic.

Masyn went on to examine the individuals who were classified differently when using the clinical definition and when using the results from the model. She found that the predictions from the latent profile model could be explained medically and may be an improvement over the clinical definitions.

References

Andrews, D. F., and A. M. Herzberg, ed. 1985. *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. New York: Springer.

Masyn, K. E. 2013. Latent class analysis and finite mixture modeling. In *The Oxford Handbook of Quantitative Methods*, ed. T. D. Little, vol. 2, 551–610. New York: Oxford University Press.

Reaven, G. M., and R. G. Miller. 1979. An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia* 16: 17–24.

Also see

[SEM] **Example 50g** — Latent class model

[SEM] **Example 51g** — Latent class goodness-of-fit statistics

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Example 53g — Finite mixture Poisson regression

Description Remarks and examples References Also see

Description

To demonstrate a finite mixture model (FMM), we use the following data:

```
. use https://www.stata-press.com/data/r16/gsem_mixture
(U.S. Medical Expenditure Panel Survey (2003))
. describe
Contains data from https://www.stata-press.com/data/r16/gsem_mixture.dta
  obs:         3,677                          U.S. Medical Expenditure Panel
                                                Survey (2003)
 vars:            12                          26 Jan 2019 08:46
                                              (_dta has notes)
```

variable name	storage type	display format	value label	variable label
drvisits	int	%9.0g		number of doctor visits
private	byte	%8.0g		has private supplementary insurance
medicaid	byte	%8.0g		has Medicaid public insurance
age	byte	%8.0g		age in years
educ	byte	%8.0g		years of education
actlim	byte	%8.0g		has activity limitations
chronic	byte	%8.0g		number of chronic conditions
income	float	%9.0g		income in $1,000s
offer	byte	%8.0g		employer offers insurance
hpvisits	int	%8.0g		number of visits to health professionals other than doctors
female	byte	%8.0g		female
phylim	byte	%8.0g		has physical limitation

```
Sorted by:
. notes
_dta:
  1.  Data on annual number of doctor visits for individuals age 65 and older
      from the U.S. Medical Expenditure Panel Survey for 2003.
  2.  Data is analyzed in Cameron, A. C. and P. K. Trivedi, 2010,
      _Microeconometrics Using Stata, Rev. Ed., College Station, TX: Stata
      Press.
  3.  Additional information on finite mixture models for count data and a
      similar example are found in Deb, P. and P. K. Trivedi, 1997, Demand for
      medical care by the elderly: A finite mixture approach, _Journal of
      Applied Econometrics_, vol. 12, 313--336.
```

See *Finite mixture models* in [SEM] **Intro 5** for background.

Remarks and examples

We are interested in fitting a Poisson regression to model the annual number of doctor visits as a function of whether an individual has private supplementary insurance, whether he or she has Medicaid, age, age squared, education level, whether he or she has activity limitations, and the number of chronic conditions. If we believed that the same model applied to the entire population, we could fit the model by typing

```
. poisson drvisits private medicaid c.age##c.age educ actlim chronic
```

or, equivalently, by using `gsem`,

```
. gsem (drvisits <- private medicaid c.age##c.age educ actlim chronic), poisson
```

However, we believe that the model may differ across groups in the population. We do not have any information that identifies what these groups are or that tells us which individuals in our sample belong to each group. We can consider a categorical latent variable that identifies these groups and refer to the levels of this latent variable as latent classes. With an FMM, we can incorporate the categorical latent variable into our model to account for differences across the latent classes.

Following Cameron and Trivedi (2010), we will fit an FMM with a Poisson regression component for each latent class. We will estimate distinct coefficients for the Poisson model in each class, and we will estimate the probability of belonging to each of these classes using a multinomial logistic regression. We fit the model as follows:

```
. gsem (drvisits <- private medicaid c.age##c.age educ actlim chronic),
> poisson lclass(C 2) startvalues(randomid, draws(5) seed(15))
Computing starting values using randomid:
  (iteration log omitted)
Generalized structural equation model           Number of obs     =      3,677
Log likelihood = -11502.686
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C	(base outcome)					
2.C						
_cons	.877227	.0494614	17.74	0.000	.7802845	.9741696

```
Class          : 1
Response       : drvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drvisits						
private	.138229	.0247626	5.58	0.000	.0896951	.1867629
medicaid	.1269723	.0341525	3.72	0.000	.0600345	.19391
age	.2628874	.0466774	5.63	0.000	.1714014	.3543735
c.age#c.age	-.0017418	.0003108	-5.60	0.000	-.002351	-.0011326
educ	.0241679	.0030705	7.87	0.000	.0181499	.030186
actlim	.1831598	.0238817	7.67	0.000	.1363525	.2299671
chronic	.1970511	.0088783	22.19	0.000	.17965	.2144523
_cons	-8.051256	1.741677	-4.62	0.000	-11.46488	-4.637632

```
Class          : 2
Response       : drvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drvisits						
private	.2077415	.0306353	6.78	0.000	.1476974	.2677856
medicaid	.1071618	.0407211	2.63	0.008	.02735	.1869736
age	.3798087	.0562035	6.76	0.000	.269652	.4899655
c.age#c.age	-.0024869	.0003736	-6.66	0.000	-.0032191	-.0017547
educ	.029099	.003972	7.33	0.000	.021314	.0368841
actlim	.1244235	.0310547	4.01	0.000	.0635574	.1852895
chronic	.3191166	.0089757	35.55	0.000	.3015247	.3367086
_cons	-14.25713	2.101964	-6.78	0.000	-18.37691	-10.13736

Notes:

1. We used the `lclass(C 2)` to specify that our categorical latent variable is named C and has two latent classes.
2. The first table in the output provides the estimated coefficients in the multinomial logit model for `C`.
3. The next two tables are the results for the Poisson regression models for the first and second classes. By default, the coefficients and intercepts vary across the classes. We can specify `lcinvariant(cons)` if we want intercepts to be constrained to be equal across classes, or we can specify `lcinvariant(coef)` if we want all coefficients constrained to be equal across classes. See [SEM] **gsem lclass options** for details on the `lcinvariant()` option.
4. We added the `startvalues(randomid), draws(5) seed(15))` option to request that starting values be computed using random class assignments. In this option, `draws(5)` specifies that five random draws be taken and that the one with the best log likelihood after the EM iterations be selected. If you fit FMMs and other models with categorical latent variables, taking multiple draws of random starting values can help to prevent convergence at a local maximum rather than the global maximum. `gsem` provides a variety of options for obtaining starting values. See [SEM] **Intro 12** and [SEM] **gsem estimation options** for more information on starting values.
5. The `fmm:` prefix can be used to fit finite mixture regression models with a single response variable. We could have fit this same model with `fmm: poisson` by typing

```
. fmm 2, startvalues(randomid, draws(5) seed(15)): ///
  poisson drvisits private medicaid c.age##c.age educ actlim chronic
```

We can use `estat lcprob` to estimate the proportion of individuals in each class.

```
. estat lcprob
Latent class marginal probabilities             Number of obs     =      3,677
```

	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
C				
1	.2937527	.0102614	.2740502	.3142586
2	.7062473	.0102614	.6857414	.7259498

We find that about 29% of the population is in class 1 and about 71% is in class 2.

To better understand these classes, we use `estat lcmean` to estimate the marginal predicted counts (means) for each class.

```
. estat lcmean
Latent class marginal means                     Number of obs     =      3,677
```

		Margin	Delta-method Std. Err.	z	P>|z|	[95% Conf.	Interval]
1							
	drvisits	13.95943	.1767506	78.98	0.000	13.613	14.30585
2							
	drvisits	3.801692	.0587685	64.69	0.000	3.686508	3.916876

Class 1 appears to represent those who visit the doctor frequently and class 2, those who visit less frequently.

We can also predict the posterior probabilities of class membership and then use those to determine the predicted class for each individual.

```
. predict postpr_dr*, classposteriorpr
. generate pclass_dr = 1 + (postpr_dr2>0.5)
. tabulate pclass_dr
```

pclass_dr	Freq.	Percent	Cum.
1	1,061	28.86	28.86
2	2,616	71.14	100.00
Total	3,677	100.00	

We see that 1,061 individuals in our sample are predicted to be in class 1, the class that frequently visits the doctor.

Our dataset also includes the variable `hpvisits`, which records the number of visits individuals make to health professionals other than doctors. We fit a similar model to the one above but with `hpvisits` as our response variable.

```
. gsem (hpvisits <- private medicaid c.age##c.age educ actlim chronic),
> poisson lclass(C 2) startvalues(classid pclass_dr)
  (iteration log omitted)
Generalized structural equation model           Number of obs     =      3,677
Log likelihood = -8510.4898
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C	(base outcome)					
2.C						
_cons	2.241837	.059523	37.66	0.000	2.125174	2.3585

```
Class          : 1
Response       : hpvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
hpvisits						
private	.3218525	.0347116	9.27	0.000	.253819	.389886
medicaid	.0715449	.0566317	1.26	0.206	-.0394511	.182541
age	.0975749	.0743567	1.31	0.189	-.0481615	.2433113
c.age#c.age	-.0004749	.0004971	-0.96	0.339	-.0014492	.0004993
educ	.0278151	.0046572	5.97	0.000	.0186872	.0369429
actlim	.7088077	.0353277	20.06	0.000	.6395666	.7780488
chronic	-.0077779	.0127981	-0.61	0.543	-.0328617	.0173059
_cons	-2.430713	2.766794	-0.88	0.380	-7.853529	2.992103

```
Class          : 2
Response       : hpvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
hpvisits						
private	.4448319	.0451971	9.84	0.000	.3562473	.5334165
medicaid	-.4490187	.074252	-6.05	0.000	-.5945499	-.3034875
age	.4160345	.0797576	5.22	0.000	.2597125	.5723565
c.age#c.age	-.0026784	.0005287	-5.07	0.000	-.0037147	-.0016421
educ	.1250644	.0062921	19.88	0.000	.1127322	.1373967
actlim	.3357366	.0442285	7.59	0.000	.2490503	.4224229
chronic	.206585	.0152161	13.58	0.000	.176762	.2364081
_cons	-18.21906	2.991859	-6.09	0.000	-24.083	-12.35513

This time, we used the `startvalues(classid pclass_dr)` option to specify how starting values are calculated. This means that we are using the variable `pclass_dr` as an initial guess of class membership to be used when computing starting values.

We again use `estat lcprob` to estimate the predicted proportion of the population in each class.

```
. estat lcprob
Latent class marginal probabilities             Number of obs     =      3,677
```

	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
C				
1	.0960559	.0051683	.0863925	.106674
2	.9039441	.0051683	.893326	.9136075

This time about 10% is in class 1, and 90% is in class 2.

We can predict the class for each individual based on this model and compare the classifications from the two models.

```
. predict postpr_hp*, classposteriorpr
. generate pclass_hp = 1 + (postpr_hp2>0.5)
. tabulate pclass_hp pclass_dr
```

pclass_hp	pclass_dr 1	2	Total
1	169	180	349
2	892	2,436	3,328
Total	1,061	2,616	3,677

Many individuals are predicted to be in class 2 based on both models, meaning that they are in the group that visits the doctor infrequently and in the group that visits other health professionals infrequently. However, there are also 892 that are classified differently by the two models. These individuals are in the class that visits the doctor frequently based on the first model but in the class that visits other healthcare professionals infrequently based on the second model.

In [SEM] **Example 54g**, we consider simultaneously modeling `drvisits` and `hpvisits` and using a single categorical latent variable that identifies groups in the population.

References

Cameron, A. C., and P. K. Trivedi. 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.

Deb, P., and P. K. Trivedi. 1997. Demand for medical care by the elderly: A finite mixture approach. *Journal of Applied Econometrics* 12: 313–336.

Also see

[SEM] **Example 54g** — Finite mixture Poisson regression, multiple responses

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat lcmean** — Latent class marginal means

[SEM] **estat lcprob** — Latent class marginal probabilities

[FMM] **fmm intro** — Introduction to finite mixture models

[FMM] **fmm: poisson** — Finite mixtures of Poisson regression models

Title

Example 54g — Finite mixture Poisson regression, multiple responses

Description Remarks and examples References Also see

Description

In this example, we demonstrate how to fit a finite mixture model with more than one response variable.

We continue with [SEM] **Example 53g**, where we fit two separate finite mixture Poisson regression models, one for `drvisits` and one for `hpvisits`. To refit these models, we could type

```
. use https://www.stata-press.com/data/r16/gsem_mixture
. gsem (drvisits <- private medicaid c.age##c.age educ actlim chronic), ///
  poisson lclass(C 2) startvalues(randomid, draws(5) seed(15))
. predict postpr_dr*, classposteriorpr
. generate pclass_dr = 1 + (postpr_dr2>0.5)
. gsem (hpvisits <- private medicaid c.age##c.age educ actlim chronic), ///
  poisson lclass(C 2) startvalues(classid pclass_dr)
```

As we pointed out in [SEM] **Example 53g**, we could have fit these finite mixture models using `fmm: poisson`. `gsem` extends the types of models that can be fit with the `fmm:` prefix, and we demonstrate one possible extension here.

See *Finite mixture models* in [SEM] **Intro 5** for background.

Remarks and examples

We fit Poisson regression models for `drvisits` and `hpvisits` simultaneously and include one categorical latent variable `C` that has three classes.

```
. gsem (drvisits hpvisits <- private medicaid c.age##c.age educ actlim chronic),
> poisson lclass(C 3) startvalues(randomid, draws(5) seed(15))
```

(*iteration log omitted*)

```
Generalized structural equation model           Number of obs     =      3,677
Log likelihood = -20557.828
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.C	(base outcome)					
2.C						
_cons	-.7734177	.0645659	-11.98	0.000	-.8999645	-.6468708
3.C						
_cons	.7802324	.0514513	15.16	0.000	.6793898	.8810751

```
Class          : 1
Response       : drvisits
Family         : Poisson
Link           : log
Response       : hpvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drvisits						
private	.1474015	.0300291	4.91	0.000	.0885456	.2062574
medicaid	.8527038	.0562373	15.16	0.000	.7424807	.9629269
age	.3724257	.0559256	6.66	0.000	.2628137	.4820378
c.age#c.age	-.0024633	.0003717	-6.63	0.000	-.0031918	-.0017348
educ	.0245594	.0040756	6.03	0.000	.0165713	.0325474
actlim	-1.245655	.0355763	-35.01	0.000	-1.315383	-1.175927
chronic	.2613438	.0110305	23.69	0.000	.2397244	.2829633
_cons	-12.35694	2.091664	-5.91	0.000	-16.45652	-8.257351
hpvisits						
private	.9375808	.0867609	10.81	0.000	.7675326	1.107629
medicaid	3.677892	.0920384	39.96	0.000	3.4975	3.858284
age	.7111631	.1290143	5.51	0.000	.4582996	.9640266
c.age#c.age	-.0047949	.0008583	-5.59	0.000	-.0064771	-.0031127
educ	.0198688	.0079941	2.49	0.013	.0042008	.0355369
actlim	-.3651267	.065141	-5.61	0.000	-.4928007	-.2374527
chronic	.005002	.0233886	0.21	0.831	-.0408387	.0508427
_cons	-26.74904	4.815349	-5.55	0.000	-36.18695	-17.31112

Class : 2

Response : drvisits
Family : Poisson
Link : log

Response : hpvisits
Family : Poisson
Link : log

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drvisits						
private	.0147941	.0388253	0.38	0.703	-.0613021	.0908902
medicaid	.3523114	.0436849	8.06	0.000	.2666905	.4379323
age	.0498856	.0732193	0.68	0.496	-.0936215	.1933927
c.age#c.age	-.0003573	.0004861	-0.73	0.462	-.0013101	.0005955
educ	.026515	.0043097	6.15	0.000	.0180681	.0349618
actlim	.4746876	.0352511	13.47	0.000	.4055968	.5437785
chronic	.1662376	.0125336	13.26	0.000	.1416721	.1908031
_cons	-.2725728	2.738699	-0.10	0.921	-5.640324	5.095179
hpvisits						
private	.2289702	.0345173	6.63	0.000	.1613176	.2966228
medicaid	-2.987888	.1067345	-27.99	0.000	-3.197084	-2.778692
age	-.0789968	.0701865	-1.13	0.260	-.2165598	.0585661
c.age#c.age	.000674	.0004644	1.45	0.147	-.0002363	.0015842
educ	.0519305	.0048815	10.64	0.000	.0423629	.0614981
actlim	.7552881	.0355466	21.25	0.000	.685618	.8249582
chronic	.0077406	.0129495	0.60	0.550	-.01764	.0331213
_cons	4.051195	2.634801	1.54	0.124	-1.112919	9.21531

```
Class          : 3
Response       : drvisits
Family         : Poisson
Link           : log
Response       : hpvisits
Family         : Poisson
Link           : log
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
drvisits						
private	.2498045	.0283265	8.82	0.000	.1942857	.3053234
medicaid	-.5869781	.0428155	-13.71	0.000	-.670895	-.5030613
age	.254306	.0491116	5.18	0.000	.1580491	.3505629
c.age#c.age	-.001653	.0003271	-5.05	0.000	-.002294	-.0010119
educ	.0337192	.0038846	8.68	0.000	.0261056	.0413328
actlim	1.197753	.0332683	36.00	0.000	1.132548	1.262957
chronic	.2395462	.0089703	26.70	0.000	.2219647	.2571277
_cons	-9.514385	1.833986	-5.19	0.000	-13.10893	-5.919839
hpvisits						
private	.2693053	.0564788	4.77	0.000	.1586089	.3800017
medicaid	-1.100159	.1030161	-10.68	0.000	-1.302066	-.8982507
age	.2981321	.1002282	2.97	0.003	.1016885	.4945758
c.age#c.age	-.0018912	.0006649	-2.84	0.004	-.0031943	-.000588
educ	.1581305	.0085644	18.46	0.000	.1413446	.1749164
actlim	1.013452	.0585852	17.30	0.000	.8986271	1.128277
chronic	.2577497	.0190401	13.54	0.000	.2204318	.2950677
_cons	-14.41321	3.755059	-3.84	0.000	-21.77299	-7.053427

To better understand the three classes, we can use `estat lcmean` to obtain the marginal counts for doctor visits and for visits to other health professionals in all three classes.

```
. estat lcmean
Latent class marginal means                     Number of obs     =      3,677
```

	Margin	Delta-method Std. Err.	z	P>|z|	[95% Conf.	Interval]
1						
drvisits	10.53552	.1875668	56.17	0.000	10.1679	10.90314
hpvisits	5.288516	.2145266	24.65	0.000	4.868052	5.70898
2						
drvisits	10.61758	.1903746	55.77	0.000	10.24445	10.99071
hpvisits	16.8819	.3076167	54.88	0.000	16.27898	17.48482
3						
drvisits	5.665389	.0735717	77.01	0.000	5.521191	5.809587
hpvisits	1.230097	.0325677	37.77	0.000	1.166265	1.293928

The first class appears to be a group that visits the doctor frequently but visits other health professionals less frequently. The second class represents those who visit both the doctor and other health professionals frequently, and the third class represents those who visit both the doctor and other health professionals less frequently.

Finally, we use `estat lcprob` to determine the expected proportions in each class.

```
. estat lcprob
Latent class marginal probabilities             Number of obs     =      3,677
```

	Margin	Delta-method Std. Err.	[95% Conf.	Interval]
C				
1	.2744679	.0097446	.2557872	.2939741
2	.1266487	.0062627	.1148739	.1394402
3	.5988834	.0108667	.5774108	.619983

Based on this model, we expect 27% of individuals to be in the first class, 13% to be in the second class, and 60% to be in the third class.

The estimation of standard errors for marginal means and marginal probabilities can be time-consuming with large models. If you are interested only in the means and probabilities, you can specify the `nose` option with `estat lcmean` and `estat lcprob` to speed up estimation. With this option, no standard errors, test statistics, or confidence intervals are reported.

References

Cameron, A. C., and P. K. Trivedi. 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.

Deb, P., and P. K. Trivedi. 1997. Demand for medical care by the elderly: A finite mixture approach. *Journal of Applied Econometrics* 12: 313–336.

Also see

[SEM] **Example 53g** — Finite mixture Poisson regression

[SEM] **Intro 5** — Tour of models

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **estat lcmean** — Latent class marginal means

[SEM] **estat lcprob** — Latent class marginal probabilities

[FMM] **fmm intro** — Introduction to finite mixture models

[FMM] **fmm: poisson** — Finite mixtures of Poisson regression models

Title

gsem — Generalized structural equation model estimation command

Description | Menu | Syntax | Options
Remarks and examples | Stored results | References | Also see

Description

`gsem` fits generalized SEMs. When you use the Builder in `gsem` mode, you are using the `gsem` command.

Menu

Statistics > SEM (structural equation modeling) > Model building and estimation

Syntax

`gsem` *paths* [*if*] [*in*] [*weight*] [, *options*]

where *paths* are the paths of the model in command-language path notation; see [SEM] **sem and gsem path notation**.

options	Description
model_description_options	fully define, along with *paths*, the model to be fit
group_options	fit model for different groups
lclass_options	fit model with latent classes
estimation_options	method used to obtain estimation results
reporting_options	reporting of estimation results
syntax_options	controlling interpretation of syntax

Factor variables and time-series operators are allowed.

`bootstrap`, `by`, `jackknife`, `permute`, `statsby`, and `svy` are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see [U] **11.1.6 weight**.

Also see [SEM] **gsem postestimation** for features available after estimation.

Options

model_description_options describe the model to be fit. The model to be fit is fully specified by *paths*—which appear immediately after `gsem`—and the options `covariance()`, `variance()`, and `means()`. See [SEM] **gsem model description options** and [SEM] **sem and gsem path notation**.

group_options allow the specified model to be fit for different subgroups of the data, with some parameters free to vary across groups and other parameters constrained to be equal across groups. See [SEM] **gsem group options**.

lclass_options allow the specified model to be fit across a specified number of latent classes, with some parameters free to vary across classes and other parameters constrained to be equal across classes. See [SEM] **gsem lclass options**.

estimation_options control how the estimation results are obtained. These options control how the standard errors (VCE) are obtained and control technical issues such as choice of estimation method. See [SEM] **gsem estimation options**.

reporting_options control how the results of estimation are displayed. See [SEM] **gsem reporting options**.

syntax_options control how the syntax that you type is interpreted. See [SEM] **sem and gsem syntax options**.

Remarks and examples

`gsem` provides important features not provided by `sem` and correspondingly omits useful features provided by `sem`. The differences in capabilities are the following:

1. `gsem` allows generalized linear response functions as well as the linear response functions allowed by `sem`.
2. `gsem` allows for multilevel models, something `sem` does not.
3. `gsem` allows for categorical latent variables, which are not allowed by `sem`.
4. `gsem` allows Stata's factor-variable notation to be used in specifying models, something `sem` does not.
5. `gsem`'s method ML is sometimes able to use more observations in the presence of missing values than can `sem`'s method ML. Meanwhile, `gsem` does not provide the MLMV method provided by `sem` for explicitly handling missing values.
6. `gsem` cannot produce standardized coefficients.
7. `gsem` cannot use summary statistic datasets (SSDs); `sem` can.

`gsem` has nearly identical syntax to `sem`. Differences in syntax arise because of differences in capabilities. The resulting differences in syntax are the following:

1. `gsem` adds new syntax to *paths* to handle latent variables associated with multilevel modeling.
2. `gsem` adds new options to handle the family and link of generalized linear responses.
3. `gsem` adds new syntax to handle categorical latent variables.
4. `gsem` deletes options related to features it does not have, such as SSDs.
5. `gsem` adds technical options for controlling features not provided by `sem`, such as numerical integration (quadrature choices), number of integration points, and a number of options dealing with starting values, which are a more difficult proposition in the generalized SEM framework.

For a readable explanation of what `gsem` can do and how to use it, see the intro sections. You might start with [SEM] **Intro 1**.

For examples of `gsem` in action, see the example sections. You might start with [SEM] **Example 1**.

For detailed syntax and descriptions, see the references below.

Remarks on three advanced topics are presented under the following headings:

Default normalization constraints
Default covariance assumptions
How to solve convergence problems

Default normalization constraints

`gsem` applies the same rules as `sem` to identify models; see [SEM] **sem** and see [SEM] **Intro 4**. Everything said there about continuous latent variables applies to multilevel latent variables such as `M1[school]` and `M2[school>teacher]`.

Default covariance assumptions

`gsem` assumes the same covariance structures as does `sem`; see [SEM] **sem** and see [SEM] **Intro 4**. `gsem`, however, treats covariances between observed exogenous variables as given. Actually, so does `sem` unless you specify an override. The override cannot be specified with `gsem`.

How to solve convergence problems

See [SEM] **Intro 12**.

Stored results

`gsem` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_clust)`	number of clusters
`e(N_groups)`	number of groups
`e(k)`	number of parameters
`e(k_cat#)`	number of categories for the #th *depvar*, ordinal
`e(k_dv)`	number of dependent variables
`e(k_eq)`	number of equations in `e(b)`
`e(k_out#)`	number of outcomes for the #th *depvar*, mlogit
`e(k_rc)`	number of covariances
`e(k_rs)`	number of variances
`e(ll)`	log likelihood
`e(n_quad)`	number of integration points
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if target model converged, `0` otherwise

Macros

e(cmd)	gsem
e(cmdline)	command as typed
e(depvar)	names of dependent variables
e(eqnames)	names of equations
e(wtype)	weight type
e(wexp)	weight expression
e(fweight*k*)	fweight variable for *k*th level, if specified
e(pweight*k*)	pweight variable for *k*th level, if specified
e(iweight*k*)	iweight variable for *k*th level, if specified
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(family#)	family for the #th *depvar*
e(link#)	link for the #th *depvar*
e(offset#)	offset for the #th *depvar*
e(intmethod)	integration method
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. Err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(method)	estimation method: ml
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(covariates)	list of covariates
e(footnote)	program used to implement the footnote display
e(groupvar)	name of group variable
e(lclass)	name of latent class variables
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved
e(marginsnotok)	predictions not allowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(marginsdefault)	default predict() specification for margins

Matrices

e(_N)	sample size for each *depvar*
e(b)	parameter vector
e(b_pclass)	parameter class
e(cat#)	categories for the #th *depvar*, ordinal
e(out#)	outcomes for the #th *depvar*, mlogit
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(nobs)	vector with number of observations per group
e(groupvalue)	vector of group values of e(groupvar)
e(lclass_k_levels)	number of levels for latent class variables
e(lclass_bases)	base levels for latent class variables

Functions

e(sample)	marks estimation sample

References

Bartus, T. 2017. Multilevel multiprocess modeling with gsem. *Stata Journal* 17: 442–461.

Canette, I. 2013. Fitting ordered probit models with endogenous covariates with Stata's gsem command. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2013/11/07/fitting-ordered-probit-models-with-endogenous-covariates-with-statas-gsem-command/.

——. 2014. Using gsem to combine estimation results. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2014/08/18/using-gsem-to-combine-estimation-results/.

Lindsey, C., and E. Pinzon. 2016. Multiple equation models: Estimation and marginal effects using gsem. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/06/07/multiple-equation-models-estimation-and-marginal-effects-using-gsem/.

Also see

[SEM] **Intro 1** — Introduction

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **gsem path notation extensions** — Command syntax for path diagrams

[SEM] **gsem model description options** — Model description options

[SEM] **gsem group options** — Fitting models on different groups

[SEM] **gsem lclass options** — Fitting models with latent classes

[SEM] **gsem estimation options** — Options affecting estimation

[SEM] **gsem reporting options** — Options affecting reporting of results

[SEM] **sem and gsem syntax options** — Options affecting interpretation of syntax

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Methods and formulas for gsem** — Methods and formulas for gsem

[SVY] **svy estimation** — Estimation commands for survey data

[U] **20 Estimation and postestimation commands**

Title

gsem estimation options — Options affecting estimation

Description Syntax Options Remarks and examples Also see

Description

These options control how results are obtained, from starting values, to numerical integration (also known as quadrature), to how variance estimates are obtained.

Syntax

```
gsem paths ..., ... estimation_options
```

estimation_options	Description
method(ml)	method used to obtain the estimated parameters; only one method available with gsem
vce(*vcetype*)	*vcetype* may be oim, opg, robust, or cluster *clustvar*
pweights(*varlist*)	sampling weight variables for each level
fweights(*varlist*)	frequency weight variables for each level
iweights(*varlist*)	importance weight variables for each level
from(*matname*)	specify starting values
startvalues(*svmethod*)	method for obtaining starting values
startgrid[(*gridspec*)]	perform a grid search to improve starting values
emopts(*maxopts*)	control EM algorithm for improved starting values
noestimate	do not fit the model; show starting values instead
intmethod(*intmethod*)	integration method
intpoints(#)	set the number of integration (quadrature) points
adaptopts(*adaptopts*)	options for adaptive quadrature
listwise	apply sem's (not gsem's) rules for omitting observations with missing values
dnumerical	use numerical derivative techniques
etolerance(#)	set the rescaling tolerance to # to prevent numerical overflow in models with continuous latent variables; rarely used
maximize_options	control maximization process for specified model; seldom used

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation

adaptopts	Description
[no]log	whether to display the iteration log for each numerical integral calculation
iterate(#)	set the maximum number of iterations of the adaptive technique; default is iterate(1001)
tolerance(#)	set tolerance for determining convergence of the adaptive parameters; default is tolerance(1e-8)

Options

method(ml) is the default and is the only method available with gsem. This option is included for compatibility with sem, which provides several methods; see [SEM] **sem option method()**.

vce(*vcetype*) specifies the technique used to obtain the variance–covariance matrix of the estimates. See [SEM] **sem option method()**.

pweights(*varlist*), fweights(*varlist*), and iweights(*varlist*) specify weight variables to be applied from the observation (first) level to the highest level groups. Only one of these options may be specified, and none of them are allowed with crossed models or intmethod(laplace).

pweights(*varlist*) specifies sampling weights and implies vce(robust) if the vce() option was not also specified.

fweights(*varlist*) specifies frequency weights.

iweights(*varlist*) specifies importance weights.

from(*matname*), startvalues(*svmethod*), startgrid[(*gridspec*)], and emopts(*maxopts*) specify overriding starting values, specify how other starting values are to be calculated, and provide the ability to improve the starting values. All of this is discussed in [SEM] **Intro 12**. Below we provide a technical description.

from(*matname*) allows you to specify starting values. See [SEM] **Intro 12** and see [SEM] **sem and gsem option from()**. We show the syntax as from(*matname*), but from() has another, less useful syntax, too. An alternative to from() is init() used in the path specifications; see [SEM] **sem and gsem path notation**.

startvalues() specifies how starting values are to be computed. Starting values specified in from() override the computed starting values, and starting values specified via init() override both.

Starting values options for models without categorical latent variables are as follows:

startvalues(zero) specifies that starting values are to be set to 0.

startvalues(constantonly) builds on startvalues(zero) by fitting a constant-only model for each response to obtain estimates of intercept and scale parameters, and it substitutes 1 for the variances of latent variables.

startvalues(fixedonly [, *maxopts*]) builds on startvalues(constantonly) by fitting a full fixed-effects model for each response variable to obtain estimates of coefficients along with intercept and scale parameters, and it continues to use 1 for the variances of latent variables.

startvalues(ivloadings [, *maxopts*]) builds on startvalues(fixedonly) by using instrumental-variable methods with the generalized residuals from the fixed-effects models to compute starting values for latent variable loadings, and still uses 1 for the variances of latent variables.

`startvalues(iv [ , `*maxopts*`])` builds on `startvalues(ivloadings)` by using instrumental-variable methods with generalized residuals to obtain variances of latent variables.

Starting values options for models with categorical latent variables are as follows:

`startvalues(factor [ , `*maxopts*`])` specifies that starting values be computed by assigning each observation to an initial latent class that is determined by running a `factor` analysis on all the observed variables in the specified model. This is the default for models with categorical latent variables.

`startvalues(classid `*varname*`[ , `*maxopts*`])` specifies that starting values be computed by assigning each observation to an initial latent class specified in *varname*. *varname* is required to have each class represented in the estimation sample.

`startvalues(classpr `*varlist*`[ , `*maxopts*`])` specifies that starting values be computed using the initial class probabilities specified in *varlist*. *varlist* is required to contain k variables for a model with k latent classes. The values in *varlist* are normalized to sum to 1 within each observation.

`startvalues(randomid [ , draws(#) seed(#) `*maxopts*`])` specifies that starting values be computed by randomly assigning observations to initial classes.

`startvalues(randompr [ , draws(#) seed(#) `*maxopts*`])` specifies that starting values be computed by randomly assigning initial class probabilities.

`startvalues(jitter [`$\#_c$ `[`$\#_v$`] , draws(#) seed(#) `*maxopts*`])` specifies that starting values be constructed by randomly perturbing the values from a Gaussian approximation to each outcome.

$\#_c$ is the magnitude for randomly perturbing coefficients, intercepts, cutpoints, and scale parameters; the default value is 1.

$\#_v$ is the magnitude for randomly perturbing variances for Gaussian outcomes; the default value is 1.

Some starting values options have suboptions that allow for tuning the starting values calculations:

maxopts is a subset of the standard *maximize_options*: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`; see [R] **Maximize**.

`draws(#)` specifies the number of random draws. For `startvalues(randomid)`, `startvalues(randompr)`, and `startvalues(jitter)`, `gsem` will generate # random draws and select the starting values from the draw with the best log-likelihood value from the EM iterations. The default is `draws(1)`.

`seed(#)` sets the random-number seed.

`startgrid()` performs a grid search on variance components of latent variables to improve starting values. This is well discussed in [SEM] **Intro 12**. No grid search is performed by default unless the starting values are found to be not feasible, in which case `gsem` runs `startgrid()` to perform a "minimal" search involving 3^L likelihood evaluations, where L is the number of latent variables. Sometimes this resolves the problem. Usually, however, there is no problem and `startgrid()` is not run by default. There can be benefits from running `startgrid()` to get better starting values even when starting values are feasible.

`emopts(`*maxopts*`)` controls maximization of the log likelihood for the EM algorithm. The EM algorithm is used only for models with categorical latent variables. *maxopts* is the same subset of *maximize_options* that are allowed in the `startvalues()` option, but some of the defaults are

different for the EM algorithm. The default maximum number of iterations is `iterate(20)`. The default coefficient vector tolerance is `tolerance(1e-4)`. The default log-likelihood tolerance is `ltolerance(1e-6)`.

`noestimate` specifies that the model is not to be fit. Instead, starting values are to be shown (as modified by the above options if modifications were made), and they are to be shown using the `coeflegend` style of output. An important use of this option is before you have modified starting values at all; you can type the following:

```
. gsem ..., ... noestimate
. matrix b = e(b)
. ... (modify elements of b) ...
. gsem ..., ... from(b)
```

`intmethod(`*intmethod*`)`, `intpoints(`*#*`)`, and `adaptopts(`*adaptopts*`)` affect how integration for the latent variables is numerically calculated.

`intmethod(`*intmethod*`)` specifies the method and defaults to `intmethod(mvaghermite)`. We recommend this method, although sometimes the more computationally intensive `intmethod(mcaghermite)` works better for multilevel models that are failing to converge. Sometimes it is useful to fall back on the less computationally intensive and less accurate `intmethod(ghermite)` and `intmethod(laplace)` to get the model to converge and then perhaps use one of the other more accurate methods. All of this is explained in [SEM] **Intro 12**. `intmethod(laplace)` is the default when fitting crossed models. Crossed models are often difficult.

`intpoints(`*#*`)` specifies the number of integration points to use and defaults to `intpoints(7)`. Increasing the number increases accuracy but also increases computational time. Computational time is roughly proportional to the number specified. See [SEM] **Intro 12**.

`adaptopts(`*adaptopts*`)` affects the adaptive part of adaptive quadrature (another term for numerical integration) and thus is relevant only for `intmethod(mvaghermite)`, `intmethod(mcaghermite)`, and `intmethod(laplace)`.

`adaptopts()` defaults to `adaptopts(nolog iterate(1001) tolerance(1e-8))`.

[`no`]`log` specifies whether iteration logs are shown each time a numerical integral is calculated.

`iterate()` specifies the maximum number of iterations of the adaptive technique.

`tolerance()` specifies the tolerance for determining convergence of the adaptive parameters. Convergence is declared when the relative change in the log likelihood is less than or equal to the tolerance.

`listwise` applies `sem`'s rules rather than `gsem`'s rules for omitting observations with missing values. By default, `gsem` is sometimes able to use observations containing missing values for fitting parts of the model. `sem`, meanwhile, applies a listwise-deletion rule unless it is using `method(mlmv)`. Specifying `listwise` allows us at StataCorp to verify that `gsem` and `sem` produce the same results. We find that reassuring. Actually, automated tests verify that results are the same before shipping. For your information, `sem` and `gsem` use different numerical machinery for obtaining results, and thus the near equality of results is a strong test that each is coded correctly. You may find `listwise` useful if you are reproducing results from another package that uses listwise deletion.

`dnumerical` specifies that during optimization, the gradient vector and Hessian matrix be computed using numerical techniques instead of analytical formulas. By default, `gsem` uses analytical formulas for computing the gradient and Hessian for all integration methods except `intmethod(laplace)`.

`etolerance(#)` specifies a positive tolerance used to determine when to rescale log-likelihood values that are too big to exponentiate. The formula for the default tolerance is

`max(2,`*nLvars*`) * log(`*p*`)`

where *nLvars* is the number of latent variables in the model and p is the number of quadrature points.

The rule `gsem` uses to determine when to rescale log-likelihood values at a given level is the following: rescale log-likelihood values outside the range

$$[-M, M - \#]$$

where M is a limit that represents values beyond which exponentiation can yield a missing value.

This option is rarely ever necessary but can be helpful for multilevel models with large group sizes.

maximize_options specify the standard and rarely specified options for controlling the maximization process; see [R] **Maximize**. The relevant options for `gsem` are `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)` and `nonrtolerance`.

Remarks and examples

For more information on `vce()`, see [SEM] **Intro 8** and [SEM] **Intro 9**.

For more information on the other options, see [SEM] **Intro 12**.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 8** — Robust and clustered standard errors

[SEM] **Intro 9** — Standard errors, the full story

[SEM] **Intro 12** — Convergence problems and how to solve them

Title

gsem family-and-link options — Family-and-link options

Description Syntax Options Remarks and examples Also see

Description

gsem not only allows models of the form $y_i = \mathbf{x}_i\boldsymbol{\beta} + u_i$, it also allows

$$g\{E(y_i)\} = \mathbf{x}_i\boldsymbol{\beta}$$
$$y_i \sim F$$

where you can choose F and $g(\cdot)$ from a menu. F is called the family, and $g(\cdot)$ is called the link. One set of choices is the Gaussian distribution for F and the identity function for $g(\cdot)$. In that case, gsem reproduces linear regression. Other combinations of $g(\cdot)$ and F produce other popular models, including logit (also known as logistic regression), probit, multinomial logit, Poisson regression, and more.

Syntax

gsem *paths* ..., ... *family_and_link_options*

family_and_link_options	Description
family(*family*)	distribution family; default is family(gaussian)
link(*link*)	link function; default varies per family
cloglog	synonym for family(bernoulli) link(cloglog)
exponential	synonym for family(exponential) link(log)
gamma	synonym for family(gamma) link(log)
logit	synonym for family(bernoulli) link(logit)
loglogistic	synonym for family(loglogistic) link(log)
lognormal	synonym for family(lognormal) link(log)
llogistic	synonym for family(llogistic) link(log)
lnormal	synonym for family(lnormal) link(log)
mlogit	synonym for family(multinomial) link(logit)
nbreg	synonym for family(nbreg mean) link(log)
ocloglog	synonym for family(ordinal) link(cloglog)
ologit	synonym for family(ordinal) link(logit)
oprobit	synonym for family(ordinal) link(probit)
poisson	synonym for family(poisson) link(log)
probit	synonym for family(bernoulli) link(probit)
regress	synonym for family(gaussian) link(identity)
weibull	synonym for family(weibull) link(log)
exposure(*varname*$_e$)	include ln(*varname*$_e$) with coefficient constrained to 1
offset(*varname*$_o$)	include *varname*$_o$ with coefficient constrained to 1

family	Description
`gaussian` [, *options*]	Gaussian (normal); the default
`bernoulli`	Bernoulli
`beta`	beta
`binomial` [# | *varname*]	binomial; default number of binomial trials is 1
`ordinal`	ordinal
`multinomial`	multinomial
`poisson` [, *poisson*]	Poisson
`nbinomial` [`mean` | `constant`]	negative binomial; default dispersion is `mean`
`exponential` [, *survival*]	exponential
`gamma` [, *survival*]	gamma
`loglogistic` [, *survival*]	loglogistic
`lognormal` [, *survival*]	lognormal
`weibull` [, *survival*]	Weibull
`pointmass` #	point-mass density at #

link	Description
`identity`	identity
`log`	log
`logit`	logit
`probit`	probit
`cloglog`	complementary log-log

options	Description
`ldepvar(`*varname*`)`	lower depvar for interval-response data
`udepvar(`*varname*`)`	upper depvar for interval-response data
`lcensored(`*varname* | #`)`	lower limit for left-censoring
`rcensored(`*varname* | #`)`	upper limit for right-censoring

Only allowed with `family(gaussian)` with `link(identity)`.

poisson	Description
`ltruncated(`*varname* | #`)`	lower limit for left-truncation

survival	Description
`ltruncated(`*varname* | #`)`	lower limit for left-truncation
`failure(`*varname*`)`	indicator for failure event
`ph`	proportional hazards parameterization
`aft`	accelerated failure-time parameterization

`ph` is allowed only with families `exponential` and `weibull`.

`aft` is allowed only with families `exponential`, `gamma`, `loglogistic`, `lognormal`, and `weibull`.

If you specify both `family()` and `link()`, not all combinations make sense. You may choose from the following combinations:

	identity	log	logit	probit	cloglog
Gaussian	D	x			
Bernoulli			D	x	x
beta			D	x	x
binomial			D	x	x
ordinal			D	x	x
multinomial			D		
Poisson		D			
negative binomial		D			
exponential		D			
Weibull		D			
gamma		D			
loglogistic		D			
lognormal		D			
pointmass	D				

D denotes the default.

Options

`family(`*family*`)` and `link(`*linkname*`)` specify F and $g(\cdot)$. If neither is specified, linear regression is assumed.

Two of the families allow optional arguments:

`family(binomial` $\left[\right.$ `#` $\left|\right.$ *varname* $\left.\right]$`)` specifies that data are in binomial form, that is, that the response variable records the number of successes from a series of Bernoulli trials. The number of trials is given either as a constant number or as a varname that allows the number of trials to vary over observations, or it is not given at all. In the last case, the number of trials is thus equivalent to specifying `family(bernoulli)`.

`family(nbinomial` $\left[\right.$ `mean` $\left|\right.$ `constant` $\left.\right]$`)` specifies a negative binomial model, a Poisson model with overdispersion. Be aware, however, that even Poisson models can have overdispersion if latent variables are included in the model. Let's use the term "conditional overdispersion" to refer to dispersion above and beyond that implied by latent variables, if any.

That conditional overdispersion can take one of two forms. In mean overdispersion, the conditional overdispersion is a linear function of the conditional (predicted) mean. Constant overdispersion refers to the conditional overdispersion being, of course, constant.

If you do not specify `mean` or `constant`, then `mean` is assumed.

`family(pointmass #)` is a special family allowed when `lclass()` is also specified.

`cloglog`, `exponential`, `gamma`, `logit`, `loglogistic`, `lognormal`, `llogistic`, `lnormal`, `mlogit`, `nbreg`, `ocloglog`, `ologit`, `oprobit`, `poisson`, `probit`, `regress`, and `weibull` are shorthands for specifying popular models.

`exposure(`$varname_e$`)` and `offset(`$varname_o$`)` are most commonly used with families `poisson` and `nbreg`, that is, they typically concern count models.

`exposure()` specifies a variable that reflects the amount of exposure—usually measured in time units—for each observation over which the responses were counted. If one observation was exposed for twice the time of another, and the observations were otherwise identical, one would expect twice as many events to be counted. To assume that, $\ln(varname_e)$ is entered into $\mathbf{x}_i\boldsymbol{\beta}$ with coefficient constrained to be 1.

`offset()` enters $varname_o$ into $\mathbf{x}_i\boldsymbol{\beta}$ with coefficient constrained to be 1. `offset()` is just another way of specifying `exposure()` where the offset variable is the log of amount of exposure.

If neither `exposure()` nor `offset()` is specified, observations are assumed to have equal amounts of exposure.

`ldepvar(`*varname*`)` and `udepvar(`*varname*`)` specify that each observation can be point data, interval data, left-censored data, or right-censored data. The type of data for a given observation is determined by the values in y_i and *varname*. The following specifications are equivalent:

depvar$_1$ `<- ... , family(gauss, udepvar(`*depvar*$_2$`)`

depvar$_2$ `<- ... , family(gauss, ldepvar(`*depvar*$_1$`)`

Thus only one of `ldepvar()` or `udepvar()` is allowed. In either case, *depvar*$_1$ and *depvar*$_2$ should have the following form:

Type of data		*var*$_1$	*var*$_2$
point data	$a = [a, a]$	a	a
interval data	$[a, b]$	a	b
left-censored data	$(-\infty, b]$	.	b
right-censored data	$[a, +\infty)$	a	.

`lcensored(`*varname*`|#)` and `rcensored(`*varname*`|#)` indicate the lower and upper limits for censoring, respectively. You may specify only one.

`lcensored(`*arg*`)` specifies that observations with $y_i \leq arg$ are left-censored and the remaining observations are not.

`rcensored(`*arg*`)` specifies that observations with $y_i \geq arg$ are right-censored and the remaining observations are not.

Neither `lcensored()` nor `rcensored()` may not be combined with `ldepvar()` or `udepvar()`.

`ltruncated(`*varname*`|#)` indicates the lower limits for truncation.

`ltruncated(`*arg*`)` specifies that the distribution is truncated on the left at *arg*, meaning that $y_i \leq arg$ is not within the support for the corresponding distribution family. This option rescales the underlying density function to accommodate the truncated support for y_i. This means that values of y_i that are less than or equal to *arg* do not contribute to the likelihood. For survival families, this means that time (time at risk) starts at *arg* instead of at 0.

`failure(`*varname*`)` specifies the failure event.

If `failure()` is not specified, all observations are assumed to indicate a failure.

If `failure(`*varname*`)` is specified, *varname* is interpreted as an indicator variable; 0 and missing mean censored, and all other values are interpreted as representing failure.

`ph` specifies the proportional hazards parameterization and is allowed with families `exponential` and `weibull`. This is the default parameterization for these families.

`aft` specifies the accelerated failure-time parameterization and is allowed with families `exponential`, `gamma`, `loglogistic`, `lognormal`, and `weibull`. This is an optional parameterization for `exponential` and `weibull` but the only parameterization for the others.

Remarks and examples

In the command language, the family-and-link options may be specified at the end of a command among the global options:

```
. gsem ..., ... family(...) link(...) ...
. gsem ..., ... poisson exposure(time) ...
```

Specified that way, the options apply to all the response variables. Alternatively, they may be specified inside paths to affect single equations:

```
. gsem (y1 <- x1 x2, family(...) poisson(...)) (y2 <- x2 L) ...
. gsem (y1 <- x1 x2, family(...) link(...))       ///
       (y2 <- x2 L,  family(...) link(...)) ...
. gsem (y1 <- x1 x2, poisson exposure(time)) (y2 <- x2 L) ...
. gsem (y1 <- x1 x2, poisson exposure(time))      ///
       (y2 <- x2 L,  family(...) link(...)) ...
```

On a different topic, it is worth nothing that you can fit exponential-regression models with `family(gamma) link(log)` if you constrain the log of the scale parameter to be 0 with `gsem`'s `constraints()` option. For instance, you might type

```
. constraint 1 _b[/y:logs] = 0
. gsem (y <- x1 x2, gamma), constraints(1)
```

The name `_b[/y:logs]` changes according to the name of the dependent variable. Had `y` instead been named `waitingtime`, the parameter would have been named `_b[/waitingtime:logs]`. Rather than remembering that, remember instead that the best way to discover the names of parameters is to type

```
. gsem (waitingtime <- x1 x2, gamma), noestimate
```

and then look at the output to discover the names. See [SEM] **sem and gsem option constraints( )**.

For examples of generalized response variables, see the following:

[SEM] **Example 27g**: Single-factor measurement model (generalized response)
[SEM] **Example 28g**: One-parameter logistic IRT (Rasch) model
[SEM] **Example 29g**: Two-parameter logistic IRT model
[SEM] **Example 30g**: Two-level measurement model (multilevel, generalized response)
[SEM] **Example 31g**: Two-factor measurement model (generalized response)
[SEM] **Example 32g**: Full structural equation model (generalized response)
[SEM] **Example 33g**: Logistic regression
[SEM] **Example 34g**: Combined models (generalized responses)
[SEM] **Example 35g**: Ordered probit and ordered logit
[SEM] **Example 36g**: MIMIC model (generalized response)
[SEM] **Example 37g**: Multinomial logistic regression
[SEM] **Example 39g**: Three-level model (multilevel, generalized response)
[SEM] **Example 41g**: Two-level multinomial logistic regression (multilevel)
[SEM] **Example 43g**: Tobit regression
[SEM] **Example 44g**: Interval regression
[SEM] **Example 45g**: Heckman selection model
[SEM] **Example 46g**: Endogenous treatment-effects model
[SEM] **Example 47g**: Exponential survival model
[SEM] **Example 48g**: Loglogistic survival model with censored and truncated data
[SEM] **Example 49g**: Multiple-group Weibull survival model
[SEM] **Example 50g**: Latent class model
[SEM] **Example 52g**: Latent profile model

Also see

[SEM] **gsem** — Generalized structural equation model estimation command
[SEM] **Intro 2** — Learning the language: Path diagrams and command language
[SEM] **sem and gsem path notation** — Command syntax for path diagrams
[SEM] **gsem path notation extensions** — Command syntax for path diagrams

Title

gsem group options — Fitting models on different groups

Description Syntax Options Remarks and examples Also see

Description

`gsem` can fit combined models across subgroups of the data while allowing some parameters to vary and constraining others to be equal across subgroups. These subgroups could be males and females, age category, and the like.

`gsem` performs such estimation when the `group(`*varname*`)` option is specified. The `ginvariant(`*pclassname*`)` option specifies which parameters are to be constrained to be equal across the groups.

Syntax

`gsem` *paths* `...,` `...` *group_options*

group_options	Description
`group(`*varname*`)`	fit model for different groups
`ginvariant(`*pclassname*`)`	specify parameters that are equal across groups

pclassname	Description
`cons`	intercepts and cutpoints
`coef`	fixed coefficients
`loading`	latent variable coefficients
`errvar`	covariances of errors
`scale`	scaling parameters
`means`	means of exogenous variables
`covex`	covariances of exogenous latent variables
`all`	all the above
`none`	none of the above

`ginvariant(cons coef loading)` is the default if `ginvariant()` is not specified.

Options

`group(`*varname*`)` specifies that the model be fit as described above. *varname* specifies the name of a numeric variable that records the group to which the observation belongs.

`ginvariant(`*pclassname*`)` specifies which classes of parameters of the model are to be constrained to be equal across groups. The classes are defined above. The default is `ginvariant(cons coef loading)` if the option is not specified.

Remarks and examples

See [SEM] **Intro 6** and [SEM] **Example 49g**.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 6** — Comparing groups

[SEM] **Example 49g** — Multiple-group Weibull survival model

Title

gsem lclass options — Fitting models with latent classes

Description Syntax Options Remarks and examples Also see

Description

`gsem` can fit models with categorical latent variables having specified numbers of latent classes. Some parameters can vary across classes while others are constrained to be equal across classes.

`gsem` performs such estimation when the `lclass()` option is specified. The `lcinvariant(`*pclassname*`)` option specifies which parameters are to be constrained to be equal across the latent classes.

Syntax

`gsem` *paths* `...,` `...` `lclass(`*lcname* `#` [`, base(#)`]`)` `lcinvariant(`*pclassname*`)`

lclass_options	Description
`lclass()`	fit model with latent classes
`lcinvariant(`*pclassname*`)`	specify parameters that are equal across latent classes

pclassname	Description
`cons`	intercepts and cutpoints
`coef`	fixed coefficients
`errvar`	covariances of errors
`scale`	scaling parameters
`all`	all the above
`none`	none of the above

`lcinvariant(errvar scale)` is the default if `lcinvariant()` is not specified.

Options

`lclass(`*lcname* `#` [`, base(#)`]`)` specifies that the model be fit as described above.

lcname specifies the name of a categorical latent variable, and # specifies the number of latent classes. The latent classes are the contiguous integers starting with `1` and ending with #.

`base(#)` specifies the class of *lcname* to be treated as the base class. The default is `base(1)`.

`lcinvariant(`*pclassname*`)` specifies which classes of parameters of the model are to be constrained to be equal across the latent classes. The classes are defined above. The default is `lcinvariant(errvar scale)`.

Remarks and examples

See [SEM] **Intro 2**, and see [SEM] **Example 50g**, [SEM] **Example 51g**, and [SEM] **Example 52g**.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **Example 50g** — Latent class model

[SEM] **Example 51g** — Latent class goodness-of-fit statistics

[SEM] **Example 52g** — Latent profile model

Title

gsem model description options — Model description options

Description Syntax Options Remarks and examples Also see

Description

paths and the options above describe the model to be fit by `gsem`.

Syntax

`gsem` *paths* `...,` `...` *model_description_options*

model_description_options	Description
`family()`, `link()`, ...	see [SEM] **gsem family-and-link options**
* `covariance()`	path notation for treatment of covariances; see [SEM] **sem and gsem path notation**
* `variance()`	path notation for treatment of variances; see [SEM] **sem and gsem path notation**
* `means()`	path notation for treatment of means; see [SEM] **sem and gsem path notation**
* `covstructure()`	alternative method to place restrictions on covariances; see [SEM] **sem and gsem option covstructure()**
`collinear`	keep collinear variables
`noconstant`	do not fit intercepts
`noasis`	omit perfect predictors from Bernoulli models
`noanchor`	do not apply default anchoring
`forcenoanchor`	programmer's option
* `reliability()`	reliability of measurement variables; see [SEM] **sem and gsem option reliability()**
`constraints()`	specify constraints; see [SEM] **sem and gsem option constraints()**
`from()`	specify starting values; see [SEM] **sem and gsem option from()**

* Option may be specified more than once.

Options

`family()` and `link()` specify the distribution and link function, such as `family(poisson)` `link(log)`, for generalized linear responses. There are lots of synonyms, so you can specify, for example, just `poisson`. In addition, there are `exposure()` and `offset()` options. See [SEM] **gsem family-and-link options**.

`covariance()`, `variance()`, and `means()` fully describe the model to be fit. See [SEM] **sem and gsem path notation**.

`covstructure()` provides a convenient way to constrain covariances in your model. Alternatively or in combination, you can place constraints by using the standard path notation. See [SEM] **sem and gsem option covstructure()**.

`collinear`; see [R] **Estimation options**.

`noconstant` specifies that all intercepts be constrained to 0. See [SEM] **sem and gsem path notation**.

This option is seldom specified.

`noasis` specifies that perfect-predictor variables be omitted from all family Bernoulli models. By default, `gsem` does not omit the variable, so one can specify tricky models where an equation contains perfect predictors that are still identified through other portions of the model.

`noanchor` specifies that `gsem` not check for lack of identification or fill in anchors where needed. `gsem` is instead to issue an error message if anchors would be needed. Specify this option when you believe you have specified the necessary normalization constraints and want to hear about it if you are wrong. See *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.

`forcenoanchor` is similar to `noanchor` except that rather than issue an error message, `gsem` proceeds to estimation. There is no reason you should specify this option. `forcenoanchor` is used in testing of `gsem` at StataCorp.

`reliability()` specifies the fraction of variance not due to measurement error for a variable. See [SEM] **sem and gsem option reliability()**.

`constraints()` specifies parameter constraints you wish to impose on your model; see [SEM] **sem and gsem option constraints()**. Constraints can also be specified as described in [SEM] **sem and gsem path notation**, and they are usually more conveniently specified using the path notation.

`from()` specifies the starting values to be used in the optimization process; see [SEM] **sem and gsem option from()**. Starting values can also be specified using the `init()` suboption as described in [SEM] **sem and gsem path notation**.

Remarks and examples

To use `gsem` successfully, you need to understand *paths*, `covariance()`, `variance()`, and `means()`; see *Using path diagrams to specify standard linear SEMs* in [SEM] **Intro 2** and see [SEM] **sem and gsem path notation**.

`covstructure()` is often convenient; see [SEM] **sem and gsem option covstructure()**.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **sem and gsem option constraints()** — Specifying constraints

[SEM] **sem and gsem option covstructure()** — Specifying covariance restrictions

[SEM] **sem and gsem option from()** — Specifying starting values

[SEM] **sem and gsem option reliability()** — Fraction of variance not due to measurement error

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

Title

gsem path notation extensions — Command syntax for path diagrams

Description Syntax Options Remarks and examples Also see

Description

This entry concerns `gsem` only.

The command syntax for describing generalized SEMs is fully specified by *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()`; see [SEM] **sem and gsem path notation** and [SEM] **sem and gsem option covstructure( )**.

With `gsem`, the notation is extended to allow for generalized linear response variables, multilevel latent variables, categorical latent variables, and comparisons of groups. That is the subject of this entry.

Syntax

`gsem` *paths* ... [`, covariance() variance() means() group() lclass()`]

`gsem` *paths* ... [`, covstructure() means() group() lclass()`]

paths specifies the direct paths between the variables of your model.

The model to be fit is fully described by *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()`.

The syntax of these elements is modified when the `group()` or `lclass()` option is specified.

Options

`covariance()`, `variance()`, and `means()` are described in [SEM] **sem and gsem path notation**.

`covstructure()` is described in [SEM] **sem and gsem option covstructure( )**.

`group(`*varname*`)` allows models specified with *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()` to be automatically interacted with the groups defined by *varname*; see [SEM] **Intro 6**. The syntax of *paths* and the arguments of `covariance()`, `variance()`, `covstructure()`, and `means()` gain an extra syntactical piece when `group()` is specified.

`lclass()` allows models specified with *paths*, `covariance()`, `variance()`, and `covstructure()` to be automatically interacted with categorical latent variables; see [SEM] **Intro 2**. The syntax of *paths* and the arguments of `covariance()`, `variance()`, and `covstructure()` gain an extra syntactical piece when `lclass()` is specified.

Remarks and examples

Remarks are presented under the following headings:

Specifying family and link
Specifying multilevel nested latent variables
Specifying multilevel crossed latent variables
Specifying paths for a specific group
Specifying paths for a specific latent class
Specifying paths for a specific group and latent class

Specifying family and link

gsem fits not only linear models but also generalized linear models. There is a set of options for specifying the specific model to be fit. These options are known as family-and-link options, which include family() and link(), but those options are seldom used in favor of other family-and-link shorthand options such as logit, which means family(bernoulli) and link(logit). These options are explained in [SEM] **gsem family-and-link options**.

In the command language, you can specify these options among the shared options at the end of a gsem command:

```
. gsem ..., ... logit ...
```

That is convenient but only if all the equations in the model are using the same specific response function. Many models include multiple equations with each using a different response function.

You can specify any of the family-and-link options within paths. For instance, typing

```
. gsem (y <- x1 x2), logit
```

has the same effect as typing

```
. gsem (y <- x1 x2, logit)
```

Thus you can type

```
. gsem (y1 <- x1 L, logit) (y2 <- x2 L, poisson) ..., ...
```

The y1 equation would be logit, and the y2 equation would be Poisson. If you wanted y2 to be linear regression, you could type

```
. gsem (y1 <- x1 L, logit) (y2 <- x2 L, regress) ..., ...
```

or you could be silent and let y2 default to linear regression,

```
. gsem (y1 <- x1 L, logit) (y2 <- x2 L) ..., ...
```

Specifying multilevel nested latent variables

Latent variables are indicated by a name in which at least the first letter is capitalized. This generic form of the name is often written *Lname*.

In regular latent variables, which we will call level-1 latent variables, the unobserved values vary observation by observation. Level-1 latent variables are the more common kinds of latent variables.

gsem allows higher-level latent variables as well as the level-1 variables. Let's consider three-level data: students at the observational level, teachers at the second level, and schools at the third. In these data, each observation is a student. We have data on students nested within teachers nested within schools.

Let's assume that we correspondingly have three identification (ID) variables. We number the following list with the nesting level of the data:

3. Variable school contains a school ID number. If two observations have the same value of school, then both of those students attended the same school.
2. Variable teacher contains a teacher ID number, or it contains a teacher-within-school ID number. That is, we do not care whether different schools assigned teachers the same ID number. It will be sufficient for us that the ID number is unique within school.

1. Variable `student` contains a student ID number, or it contains a student-within-school ID number, or even a student-within-teacher-within-school ID number. That is, we do not care whether different observations have the same student ID as long as they have different teacher IDs or different school IDs.

Here is how you write latent variable names at each level of the model:

3. Level 3 is the school level. Latent variables are written as

   Lname`[school]`

   An example would be `SchQuality[school]`.

   The unobserved values of *Lname*`[school]` vary across schools and are constant within school.

   If *Lname*`[school]` is endogenous, its error variable is `e.`*Lname*`[school]`.

   You must refer to *Lname*`[school]` without omitting the `[school]` part. *Lname* by itself looks like another latent variable to `gsem`.

2. Level 2 is the teacher-within-school level. Latent variables are written as

   Lname`[school>teacher]` or
   Lname`[teacher<school]`

   An example would be `TeachQuality[school>teacher]` or
   `TeachQuality[teacher<school]`.

   To `gsem`, *Lname*`[school>teacher]` and *Lname*`[teacher<school]` mean the same thing. You can even refer to `TeachQuality[school>teacher]` in one place and refer to `TeachQuality[teacher<school]` in another, and there will be no confusion.

   The unobserved values of *Lname*`[school>teacher]` vary across schools and teachers, and they are constant within teacher.

   If *Lname*`[school>teacher]` is endogenous, its error variable is
   `e.`*Lname*`[school>teacher]` or, equivalently, `e.`*Lname*`[teacher<school]`.

1. Level 1 is the student or observational level. Latent variables are written as

   Lname`[school>teacher>student]` or
   Lname`[student<teacher<school]` or
   Lname

   Everybody just writes *Lname*. These are the latent variables that correspond to the latent variables that `sem` provides. Unobserved values within the latent variable vary observation by observation.

   If *Lname* is endogenous, its error variable is `e.`*Lname*.

You can use multilevel latent variables in paths and options just as you would use any other latent variable; see [SEM] **sem and gsem path notation**. Remember, however, that you must type out the full name of all but the first-level latent variables. You type, for instance, `SchQual[school>teacher]`. There is a real tendency to type just `SchQual` when the name is unique.

Changing the subject, we see that the names by which effects are referred to are a function of the top level. We just discussed a three-level model. The three levels of the model were

(3) `school`
(2) `school>teacher`
(1) `school>teacher>student`

If we had a two-level model, the levels would be

(2) `teacher`
(1) `teacher>student`

Thus, if we had started with a two-level model and then wanted to add a third, higher level onto it, latent variables that were previously referred to as, say, `TeachQual[teacher]` would now be referred to as `TeachQual[school>teacher]`.

Specifying multilevel crossed latent variables

In our previous example, we had a three-level nested model in which student was nested within teacher, which was nested within school.

Let's consider data on employees that also have the characteristics of working in an occupation and working in an industry. These variables are not nested. Just as before, we will assume we have variable `employee` containing an employee ID and variables `industry` and `occupation`. The latent variables associated with this model could be the following:

Level	Latent-variable name
occupation	*Lname*`[occupation]`
industry	*Lname*`[industry]`
employee (observational)	*Lname*

Specifying paths for a specific group

The `group(`*varname*`)` option,

```
. gsem ..., ... group(varname)
```

specifies that the model be fit separately for the different values of *varname*. *varname* might be `sex`, and then the model would be fit separately for males and females, or *varname* might be something else and perhaps take on more than two values.

Whatever *varname* is, `group(`*varname*`)` defaults to letting some of the path coefficients, covariances, variances, and means of your model vary across the groups and constraining others to be equal. Which parameters vary and which are constrained is described in [SEM] **gsem group options**, but that is a minor detail right now.

In what follows, we will assume that *varname* is `mygrp` and takes on three values. Those values are 1, 2, and 3, but they could just as well be 2, 9, and 12.

Consider typing

```
. gsem ..., ...
```

and typing

```
. gsem ..., ... group(mygrp)
```

Whatever *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()` are that describe the model, there are now three times as many parameters because each group has its own unique set. In fact, when you give the second command, you are not merely asking for three times the parameters, you are specifying three models, one for each group! In this case, you specified the same model three times without knowing it.

You can vary the model specified across groups.

1. Let's write the model you wish to fit as

```
. gsem (a) (b) (c), cov(d) cov(e) var(f)
```

where a, b, ..., f stand for what you type. In this generic example, we have two `cov()` options just because multiple `cov()` options often occur in real models. When you type

```
. gsem (a) (b) (c), cov(d) cov(e) var(f) group(mygrp)
```

results are as if you typed

```
. gsem (1: a) (2: a) (3: a)                                  ///
       (1: b) (2: b) (3: b)                                  ///
       (1: c) (2: c) (3: c),                                 ///
                cov(1: d) cov(2: d) cov(3: d)                ///
                cov(1: e) cov(2: e) cov(3: e)                ///
                var(1: f) cov(2: f) cov(3: f)  group(mygrp)
```

The `1:`, `2:`, and `3:` identify the groups for which paths, covariances, or variances are being added, modified, or constrained.

If `mygrp` contained the unique values 5, 8, and 10 instead of 1, 2, and 3, then `5:` would appear in place of `1:`, `8:` would appear in place of `2:`, and `10:` would appear in place of `3:`.

2. Consider the model

```
. gsem (y <- x) (b) (c), cov(d) cov(e) var(f) group(mygrp)
```

The default `ginvariant()` option constrains all intercepts, coefficients, and loadings to be the same across all groups. If you wanted to constrain the path coefficient `(y <- x)` to be the same across all three groups, but let all other parameters be group specific, you could type

```
. gsem (y <- x@c1) (b) (c), cov(d) cov(e) var(f) group(mygrp) ginvariant(none)
```

This works because the expansion of `(y <- x@c1)` is

```
(1: y <- x@c1) (2: y <- x@c1) (3: y <- x@c1)
```

See item 12 in [SEM] **sem and gsem path notation** for more examples of specifying constraints.

3. Consider the model

```
. gsem (y <- x) (b) (c), cov(d) cov(e) var(f) group(mygrp)
```

If you wanted to constrain the path coefficient `(y <- x)` to be the same in groups 2 and 3, you could type

```
. gsem (1: y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),      ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The default `ginvariant()` option still constrains the other coefficients to be equal to each other, but not necessarily equal to the coefficient in groups 2 and 3. In our example, `mygrp` has 3 levels, so the above effectively removed the default constraint on the group 1 coefficient.

4. Instead of following item 3, you could type

```
. gsem (y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),        ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The part `(y <- x) (2: y <- x@c1) (3: y <- x@c1)` expands to

```
(1: y <- x) (2: y <- x) (3: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

and thus the path is defined twice for group 2 and twice for group 3. When a path is defined more than once, the definitions are combined. In this case, the second definition adds more information, so the result is as if you typed

```
(1: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

5. Instead of following item 3 or item 4, you could type

```
. gsem (y <- x@c1) (1: y <- x@c2) (b) (c),      ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The part `(y <- x@c1) (1: y <- x@c2)` expands to

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x@c2)
```

When results are combined from repeated definitions, then definitions that appear later take precedence. In this case, results are as if the expansion read

```
(1: y <- x@c2) (2: y <- x@c1)  (3: y <- x@c1)
```

Thus coefficients for groups 2 and 3 are constrained. The group-1 coefficient is constrained to `c2`. If `c2` appears nowhere else in the model specification, then results are as if the path for group 1 were unconstrained.

6. Instead of following item 3, item 4, or item 5, you could not type

```
. gsem (y <- x@c1) (1: y <- x) (b) (c),                ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The expansion of `(y <- x@c1) (1: y <- x)` reads

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x)
```

and you might think that `1: y <- x` would replace `1: y <- x@c1`. Information, however, is combined, and even though precedence is given to information appearing later, silence does not count as information. Thus the expanded and reduced specification reads the same as if `1: y <- x` was never specified:

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)
```

7. Items 1–6, stated in terms of *paths*, apply equally to what is typed inside the `means()`, `variance()`, `covariance()`, and `covstructure()` options. For instance, if you typed

```
. gsem (a) (b) (c), var(e.y@c1) group(mygrp)
```

then you are constraining the variance to be equal across all three groups.

If you wanted to constrain the variance to be equal in groups 2 and 3, you could type

```
. gsem (a) (b) (c), var(e.y) var(2: e.y@c1) var(3: e.y@c1), group(mygrp)
```

You could omit typing `var(e.y)` because it is implied. Alternatively, you could type

```
. gsem (a) (b) (c), var(e.y@c1) var(1: e.y@c2) group(mygrp)
```

You could not type

```
. gsem (a) (b) (c), var(e.y@c1) var(1: e.y) group(mygrp)
```

because silence does not count as information when specifications are combined.

Similarly, if you typed

```
. gsem (a) (b) (c), cov(e.y1*e.y2@c1) group(mygrp)
```

then you are constraining the covariance to be equal across all groups. If you wanted to constrain the covariance to be equal in groups 2 and 3, you could type

```
. gsem (a) (b) (c), cov(e.y1*e.y2)                                    ///
                    cov(2: e.y1*e.y2@c1) cov(3: e.y1*e.y2@c1) ///
                    group(mygrp)
```

You could not omit `cov(e.y1*e.y2)` because it is not assumed. By default, error variables are assumed to be uncorrelated. Omitting the option would constrain the covariance to be 0 in group 1 and to be equal in groups 2 and 3.

Alternatively, you could type

```
. gsem (a) (b) (c), cov(e.y1*e.y2@c1)                    ///
                    cov(1: e.y1*e.y2@c2)                 ///
                    group(mygrp)
```

8. In the examples above, we have referred to the groups with their numeric values, 1, 2, and 3. Had the values been 5, 8, and 10, then we would have used those values.

   If the group variable `mygrp` has a value label, you can use the label to refer to the group. For instance, imagine `mygrp` is labeled as follows:

   ```
   . label define grpvals 1 Male  2 Female  3 "Unknown sex"
   . label values mygrp grpvals
   ```

   We could type

   ```
   . gsem (y <- x) (Female: y <- x@c1) (Unknown sex: y <- x@c1) ..., ...
   ```

   or we could type

   ```
   . gsem (y <- x) (2: y <- x@c1) (3: y <- x@c1) ..., ...
   ```

Specifying paths for a specific latent class

The `lclass()` option in

```
. gsem ..., ... lclass(C 3)
```

specifies that the model be fit separately for each of 3 classes of the categorical latent variable `C`. The classes of `C` are 1, 2, and 3. `gsem` allows for more than one categorical latent variable to be specified by allowing more than one `lclass()` option. The two `lclass()` options in

```
. gsem ..., ... lclass(C 3) lclass(D 2)
```

specify that the model be fit separately for each of the 6 latent classes defined by the interaction between the categorical latent variables `C` and `D`.

Whatever the number of classes and latent variables, `gsem` defaults to letting some of the path coefficients, covariances, and variances of your model to vary across the latent classes and constraining others to be equal. Which parameters vary and which are constrained is described in [SEM] **gsem lclass options**, but that is a minor detail right now.

Consider typing

```
. gsem ..., ...
```

then typing

```
. gsem ..., ... lclass(C 3)
```

Whatever *paths*, `covariance()`, `variance()`, and `covstructure()` are that describe the model, there are now three times as many parameters because each latent class has its own unique set. In fact, when you give the second command, you are not merely asking for three times the parameters, you are specifying three models, one for each latent class, and a multinomial model for latent class probabilities.

You can vary the model specified across latent classes.

1. Let's write the model you wish to fit as

   ```
   . gsem (a) (b) (c), cov(d) cov(e) var(f)
   ```

   where a, b, ..., f stand for what you type. In this generic example, we have two `cov()` options just because multiple `cov()` options can occur in real models. When you type

   ```
   . gsem (a) (b) (c), cov(d) cov(e) var(f) lclass(C 3)
   ```

   results are as if you typed

   ```
   . gsem (1: a) (2: a) (3: a)                                  ///
          (1: b) (2: b) (3: b)                                  ///
          (1: c) (2: c) (3: c)                                  ///
          (1.C <- _cons@0)                                      ///
          (2.C <- _cons)                                        ///
          (3.C <- _cons),                                       ///
                   cov(1: d) cov(2: d) cov(3: d)                ///
                   cov(1: e) cov(2: e) cov(3: e)                ///
                   var(1: f) cov(2: f) cov(3: f)                ///
                   lclass(C 3)
   ```

   The `1:`, `2:`, and `3:` identify the latent classes for which paths, covariances, or variances are being added, modified, or constrained.

   The paths to `1.C`, `2.C`, and `3.C` identify linear predictions for the multinomial probabilities for the categorical latent variable `C`.

2. Consider the model

   ```
   . gsem (y <- x) (b) (c), cov(d) cov(e) var(f) lclass(C 3)
   ```

   If you wanted to constrain the path coefficient `(y <- x)` to be the same across all three latent classes, you could type

   ```
   . gsem (y <- x@c1) (b) (c), cov(d) cov(e) var(f) lclass(C 3)
   ```

   This works because the expansion of `(y <- x@c1)` is

   ```
   (1: y <- x@c1) (2: y <- x@c1) (3: y <- x@c1)
   ```

   See item 12 in [SEM] **sem and gsem path notation** for more examples of specifying constraints.

3. Consider the model

   ```
   . gsem (y <- x) (b) (c), cov(d) cov(e) var(f) lclass(C 3)
   ```

If you wanted to constrain the path coefficient `(y <- x)` to be the same in latent classes 2 and 3, you could type

```
. gsem (1: y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),      ///
                        cov(d) cov(e) var(f) lclass(C 3)
```

4. Instead of following item 3, you could type

```
. gsem (y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),        ///
                        cov(d) cov(e) var(f) lclass(C 3)
```

The part `(y <- x) (2: y <- x@c1) (3: y <- x@c1)` expands to

```
(1: y <- x) (2: y <- x) (3: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

and thus the path is defined twice for latent class 2 and twice for latent class 3. When a path is defined more than once, the definitions are combined. In this case, the second definition adds more information, so the result is as if you typed

```
(1: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

5. Instead of following item 3 or item 4, you could type

```
. gsem (y <- x@c1) (1: y <- x@c2) (b) (c),      ///
                        cov(d) cov(e) var(f) lclass(C 3)
```

The part `(y <- x@c1) (1: y <- x@c2)` expands to

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x@c2)
```

When results are combined from repeated definitions, then definitions that appear later take precedence. In this case, results are as if the expansion read

```
(1: y <- x@c2) (2: y <- x@c1)  (3: y <- x@c1)
```

Thus coefficients for latent classes 2 and 3 are constrained. The first latent class coefficient is constrained to `c2`. If `c2` appears nowhere else in the model specification, then results are as if the path for the first latent class were unconstrained.

6. Instead of following item 3, item 4, or item 5, you could not type

```
. gsem (y <- x@c1) (1: y <- x) (b) (c),                ///
                        cov(d) cov(e) var(f) lclass(C 3)
```

The expansion of `(y <- x@c1) (1: y <- x)` reads

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x)
```

and you might think that `1: y <- x` would replace `1: y <- x@c1`. Information, however, is combined, and even though precedence is given to information appearing later, silence does not count as information. Thus the expanded and reduced specification reads the same as if `1: y <- x` was never specified:

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)
```

7. Items 1–6, stated in terms of *paths*, apply equally to what is typed inside the `variance()`, `covariance()`, and `covstructure()` options. For instance, if you typed

```
. gsem (a) (b) (c), var(e.y@c1) lclass(C 3) lcinvariant(none)
```

then you are constraining the error variance of `y` to be equal across all three latent classes. Without the `lcinvariant(none)` option, `gsem` would constrain all error variances to be equal across all groups.

If you wanted to constrain the variance to be equal in latent classes 2 and 3, you could type

```
. gsem (a) (b) (c), var(e.y) var(2: e.y@c1) var(3: e.y@c1), lclass(C 3)
```

You could omit typing `var(e.y)` because it is implied. Alternatively, you could type

```
. gsem (a) (b) (c), var(e.y@c1) var(1: e.y@c2) lclass(C 3)
```

You could not type

```
. gsem (a) (b) (c), var(e.y@c1) var(1: e.y) lclass(C 3)
```

because silence does not count as information when specifications are combined.

Similarly, if you typed

```
. gsem (a) (b) (c), cov(e.y1*e.y2@c1) lclass(C 3)
```

then you are constraining the covariance to be equal across all latent classes. If you wanted to constrain the covariance to be equal in latent classes 2 and 3, you could type

```
. gsem (a) (b) (c), cov(e.y1*e.y2)                                  ///
                    cov(2: e.y1*e.y2@c1) cov(3: e.y1*e.y2@c1) ///
                    lclass(C 3)
```

You could not omit `cov(e.y1*e.y2)` because it is not assumed. By default, error variables are assumed to be uncorrelated. Omitting the option would constrain the covariance to be 0 in latent class 1 and to be equal in latent classes 2 and 3.

Alternatively, you could type

```
. gsem (a) (b) (c), cov(e.y1*e.y2@c1) cov(1: e.y1*e.y2@c2) lclass(C 3)
```

8. The above examples focused on models with a single categorical latent variable. The same rules apply for models with two or more categorical latent variables, with the added requirement that the latent class prefix is specified using factor-variable notation.

   For example, suppose your model is

   ```
   . gsem (a), cov(b) var(c) lclass(C 3) lclass(D 2)
   ```

   The result is as if you typed

   ```
   . gsem (1.C#1.D: a) (1.C#2.D: a)                          ///
          (2.C#1.D: a) (2.C#2.D: a)                          ///
          (3.C#1.D: a) (3.C#2.D: a)                          ///
          (1.C <- _cons@0) (2.C <- _cons) (3.C <- _cons)     ///
          (1.D <- _cons@0) (2.D <- _cons),                   ///
                   cov(1.C#1.D: b) cov(1.C#2.D: b)           ///
                   cov(2.C#1.D: b) cov(2.C#2.D: b)           ///
                   cov(3.C#1.D: b) cov(3.C#2.D: b)           ///
                   var(1.C#1.D: c) var(1.C#2.D: c)           ///
                   var(2.C#1.D: c) var(2.C#2.D: c)           ///
                   var(3.C#1.D: c) var(3.C#2.D: c)           ///
                   lclass(C 3) lclass(D 2)
   ```

   Each of the *paths*, `variance()`, `covariance()`, and `covstructure()` options is now targeted to a specific latent class.

9. From the above example, we see that the multinomial probabilities for the latent classes are modeled as if `C` and `D` were independent. For example, the linear prediction for the latent class `3.C#2.D` is composed of the intercept for `3.C` plus the intercept for `2.D`.

   Add the path `3.C#2.D <- _cons` to the above model specification, and the linear prediction for the latent class `3.C#2.D` will be composed of the intercept for `3.C`, plus the intercept for `2.D`, plus the new specified intercept for `3.C#2.D`. With this new intercept, `C` and `D` are no longer modeled independently.

10. Consider the model

```
. gsem (y <- x), lclass(C 3) lclass(D 2)
```

If you wanted to constrain the path coefficient `(y <- x)` to be the same in all latent classes with `1.C`, then add the path `(1.C: y <- x@c1)` so that the model is

```
. gsem (y <- x) (1.C: y <- x@c1), lclass(C 3) lclass(D 2)
```

This essentially expands to

```
. gsem (1.C#1.D: y <- x@c1) (1.C#2.D: y <- x@c1)         ///
       (2.C#1.D: y <- x)    (2.C#2.D: y <- x)            ///
       (3.C#1.D: y <- x)    (3.C#2.D: y <- x)            ///
       (1.C <- _cons@0) (2.C <- _cons) (3.C <- _cons)    ///
       (1.D <- _cons@0) (2.D <- _cons),                  ///
                lclass(C 3) lclass(D 2)
```

This shortcut syntax works similarly for options `covariance()`, `variance()`, and `covstructure()`.

Specifying paths for a specific group and latent class

`gsem` allows models where both option `group()` and option `lclass()` are specified. Building on the examples from the above sections, the model

```
. gsem ..., ... group(mygrp) lclass(D 2)
```

specifies that the model with categorical latent variable `D` be fit separately for each level of `mygrp`. The number of model parameters multiplies when options `group()` and `lclass()` are specified.

For example, the model

```
. gsem (y <- x), group(mygrp) lclass(D 2)
```

with `mygrp` having levels 1, 2, and 3, is equivalent to

```
. gsem (1.mygrp#1.D: y <- x) (1.mygrp#2.D: y <- x)          ///
       (2.mygrp#1.D: y <- x) (2.mygrp#2.D: y <- x)          ///
       (3.mygrp#1.D: y <- x) (3.mygrp#2.D: y <- x)          ///
       (1.mygrp: 1.D <- _cons@0) (1.mygrp: 2.D <- _cons)    ///
       (2.mygrp: 1.D <- _cons@0) (2.mygrp: 2.D <- _cons)    ///
       (3.mygrp: 1.D <- _cons@0) (3.mygrp: 2.D <- _cons),   ///
                group(mygrp) lclass(D 2)
```

The path `(y <- x)` expands to 6 paths, one for each group and latent class combination. The number of covariances and variances is similarly multiplied by 6. The linear predictions for `D` expand by the number of group levels.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem group options** — Fitting models on different groups

[SEM] **gsem lclass options** — Fitting models with latent classes

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

Title

gsem postestimation — Postestimation tools for gsem

Postestimation commands margins Remarks and examples Also see

Postestimation commands

The following are the postestimation commands that you can use after estimation by `gsem`:

Command	Description
`estat eform`	display exponentiated coefficients
`estat ic`	Akaike's and Schwarz's Bayesian information criteria (AIC and BIC)
`estat lcgof`	latent class goodness-of-fit statistic
`estat lcmean`	latent class marginal means
`estat lcprob`	latent class marginal probabilities
`estat sd`	display variance components as standard deviations and correlations
* `hausman`	Hausman's specification test
* `lrtest`	likelihood-ratio tests
`test`	Wald tests
`lincom`	linear combination of parameters
`nlcom`	nonlinear combination of parameters
`testnl`	Wald tests of nonlinear hypotheses
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`predict`	generalized linear predictions, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`contrast`	contrasts and linear hypothesis tests
`pwcompare`	pairwise comparisons
`estimates`	cataloging estimation results

* `hausman` and `lrtest` are not appropriate with `svy` estimation results.

For a summary of postestimation features, see [SEM] **Intro 7**.

Postestimation commands such `lincom` and `nlcom` require referencing estimated parameter values, which are accessible via `_b[`*name*`]`. To find out what the names are, type `sem, coeflegend`.

margins

Description for margins

`margins` estimates margins of response for expected values, probabilities, and predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
default	calculate expected values for each *depvar*
`mu`	calculate expected value of *depvar*
`pr`	calculate probability (synonym for `mu` when μ is a probability)
`eta`	calculate expected value of linear prediction of *depvar*
`expression(`*exp*`)`	calculate prediction using *exp*
`classpr`	calculate latent class probabilities
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`survival`	not allowed with `margins`
`latent`	not allowed with `margins`
`latent(`*varlist*`)`	not allowed with `margins`
`classposteriorpr`	not allowed with `margins`

`mu` defaults to the first *depvar* if option `outcome()` is not specified. If *depvar* is `family(multinomial)` or `family(ordinal)` the default is the first level of the outcome.

`pr` defaults to the first *depvar* that allows predicted probabilities if option `outcome()` is not specified. If *depvar* is `family(multinomial)` or `family(ordinal)` the default is the first level of the outcome.

`eta` defaults to the first *depvar* if option `outcome()` is not specified. If *depvar* is `family(multinomial)` the default is the first level of the outcome.

`classpr` defaults to the first latent class if option `class()` is not specified.

`predict`'s option `marginal` is assumed if `predict`'s options `conditional(fixedonly)` and `class()` are not specified; see [SEM] **predict after gsem**.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

This manual entry concerns `gsem`. For information on postestimation features available after `sem`, see [SEM] **sem postestimation**.

Also see

[SEM] **gsem reporting options** — Options affecting reporting of results

[U] 20 Estimation and postestimation commands

Title

gsem reporting options — Options affecting reporting of results

Description Syntax Options Remarks and examples Also see

Description

These options control how gsem displays estimation results.

Syntax

gsem *paths* ..., ... *reporting_options*

gsem, *reporting_options*

reporting_options	Description
level(#)	set confidence level; default is level(95)
coeflegend	display coefficient legend
nocnsreport	do not display constraints
noheader	do not display header above parameter table
nodvheader	do not display dependent variables information in the header
notable	do not display parameter table
byparm	display results in a single table with rows arranged by parameter
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

Options

level(#); see [R] **Estimation options**.

coeflegend displays the legend that reveals how to specify estimated coefficients in _b[] notation, which you are sometimes required to type in specifying postestimation commands.

nocnsreport suppresses the display of the constraints. Fixed-to-zero constraints that are automatically set by gsem are not shown in the report to keep the output manageable.

noheader suppresses the header above the parameter table, the display that reports the final log-likelihood value, number of observations, etc.

nodvheader suppresses the dependent variables information from the header above the parameter table.

notable suppresses the parameter table.

byparm specifies that estimation results with multiple groups or latent classes be reported in a single table with rows arranged by parameter. The default is to report results in separate tables for each group and latent class combination.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Remarks and examples

Any of the above options may be specified when you fit the model or when you redisplay results, which you do by specifying nothing but options after the gsem command:

```
. gsem (...) (...), ...
(original output displayed)
. gsem
(output redisplayed)
. gsem, coeflegend
(coefficient-name table displayed)
. gsem
(output redisplayed)
```

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **Example 29g** — Two-parameter logistic IRT model

Title

lincom — Linear combinations of parameters

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`lincom` is a postestimation command for use after `sem`, `gsem`, and nearly all Stata estimation commands.

`lincom` computes point estimates, standard errors, z statistics, p-values, and confidence intervals for linear combinations of the estimated parameters.

After `sem` and `gsem`, you must use the `_b[]` coefficient notation; you cannot refer to variables by using shortcuts to obtain coefficients on variables.

See [R] **lincom**.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Linear combinations of parameters

Syntax

`lincom` *exp* [, *options*]

Options

See *Options* in [R] **lincom**.

Remarks and examples

`lincom` works in the metric of SEM, which is to say path coefficients, variances, and covariances. If you want to frame your linear combinations in terms of standardized coefficients and correlations, and you fit the model with `sem`, not `gsem`, then prefix `lincom` with `estat stdize:`; see [SEM] **estat stdize**.

Stored results

See *Stored results* in [R] **lincom**.

Also see

[R] **lincom** — Linear combinations of parameters

[SEM] **estat stdize** — Test standardized parameters

[SEM] **nlcom** — Nonlinear combinations of parameters

[SEM] **test** — Wald test of linear hypotheses

Title

lrtest — Likelihood-ratio test of linear hypothesis

Description Menu Syntax Remarks and examples
Stored results Also see

Description

`lrtest` is a postestimation command for use after `sem`, `gsem`, and other Stata estimation commands.

`lrtest` performs a likelihood-ratio test comparing models. See [R] **lrtest**.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Likelihood-ratio test

Syntax

`{sem|gsem} ..., ...` (fit model 1)

`estimates store` *modelname1*

`{sem|gsem} ..., ...` (fit model 2)

`estimates store` *modelname2*

`lrtest` *modelname1* *modelname2* (compare models)

where one of the models is constrained and the other is unconstrained. `lrtest` counts parameters to determine which model is constrained and which is unconstrained, so it does not matter which model is which.

Warning concerning use with `sem`: Do not omit variables, observed or latent, from the model. Constrain their coefficients to be 0 instead. The models being compared must contain the same set of variables. This is because the standard SEM likelihood value is a function of the variables appearing in the model. Despite this fact, use of `lrtest` is appropriate under the relaxed conditional normality assumption.

Note concerning `gsem`: The above warning does not apply to `gsem` just as it does not apply to other Stata estimation commands. Whether you omit variables or constrain coefficients to 0, results will be the same. The generalized SEM likelihood is conditional on the exogenous variables.

Remarks and examples

See [SEM] **Example 10** and [SEM] **Example 39g**.

When using `lrtest` after `sem`, you must be careful that the models being compared have the same observed and latent variables. For instance, the following is allowed:

```
. sem (L1 -> x1 x2 x3) (L1 <- x4 x5) (x1 <- x4) (x2 <- x5)
. estimates store m1
. sem (L1 -> x1 x2 x3) (L1 <- x4 x5)
. estimates store m2
. lrtest m1 m2
```

This is allowed because both models contain the same variables, namely, `L1`, `x1`, . . . , `x5`, even though the second model omitted some paths.

The following would produce invalid results:

```
. sem (L1 -> x1 x2 x3) (L1 <- x4 x5) (x1 <- x4) (x2 <- x5)
. estimates store m1
. sem (L1 -> x1 x2 x3) (L1 <- x4)
. estimates store m2
. lrtest m1 m2
```

It produces invalid results because the second model does not include variable `x5` and the first model does. To run this test correctly, you type

```
. sem (L1 -> x1 x2 x3) (L1 <- x4 x5) (x1 <- x4) (x2 <- x5)
. estimates store m1
. sem (L1 -> x1 x2 x3) (L1 <- x4 x5@0)
. estimates store m2
. lrtest m1 m2
```

If we were using `gsem` rather than `sem`, all the above would still be valid.

Stored results

See *Stored results* in [R] **lrtest**.

Also see

[SEM] **Example 10** — MIMIC model

[SEM] **Example 39g** — Three-level model (multilevel, generalized response)

[R] **lrtest** — Likelihood-ratio test after estimation

[SEM] **test** — Wald test of linear hypotheses

[SEM] **estat stdize** — Test standardized parameters

[SEM] **estat eqtest** — Equation-level tests that all coefficients are zero

Title

Methods and formulas for gsem — Methods and formulas for gsem

Description Remarks and examples References Also see

Description

The methods and formulas for the `gsem` command are presented below.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Families of distributions
- *The Bernoulli family*
- *The beta family*
- *The binomial family*
- *The ordinal family*
- *The multinomial family*
- *The Poisson family*
- *The negative binomial family*
- *The Gaussian family*
- *Reliability*
- *Point mass*

Link functions
- *The logit link*
- *The probit link*
- *The complementary log-log link*
- *The log link*
- *The identity link*

Survival distributions
- *The exponential distribution*
- *The Weibull distribution*
- *The gamma distribution*
- *The loglogistic distribution*
- *The lognormal distribution*

Models with continuous latent variables
- *The likelihood*
- *Gauss–Hermite quadrature*
- *Adaptive quadrature*
- *Laplacian approximation*
- *Survey data*
- *Predictions*
  - *Empirical Bayes*
  - *Other predictions*

Models with categorical latent variables
- *The likelihood*
- *The EM algorithm*
- *Survey data*
- *Predictions*

Introduction

gsem fits generalized linear models with categorical or continuous latent variables via maximum likelihood. Here is a table identifying the family/link combinations that gsem allows.

	logit	probit	cloglog	log	identity
Bernoulli	x	x	x		
beta	x	x	x		
binomial	x	x	x		
ordinal	x	x	x		
multinomial	x				
Poisson				x	
negative binomial				x	
Gaussian				x	x
exponential				x	
Weibull				x	
gamma				x	
loglogistic				x	
lognormal				x	
pointmass					x

Log-likelihood calculations for fitting any model with continuous latent variables require integrating out the latent variables. One widely used modern method is to directly estimate the integral required to calculate the log likelihood by Gauss–Hermite quadrature or some variation thereof. gsem implements four different methods for numerically evaluating the integrals.

1. Gauss–Hermite quadrature (GHQ)
2. Mean-variance adaptive quadrature (MVAGH)
3. Mode-curvature adaptive quadrature (MCAGH)
4. Laplacian approximation

The default method is MVAGH. The numerical integration method for MVAGH is based on Rabe-Hesketh, Skrondal, and Pickles (2005), and the other numerical integration methods described in this manual entry are based on Skrondal and Rabe-Hesketh (2004, chap. 6.3).

Families of distributions

gsem implements the most commonly used distribution families associated with generalized linear models. gsem also implements distributions for ordinal and multinomial outcomes.

In this manual entry, observed endogenous variables are also known as generalized responses or generalized outcomes, but we will simply refer to them as responses or outcomes. The random variable corresponding to a given response will be denoted by Y. An observed value of Y will be denoted by y, and the expected value of Y by μ. For the ordinal and multinomial families, we will refer to a linear prediction, denoted by z, instead of the expected value.

The Bernoulli family

The Bernoulli family is a binary response model. The response Y is assumed to take on the values 0 or 1; however, `gsem` allows any nonzero and nonmissing value to mean 1.

The log of the conditional probability mass function is

$$\log f(y|\mu) = y\log\mu + (1-y)\log(1-\mu)$$

where μ is also known as the probability of a success. The default link for the Bernoulli family is the `logit` link.

The beta family

The beta family is a fractional response model. The response Y is assumed to take on the real values between 0 and 1.

The log of the conditional probability density function is

$$\begin{aligned}\log f(y|\mu,s) = &\log\{\Gamma(s)\} - \log\{\Gamma(\mu s)\} - \log\{\Gamma(s-\mu s)\}\\ &+ (\mu s-1)\log y + (s-\mu s-1)\log(1-y)\end{aligned}$$

where μ is the mean of Y, and s is the scale parameter. `gsem` fits s in the log scale. The default link for the beta family is the `logit` link.

The binomial family

The binomial family is a count response model and generalizes the Bernoulli family by taking the sum of k independent Bernoulli outcomes. The response Y is assumed to take on the values $0, 1, \ldots, k$.

The log of the conditional probability mass function is

$$\begin{aligned}\log f(y|\mu) = &\log\{\Gamma(k+1)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(k-y+1)\}\\ &+ y\log\mu + (1-y)\log(1-\mu)\end{aligned}$$

where μ is the expected value for a single Bernoulli outcome. The default link for the binomial family is the `logit` link.

The ordinal family

The ordinal family is a discrete response model. The response Y is assumed to take on one of k unique values. The actual values are irrelevant except that higher values are assumed to correspond to "higher" outcomes. Without loss of generality, we will assume that Y takes on the values $1, \ldots, k$. The ordinal family with k outcomes has cutpoints $\kappa_0, \kappa_1, \ldots, \kappa_k$, where $\kappa_0 = -\infty$, $\kappa_y < \kappa_{y+1}$, and $\kappa_k = +\infty$.

Given a linear prediction z, the probability that a random response Y takes the value y is

$$\Pr(Y = y|z) = \Pr(Y^* < \kappa_y - z) - \Pr(Y^* < \kappa_{y-1} - z)$$

where Y^* is the underlying stochastic component for Y. The distribution for Y^* is determined by the link function. `gsem` allows `logit`, `probit`, and `cloglog` for the ordinal family. The `logit` link assigns Y^* the extreme value distribution that is synonymous with the logit link for Bernoulli outcomes. The `probit` link assigns Y^* the standard normal distribution that is synonymous with the probit link for Bernoulli outcomes. The `cloglog` link assigns Y^* the distribution that is synonymous with the complementary log-log link for Bernoulli outcomes. The default link for the ordinal family is the `logit` link.

The multinomial family

The multinomial family is a discrete response model. The response Y is assumed to take on one of k unique values. The actual values are irrelevant and order does not matter; however, `gsem` requires that the values are nonnegative integers. Without loss of generality, we will assume that Y takes on the values $1, \ldots, k$. Each of the k outcomes has its own linear prediction. For the model to be identified, one of the outcomes is chosen to be the base or reference. The linear prediction for the base outcome is constrained to be 0 for all observations. Without loss of generality, we will assume the base outcome is the first outcome. Let z_i be the prediction for outcome i, where $z_1 = 0$ for the base outcome.

Given the k linear predictions $\boldsymbol{z}' = (z_1, z_2, \ldots, z_k)$, the log of the conditional probability mass function is

$$\log f(y|\boldsymbol{z}) = z_y - \log\left\{\sum_{i=1}^{k} \exp(z_i)\right\}$$

The only link allowed for the multinomial family is the `logit` link.

The Poisson family

The Poisson family is a count-data response model. The response Y is assumed to take on nonnegative integer values.

The log of the conditional probability mass function is

$$\log f(y|\mu) = -\mu + y\log\mu - \log\Gamma(y+1)$$

The only link allowed for the Poisson family is the `log` link.

The negative binomial family

The negative binomial family is another count-data response model. It is commonly thought of as a Poisson family with overdispersion. `gsem` allows two parameterizations for the dispersion in this family: mean dispersion and constant dispersion.

The log of the conditional probability mass function is

$$\begin{aligned}\log f(y|\mu,\alpha) = &\log\{\Gamma(y+m)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(m)\} \\ &+ m\log p + y\log(1-p)\end{aligned}$$

where m and p depend on the form of dispersion.

The only link allowed for the negative binomial family is the `log` link.

For mean dispersion, we have

$$m = 1/\alpha$$
$$p = \frac{1}{1+\alpha\mu}$$

where μ is the expected value of Y and α is the scale parameter. `gsem` fits α in the log scale.

For constant dispersion, we have

$$m = \exp(\log\mu - \log\delta)$$
$$p = \frac{1}{1+\delta}$$

where μ is the expected value of Y and δ is the scale parameter. `gsem` fits δ in the log scale.

The Gaussian family

The Gaussian family is a continuous response model and is synonymous with the normal distribution.

When the Gaussian family is specified with the `identity` link but no censoring, `gsem` fits this family by using a single multivariate density function and allows the following two special features:

1. `gsem` can fit covariances between the Gaussian error variables.
2. `gsem` can fit paths between Gaussian responses, including nonrecursive systems.

The log of the conditional probability density function is

$$\log f(\boldsymbol{y}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{1}{2}\left\{d\log 2\pi + \log|\boldsymbol{\Sigma}| + (\boldsymbol{y}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right\}$$

where d is the dimension of the observed response vector $\boldsymbol{y}$, $\boldsymbol{\mu}$ is the mean of the responses, and $\boldsymbol{\Sigma}$ is the variance matrix of their unexplained errors.

When the Gaussian family is specified with the `log` link or censoring, the two special features described above no longer apply. In addition, the multivariate density function is no longer used. Instead, for each response using the `log` link, the log of the conditional probability density function corresponds to the formula above with $d = 1$. For censored responses, the log likelihood corresponds to the one in the *Methods and formulas* for [R] **intreg**.

Reliability

For a given Gaussian response variable with the identity link, the reliability Y may be specified as p or $100 \times p\%$. The variance of Y's associated error variable is then constrained to $(1-p)$ times the observed variance of Y.

Point mass

gsem, with categorical latent variables, allows an outcome to have a unit probability mass at a single integer value. This feature is useful for fitting models like the zero-inflated Poisson model.

Link functions

Except for the ordinal and multinomial families, the link function defines the transformation between the mean and the linear prediction for a given response. If Y is the random variable corresponding to an observed response variable y, then the link function performs the transformation

$$g(\mu) = z$$

where $\mu = E(Y)$ and z is the linear prediction. In practice, the likelihood evaluator function uses the inverse of the link function to map the linear prediction to the mean.

The logit link

The logit link is

$$g(\mu) = \log\mu - \log(1-\mu)$$

and its inverse is

$$\mu = g^{-1}(z) = \frac{1}{1+e^z}$$

The probit link

The probit link is

$$g(\mu) = \Phi^{-1}(\mu)$$

and its inverse is

$$\mu = g^{-1}(z) = \Phi(z)$$

where $\Phi(\cdot)$ is the cumulative distribution function for the standard normal distribution and $\Phi^{-1}(\cdot)$ is its inverse.

The complementary log-log link

The complementary log-log link is

$$g(\mu) = \log\{-\log(1-\mu)\}$$

and its inverse is

$$\mu = g^{-1}(z) = 1 - \exp\{-\exp(z)\}$$

The log link

The log link is

$$g(\mu) = \log\mu$$

and its inverse is

$$\mu = g^{-1}(z) = e^z$$

The identity link

The identity link is $g(\mu) = \mu$.

Survival distributions

`gsem` also implements the most commonly used parametric distributions associated with survival analysis. This includes support for time-varying covariates and right-censoring. See [ST] **streg** for more background information on parametric survival models.

In this section, the observed endogenous variables are survival times. The random variable corresponding to a given response will continue to be denoted by Y, with observed value denoted by y and expected value denoted by μ. The response Y is assumed to be a nonnegative real value. The starting time under observation is denoted by y_0, where $y_0 \geq 0$. The indicator for failure is denoted by d, where $d = 0$ indicates a right-censored survival time, and $d = 1$ indicates failure at observed time y.

The log likelihood for these models is given by

$$l(y, y_0, d) = d\log f(y) + (1 - d)\log S(y) - \log S(y_0)$$

where $S(\cdot)$ is the survivor function for Y, and $f(\cdot)$ is the corresponding density function. The relationship between the survivor function and the density is

$$f(y) = -\frac{\partial S(y)}{\partial y}$$

In the following, we provide formulas for the survivor functions with respect to a linear prediction, denoted by z.

The `log` link is the only link allowed with the survival distributions. Except for the gamma distribution, the use of the `log` link is merely a nod to the fact that the expected value of the outcome involves exponentiating the linear prediction. A formula for the expected value of Y, as a function of the linear prediction, is provided for each survival distribution.

The exponential distribution

The survivor function for the exponential distribution is

$$S(y) = \exp(-\lambda y)$$

where λ is a function of z and is determined by one of the two parameterizations implemented in `gsem`. The proportional hazards parameterization is

$$\lambda = \exp(z)$$

and the accelerated failure-time parameterization is

$$\lambda = \exp(-z)$$

The expected value of Y is

$$\mu = \frac{1}{\lambda}$$

The Weibull distribution

The survivor function for the Weibull distribution is

$$S(y) = \exp(-\lambda y^s)$$

where s is a scale parameter, and λ is a function of z that is determined by one of the two parameterizations implemented in `gsem`. `gsem` fits s in the log scale. The proportional hazards parameterization is

$$\lambda = \exp(z)$$

with expected value

$$\mu = \Gamma(1 + 1/s)\exp(-z/s)$$

and the accelerated failure-time parameterization is

$$\lambda = \exp(-sz)$$

with expected value

$$\mu = \Gamma(1 + 1/s)\exp(z)$$

The gamma distribution

The survivor function for the gamma distribution is

$$S(y) = \int_y^\infty \frac{1}{\Gamma(s^{-2})\left\{e^z s^2\right\}^{s^{-2}}} t^{s^{-2}-1} \exp\left(-\frac{t}{e^z s^2}\right) \partial t$$

where s is a scale parameter. `gsem` fits s in the log scale.

The expected value for this distribution is

$$\mu = \exp(z)$$

The loglogistic distribution

The survivor function for the loglogistic distribution is

$$S(y) = \left\{1 + \exp\left(\frac{\log y - z}{s}\right)\right\}^{-1}$$

where s is a scale parameter. `gsem` fits s in the log scale.

The expected value for this distribution is

$$\mu = \frac{\pi s \exp(z)}{\sin(\pi s)}$$

The lognormal distribution

The survivor function for the lognormal distribution is

$$S(y) = 1 - \Phi\left(\frac{\log y - z}{s}\right)$$

where s is a scale parameter. `gsem` fits s in the log scale.

The expected value for this distribution is

$$\mu = \exp\left(z + \frac{1}{2}s^2\right)$$

Models with continuous latent variables

The likelihood

gsem fits generalized linear models with continuous latent variables via maximum likelihood. The likelihood for the specified model is derived under the assumption that each response variable is independent and identically distributed across the estimation sample. The response variables are also assumed to be independent of each other. These assumptions are conditional on the latent variables and the observed exogenous variables.

The likelihood is computed by integrating out the latent variables. Let $\boldsymbol{\theta}$ be the vector of model parameters, $\boldsymbol{y}$ be the vector of observed response variables, $\boldsymbol{x}$ be the vector of observed exogenous variables, and $\boldsymbol{u}$ be the $r \times 1$ vector of latent variables. Then, the marginal likelihood looks something like

$$\mathcal{L}(\boldsymbol{\theta}) = \int_{\Re^r} f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta})\phi(\boldsymbol{u}|\boldsymbol{\mu}_u,\boldsymbol{\Sigma}_u)\partial\boldsymbol{u}$$

where $\Re$ denotes the set of values on the real line, $\Re^r$ is the analog in r-dimensional space, $\boldsymbol{\theta}$ is a vector of the unique model parameters, $f(\cdot)$ is the conditional probability density function for the observed response variables, $\phi(\cdot)$ is the multivariate normal density for $\boldsymbol{u}$, $\boldsymbol{\mu}_u$ is the expected value of $\boldsymbol{u}$, and $\boldsymbol{\Sigma}_u$ is the covariance matrix for $\boldsymbol{u}$. All auxiliary parameters are fit directly without any further parameterization, so we simply acknowledge that the auxiliary parameters are among the elements of $\boldsymbol{\theta}$.

The $\boldsymbol{y}$ variables are assumed to be independent, conditionally on $\boldsymbol{x}$ and $\boldsymbol{u}$, so $f(\cdot)$ is the product of the individual conditional densities. One exception to this is when $\boldsymbol{y}$ contains two or more Gaussian response variables with the identity link, in which case the Gaussian responses are actually modeled using a multivariate normal density to allow for correlated errors and nonrecursive systems among Gaussian responses. This one exception does not change how the integral is numerically evaluated, so we make no effort to represent this distinction in the formulas.

For a single-level model with n response variables, the conditional joint density function for a given observation is

$$f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) = \prod_{i=1}^{n} f_i(y_i|\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta})$$

For a two-level model, the likelihood is computed at the cluster level, so the conditional density is also a product of the observation-level density contributions within a given cluster,

$$f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) = \prod_{i=1}^{n}\prod_{j=1}^{t} f_i(y_{ij}|\boldsymbol{x}_j,\boldsymbol{u},\boldsymbol{\theta})$$

where t is the number of individuals within the cluster. This extends to more levels by expanding the products down to the observations nested within the hierarchical groups. Because the single-level model is a special case of a two-level model where all the groups have a single observation, we will now use the two-level notation and subscripts.

Except for the ordinal and multinomial families, we use the link function to map the conditional mean

$$\mu_{ij} = E(y_{ij}|\boldsymbol{x}_j,\boldsymbol{u})$$

to the linear prediction

$$z_{ij} = \boldsymbol{x}_j'\boldsymbol{\beta}_i + \boldsymbol{x}_j'\boldsymbol{\Lambda}_i\boldsymbol{u}$$

where $\boldsymbol{\beta}_i$ is the vector of the fixed-effect coefficients and $\boldsymbol{\Lambda}_i$ is the matrix of the latent loadings for y_{ij}. For notational convenience, we will overload the definitions of $f(\cdot)$ and $f_i(\cdot)$ so that they are functions of the responses and model parameters through the linear predictions $\boldsymbol{z}' = (z_1, \ldots, z_n)$. Thus $f(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta})$ is equivalently specified as $f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta})$, and $f_i(y_{ij}|\boldsymbol{x}_j, \boldsymbol{u}, \boldsymbol{\theta})$ is equivalently specified as $f_i(y_{ij}, z_{ij}, \boldsymbol{\theta})$. In this new notation, the likelihood for a given cluster is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \int_{\Re^r} f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta})\phi(\boldsymbol{u}|\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u)\partial\boldsymbol{u} \\ &= \frac{1}{(2\pi)^{r/2}\sqrt{|\boldsymbol{\Sigma}_u|}} \int_{\Re^r} \exp\left\{\log f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta}) - \frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu}_u)'\boldsymbol{\Sigma}_u^{-1}(\boldsymbol{u} - \boldsymbol{\mu}_u)\right\}\partial\boldsymbol{u}\end{aligned} \tag{2}$$

`gsem` allows nonrecursive systems between Gaussian response variables with the identity link, but non-Gaussian responses and Gaussian responses with the log link are not allowed to participate in any nonrecursive systems. This means that if a given response y is specified with a family other than Gaussian or a link other than identity, then y cannot have a path that ultimately leads back to itself. Any response may participate in a recursive system because the participating responses may be treated as exogenous variables when predicting other responses in a recursive system.

The latent vector $\boldsymbol{u}$ consists of stacked collections of the latent variables from each level. Within each level, the latent endogenous variables $\boldsymbol{\eta}$ are stacked over the latent exogenous variables $\boldsymbol{\xi}$. Within a given level, the latent exogenous variables and latent endogenous errors are assumed independent and multivariate normal,

$$\begin{aligned}\boldsymbol{\xi} &\sim N(\boldsymbol{\kappa}, \boldsymbol{\Phi}) \\ \boldsymbol{\epsilon} &\sim N(\mathbf{0}, \boldsymbol{\Psi})\end{aligned}$$

so according to the linear relationship

$$\boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \mathbf{A}\boldsymbol{x} + \boldsymbol{\epsilon}$$

we have that the latent variables are jointly multivariate normal. This linear relationship implies that `gsem` allows latent variables to predict each other but only within level. It also means that `gsem` allows paths from observed variables to latent variables; however, the observed variable must be constant within group if the path is to a group-level latent variable.

For our two-level model, we have

$$\boldsymbol{u} \sim N(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u)$$

where

$$\boldsymbol{\mu}_u = \begin{pmatrix}\boldsymbol{\mu}_\eta \\ \boldsymbol{\kappa}\end{pmatrix} \qquad \boldsymbol{\Sigma}_u = \begin{pmatrix}\boldsymbol{\Sigma}_{\eta\eta} & \boldsymbol{\Sigma}_{\eta\xi} \\ \boldsymbol{\Sigma}_{\xi\eta} & \boldsymbol{\Phi}\end{pmatrix}$$

$$\begin{aligned}\boldsymbol{\mu}_\eta &= (\mathbf{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma}\boldsymbol{\kappa} + \mathbf{A}\boldsymbol{x}) \\ \boldsymbol{\Sigma}_{\eta\eta} &= (\mathbf{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}' + \boldsymbol{\Psi})\left\{(\mathbf{I} - \mathbf{B})^{-1}\right\}' \\ \boldsymbol{\Sigma}_{\eta\xi} &= (\mathbf{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}\end{aligned}$$

The vector $\boldsymbol{\theta}$ is therefore the set of unique model parameters taken from the following:

$\boldsymbol{\beta}_i$ is the vector of fixed-effect coefficients for y_{ij}.

$\boldsymbol{\Lambda}_i$ is the matrix of latent loadings for y_{ij}.

$\mathbf{B}$ is the matrix of latent endogenous coefficients.

$\boldsymbol{\Gamma}$ is the matrix of latent exogenous coefficients.

$\mathbf{A}$ is the matrix of latent fixed-effect coefficients.

$\boldsymbol{\kappa}$ is the vector of latent exogenous means.

$\boldsymbol{\Phi}$ is the matrix of latent exogenous variances and covariances.

$\boldsymbol{\Psi}$ is the matrix of latent endogenous error variances and covariances.

Auxiliary parameters are the result from some of the distribution families.

Each level of a multilevel model will have its own set of the following parameters: $\mathbf{B}$, $\boldsymbol{\Gamma}$, $\mathbf{A}$, $\boldsymbol{\kappa}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\Psi}$. For multilevel models, $\boldsymbol{\Sigma}_u$ is a block-diagonal matrix with a block for each level.

Gauss–Hermite quadrature

The integral in (2) is generally not tractable, so we must use numerical methods. In the univariate case, the integral of a function multiplied by the kernel of the standard normal distribution can be approximated using Gauss–Hermite quadrature (GHQ). For q-point GHQ, let the abscissa and weight pairs be denoted by (a_k^*, w_k^*), $k = 1, \ldots, q$. The GHQ approximation is then

$$\int_{-\infty}^{\infty} f(x) \exp(-x^2)\, dx \approx \sum_{k=1}^{q} w_k^* f(a_k^*)$$

Using the standard normal distribution yields the approximation

$$\int_{-\infty}^{\infty} f(x)\phi(x)\, dx \approx \sum_{k=1}^{q} w_k f(a_k)$$

where $a_k = \sqrt{2}a_k^*$ and $w_k = (w_k^*)/\sqrt{\pi}$.

We can use a change-of-variables technique to transform the multivariate integral (2) into a set of nested univariate integrals. Each univariate integral can then be evaluated using GHQ. Let $\boldsymbol{v}$ be a random vector whose elements are independently standard normal, and let $\mathbf{L}$ be the Cholesky decomposition of $\boldsymbol{\Sigma}_u$; that is, $\boldsymbol{\Sigma}_u = \mathbf{L}\mathbf{L}'$. In the distribution, we have that $\boldsymbol{u} = \boldsymbol{\mu}_u + \mathbf{L}\boldsymbol{v}$, and the linear predictions vector as a function of $\boldsymbol{v}$ is

$$z_{ij} = \boldsymbol{x}_j'\boldsymbol{\beta}_i + \boldsymbol{x}_j'\boldsymbol{\Lambda}_i(\boldsymbol{\mu}_u + \mathbf{L}\boldsymbol{v})$$

so the likelihood for a given cluster is

$$\mathcal{L}(\boldsymbol{\theta}) = (2\pi)^{-r/2} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp\left\{ \log f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta}) - \frac{1}{2}\sum_{k=1}^{r} v_k^2 \right\} dv_1 \ldots dv_r \tag{3}$$

where r is the number of latent variables.

Consider an r-dimensional quadrature grid containing q quadrature points in each dimension. Let the vector of abscissas $\boldsymbol{a}_{\mathbf{k}} = (a_{k_1}, \ldots, a_{k_r})'$ be a point in this grid, and let $\boldsymbol{w}_{\mathbf{k}} = (w_{k_1}, \ldots, w_{k_r})'$ be the vector of corresponding weights. The GHQ approximation to the likelihood for a given cluster is

$$\mathcal{L}^{\text{GHQ}}(\boldsymbol{\theta}) = \sum_{k_1=1}^{q} \cdots \sum_{k_r=1}^{q} \left[\exp\left\{ \sum_{i=1}^{n} \log f_i(y_{ij}, z_{ij\mathbf{k}}, \boldsymbol{\theta}) \right\} \prod_{s=1}^{r} w_{k_s} \right]$$

where

$$z_{ij\mathbf{k}} = \boldsymbol{x}_j'\boldsymbol{\beta} + \boldsymbol{x}_j'\boldsymbol{\Lambda}_i(\boldsymbol{\mu}_u + \mathbf{L}\boldsymbol{a}_{\mathbf{k}})$$

Adaptive quadrature

This section sets the stage for mean–variance adaptive Gauss–Hermite quadrature (MVAGH) and mode-curvature adaptive Gauss–Hermite quadrature (MCAGH).

Let's reconsider the likelihood in (3). If we fix the observed variables and the model parameters, we see that the posterior density for $\boldsymbol{v}$ is proportional to

$$\phi(\boldsymbol{v}) f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta})$$

It is reasonable to assume that this posterior density can be approximated by a multivariate normal density with mean vector $\boldsymbol{\mu}_v$ and variance matrix $\boldsymbol{\tau}_v$. Instead of using the prior density of $\boldsymbol{v}$ as the weighting distribution in the integral, we can use our approximation for the posterior density,

$$\mathcal{L}(\boldsymbol{\theta}) = \int_{\Re^r} \frac{f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta})\phi(\boldsymbol{v})}{\phi(\boldsymbol{v}, \boldsymbol{\mu}_v, \boldsymbol{\tau}_v)} \phi(\boldsymbol{v}, \boldsymbol{\mu}_v, \boldsymbol{\tau}_v)\, d\boldsymbol{v}$$

The likelihood is then approximated with

$$\mathcal{L}^{*}(\boldsymbol{\theta}) = \sum_{k_1=1}^{q} \cdots \sum_{k_r=1}^{q} \left[\exp\left\{ \sum_{i=1}^{n} \log f_i(y_{ij}, z^{*}_{ij\mathbf{k}}, \boldsymbol{\theta}) \right\} \prod_{s=1}^{r} \omega_{k_s} \right]$$

where

$$z^{*}_{ij\mathbf{k}} = \boldsymbol{x}_j'\boldsymbol{\beta} + \boldsymbol{x}_j'\boldsymbol{\Lambda}_i(\boldsymbol{\mu}_u + \mathbf{L}\boldsymbol{\alpha}_{\mathbf{k}})$$

and $\boldsymbol{\alpha}_{\mathbf{k}}$ and the ω_{k_s} are the adaptive versions of the abscissas and weights after an orthogonalizing transformation, which eliminates posterior covariances between the latent variables. $\boldsymbol{\alpha}_{\mathbf{k}}$ and the ω_{k_s} are functions of $\boldsymbol{a}_{\mathbf{k}}$ and $\boldsymbol{w}_{\mathbf{k}}$ and the adaptive parameters $\boldsymbol{\mu}_v$ and $\boldsymbol{\tau}_v$.

For MVAGH, $\boldsymbol{\mu}_v$ is the posterior mean and $\boldsymbol{\tau}_v$ is the posterior variance of $\boldsymbol{v}$. They are computed iteratively by updating the posterior moments by using the MVAGH approximation, starting with a 0 mean vector and identity variance matrix.

For MCAGH, $\boldsymbol{\mu}_v$ is the posterior mode for $\boldsymbol{v}$, and $\boldsymbol{\tau}_v$ is the curvature at the mode. They are computed by optimizing the integrand in (3) with respect to $\boldsymbol{v}$.

Laplacian approximation

Let's reconsider the likelihood in (2) and denote the argument in the exponential function by

$$\begin{aligned} h(\boldsymbol{u}) &= \log f(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\theta}) - \frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu}_u)'\boldsymbol{\Sigma}_u^{-1}(\boldsymbol{u}-\boldsymbol{\mu}_u) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{t} \log f_i(y_{ij}, z_{ij}, \boldsymbol{\theta}) - \frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu}_u)'\boldsymbol{\Sigma}_u^{-1}(\boldsymbol{u}-\boldsymbol{\mu}_u) \end{aligned}$$

where

$$z_{ij} = \boldsymbol{x}_j'\boldsymbol{\beta}_i + \boldsymbol{x}_j'\boldsymbol{\Lambda}_i\boldsymbol{u}$$

The Laplacian approximation is based on a second-order Taylor expansion of $h(\boldsymbol{u})$ about the value of $\boldsymbol{u}$ that maximizes it. The first and second partials with respect to $\boldsymbol{u}$ are

$$h'(\boldsymbol{u}) = \frac{\partial h(\boldsymbol{u})}{\partial \boldsymbol{u}} = \sum_{i=1}^{n}\sum_{j=1}^{t} \frac{\partial \log f_i(y_{ij}, z_{ij}, \boldsymbol{\theta})}{\partial z_{ij}} \boldsymbol{\Lambda}_i'\boldsymbol{x}_j - \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}-\boldsymbol{\mu}_u)$$

$$\mathbf{H}(\boldsymbol{u}) = \frac{\partial^2 h(\boldsymbol{u})}{\partial \boldsymbol{u}\partial \boldsymbol{u}'} = \sum_{i=1}^{n}\sum_{j=1}^{t} \boldsymbol{x}_j'\boldsymbol{\Lambda}_i \frac{\partial^2 \log f_i(y_{ij}, z_{ij}, \boldsymbol{\theta})}{\partial z_{ij}\partial z_{ij}} \boldsymbol{\Lambda}_i'\boldsymbol{x}_j - \boldsymbol{\Sigma}^{-1}$$

The maximizer of $h(\boldsymbol{u})$ is $\widehat{\boldsymbol{u}}$ such that $h'(\widehat{\boldsymbol{u}}) = \mathbf{0}$. The integral in (2) is proportional to the posterior density of $\boldsymbol{u}$ given the data, so $\widehat{\boldsymbol{u}}$ is also the posterior mode.

The second-order Taylor approximation then takes the form

$$h(\boldsymbol{u}) \approx h(\widehat{\boldsymbol{u}}) + \frac{1}{2}(\boldsymbol{u}-\widehat{\boldsymbol{u}})'\mathbf{H}(\widehat{\boldsymbol{u}})(\boldsymbol{u}-\widehat{\boldsymbol{u}}) \tag{4}$$

because the first-order derivative term is 0. The integral is approximated by

$$\int_{\Re^r} \exp\{h(\boldsymbol{u})\}\, d\boldsymbol{u} \approx \exp\{h(\widehat{\boldsymbol{u}})\}(2\pi)^{r/2}\, |-\mathbf{H}(\widehat{\boldsymbol{u}})|^{-1/2}$$

because the second term in (4) is the kernel of a multivariate normal density once it is exponentiated. The Laplacian approximation for the log likelihood is

$$\log \mathcal{L}^{\text{Lap}}(\boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_u| - \frac{1}{2}\log|-\mathbf{H}(\widehat{\boldsymbol{u}})| + h(\widehat{\boldsymbol{u}})$$

Survey data

In the presence of sampling weights, following Rabe-Hesketh and Skrondal (2006), the weighted log pseudolikelihood for a two-level model is given as

$$\log\mathcal{L}(\boldsymbol{\theta}) = \sum_{j=1}^{M} w_j \log \int_{-\infty}^{\infty} \exp\left\{\sum_{i=1}^{n_j} w_{i|j} \log f(y_{ij}|\eta_{ij})\right\} \phi(\mathbf{v}_{j1})\, d\mathbf{v}_{j1}$$

where w_j is the inverse of the probability of selection for the jth cluster; $w_{i|j}$ is the inverse of the conditional probability of selection of individual i, given the selection of cluster j; $f(\cdot)$ is as defined previously; and $\phi(\cdot)$ is the standard multivariate normal density.

Weighted estimation is achieved through the direct application of w_j and $w_{i|j}$ into the likelihood calculations as detailed above to reflect replicated clusters for w_j and replicated observations within clusters for $w_{i|j}$. Because this estimation is based on replicated clusters and observations, frequency weights are handled similarly.

Weights are not allowed with crossed models or the Laplacian approximation.

Predictions

We begin by considering the prediction of the latent variables $\boldsymbol{u}$ for a given cluster in a two-level model. Prediction of latent variables in multilevel generalized linear models involves assigning values to the latent variables, and there are many methods for doing so; see Skrondal and Rabe-Hesketh (2009) and Skrondal and Rabe-Hesketh (2004, chap. 7) for a comprehensive review. Stata offers two methods of predicting latent variables: empirical Bayes means (also known as posterior means) and empirical Bayes modes (also known as posterior modes). Below we provide more details about the two methods.

Empirical Bayes

Let $\widehat{\boldsymbol{\theta}}$ denote the estimated model parameters. Empirical Bayes (EB) predictors of the latent variables are the means or modes of the empirical posterior distribution with the parameter estimates $\boldsymbol{\theta}$ replaced with their estimates $\widehat{\boldsymbol{\theta}}$. The method is called "empirical" because $\widehat{\boldsymbol{\theta}}$ is treated as known. EB combines the prior information about the latent variables with the likelihood to obtain the conditional posterior distribution of latent variables. Using Bayes's theorem, the empirical conditional posterior distribution of the latent variables for a given cluster is

$$\begin{aligned}\omega(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{x};\widehat{\boldsymbol{\theta}}) &= \frac{f(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\,\phi(\boldsymbol{u};\widehat{\boldsymbol{\mu}}_u,\widehat{\boldsymbol{\Sigma}}_u)}{\int f(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\,\phi(\boldsymbol{u};\widehat{\boldsymbol{\mu}}_u,\widehat{\boldsymbol{\Sigma}}_u)\,d\boldsymbol{u}} \\ &= \frac{f(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\,\phi(\boldsymbol{u};\widehat{\boldsymbol{\mu}}_u,\widehat{\boldsymbol{\Sigma}}_u)}{\mathcal{L}(\widehat{\boldsymbol{\theta}})}\end{aligned}$$

The denominator is just the likelihood contribution of the given cluster.

EB mean predictions of latent variables, $\widetilde{\boldsymbol{u}}$, also known as posterior means, are calculated as

$$\widetilde{\boldsymbol{u}} = \int_{\Re^r} \boldsymbol{u}\,\omega(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\,d\boldsymbol{u}$$

where we use the notation $\widetilde{\boldsymbol{u}}$ rather than $\widehat{\boldsymbol{u}}$ to distinguish predicted values from estimates. This multivariate integral is approximated by MVAGH. If you have multiple latent variables within a level or latent variables across levels, the calculation involves orthogonalizing transformations with the Cholesky transformation because the latent variables are no longer independent under the posterior distribution.

When all the response variables are normal, the posterior density is multivariate normal, and EB means are also best linear unbiased predictors (BLUPs); see Skrondal and Rabe-Hesketh (2004, 227). In generalized mixed-effects models, the posterior density tends to multivariate normal as cluster size increases.

EB modal predictions can be approximated by solving for $\widetilde{\widetilde{\boldsymbol{u}}}$ such that

$$\frac{\partial}{\partial \boldsymbol{u}} \log\omega(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\Big|_{\boldsymbol{u}=\widetilde{\widetilde{\boldsymbol{u}}}} = \boldsymbol{0}$$

Because the denominator in $\omega(\cdot)$ does not depend on $\boldsymbol{u}$, we can omit it from the calculation to obtain the EB mode. The calculation of EB modes does not require numerical integration; thus they are often used in place of EB means. As the posterior density gets closer to being multivariate normal, EB modes get closer and closer to EB means.

Just like there are many methods of assigning values to the random effects, there are many methods of calculating standard errors of the predicted random effects; see Skrondal and Rabe-Hesketh (2009) for a comprehensive review.

Stata uses the posterior standard deviation as the standard error of the posterior means predictor of random effects. For a given level, the EB posterior covariance matrix of the random effects is given by

$$\mathrm{Cov}(\widetilde{\boldsymbol{u}}|\boldsymbol{y},\boldsymbol{x};\widehat{\boldsymbol{\theta}}) = \int_{\Re^r} (\boldsymbol{u}-\widetilde{\boldsymbol{u}})(\boldsymbol{u}-\widetilde{\boldsymbol{u}})'\,\omega(\boldsymbol{u}|\boldsymbol{y},\boldsymbol{x};\widehat{\boldsymbol{\theta}})\,d\boldsymbol{u}$$

The posterior covariance matrix and the integrals are approximated by MVAGH.

Conditional standard errors for the estimated posterior modes are derived from standard theory of maximum likelihood, which dictates that the asymptotic variance matrix of $\widetilde{\widetilde{\boldsymbol{u}}}$ is the negative inverse of the Hessian matrix.

Other predictions

In what follows, we show formulas with the posterior means estimates of random effects $\widetilde{\boldsymbol{u}}$, which are used by default or if the `conditional(ebmeans)` option is specified. If the `conditional(ebmodes)` option is specified, $\widetilde{\boldsymbol{u}}$ are simply replaced with the posterior modes $\widetilde{\widetilde{\boldsymbol{u}}}$ in these formulas.

For the ith response in the jth observation within a given cluster in a two-level model, the linear predictor is computed as

$$\widehat{z}_{ij} = \boldsymbol{x}_j'\widehat{\boldsymbol{\beta}} + \boldsymbol{x}_j'\widehat{\boldsymbol{\Lambda}}_i\widetilde{\boldsymbol{u}}$$

The linear predictor includes the offset or exposure variable if one was specified during estimation, unless the `nooffset` option is specified. If the `fixedonly` option is specified, the linear predictor is computed as

$$\widehat{z}_{ij} = \boldsymbol{x}_j'\widehat{\boldsymbol{\beta}}$$

The predicted mean, conditional on the predicted latent variables, is

$$\widehat{\mu}_{ij} = g^{-1}(\widehat{z}_{ij})$$

where $g^{-1}(\cdot)$ is the inverse link function defined in *Link functions* above. For the ordinal and multinomial families, the predicted mean is actually a probability, and `gsem` can produce a probability for each outcome value as described in *The ordinal family* and *The multinomial family* above. For the survival distributions, the predicted mean is computed using the formulas provided in *Survival distributions* above.

Models with categorical latent variables

The likelihood

`gsem` fits generalized linear models with categorical latent variables via maximum likelihood. The likelihood for the specified model is derived under the assumption that each response variable is independent and identically distributed across the estimation sample. The response variables are also assumed to be independent of each other. These assumptions are conditional on the categorical latent variables and the observed exogenous variables.

The likelihood is computed by combining the conditional likelihoods from each latent class (level combinations of the categorical latent variables) weighted by the associated latent class probabilities. Let $\boldsymbol{\theta}$ be the vector of model parameters. For a given observation, let $\boldsymbol{y}$ be the vector of observed response variables and $\boldsymbol{x}$ be the vector of observed exogenous variables. Without loss of generality, we will assume a single categorical latent variable C with k levels $1, \ldots, k$. The marginal likelihood for a given observation looks something like

$$\mathcal{L}_C(\boldsymbol{\theta}) = \sum_{i=1}^{k} \pi_i f_i(\boldsymbol{y}|\boldsymbol{x}, c_i = 1, \boldsymbol{\theta})$$

where π_i is the probability for the ith latent class, $f_i(\cdot)$ is the conditional probability density function for the observed response variables in the ith latent class, and $\boldsymbol{c}' = (c_1, \ldots, c_k)$ is the vector of latent class indicators. When $c_i = 1$, all other elements of $\boldsymbol{c}$ are 0. All auxiliary parameters are fit directly without any further parameterization, so we simply acknowledge that the auxiliary parameters are among the elements of $\boldsymbol{\theta}$.

The $\boldsymbol{y}$ variables are assumed to be independent, conditionally on $\boldsymbol{x}$ and C, so $f_i(\cdot)$ is the product of the individual conditional densities. One exception to this is when $\boldsymbol{y}$ contains two or more Gaussian response variables with the identity link, in which case the Gaussian responses are actually modeled using a multivariate normal density to allow for correlated errors and nonrecursive systems among Gaussian responses. This one exception does not meaningfully change the following discussion, so we make no effort to represent this distinction in the formulas.

For the ith latent class with n response variables, the conditional joint density function for a given observation is

$$f_i(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{n} f_{ij}(y_{ij}|\boldsymbol{x}, \boldsymbol{\theta})$$

Except for the ordinal and multinomial families, we use the link function to map the conditional mean

$$\mu_{ij} = E(y_{ij}|\boldsymbol{x}, c_i = 1)$$

to the linear prediction

$$z_{ij} = \boldsymbol{x}'\boldsymbol{\beta}_{ij}$$

where $\boldsymbol{\beta}_{ij}$ is the vector of the coefficients for y_{ij}. For notational convenience, we will overload the definitions of $f_i(\cdot)$ and $f_{ij}(\cdot)$ so that they are functions of the responses and model parameters through the linear predictions $\boldsymbol{z}_i' = (z_{i1}, \ldots, z_{in})$. Thus $f_i(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ is equivalently specified as $f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta})$, and $f_{ij}(y_{ij}|\boldsymbol{x}, \boldsymbol{\theta})$ is equivalently specified as $f_{ij}(y_{ij}, z_{ij}, \boldsymbol{\theta})$. In this new notation, the likelihood for a given observation is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{k} \pi_i \prod_{j=1}^{n} f_{ij}(y_{ij}, z_{ij}, \boldsymbol{\theta}) \tag{1}$$

gsem uses the multinomial distribution to model the probabilities for the latent classes. For the ith latent class, the probability is given by

$$\pi_i = \Pr(c_i = 1|\boldsymbol{x}) = \frac{\exp(z_i)}{\sum_a \exp(z_a)}$$

where the linear prediction for the ith latent class is

$$z_i = \boldsymbol{x}'\boldsymbol{\gamma}_i$$

and $\boldsymbol{\gamma}_i$ is the associated vector of coefficients. gsem assumes the first latent class is the base level; $\boldsymbol{\gamma}_1$ is a vector of zeros so that $z_1 = 0$ and $\exp(z_1) = 1$.

The vector $\boldsymbol{\theta}$ is therefore the set of unique model parameters taken from the following:

$\boldsymbol{\gamma}_i$ is the vector of coefficients for the ith latent class.

$\boldsymbol{\beta}_{ij}$ is the vector of coefficients for y_{ij}.

Auxiliary parameters that result from some of the distribution families.

Each latent class will have its own set of these parameters.

The EM algorithm

gsem uses the EM algorithm to refine starting values before maximizing the likelihood in (1).

The EM algorithm uses the complete-data likelihood, a likelihood where it is as if we have observed values for the latent class indicator variables $\boldsymbol{c}$. In the complete-data case, the likelihood for a given observation is

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{k} \{\pi_i f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta})\}^{c_i}$$

so the complete-data log likelihood is

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{k} c_i \left\{ \log \pi_i + \log f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta}) \right\}$$

We intend to maximize the expected complete-data log likelihood given the observed variables $\boldsymbol{y}$ and $\boldsymbol{x}$. This is an iterative process in which we use the gth guess of the model parameters, denoted $\boldsymbol{\theta}_{(g)}$, then compute the next guess $\boldsymbol{\theta}_{(g+1)}$.

In the expectation (E) step, we derive the functional form of the expected complete-data log likelihood. The complete-data log likelihood is a linear function of the latent class indicator variables, so

$$\mathrm{E}(c_i|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}_{(g)}) = \frac{\pi_i f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta}_{(g)})}{\sum_{j=1}^{k} \pi_j f_j(\boldsymbol{y}, \boldsymbol{z}_j, \boldsymbol{\theta}_{(g)})}$$

We denote this posterior probability by p_i, so the expected complete-data log likelihood for a given observation is given by

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(g)}) = \sum_{i=1}^{k} p_i \left\{ \log \pi_i + \log f_i(\boldsymbol{y}, \boldsymbol{z}_i, \boldsymbol{\theta}) \right\}$$

Note that $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(g)})$ is a function of $\boldsymbol{\theta}_{(g)}$ solely through the posterior probabilities p_i.

Now that we have the conditional complete-data log likelihood, the maximization (M) step is to maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_{(g)})$ with respect to $\boldsymbol{\theta}$ to find $\boldsymbol{\theta}_{(g+1)}$.

Survey data

`gsem` with categorical latent variables also supports estimation with survey data; however, only the linearized variance estimator is supported. For details on VCEs with survey data, see [SVY] **Variance estimation**.

Predictions

The predicted mean for a given response within a latent class is computed by applying the associated link function to the linear prediction; see *Link functions* above. For ordinal and multinomial responses, the predicted mean for a given response level is the predicted probability for that level. For survival outcomes, the formulas for predicted means are provided in *Survival distributions* above.

Let $\widehat{z}_i$ be the linear prediction for the ith latent class; then, the predicted probability for the ith latent class is given by

$$\widehat{\pi}_i = \frac{\exp(\widehat{z}_i)}{\sum_{a=1}^{k} \exp(\widehat{z}_a)}$$

The predicted posterior probability for the ith latent class is given by

$$\widetilde{\pi}_i = \frac{\widehat{\pi}_i f_i(\boldsymbol{y}, \widehat{\boldsymbol{z}}_i, \widehat{\boldsymbol{\theta}})}{\sum_{j=1}^{k} \widehat{\pi}_j f_j(\boldsymbol{y}, \widehat{\boldsymbol{z}}_j, \widehat{\boldsymbol{\theta}})}$$

Let $\widehat{\mu}_i$ be the predicted mean of response y in the ith latent class. The predicted overall mean of y, using the fitted latent class probabilities, is given by

$$\widehat{\mu} = \sum_{i=1}^{k} \widehat{\pi}_i \widehat{\mu}_i$$

The predicted overall mean of y, using the posterior latent class probabilities, is given by

$$\widetilde{\mu} = \sum_{i=1}^{k} \widetilde{\pi}_i \widehat{\mu}_i$$

References

Goodman, L. A. 2002. Latent class analysis: The empirical study of latent types, latent variables, and latent structures. In *Applied Latent Class Analysis*, ed. J. A. Hagenaars and A. L. McCutcheon, 3–55. Cambridge: Cambridge University Press.

Masyn, K. E. 2013. Latent class analysis and finite mixture modeling. In *The Oxford Handbook of Quantitative Methods*, ed. T. D. Little, vol. 2, 551–610. New York: Oxford University Press.

Rabe-Hesketh, S., and A. Skrondal. 2006. Multilevel modelling of complex survey data. *Journal of the Royal Statistical Society, Series A* 169: 805–827.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2005. Maximum likelihood estimation of limited and discrete dependent variable models with nested random effects. *Journal of Econometrics* 128: 301–323.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.

——. 2009. Prediction in multilevel generalized linear models. *Journal of the Royal Statistical Society, Series A* 172: 659–687.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

Title

Methods and formulas for sem — Methods and formulas for sem

Description Remarks and examples References Also see

Description

The methods and formulas for the `sem` commands are presented below.

Remarks and examples

Remarks are presented under the following headings:

Variable notation
Model and parameterization
Summary data
Maximum likelihood
Weighted least squares
Groups
Fitted parameters
Satorra–Bentler variance estimation
Standardized parameters
Reliability
Postestimation
Model framework
Goodness of fit
Group goodness of fit
Equation-level goodness of fit
Wald tests
Score tests
Residuals
Testing standardized parameters
Stability of nonrecursive systems
Direct, indirect, and total effects
Predictions

Variable notation

We will use the following convention to keep track of the five variable types recognized by the `sem` estimation command:

1. Observed endogenous variables are denoted y.
2. Observed exogenous variables are denoted x.
3. Latent endogenous variables are denoted η.
4. Latent exogenous variables are denoted ξ.
5. Error variables are denoted with prefix $e.$ on the associated endogenous variable.
    a. Error variables for observed endogenous are denoted $e.y$.
    b. Error variables for latent endogenous are denoted $e.\eta$.

In any given analysis, there are typically several variables of each type. Vectors of the four main variable types are denoted $\boldsymbol{y}$, $\boldsymbol{x}$, $\boldsymbol{\eta}$, and $\boldsymbol{\xi}$. The vector of all endogenous variables is

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{\eta} \end{pmatrix}$$

The vector of all exogenous variables is

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\xi} \end{pmatrix}$$

The vector of all error variables is

$$\boldsymbol{\zeta} = \begin{pmatrix} e.\boldsymbol{y} \\ e.\boldsymbol{\eta} \end{pmatrix}$$

Model and parameterization

`sem` can fit models of the form

$$\boldsymbol{Y} = \mathbf{B}\boldsymbol{Y} + \boldsymbol{\Gamma}\boldsymbol{X} + \boldsymbol{\alpha} + \boldsymbol{\zeta}$$

where $\mathbf{B} = [\beta_{ij}]$ is the matrix of coefficients on endogenous variables predicting other endogenous variables, $\boldsymbol{\Gamma} = [\gamma_{ij}]$ is the matrix of coefficients on exogenous variables, $\boldsymbol{\alpha} = [\alpha_i]$ is the vector of intercepts for the endogenous variables, and $\boldsymbol{\zeta}$ is assumed to have mean 0 and

$$\mathrm{Cov}(\boldsymbol{X}, \boldsymbol{\zeta}) = \mathbf{0}$$

Let

$$\boldsymbol{\kappa} = [\kappa_j] = E(\boldsymbol{X})$$
$$\boldsymbol{\Phi} = [\phi_{ij}] = \mathrm{Var}(\boldsymbol{X})$$
$$\boldsymbol{\Psi} = [\psi_{ij}] = \mathrm{Var}(\boldsymbol{\zeta})$$

Then the mean vector of the endogenous variables is

$$\boldsymbol{\mu}_Y = E(\boldsymbol{Y}) = (\boldsymbol{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma}\boldsymbol{\kappa} + \boldsymbol{\alpha})$$

the variance matrix of the endogenous variables is

$$\boldsymbol{\Sigma}_{YY} = \mathrm{Var}(\boldsymbol{Y}) = (\boldsymbol{I} - \mathbf{B})^{-1}(\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}' + \boldsymbol{\Psi})\left\{(\boldsymbol{I} - \mathbf{B})^{-1}\right\}'$$

and the covariance matrix between the endogenous variables and the exogenous variables is

$$\boldsymbol{\Sigma}_{YX} = \mathrm{Cov}(\boldsymbol{Y}, \boldsymbol{X}) = (\boldsymbol{I} - \mathbf{B})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}$$

Let $\boldsymbol{Z}$ be the vector of all variables:

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{pmatrix}$$

Then its mean vector is

$$\boldsymbol{\mu} = E(\boldsymbol{Z}) = \begin{pmatrix} \boldsymbol{\mu}_Y \\ \boldsymbol{\kappa} \end{pmatrix}$$

and its variance matrix is

$$\boldsymbol{\Sigma} = \mathrm{Var}(\boldsymbol{Z}) = \begin{pmatrix} \boldsymbol{\Sigma}_{YY} & \boldsymbol{\Sigma}_{YX} \\ \boldsymbol{\Sigma}'_{YX} & \boldsymbol{\Phi} \end{pmatrix}$$

Summary data

Let $\boldsymbol{z}_t$ be the vector of all observed variables for the tth observation,

$$\boldsymbol{z}_t = \begin{pmatrix} \boldsymbol{y}_t \\ \boldsymbol{x}_t \end{pmatrix}$$

and let w_t be the corresponding weight value, where $t = 1, \ldots, N$. If no weights were specified, then $w_t = 1$. Let $w_.$ be the sum of the weights; then the sample mean vector is

$$\overline{\boldsymbol{z}} = \frac{1}{w_.} \sum_{t=1}^{N} w_t \boldsymbol{z}_t$$

and the sample variance matrix is

$$\boldsymbol{S} = \frac{1}{w_. - 1} \sum_{t=1}^{N} w_t (\boldsymbol{z}_t - \overline{\boldsymbol{z}})(\boldsymbol{z}_t - \overline{\boldsymbol{z}})'$$

Maximum likelihood

Let $\boldsymbol{\theta}$ be the vector of unique model parameters, such as

$$\boldsymbol{\theta} = \begin{pmatrix} \mathrm{vec}(\mathbf{B}) \\ \mathrm{vec}(\boldsymbol{\Gamma}) \\ \mathrm{vech}(\boldsymbol{\Psi}) \\ \mathrm{vech}(\boldsymbol{\Phi}) \\ \boldsymbol{\alpha} \\ \boldsymbol{\kappa} \end{pmatrix}$$

Then under the assumption of the multivariate normal distribution, the overall log likelihood for $\boldsymbol{\theta}$ is

$$\log L(\boldsymbol{\theta}) = -\frac{w_.}{2} \left[k \log(2\pi) + \log \{\det(\boldsymbol{\Sigma}_o)\} + \mathrm{tr} \left(\boldsymbol{D} \boldsymbol{\Sigma}_o^{-1} \right) \right]$$

where k is the number of observed variables, $\boldsymbol{\Sigma}_o$ is the submatrix of $\boldsymbol{\Sigma}$ corresponding to the observed variables, and

$$\boldsymbol{D} = f\boldsymbol{S} + (\overline{\boldsymbol{z}} - \boldsymbol{\mu}_o)(\overline{\boldsymbol{z}} - \boldsymbol{\mu}_o)'$$

where

$$f = \begin{cases} 1, & \text{if } \texttt{nm1} \text{ is specified} \\ \dfrac{w_. - 1}{w_.}, & \text{otherwise} \end{cases}$$

and $\boldsymbol{\mu}_o$ is the subvector of $\boldsymbol{\mu}$ corresponding to the observed variables.

For the BHHH optimization technique and when computing observation-level scores, the log likelihood for $\boldsymbol{\theta}$ is computed as

$$\log L(\boldsymbol{\theta}) = -\sum_{t=1}^{N} \frac{w_t}{2} \left[k \log(2\pi) + \log \{\det(\boldsymbol{\Sigma}_o)\} + (\boldsymbol{z}_t - \boldsymbol{\mu}_o)' \boldsymbol{\Sigma}_o^{-1} (\boldsymbol{z}_t - \boldsymbol{\mu}_o) \right]$$

and the `nm1` option is ignored.

When `method(mlmv)` is specified, `sem` groups the data according to missing-value patterns. Each missing-value pattern will have its own summary data: k, $\overline{\boldsymbol{z}}$, and $\boldsymbol{S}$. The log likelihood for a missing-value pattern is computed using this summary data and the corresponding elements of $\boldsymbol{\mu}_o$ and $\boldsymbol{\Sigma}_o$. The overall log likelihood is computed by summing the log-likelihood values from each missing-value pattern.

Weighted least squares

Let $\boldsymbol{v}$ be the vector of unique sample moments of the observed variables and $\boldsymbol{\tau}$ be the corresponding vector of population moments. Then

$$\boldsymbol{v} = \begin{bmatrix} \overline{\boldsymbol{z}} \\ \text{vech}(f\boldsymbol{S}) \end{bmatrix}$$

and

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\mu}_o \\ \text{vech}(\boldsymbol{\Sigma}_o) \end{bmatrix}$$

The weighted least-squares (WLS) criterion function to minimize is the quadratic form

$$F_{\text{WLS}}(\boldsymbol{\theta}) = (\boldsymbol{v} - \boldsymbol{\tau})' \boldsymbol{W}^{-1} (\boldsymbol{v} - \boldsymbol{\tau})$$

where $\boldsymbol{W}$ is the least-squares weight matrix. For unweighted least squares (ULS), the weight matrix is the identity matrix $\boldsymbol{W} = \boldsymbol{I}$. Other weight matrices are mentioned in Bollen (1989).

The weight matrix implemented in `sem` is an estimate of the asymptotic covariance matrix of $\boldsymbol{v}$. This weight matrix is derived without any distributional assumptions and is often referred to as derived from an arbitrary distribution function or is asymptotic distribution free (ADF), thus the option `method(adf)`.

Groups

When the `group()` option is specified, each group has its own summary data and model parameters. The entire collection of model parameters is

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_G \end{pmatrix}$$

where G is the number of groups. The group-level criterion values are combined to produce an overall criterion value.

For `method(ml)` and `method(mlmv)`, the overall log likelihood is

$$\log L(\boldsymbol{\theta}) = \sum_{g=1}^{G} \log L(\boldsymbol{\theta}_g)$$

For `method(adf)`, the overall criterion function to minimize is

$$F_{\text{WLS}}(\boldsymbol{\theta}) = \sum_{g=1}^{G} F_{\text{WLS}}(\boldsymbol{\theta}_g)$$

Fitted parameters

`sem` fits the specified model by maximizing the log likelihood or minimizing the WLS criterion. If $\boldsymbol{\theta}$ is the vector of model parameters, then the fitted parameter vector is denoted by $\widehat{\boldsymbol{\theta}}$, and similarly for $\widehat{\mathbf{B}}$, $\widehat{\boldsymbol{\Gamma}}$, $\widehat{\boldsymbol{\Psi}}$, $\widehat{\boldsymbol{\Phi}}$, $\widehat{\boldsymbol{\alpha}}$, $\widehat{\boldsymbol{\kappa}}$, $\widehat{\boldsymbol{\Sigma}}$, $\widehat{\boldsymbol{\mu}}$, and their individual elements.

Satorra–Bentler variance estimation

When `vce(sbentler)` is specified, the variance of $\widehat{\boldsymbol{\theta}}$ is estimated as

$$\widehat{V}_{\mathrm{sb}}(\widehat{\boldsymbol{\theta}}) = \frac{1}{w_{\cdot}}(\Delta'\boldsymbol{H}\Delta)^{-1}\Delta'\boldsymbol{H}\boldsymbol{W}\boldsymbol{H}\Delta(\Delta'\boldsymbol{H}\Delta)^{-1}$$

where $\boldsymbol{W}$ is the asymptotic covariance matrix of $\boldsymbol{v}$ used in the WLS criterion function, and Δ is the partial derivative matrix of $\boldsymbol{\tau}$ with respect to $\boldsymbol{\theta}$ and uses the maximum likelihood fitted values $\widehat{\boldsymbol{\theta}}$. $\boldsymbol{H}$ is computed as

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{S}^{-1} & \mathbf{o} \\ \mathbf{o} & \boldsymbol{A} \end{pmatrix}$$

where

$$\boldsymbol{A} = .5\boldsymbol{D}'(\boldsymbol{S}^{-1} \otimes \boldsymbol{S}^{-1})\boldsymbol{D}$$

and $\boldsymbol{D}$ is a duplication matrix. See Satorra and Bentler (1994, eq. 16.10).

Standardized parameters

Let $\widehat{\sigma}_{ii}$ be the ith diagonal element of $\widehat{\boldsymbol{\Sigma}}_{YY}$. Then the standardized parameter estimates are

$$\widetilde{\beta}_{ij} = \widehat{\beta}_{ij}\sqrt{\frac{\widehat{\sigma}_{ii}}{\widehat{\sigma}_{jj}}}$$

$$\widetilde{\gamma}_{ij} = \widehat{\gamma}_{ij}\sqrt{\frac{\widehat{\phi}_{ii}}{\widehat{\sigma}_{jj}}}$$

$$\widetilde{\psi}_{ij} = \begin{cases} \widehat{\psi}_{ij}/\sqrt{\widehat{\psi}_{ii}\widehat{\psi}_{jj}}, & \text{if } i \neq j \\ \widehat{\psi}_{ii}/\sqrt{\widehat{\sigma}_{ii}}, & \text{if } i = j \end{cases}$$

$$\widetilde{\phi}_{ij} = \frac{\widehat{\phi}_{ij}}{\sqrt{\widehat{\phi}_{ii}\widehat{\phi}_{jj}}}$$

$$\widetilde{\alpha}_{i} = \frac{\widehat{\alpha}_{i}}{\sqrt{\widehat{\sigma}_{ii}}}$$

$$\widetilde{\kappa}_{j} = \frac{\widehat{\kappa}_{j}}{\sqrt{\widehat{\phi}_{jj}}}$$

The variance matrix of the standardized parameters is estimated using the delta method.

Reliability

For an observed endogenous variable, y, the reliability may be specified as p or $100 \times p\%$. The variance of $e.y$ is then constrained to $(1 - p)$ times the observed variance of y.

Postestimation

Model framework

`estat framework` reports the fitted parameters in their individual matrix forms as introduced in *Fitted parameters*.

Goodness of fit

`estat gof` reports the following goodness-of-fit statistics.

Let the degrees of freedom for the specified model be denoted by df_m. In addition to the specified model, `sem` fits saturated and baseline models corresponding to the observed variables in the specified model. The saturated model fits a full covariance matrix for the observed variables and has degrees of freedom

$$\mathrm{df}_s = \binom{p+q+1}{2} + p + q$$

where p is the number of observed endogenous variables and q is the number of observed exogenous variables in the model. The baseline model fits a reduced covariance matrix for the observed variables depending on the presence of endogenous variables. If there are no endogenous variables, all variables are uncorrelated in the baseline model; otherwise, only exogenous variables are correlated in the baseline model. The degrees of freedom for the baseline model is

$$\mathrm{df}_b = \begin{cases} 2q, & \text{if } p = 0 \\ 2p + q + \binom{q+1}{2}, & \text{if } p > 0 \end{cases}$$

For `method(ml)` and `method(mlmv)`, let the saturated log likelihood be denoted by $\log L_s$ and the baseline log likelihood be denoted by $\log L_b$. The likelihood-ratio test of the baseline versus saturated models is computed as

$$\chi^2_{\mathrm{bs}} = 2(\log L_s - \log L_b)$$

with degrees of freedom $\mathrm{df}_{\mathrm{bs}} = \mathrm{df}_s - \mathrm{df}_b$. The likelihood-ratio test of the specified model versus the saturated model is computed as

$$\chi^2_{\mathrm{ms}} = 2\{\log L_s - \log L(\widehat{\boldsymbol{\theta}})\}$$

with degrees of freedom $\mathrm{df}_{\mathrm{ms}} = \mathrm{df}_s - \mathrm{df}_m$.

The Satorra–Bentler (1994) scaled χ^2 test is computed as

$$\chi^2_{\text{sb,ms}} = \chi^2_{\text{ms}}/c$$

where $c = \text{trace}(\boldsymbol{UW}/\text{df}_{\text{ms}})$. $\boldsymbol{U}$ is computed as

$$\boldsymbol{U} = \boldsymbol{H} - \boldsymbol{H}\Delta(\Delta'\boldsymbol{H}\Delta)^{-1}\Delta'\boldsymbol{H}$$

where Δ and $\boldsymbol{H}$ are defined as in *Satorra–Bentler variance estimation* above.

The Satorra–Bentler scaled χ^2 test for the baseline versus saturated test, $\chi^2_{\text{sb,bs}}$, is defined as above, but Δ is replaced for the specified model with Δ for the baseline model.

For `method(adf)`, the saturated criterion value is 0, $F_s = 0$. Let the baseline criterion value be denoted by F_b. The χ^2 test of the baseline versus saturated models is computed as

$$\chi^2_{\text{bs}} = NF_b$$

with degrees of freedom $\text{df}_{\text{bs}} = \text{df}_s - \text{df}_b$. The chi-squared test of the specified model versus the saturated model is computed as

$$\chi^2_{\text{ms}} = NF_{\text{WLS}}(\widehat{\boldsymbol{\theta}})$$

with degrees of freedom $\text{df}_{\text{ms}} = \text{df}_s - \text{df}_m$.

The Akaike information criterion (Akaike 1974) is defined as

$$\text{AIC} = -2\log L(\widehat{\boldsymbol{\theta}}) + 2\text{df}_m$$

The Bayesian information criterion (Schwarz 1978) is defined as

$$\text{BIC} = -2\log L(\widehat{\boldsymbol{\theta}}) + N\text{df}_m$$

See [R] **BIC note** for additional information on calculating and interpreting the BIC.

The overall coefficient of determination is computed as

$$\text{CD} = 1 - \frac{\det(\widehat{\boldsymbol{\Psi}})}{\det(\widehat{\boldsymbol{\Sigma}}_{YY})}$$

This value is also referred to as the overall R^2 in `estat eqgof` (see [SEM] **estat eqgof**).

The root mean squared error of approximation (Browne and Cudeck 1993) is computed as

$$\text{RMSEA} = \left\{\frac{(\chi^2_{\text{ms}} - \text{df}_{\text{ms}})G}{(N-1)\text{df}_{\text{ms}}}\right\}^{1/2}$$

The 90% confidence interval for RMSEA is

$$\left(\sqrt{\frac{G\lambda_L}{(N-1)\text{df}_{\text{ms}}}}, \sqrt{\frac{G\lambda_U}{(N-1)\text{df}_{\text{ms}}}}\right)$$

where λ_L and λ_U are the noncentrality parameters corresponding to a noncentral chi-squared distribution with df_{ms} degrees of freedom in which the noncentral chi-squared random variable has a cumulative distribution function equal to 0.95 and 0.05, respectively.

The Browne and Cudeck (1993) p-value for the test of close fit with null hypothesis

$$H_0 : \text{RMSEA} \leq 0.05$$

is computed as

$$p = 1 - \Pr(\chi^2 < \chi^2_{\text{ms}} | \lambda, \text{df}_{\text{ms}})$$

where χ^2 is distributed noncentral chi-squared with noncentrality parameter $\lambda = (0.05)^2(N-1)\text{df}_{\text{ms}}$ and df_{ms} degrees of freedom. This p-value is not computed when the `group()` option is specified.

The comparative fit index (Bentler 1990) is computed as

$$\text{CFI} = 1 - \left[\frac{(\chi^2_{\text{ms}} - \text{df}_{\text{ms}})}{\max\{(\chi^2_{\text{bs}} - \text{df}_{\text{bs}}), (\chi^2_{\text{ms}} - \text{df}_{\text{ms}})\}} \right]$$

The Tucker–Lewis index (Bentler 1990) is computed as

$$\text{TLI} = \frac{(\chi^2_{\text{bs}}/\text{df}_{\text{bs}}) - (\chi^2_{\text{ms}}/\text{df}_{\text{ms}})}{(\chi^2_{\text{bs}}/\text{df}_{\text{bs}}) - 1}$$

Let k be the number of observed variables in the model. The standardized root mean squared residual is computed as

$$\text{SRMR} = \left\{ \frac{2\sum_{i=1}^{k}\sum_{j\leq i} r_{ij}^2}{k(k+1)G} \right\}^{1/2}$$

where r_{ij} is the standardized covariance residual

$$r_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}} - \frac{\widehat{\sigma}_{ij}}{\sqrt{\widehat{\sigma}_{ii}\widehat{\sigma}_{jj}}}$$

These standardized residuals are not the same as those reported by `estat residuals`; see *Residuals* below.

The RMSEA, CFI, and TLI are also reported using Satorra–Bentler scaled statistics when the model is fit using `vce(sbentler)`. In the formulas above, χ^2_{ms} and χ^2_{bs} are replaced, respectively, with $\chi^2_{\text{sb,ms}}$ and $\chi^2_{\text{sb,bs}}$.

Group goodness of fit

`estat ggof` reports CD, SRMR, and model versus saturated χ^2 values for each group separately. The group-level formulas are the same as those computed for a single group analysis; see *Goodness of fit* above.

Equation-level goodness of fit

`estat eqgof` reports goodness-of-fit statistics for each endogenous variable in the specified model. The coefficient of determination for the ith endogenous variable is computed as

$$R_i^2 = 1 - \frac{\widehat{\psi}_{ii}}{\widehat{\sigma}_{ii}}$$

The Bentler–Raykov (Bentler and Raykov 2000) squared multiple correlation for the ith endogenous variable is computed as

$$\text{mc}_i^2 = \frac{\widehat{\text{Cov}}(y_i, \widehat{y}_i)}{\sqrt{\widehat{\sigma}_{ii}\widehat{\text{Var}}(\widehat{y}_i)}}$$

where $\widehat{\sigma}_{ii}$ is a diagonal element of $\widehat{\boldsymbol{\Sigma}}$, $\widehat{\text{Var}}(\widehat{y}_i)$ is a diagonal element of

$$\widehat{\text{Var}}(\widehat{\boldsymbol{Y}}) = (\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}\widehat{\boldsymbol{\Gamma}}\widehat{\boldsymbol{\Phi}}\widehat{\boldsymbol{\Gamma}}'\left\{(\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}\right\}' + \left\{(\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}-\boldsymbol{I}\right\}\widehat{\boldsymbol{\Psi}}\left\{(\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}-\boldsymbol{I}\right\}'$$

and $\widehat{\text{Cov}}(y_i, \widehat{y}_i)$ is a diagonal element of

$$\widehat{\text{Cov}}(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}) = (\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}\widehat{\boldsymbol{\Gamma}}\widehat{\boldsymbol{\Phi}}\widehat{\boldsymbol{\Gamma}}'\left\{(\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}\right\}' + (\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}\widehat{\boldsymbol{\Psi}}\left\{(\boldsymbol{I}-\widehat{\mathbf{B}})^{-1}-\boldsymbol{I}\right\}'$$

Wald tests

`estat eqtest` performs Wald tests on the coefficients for each endogenous equation in the model. `estat ginvariant` computes a Wald test of group invariance for each model parameter that is free to vary across all groups. See [R] **test**.

Score tests

`estat mindices` computes modification indices for each constrained parameter in the model, including paths and covariances that were not even part of the model specification. Modification indices are score tests, which are also known as Lagrange multiplier tests. `estat scoretests` performs a score test for each user-specified linear constraint. `estat ginvariant` performs a score test of group invariance for each model parameter that is constrained to be equal across all groups.

A score test compares a constrained model fit to the same model without one or more constraints. The score test is computed as

$$\chi^2 = \boldsymbol{g}(\widehat{\boldsymbol{\theta}})'\boldsymbol{V}(\widehat{\boldsymbol{\theta}})\boldsymbol{g}(\widehat{\boldsymbol{\theta}})$$

where $\widehat{\boldsymbol{\theta}}$ is the fitted parameter vector from the constrained model, $\boldsymbol{g}(\boldsymbol{\theta})$ is the gradient vector function for the unconstrained model, and $\boldsymbol{V}(\boldsymbol{\theta})$ is the variance matrix function computed from the expected information matrix function for the unconstrained model. For `method(ml)` and `method(mlmv)`,

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial \log L^*(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{V}(\boldsymbol{\theta}) = \left[E\left\{-\frac{\partial^2 \log L^*(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}'}\right\}\right]^{-1}$$

where $\log L^*(\boldsymbol{\theta})$ is the log-likelihood function for the unconstrained model. For `method(adf)`,

$$\boldsymbol{g}(\boldsymbol{\theta}) = -\frac{\partial F^*_{\text{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$\boldsymbol{V}(\boldsymbol{\theta}) = \left[E\left\{\frac{\partial^2 F^*_{\text{WLS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\partial \boldsymbol{\theta}'}\right\}\right]^{-1}$$

where $F^*(\boldsymbol{\theta})$ is the WLS criterion function for the unconstrained model.

The score test is computed as described in Wooldridge (2010) when `vce(robust)` or `vce(cluster` *clustvar*`)` is specified.

Residuals

`estat residuals` reports raw, normalized, and standardized residuals for means and covariances of the observed variables.

The raw residual for the mean of the ith observed variable is

$$\overline{z}_i - \widehat{\mu}_i$$

The raw residual for the covariance between the ith and jth observed variables is

$$S_{ij} - \widehat{\sigma}_{ij}$$

The normalized residual for the mean of the ith observed variable is

$$\frac{\overline{z}_i - \widehat{\mu}_i}{\sqrt{\widehat{\mathrm{Var}}(\overline{z}_i)}}$$

where

$$\widehat{\mathrm{Var}}(\overline{z}_i) = \begin{cases} \dfrac{S_{ii}}{N}, & \text{if the } \texttt{sample} \text{ option is specified} \\ \dfrac{\widehat{\sigma}_{ii}}{N}, & \text{otherwise} \end{cases}$$

The normalized residual for the covariance between the ith and jth observed variables is

$$\frac{S_{ij} - \widehat{\sigma}_{ij}}{\sqrt{\widehat{\mathrm{Var}}(S_{ij})}}$$

where

$$\widehat{\mathrm{Var}}(S_{ij}) = \begin{cases} \dfrac{S_{ii}S_{jj} + S_{ij}}{N}, & \text{if the } \texttt{sample} \text{ option is specified} \\ \dfrac{\widehat{\sigma}_{ii}\widehat{\sigma}_{jj} + \widehat{\sigma}_{ij}}{N}, & \text{otherwise} \end{cases}$$

If the `nm1` option is specified, the denominator in the variance estimates is $N - 1$ instead of N.

The standardized residual for the mean of the ith observed variable is

$$\frac{\overline{z}_i - \widehat{\mu}_i}{\sqrt{\widehat{\mathrm{Var}}(\overline{z}_i - \widehat{\mu}_i)}}$$

where

$$\widehat{\mathrm{Var}}(\overline{z}_i - \widehat{\mu}_i) = \widehat{\mathrm{Var}}(\overline{z}_i) - \widehat{\mathrm{Var}}(\widehat{\mu}_i)$$

and $\widehat{\mathrm{Var}}(\widehat{\mu}_i)$ is computed using the delta method. Missing values are reported when the computed value of $\widehat{\mathrm{Var}}(\overline{z}_i)$ is less than $\widehat{\mathrm{Var}}(\widehat{\mu}_i)$. The standardized residual for the covariance between the ith and jth observed variables is

$$\frac{S_{ij} - \widehat{\sigma}_{ij}}{\sqrt{\widehat{\mathrm{Var}}(S_{ij} - \widehat{\sigma}_{ij})}}$$

where

$$\widehat{\text{Var}}(S_{ij} - \widehat{\sigma}_{ij}) = \widehat{\text{Var}}(S_{ij}) - \widehat{\text{Var}}(\widehat{\sigma}_{ij})$$

and $\widehat{\text{Var}}(\widehat{\sigma}_{ij})$ is computed using the delta method. Missing values are reported when the computed value of $\widehat{\text{Var}}(S_{ij})$ is less than $\widehat{\text{Var}}(\widehat{\sigma}_{ij})$. The variances of the raw residuals used in the standardized residual calculations are derived in Hausman (1978).

Testing standardized parameters

`estat stdize` provides access to tests on the standardized parameter estimates. `estat stdize` can be used as a prefix to `lincom` (see [R] **lincom**), `nlcom` (see [R] **nlcom**), `test` (see [R] **test**), and `testnl` (see [R] **testnl**).

Stability of nonrecursive systems

`estat stable` reports a stability index for nonrecursive systems. The stability index is calculated as the maximum of the modulus of the eigenvalues of $\mathbf{B}$. The nonrecursive system is considered stable if the stability index is less than 1.

Direct, indirect, and total effects

`estat teffects` reports direct, indirect, and total effects for the fitted model. The direct effects are

$$\boldsymbol{E}_d = \begin{bmatrix} \widehat{\mathbf{B}} & \widehat{\boldsymbol{\Gamma}} \end{bmatrix}$$

the total effects are

$$\boldsymbol{E}_t = \begin{bmatrix} (\boldsymbol{I} - \widehat{\mathbf{B}})^{-1} - \boldsymbol{I} & , & (\boldsymbol{I} - \widehat{\mathbf{B}})^{-1}\widehat{\boldsymbol{\Gamma}} \end{bmatrix}$$

and the indirect effects are $\boldsymbol{E}_i = \boldsymbol{E}_t - \boldsymbol{E}_d$. The standard errors of the effects are computed using the delta method.

Let $\boldsymbol{D}$ be the diagonal matrix whose elements are the square roots of the diagonal elements of $\widehat{\boldsymbol{\Sigma}}$, and let $\boldsymbol{D}_Y$ be the submatrix of $\boldsymbol{D}$ associated with the endogenous variables. Then the standardized effects are

$$\widetilde{\boldsymbol{E}}_d = \boldsymbol{D}_Y^{-1}\boldsymbol{E}_d\boldsymbol{D}$$
$$\widetilde{\boldsymbol{E}}_i = \boldsymbol{D}_Y^{-1}\boldsymbol{E}_i\boldsymbol{D}$$
$$\widetilde{\boldsymbol{E}}_t = \boldsymbol{D}_Y^{-1}\boldsymbol{E}_t\boldsymbol{D}$$

Predictions

`predict` computes factor scores and linear predictions.

Factor scores are computed with a linear regression by using the mean vector and variance matrix from the fitted model. For notational convenience, let

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{l} \end{pmatrix}$$

where

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{pmatrix}$$

and

$$\boldsymbol{l} = \begin{pmatrix} \boldsymbol{\eta} \\ \boldsymbol{\xi} \end{pmatrix}$$

The fitted mean of $\boldsymbol{Z}$ is

$$\widehat{\boldsymbol{\mu}}_Z = \begin{pmatrix} \widehat{\boldsymbol{\mu}}_z \\ \widehat{\boldsymbol{\mu}}_l \end{pmatrix}$$

and fitted variance of $\boldsymbol{Z}$ is

$$\widehat{\boldsymbol{\Sigma}}_Z = \begin{pmatrix} \widehat{\boldsymbol{\Sigma}}_{zz} & \widehat{\boldsymbol{\Sigma}}_{zl} \\ \widehat{\boldsymbol{\Sigma}}_{zl}' & \widehat{\boldsymbol{\Sigma}}_{ll} \end{pmatrix}$$

The factor scores are computed as

$$\widetilde{\boldsymbol{l}} = \begin{pmatrix} \widetilde{\boldsymbol{\eta}} \\ \widetilde{\boldsymbol{\xi}} \end{pmatrix} = \widehat{\boldsymbol{\Sigma}}_{zl}' \widehat{\boldsymbol{\Sigma}}_{zz} \widehat{\boldsymbol{\mu}}_z + \widehat{\boldsymbol{\mu}}_l$$

The linear prediction for the endogenous variables in the tth observation is computed as

$$\widehat{\boldsymbol{Y}}_t = \widehat{\mathbf{B}} \widetilde{\boldsymbol{Y}}_t + \widehat{\boldsymbol{\Gamma}} \widetilde{\boldsymbol{X}}_t + \widehat{\boldsymbol{\alpha}}$$

where

$$\widetilde{\boldsymbol{Y}}_t = \begin{pmatrix} \boldsymbol{y}_t \\ \widetilde{\boldsymbol{\eta}} \end{pmatrix}$$

and

$$\widetilde{\boldsymbol{X}}_t = \begin{pmatrix} \boldsymbol{x}_t \\ \widetilde{\boldsymbol{\xi}} \end{pmatrix}$$

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19: 716–723.

Bentler, P. M. 1990. Comparative fit indexes in structural models. *Psychological Bulletin* 107: 238–246.

Bentler, P. M., and T. Raykov. 2000. On measures of explained variance in nonrecursive structural equation models. *Journal of Applied Psychology* 85: 125–131.

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Browne, M. W., and R. Cudeck. 1993. Alternative ways of assessing model fit. Reprinted in *Testing Structural Equation Models*, ed. K. A. Bollen and J. S. Long, pp. 136–162. Newbury Park, CA: SAGE.

Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271.

Satorra, A., and P. M. Bentler. 1994. Corrections to test statistics and standard errors in covariance structure analysis. In *Latent Variables Analysis: Applications for Developmental Research*, ed. A. von Eye and C. C. Clogg, 399–419. Thousand Oaks, CA: SAGE.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics* 6: 461–464.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

Also see

[SEM] **sem** — Structural equation model estimation command

Title

nlcom — Nonlinear combinations of parameters

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`nlcom` is a postestimation command for use after `sem`, `gsem`, and other Stata estimation commands.

`nlcom` computes point estimates, standard errors, z statistics, p-values, and confidence intervals for (possibly) nonlinear combinations of the estimated parameters. See [R] **nlcom**.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Nonlinear combinations of parameters

Syntax

`nlcom` *exp* [, *options*]

Options

See *Options* in [R] **nlcom**.

Remarks and examples

See [SEM] **Example 42g**.

`nlcom` works in the metric of SEM, which is to say path coefficients, variances, and covariances. If you want to frame your nonlinear combinations in terms of standardized coefficients and correlations and you fit the model with `sem`, not `gsem`, then prefix `nlcom` with `estat stdize:`; see [SEM] **estat stdize**.

❑ Technical note

`estat stdize:` is, strictly speaking, unnecessary because everywhere you wanted a standardized coefficient or correlation, you could just type the formula. If you did that, you would get the same results except for numerical precision. The answer produced with the `estat stdize:` prefix will be a little more accurate because `estat stdize:` is able to substitute an analytic derivative in one part of the calculation where `nlcom`, doing the whole thing itself, would be forced to use a numeric derivative.

❑

Stored results

See *Stored results* in [R] **nlcom**.

Also see

[R] **nlcom** — Nonlinear combinations of estimators

[SEM] **estat stdize** — Test standardized parameters

[SEM] **lincom** — Linear combinations of parameters

[SEM] **test** — Wald test of linear hypotheses

[SEM] **testnl** — Wald test of nonlinear hypotheses

[SEM] Example 42g — One- and two-level mediation models (multilevel)

Title

predict after gsem — Generalized linear predictions, etc.

Description Menu Syntax Options
Remarks and examples Reference Also see

Description

`predict` is a standard postestimation command of Stata. This entry concerns use of `predict` after `gsem`. See [SEM] **predict after sem** if you fit your model with `sem`.

`predict` after `gsem` creates new variables containing observation-by-observation values of estimated observed response variables, linear predictions of observed response variables, latent class probabilities, or endogenous or exogenous continuous latent variables.

Menu

Statistics > SEM (structural equation modeling) > Predictions

Syntax

Syntax for predicting observed endogenous outcomes and other statistics

`predict` [*type*] { *stub*`*` | *newvarlist* } [*if*] [*in*] [`,` *statistic options*]

Syntax for obtaining estimated continuous latent variables and their standard errors

`predict` [*type*] { *stub*`*` | *newvarlist* } [*if*] [*in*]`,` *lstatistic* [*loptions*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvarlist* } [*if*] [*in*]`,` `scores`

The default is to predict observed endogenous variables with empirical Bayes means predictions of the continuous latent variables. If the model includes a categorical latent variable, the default is class-specific predictions of the observed endogenous variables.

statistic	Description
Main	
`mu`	expected value of *depvar*; the default
`pr`	probability (synonym for `mu` when μ is a probability)
`eta`	expected value of linear prediction of *depvar*
`density`	density function at *depvar*
`distribution`	distribution function at *depvar*
`survival`	survivor function at *depvar*
`expression(`*exp*`)`	calculate prediction using *exp*
`classpr`	latent class probability
`classposteriorpr`	posterior latent class probability

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated continuous latent variables; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the latent variables
`pmarginal`	compute `mu` marginally with respect to the posterior latent class probabilities
`nooffset`	make calculation ignoring offset or exposure
† `outcome(`*depvar* [*#*]`)`	specify observed response variable (default all)
* `class(`*lclspec*`)`	specify latent class (default all)
Integration	
int_options	integration options

† `outcome(`*depvar #*`)` is allowed only if *depvar* has family `multinomial`, `ordinal`, or `bernoulli`. Predicting other generalized responses requires specifying only `outcome(`*depvar*`)`.
`outcome(`*depvar #*`)` may also be specified as `outcome(`*#.depvar*`)` or `outcome(`*depvar* `##`*`)`*.
`outcome(`*depvar* `#3`*`)`* means the third outcome value. `outcome(`*depvar* `#3`*`)`* would mean the same as `outcome(`*depvar* `4`*`)`* if outcomes were 1, 3, and 4.

* `class(`*lclspec*`)` is allowed only for models with categorical latent variables. For models with one categorical latent variable, *lclspec* can be a class value, such as `class(2)` or its equivalent factor-variable notation `class(2.C)`, assuming the categorical latent variable is `C`. For models with two or more categorical latent variables, *lclspec* may only be in factor-variable notation, such as `class(2.C#1.D)` for categorical latent variables `C` and `D`.

ctype	Description
`ebmeans`	empirical Bayes means of latent variables; the default
`ebmodes`	empirical Bayes modes of latent variables
`fixedonly`	prediction for the fixed portion of the model only

lstatistic	Description
Main	
`latent`	empirical Bayes prediction of all latent variables
`latent(`*varlist*`)`	empirical Bayes prediction of specified latent variables

loptions	Description
Main	
`ebmeans`	empirical Bayes means of latent variables; the default
`ebmodes`	empirical Bayes modes of latent variables
`se(`*stub*`*` | *newvarlist*`)`	standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
intpoints(#)	use # quadrature points to compute marginal predictions and empirical Bayes means
iterate(#)	set maximum number of iterations in computing statistics involving empirical Bayes estimators
tolerance(#)	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options

Main

mu, the default, calculates the expected value of the outcomes.

pr calculates predicted probabilities and is a synonym for mu. This option is available only for multinomial, ordinal, and Bernoulli outcomes.

eta calculates the fitted linear prediction.

density calculates the density function. This prediction is computed using the current values of the observed variables, including the dependent variable.

distribution calculates the distribution function. This prediction is computed using the current values of the observed variables, including the dependent variable. This option is not allowed for multinomial outcomes.

survival calculates the survivor function. This prediction is computed using the current values of the observed variables, including the dependent variable. This option is only allowed for exponential, gamma, loglogistic, lognormal, and Weibull outcomes.

expression(*exp*) specifies the prediction as an expression. *exp* is any valid Stata expression, but the expression must contain a call to one of the two special functions unique to this option:

1. mu(*outcome*): The mu() function specifies the calculation of the mean prediction for *outcome*. If mu() is specified without *outcome*, the mean prediction for the first outcome is implied.

   pr(*outcome*): The pr() function is a synonym for mu(*outcome*) when *outcome* identifies a multinomial, ordinal, or Bernoulli outcome.

2. eta(*outcome*): The eta() function specifies the calculation of the linear prediction for *outcome*. If eta() is specified without *outcome*, the linear predictor for the first outcome is implied.

   When you specify *exp*, both of these functions may be used repeatedly, in combination, and in combination with other Stata functions and expressions.

classpr calculates predicted probabilities for each latent class.

classposteriorpr calculates predicted posterior probabilities for each latent class. The posterior probabilities are a function of the latent class predictors and the fitted outcome densities.

conditional(*ctype*), marginal, and pmarginal specify how latent variables are handled in computing *statistic*.

conditional() specifies that *statistic* will be computed conditional on specified or estimated continuous latent variables.

`conditional(ebmeans)`, the default, specifies that empirical Bayes means be used as the estimates of the latent variables. These estimates are also known as posterior mean estimates of the latent variables.

`conditional(ebmodes)` specifies that empirical Bayes modes be used as the estimates of the latent variables. These estimates are also known as posterior mode estimates of the latent variables.

`conditional(fixedonly)` specifies that all latent variables be set to zero, equivalent to using only the fixed portion of the model.

`marginal` specifies that the predicted *statistic* be computed marginally with respect to the latent variables.

Although this is not the default, marginal predictions are often very useful in applied analysis. They produce what are commonly called population-averaged estimates. They are also required by `margins` for models with continuous latent variables.

For models with continuous latent variables, the *statistic* is calculated by integrating the prediction function with respect to all the latent variables over their entire support.

For models with categorical latent variables, `mu` is the only supported *statistic*. The overall expected value of each outcome is predicted by combining the class-specific expected values using the latent class probabilities.

`pmarginal` specifies that the overall expected value of each outcome be predicted by combining the class-specific expected values using the posterior latent class probabilities. This option is allowed only with the default *statistic*, `mu`.

`nooffset` is relevant only if option `offset()` or `exposure()` was specified at estimation time. `nooffset` specifies that `offset()` or `exposure()` be ignored, which produces predictions as if all subjects had equal exposure.

`outcome(`*depvar* [*#*]`)` specifies that predictions for *depvar* be calculated. Predictions for all observed response variables are computed by default. If *depvar* is a multinomial or an ordinal outcome, then *#* optionally specifies which outcome level to predict.

`class(`*lclspec*`)` specifies that predictions for latent class *lclspec* be calculated. Predictions for all latent classes are computed by default. For models with one categorical latent variable, such as `C`, *lclspec* can be a class value, such as `class(2)` or its equivalent factor-variable notation, `class(2.C)`. For models with two or more categorical latent variables, such as `C` and `D`, *lclspec* may only be in factor-variable notation, such as `class(2.C)` or `class(2.C#1.D)`.

`latent` and `latent(`*varlist*`)` specify that the continuous latent variables be estimated using empirical Bayes predictions. By default or if the `ebmeans` option is specified, empirical Bayes means are computed. With the `ebmodes` option, empirical Bayes modes are computed.

`latent` requests empirical Bayes estimates for all latent variables.

`latent(`*varlist*`)` requests empirical Bayes estimates for the specified latent variables.

`ebmeans` specifies that empirical Bayes means be used to predict the latent variables.

`ebmodes` specifies that empirical Bayes modes be used to predict the latent variables.

`se(`*stub*`*` | *newvarlist*`)` calculates standard errors of the empirical Bayes estimators and stores the result in *newvarlist*. This option requires the `latent` or `latent()` option.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use *stub*`*` to have `predict` generate enumerated variables with prefix *stub*.

Integration

`intpoints(#)` specifies the number of quadrature points used to compute marginal predictions and the empirical Bayes means; the default is the value from estimation.

`iterate(#)` specifies the maximum number of iterations when computing statistics involving empirical Bayes estimators; the default is the value from estimation.

`tolerance(#)` specifies convergence tolerance when computing statistics involving empirical Bayes estimators; the default is the value from estimation.

Remarks and examples

Out-of-sample prediction is allowed for all `predict` options except `scores`.

`predict` has two ways of specifying the names of the variables to be created:

```
. predict stub*, ...
```

or

```
. predict firstname secondname ..., ...
```

The first creates variables named *stub*`1`, *stub*`2`, ... The second creates variables with names that you specify. We strongly recommend using the *stub*`*` syntax when creating multiple variables because you have no way of knowing the order in which to specify the individual variable names to correspond to the order in which `predict` will make the calculations. If you use *stub*`*`, the variables will be labeled and you can rename them.

The second syntax is useful when you create one variable and specify `outcome()`, `expression()`, `class()`, or `latent()`.

See [SEM] **Intro 7**, [SEM] **Example 28g**, [SEM] **Example 29g**, [SEM] **Example 50g**, and [SEM] **Example 52g**.

Reference

Skrondal, A., and S. Rabe-Hesketh. 2009. Prediction in multilevel generalized linear models. *Journal of the Royal Statistical Society, Series A* 172: 659–687.

Also see

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem postestimation** — Postestimation tools for gsem

[SEM] **Intro 7** — Postestimation tests and predictions

[SEM] **Example 28g** — One-parameter logistic IRT (Rasch) model

[SEM] **Example 29g** — Two-parameter logistic IRT model

[SEM] **Example 50g** — Latent class model

[SEM] **Example 52g** — Latent profile model

[SEM] **Methods and formulas for gsem** — Methods and formulas for gsem

Title

predict after sem — Factor scores, linear predictions, etc.

Description Menu Syntax Options
Remarks and examples Reference Also see

Description

`predict` is a standard postestimation command of Stata. This entry concerns use of `predict` after `sem`. See [SEM] **predict after gsem** if you fit your model with `gsem`.

`predict` after `sem` creates new variables containing observation-by-observation values of estimated factor scores (meaning predicted values of latent variables) and predicted values for latent and observed endogenous variables. Out-of-sample prediction is allowed.

When `predict` is used on a model fit by `sem` with the `group()` option, results are produced with the appropriate group-specific estimates. Out-of-sample prediction is allowed; missing values are filled in for groups not included at the time the model was fit.

`predict` allows two syntaxes. You can type

```
. predict stub*, ...
```

to create variables named *stub*`1`, *stub*`2`, ..., or you can type

```
. predict var1 var2 ..., ...
```

to create variables named *var1*, *var2*, ...

`predict` may not be used with summary statistics data.

Menu

Statistics > SEM (structural equation modeling) > Predictions

Syntax

`predict` [*type*] { *stub*`*` | *newvarlist* } [*if*] [*in*] [`,` *options*]

options	Description
`xb`	linear prediction for all OEn variables; the default
`xb(`*varlist*`)`	linear prediction for specified OEn variables
`xblatent`	linear prediction for all LEn variables
`xblatent(`*varlist*`)`	linear prediction for specified LEn variables
`latent`	factor scores for all latent variables
`latent(`*varlist*`)`	factor scores for specified latent variables
`scores`	calculate first derivative of the log likelihood

Key: OEn = observed endogenous; LEn = latent endogenous

Options

`xb` calculates the linear prediction for all observed endogenous variables in the model. `xb` is the default if no option is specified.

`xb(`*varlist*`)` calculates the linear prediction for the variables specified, all of which must be observed endogenous variables.

`xblatent` and `xblatent(`*varlist*`)` calculate the linear prediction for all or the specified latent endogenous variables, respectively.

`latent` and `latent(`*varlist*`)` calculate the factor scores for all or the specified latent variables, respectively. The calculation method is an analog of regression scoring; namely, it produces the means of the latent variables conditional on the observed variables used in the model. If missing values are found among the observed variables, conditioning is on the variables with observed values only.

`scores` is for use by programmers. It provides the first derivative of the observation-level log likelihood with respect to the parameters.

Programmers: In single-group `sem`, each parameter that is not constrained to be 0 has an associated equation. As a consequence, the number of equations, and hence the number of score variables created by `predict`, may be large.

Remarks and examples

See [SEM] **Example 14**.

Factor scoring for latent variables can be interpreted as a form of missing-value imputation—think of each latent variable as an observed variable that has only missing values.

When latent variables are present in the model, linear predictions from `predict, xb` are computed by substituting the factor scores in place of each latent variable before computing the linear combination of coefficients. This method will lead to inconsistent coefficient estimates when the factor score contains measurement error; see Bollen (1989, 305–306).

Reference

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Also see

[SEM] **Example 14** — Predicted values

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

[SEM] **sem postestimation** — Postestimation tools for sem

Title

sem — Structural equation model estimation command

Description | Menu | Syntax | Options
Remarks and examples | Stored results | References | Also see

Description

`sem` fits structural equation models. Even when you use the SEM Builder, you are using the `sem` command.

Menu

Statistics > SEM (structural equation modeling) > Model building and estimation

Syntax

`sem` *paths* [*if*] [*in*] [*weight*] [`,` *options*]

where *paths* are the paths of the model in command-language path notation; see [SEM] **sem and gsem path notation**.

options	Description
model_description_options	fully define, along with *paths*, the model to be fit
group_options	fit model for different groups
ssd_options	for use with summary statistics data
estimation_options	method used to obtain estimation results
reporting_options	reporting of estimation results
syntax_options	controlling interpretation of syntax

Time-series operators are allowed.
`bootstrap`, `by`, `jackknife`, `permute`, `statsby`, and `svy` are allowed; see [U] **11.1.10 Prefix commands**.
Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.
`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.
`fweights`, `iweights`, and `pweights` are allowed; see [U] **11.1.6 weight**.
Also see [SEM] **sem postestimation** for features available after estimation.

Options

model_description_options describe the model to be fit. The model to be fit is fully specified by *paths*—which appear immediately after `sem`—and the options `covariance()`, `variance()`, and `means()`. See [SEM] **sem model description options** and [SEM] **sem and gsem path notation**.

group_options allow the specified model to be fit for different subgroups of the data, with some parameters free to vary across groups and other parameters constrained to be equal across groups. See [SEM] **sem group options**.

ssd_options allow models to be fit using summary statistics data (SSD), meaning data on means, variances (standard deviations), and covariances (correlations). See [SEM] **sem ssd options**.

estimation_options control how the estimation results are obtained. These options control how the standard errors (VCE) are obtained and control technical issues such as choice of estimation method. See [SEM] **sem estimation options**.

reporting_options control how the results of estimation are displayed. See [SEM] **sem reporting options**.

syntax_options control how the syntax that you type is interpreted. See [SEM] **sem and gsem syntax options**.

Remarks and examples

For a readable explanation of what `sem` can do and how to use it, see any of the intro sections. You might start with [SEM] **Intro 1**.

For examples of `sem` in action, see any of the example sections. You might start with [SEM] **Example 1**.

For detailed syntax and descriptions, see the references below.

Remarks on three advanced topics are presented under the following headings:

Default normalization constraints
Default covariance assumptions
How to solve convergence problems

Default normalization constraints

`sem` applies the following rules as necessary to identify the model:

1. `means(1: LatentExogenous@0)`
   `sem` constrains all latent exogenous variables to have mean 0. When the `group()` option is specified, this rule is applied to the first group only.
2. `(LatentEndogenous <- _cons@0)`
   `sem` sets all latent endogenous variables to have intercept 0.
3. `(first <- Latent@1)`
   If latent variable `Latent` is measured by observed endogenous variables, then `sem` sets the path coefficient of `(first<-Latent)` to be 1; `first` is the first observed endogenous variable.
4. `(First<-Latent@1)`
   If item 3 does not apply—if latent variable `Latent` is measured by other latent endogenous variables only—then `sem` sets the path coefficient of `First<-Latent` to be 1; `First` is the first latent variable.

The above constraints are applied as needed. Here are the available overrides:

1. To override the normalization constraints, specify your own constraints. Most normalization constraints are added by `sem` as needed. See *How sem (gsem) solves the problem for you* under *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.

2. To override means() constraints, you must use the means() option to free the parameter. To override that the mean of latent exogenous variable MyLatent has mean 0, specify the means(MyLatent) option. See [SEM] **sem and gsem path notation**.
3. To override constrained path coefficients from _cons, such as (LatentEndogenous <- _cons@0), you must explicitly specify the path without a constraint (LatentEndogenous <- _cons). See [SEM] **sem and gsem path notation**.

Default covariance assumptions

sem assumes the following covariance structure:

1. covstructure(_Ex, unstructured)
   All exogenous variables (observed and latent) are assumed to be correlated with each other.
2. covstructure(e._En, diagonal)
   The error variables associated with all endogenous variables are assumed to be uncorrelated with each other.

You can override these assumptions by

1. Constraining or specifying the relevant covariance to allow e.y and e.Ly to be correlated (specify the covariance(e.y*e.Ly) option); see [SEM] **sem model description options**.
2. Using the covstructure() option; see [SEM] **sem and gsem option covstructure( )**.

How to solve convergence problems

See [SEM] **Intro 12**.

Stored results

sem stores the following in e():

Scalars

e(N)	number of observations
e(N_clust)	number of clusters
e(N_groups)	number of groups
e(N_missing)	number of missing values in the sample for method(mlmv)
e(ll)	log likelihood of model
e(df_m)	model degrees of freedom
e(df_b)	baseline model degrees of freedom
e(df_s)	saturated model degrees of freedom
e(chi2_ms)	test of target model against saturated model
e(df_ms)	degrees of freedom for e(chi2_ms)
e(p_ms)	p-value for e(chi2_ms)
e(chi2sb_ms)	Satorra–Bentler scaled test of target model against saturated model
e(psb_ms)	p-value for e(chi2sb_ms)
e(sbc_ms)	Satorra–Bentler correction factor for e(chi2sb_ms)
e(chi2_bs)	test of baseline model against saturated model
e(df_bs)	degrees of freedom for e(chi2_bs)
e(p_bs)	p-value for e(chi2_bs)
e(chi2sb_bs)	Satorra–Bentler scaled test of baseline model against saturated model
e(psb_bs)	p-value for e(chi2sb_bs)
e(sbc_bs)	Satorra–Bentler correction factor for e(chi2_bs)
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code

e(converged)	1 if target model converged, 0 otherwise
e(critvalue)	log likelihood or discrepancy of fitted model
e(critvalue_b)	log likelihood or discrepancy of baseline model
e(critvalue_s)	log likelihood or discrepancy of saturated model
e(modelmeans)	1 if fitting means and intercepts, 0 otherwise

Macros

e(cmd)	sem
e(cmdline)	command as typed
e(data)	raw or ssd if SSD were used
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. Err.
e(method)	estimation method: ml, mlmv, or adf
e(technique)	maximization technique
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(lyvars)	names of latent y variables
e(oyvars)	names of observed y variables
e(lxvars)	names of latent x variables
e(oxvars)	names of observed x variables
e(groupvar)	name of group variable
e(xconditional)	empty if noxconditional specified, xconditional otherwise
e(marginsnotok)	predictions not allowed by margins
e(marginsdefault)	default predict() specification for margins

Matrices

e(b)	parameter vector
e(b_std)	standardized parameter vector
e(b_pclass)	parameter class
e(Cns)	constraints matrix
e(admissible)	admissibility of Σ, Ψ, Φ
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	covariance matrix of the estimators
e(V_std)	standardized covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(nobs)	vector with number of observations per group
e(groupvalue)	vector of group values of e(groupvar)

Functions

e(sample)	marks estimation sample (not with SSD)

References

MacDonald, K. 2016. Group comparisons in structural equation models: Testing measurement invariance. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/08/23/group-comparisons-in-structural-equation-models-testing-measurement-invariance/.

Wiggins, V. L. 2011. Multilevel random effects in xtmixed and sem—the long and wide of it. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2011/09/28/multilevel-random-effects-in-xtmixed-and-sem-the-long-and-wide-of-it/.

Also see

[SEM] **Intro 1** — Introduction

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **sem path notation extensions** — Command syntax for path diagrams

[SEM] **sem model description options** — Model description options

[SEM] **sem group options** — Fitting models on different groups

[SEM] **sem ssd options** — Options for use with summary statistics data

[SEM] **sem estimation options** — Options affecting estimation

[SEM] **sem reporting options** — Options affecting reporting of results

[SEM] **sem and gsem syntax options** — Options affecting interpretation of syntax

[SEM] **sem postestimation** — Postestimation tools for sem

[SEM] **Methods and formulas for sem** — Methods and formulas for sem

[SVY] **svy estimation** — Estimation commands for survey data

Title

sem and gsem option constraints() — Specifying constraints

Description Syntax Remarks and examples Also see

Description

Constraints refer to constraints to be imposed on the estimated parameters of a model. These constraints usually come in one of three forms:

1. Constraints that a parameter such as a path coefficient or variance is equal to a fixed value such as 1.
2. Constraints that two or more parameters are equal.
3. Constraints that two or more parameters are related by a linear equation.

It is usually easier to specify constraints with `sem`'s and `gsem`'s path notation; see [SEM] **sem and gsem path notation**.

`sem`'s and `gsem`'s `constraints()` option provides an alternative way of specifying constraints.

Syntax

```
 sem ... [, ... constraints(# [# ...]) ...]
gsem ... [, ... constraints(# [# ...]) ...]
```

where # are constraint numbers. Constraints are defined by the `constraint` command; see [R] **constraint**.

Remarks and examples

Remarks are presented under the following headings:

Use with sem
Use with gsem

Also see [R] **constraint**.

Use with sem

There is only one case where `constraints()` might be easier to use with `sem` instead of specifying constraints in the path notation. You wish to specify that two or more parameters are related and then decide you would like to fix the value at which they are related.

For example, if you wanted to specify that parameters are equal, you could type

```
. sem ... (y1 <- x@c1) (y2 <- x@c1)      (y3 <- x@c1)       ...
```

Using the path notation, you can specify more general relationships, too, such as

```
. sem ... (y1 <- x@c1) (y2 <- x@(2*c1)) (y3 <- x@(3*c1+1)) ...
```

Say you now decide you want to fix *c1* at value 1. Using the path notation, you modify what you previously typed:

```
. sem ... (y1 <- x@1) (y2 <- x@2)     (y3 <- x@4)       ...
```

Alternatively, you could do the following:

```
. constraint 1 _b[y2:x] = 2*_b[y1:x]
. constraint 2 _b[y3:x] = 3*_b[y1:x] + 1
. sem ..., ... constraints(1 2)
. constraint 3 _b[y1:x] = 1
. sem .., ... constraints(1 2 3)
```

Use with gsem

Gamma regression can produce exponential regression estimates if you constrain the log of the scale parameter to 0. Parameters associated with particular generalized linear families, such as scalar parameters, cutpoints, and the like, cannot be constrained using the `@` notation in paths. You must use Stata's constraints.

Say we wish to fit the model `y <- x1` with exponential regression. We admit that we do not remember the name under which `gsem` stores the scalar parameter, so first we type

```
. gsem (y <- x1, gamma), noestimate
```

From the output, we quickly discover that the log of the scale parameter is stored as `_b[/y:logs]`. With that information, to obtain the constrained results, we type

```
. constraint 1 _b[/y:logs] = 0
. gsem (y <- x1, gamma), constraints(1)
```

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **gsem model description options** — Model description options

[SEM] **sem model description options** — Model description options

[R] **constraint** — Define and list constraints

Title

sem and gsem option covstructure( ) — Specifying covariance restrictions

Description Syntax Option Remarks and examples Also see

Description

Option `covstructure()` provides a sometimes convenient way to constrain the covariances of your model.

Alternatively or in combination, you can place constraints on the covariances by using the standard path notation, such as

```
. sem  ..., ... cov(name1*name2@c1 name3*name4@c1) ...
. gsem ..., ... cov(name1*name2@c1 name3*name4@c1) ...
```

See [SEM] **sem and gsem path notation**.

If you are using `sem`, also see [SEM] **sem path notation extensions** for documentation on how the syntax of `covstructure()` is modified when the `group()` option is specified.

If you are using `gsem`, also see [SEM] **gsem path notation extensions** for documentation on how the syntax of `covstructure()` is modified when the `group()` option or the `lclass()` options are specified.

Syntax

```
sem ... [, ... covstructure(variables, structure) ... ]
gsem ... [, ... covstructure(variables, structure) ... ]
```

where *variables* is one of

1. a list of (a subset of the) exogenous variables (`sem`) or latent exogenous variables (`gsem`) in your model, for instance,

   ```
   . sem  ..., ... covstruct(x1 x2, structure)
   . sem  ..., ... covstruct(L1 L2, structure)
   . gsem ..., ... covstruct(L1 L2, structure)
   ```

2. `_OEx`, meaning all observed exogenous variables in your model (`sem` only)
3. `_LEx`, meaning all latent exogenous variables in your model (including any multilevel latent exogenous variables in the case of `gsem`)
4. `_Ex`, meaning all exogenous variables in your model (`sem` only)

or where *variables* is one of

1. a list of (a subset of the) error variables in your model, for example,

   ```
   . sem ..., ... covstruct(e.y1 e.y2 e.Aspect, structure)
   ```

2. `e._OEn`, meaning all error variables associated with observed endogenous variables in your model
3. `e._LEn`, meaning all error variables associated with latent endogenous variables in your model
4. `e._En`, meaning all error variables in your model

and where *structure* is

structure	Description	Notes
diagonal	all variances unrestricted all covariances fixed at 0	
unstructured	all variances unrestricted all covariances unrestricted	
identity	all variances equal all covariances fixed at 0	
exchangeable	all variances equal all covariances equal	
zero	all variances fixed at 0 all covariances fixed at 0	
pattern(*matname*)	covariances (variances) unrestricted if *matname*$[i, j] \geq$. covariances (variances) equal if *matname*$[i, j] =$ *matname*$[k, l]$	(1)
fixed(*matname*)	covariances (variances) unrestricted if *matname*$[i, j] \geq$. covariances (variances) fixed at *matname*$[i, j]$ otherwise	(2)

Notes:

(1) Only elements in the lower triangle of *matname* are used. All values in *matname* are interpreted as the floor() of the value if noninteger values appear. Row and column stripes of *matname* are ignored.

(2) Only elements on the lower triangle of *matname* are used. Row and column stripes of *matname* are ignored.

Option

covstructure(*variables*, *structure*) is used either to modify the covariance structure among the exogenous variables of your model or to modify the covariance structure among the error variables of your model.

You may specify the covstructure() option multiple times.

The default covariance structure for exogenous variables is covstructure(_Ex, unstructured) for sem. There is no simple way in this notation to write the default for gsem.

The default covariance structure for error variables is covstructure(e._En, diagonal) for sem and gsem.

Remarks and examples

See [SEM] **Example 17**.

Standard linear structural equation modeling allows covariances among exogenous variables, both latent and observed, and allows covariances among the error variables. Covariances between exogenous variables and error variables are disallowed (assumed to be 0).

Some authors refer to the covariances among the exogenous variables in standard SEMs as matrix $\mathbf{\Phi}$ and to the covariances among the error variables as matrix $\mathbf{\Psi}$.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **Example 17** — Correlated uniqueness model

Title

sem and gsem option from() — Specifying starting values

Description Syntax Option Remarks and examples Also see

Description

See [SEM] **Intro 12** for a description of starting values.

Starting values are usually not specified. When there are convergence problems, it is often necessary to specify starting values. You can specify starting values by using

1. suboption `init()` as described in [SEM] **sem and gsem path notation**, or by using
2. option `from()` as described here.

Option `from()` is especially convenient for using the solution of one model as starting values for another.

Syntax

{`sem`|`gsem`} ... [, ... `from(`*matname*[, `skip`]`)` ...]

{`sem`|`gsem`} ... [, ... `from(`*svlist*`)` ...]

where *matname* is the name of a Stata matrix and

where *svlist* is a space-separated list of the form

eqname`:`*name* `=` *#*

Option

`skip` is an option of `from(`*matname*`)`. It specifies to ignore any parameters in *matname* that do not appear in the model being fit. If this option is not specified, the existence of such parameters causes `sem` (`gsem`) to issue an error message.

Remarks and examples

Remarks are presented under the following headings:

Syntax 1, especially useful when dealing with convergence problems
Syntax 2, seldom used

Option `from()` can be used with `sem` or `gsem`. We illustrate below using `sem`.

Syntax 1, especially useful when dealing with convergence problems

Say you are attempting to fit

```
. sem your_full_model, ...
```

and are having difficulty with convergence. Following the advice in [SEM] **Intro 12**, you have simplified your model,

```
. sem your_simple_model, ...
```

and that does converge. You now want to use those values as starting values for the full model. Let's imagine that there are 47 estimated parameters in the simple model.

Using the standard `init()` method for specifying starting values, you now have a real job in front of you. You have to type your full model, find all the places where you want to add starting values, and add an `init()` suboption. Just a piece of your full model might read

```
... (y<-L1 L2) (L1->x1 x2) (L2->x3 L4) ...
```

and you need to modify that to read

```
... (y<-(L1, init(14.283984)) L2)                    ///
    (L1->(x1, init(2.666532)) (x2, init(-6.39499)))  ///
    (L2->x3 L4) ...
```

That change handles 3 of the 47 parameters you need to specify.

There is an easier way. Type

```
. sem your_simple_model, ...
. matrix b = e(b)
. sem your_full_model, ... from(b)
```

Here is how this works:

1. You fit the simple model. `sem` stores the resulting parameters in `e(b)`.
2. You store the fitted parameters in Stata matrix `b`.
3. To fit your full model, type the model just as you would usually, and add option `from(b)`. That option tells `sem` to get any starting values it can from Stata matrix `b`. `sem` does that and then follows its usual logic for producing starting values for the remaining parameters.

Just because you use `from(b)` does not mean you cannot specify starting values with `init()`. You can even specify starting values for some of the same parameters. Starting values specified by `init()` take precedence over those obtained from `from()`.

Syntax 2, seldom used

In syntax 2, you specify

```
. sem ..., ... from(eqname:name = # eqname:name = # ...)
```

For instance, you could type

```
. sem ..., ... from(var(X):_cons=10)
```

or you could type

```
. sem ..., ... var(X, init(10))
```

It is usually easier to type the second. See [SEM] **sem and gsem path notation**.

You may combine the two notations. If starting values are specified for a parameter both ways, those specified by `init()` take precedence.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **gsem model description options** — Model description options

[SEM] **sem model description options** — Model description options

[SEM] **gsem estimation options** — Options affecting estimation

[R] **Maximize** — Details of iterative maximization

Title

sem and gsem option reliability() — Fraction of variance not due to measurement error

Description Syntax Option Remarks and examples Also see

Description

Option `reliability()` allows you to specify the fraction of variance not due to measurement error for measurement variables.

Syntax

{`sem`|`gsem`} ... [, ... `reliability(`*varname* # [*varname* # [...]] `)`]

where *varname* is the name of an observed endogenous variable and # is the fraction or percentage of variance not due to measurement error:

```
. {sem|gsem} ..., ... reliability(x1 .8  x2 .9)
. {sem|gsem} ..., ... reliability(x1 80%  x2 90%)
```

Option

`reliability(`*varname* # [...] `)` specifies the reliability for variable *varname*. Reliability is bounded by 0 and 1 and is equal to

$$1 - \frac{\text{noise variance}}{\text{total variance}}$$

The reliability is assumed to be 1 when not specified.

Remarks and examples

See [SEM] **Example 24**.

Remarks are presented under the following headings:

Background
Dealing with measurement error of exogenous variables
Dealing with measurement error of endogenous variables
What can go wrong

Background

Option `reliability()` may be used with `sem` and may be used with `gsem` for Gaussian response variables with the identity link but only in the absence of censoring. We illustrate using `sem`, but we could just as well have used `gsem`.

Variables measured with error have attenuated path coefficients. If we had the model

```
. sem (y<-x)
```

and x were measured with error, then the estimated path coefficient would be biased toward 0. The usual solution to such measurement problems is to find multiple measurements and develop a latent variable from them:

```
. sem (x1 x2 x3<-X) (y<-X)
```

Another solution is available if we know the reliability of x. In that case, we can fit the model

```
. sem (x<-X) (y<-X), reliability(x .9)
```

The two solutions can even be combined:

```
. sem (x1 x2 x3<-X) (y<-X), reliability(x1 .9  x2 .8  x3 .9)
```

Even if you do not know the reliability, you can experiment using different but reasonable values for the reliability and thus determine the sensitivity of your estimation results to the measurement problem.

Dealing with measurement error of exogenous variables

Measurement error is most important when it occurs in exogenous variables, yet the `reliability()` option deals with measurement error of endogenous variables only. By creation of a latent variable, `reliability()` can deal with the measurement error of exogenous variables.

To fit the model `(y<-x)` where x is measured with error, you must introduce a latent variable corresponding to x measured without error. That is, the model `(y<-x)` can be converted into the model `(x<-X)` and `(y<-X)`:

$$x = \alpha_0 + \beta_0 X + e.x$$

$$y = \alpha_1 + \beta_1 X + e.y$$

To fit this model, you type

```
. sem (x<-X) (y<-X), reliability(x .9)
```

`sem` will introduce a normalization constraint, namely, that the path coefficient β_0 for `x<-X` is 1, but that is of no importance. What is important is that the estimate of that path coefficient β_1 of `y<-X` is the coefficient that would be obtained from `y<-x` were x measured without error.

In the above, we specified the measurement part of the model first. Be sure to do that. You might think you could equally well reverse the two terms so that, rather than writing

`(x<-X) (y<-X)` *(correct)*

you could write

`(y<-X) (x<-X)` *(incorrect)*

But you cannot do that unless you write

`(y<-X) (x<-X@1)` *(correct)*

because otherwise results are as if you typed

`(y<-X@1) (x<-X)` *(incorrect)*

All of that is because `sem` places its normalization constraint from the latent variable to the first observed endogenous variable. There is no real error if the terms are interchanged except that you will be surprised by the coefficient of 1 for `y<-X` and (the reciprocal of) the coefficient of interest will be on `x<-X`.

See *How sem (gsem) solves the problem for you* in [SEM] **Intro 4** and see *Default normalization constraints* in [SEM] **sem**.

Dealing with measurement error of endogenous variables

When a variable would already be endogenous before you add the `reliability()` option, it really makes little difference whether you add the `reliability()` option. That is because endogenous variables are assumed to contain error, and if some of that error is measurement error, it is still just an error. Coefficients will be unchanged by the inclusion of the `reliability()` option.

Some variances and covariances will change, but the changes are offsetting in the calculation of path coefficients.

What will change are the standardized coefficients should you ask to see them. That is because the variances are changed.

What can go wrong

Consider a model of `y` on `x`. Say we fit the model with linear regression. If the R^2 of the fit is 0.6, then we know the reliability must be greater than 0.6. R^2 measures the fraction of variance of `y` that is explained by `x`, and the reliability of `x` measures the fraction of the variance of `x` that is not due to measurement error. Measurement error is assumed to be pure noise. It is just not possible that we could explain 0.6 of the variance of `y` by using a variable with reliability of, say, 0.5.

Well, in fact, it is because there is always a chance that the pure noise will correlate with `y`, too. Asymptotically, that probability vanishes, but in finite—especially small—samples, it could happen. Even so, the calculation of the corrected SEM estimates blows up.

If you have convergence problems, you need to check for this. Specify a larger value for the reliability. The problem is you cannot specify a value of 1, and large values such as 0.99999 can lead to a lack of identification. In most cases, you will be able to find a value in between, but the only way to be sure is to remove the `reliability()` option and, if necessary, simplify your model by removing any intermediary latent variables you had to add because of reliability.

If your model converges without reliability, then your measure of reliability is too low. At this point, we have little useful advice for you. Check whether you have the right value, of course. If you do, then there are two possibilities: either the experts who provided that estimate are wrong or you got unlucky in that the measurement error did just happen to correlate with the rest of your data. You will need to evaluate the chances of that for yourself. In any case, you can experiment with higher values of the reliability and at least provide an idea of the sensitivity of your estimates to differing assumptions.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **gsem model description options** — Model description options

[SEM] **sem model description options** — Model description options

[SEM] **Example 24** — Reliability

Title

sem and gsem path notation — Command syntax for path diagrams

Description Syntax Options Remarks and examples Also see

Description

The command syntax for describing your SEM is fully specified by *paths*, `covariance()`, `variance()`, and `means()`. How this works is described below.

If you are using `sem`, also see [SEM] **sem path notation extensions** for documentation of the `group()` option for comparing different groups in the data. The syntax of the elements described below is modified when `group()` is specified.

If you are using `gsem`, also see [SEM] **gsem path notation extensions** for documentation on specification of family-and-link for generalized (nonlinear) response variables, specification of multilevel latent variables, specification of categorical latent variables, and specification of multiple-group models. The syntax of the elements described below is modified when the `group()` option for comparing different groups or the `lclass()` option for categorical latent variables is specified.

Either way, read this section first.

Syntax

`sem` *paths* ... [`, covariance() variance() means()`]

`gsem` *paths* ... [`, covariance() variance() means()`]

paths specifies the direct paths between the variables of your model.

The model to be fit is fully described by *paths*, `covariance()`, `variance()`, and `means()`.

Options

`covariance()` is used to

1. specify that a particular covariance path of your model that usually is assumed to be 0 be estimated,
2. specify that a particular covariance path that usually is assumed to be nonzero is not to be estimated (to be constrained to be 0),
3. constrain a covariance path to a fixed value, such as 0, 0.5, 1, etc., and
4. constrain two or more covariance paths to be equal.

`variance()` does the same as `covariance()` except it does it with variances.

`means()` does the same as `covariance()` except it does it with means.

Remarks and examples

Path notation is used by the `sem` and `gsem` commands to specify the model to be fit, for example,

```
. sem (x1 x2 x3 x4 <- X)
. gsem (L1 -> x1 x2 x3 x4 x5, logit) (L2 -> x6 x7 x8 x9 x10)
```

In the path notation,

1. Latent variables are indicated by a *name* in which at least the first letter is capitalized.
2. Observed variables are indicated by a *name* in which at least the first letter is lowercased. Observed variables correspond to variable names in the dataset.
3. Error variables, while mathematically a special case of latent variables, are considered in a class by themselves. For `sem`, every endogenous variable (whether observed or latent) automatically has an error variable associated with it. For `gsem`, the same is true of Gaussian endogenous variables (and latent variables, which are Gaussian). The error variable associated with endogenous variable *name* is `e.`*name*.
4. Paths between variables are written as

   (*name1* <- *name2*)

   or

   (*name2* -> *name1*)

   There is no significance to which coding is used.
5. Paths between the same variables can be combined: The paths

   (*name1* <- *name2*) (*name1* <- *name3*)

   can be combined as

   (*name1* <- *name2 name3*)

   or as

   (*name2 name3* -> *name1*)

   The paths

   (*name1* <- *name3*) (*name2* <- *name3*)

   can be combined as

   (*name1 name2* <- *name3*)

   or as

   (*name3* -> *name1 name2*)

   The paths

   (*name1* <- *name2 name3*)

   (*name4* <- *name2 name3*)

   may be written as

   (*name1 name4* <- *name2 name3*)

   or as

   (*name2 name3* -> *name1 name4*)

6. Variances and covariances (curved paths) between variables are indicated by options. Variances are indicated by

   `..., ... var(`*name1*`)`

   Covariances are indicated by

   `..., ... cov(`*name1*`*`*name2*`)`

   `..., ... cov(`*name2*`*`*name1*`)`

   There is no significance to the order of the names.

   The actual names of the options are `variance()` and `covariance()`, but they are invariably abbreviated as `var()` and `cov()`, respectively.

   The `var()` and `cov()` options are the same option, so a variance can be typed as

   `..., ... cov(`*name1*`)`

   and a covariance can be typed as

   `..., ... var(`*name1*`*`*name2*`)`

7. Variances may be combined, covariances may be combined, and variances and covariances may be combined.

   If you have

   `..., ... var(`*name1*`) var(`*name2*`)`

   you may code this as

   `..., ... var(`*name1 name2*`)`

   If you have

   `..., ... cov(`*name1*`*`*name2*`) cov(`*name2*`*`*name3*`)`

   you may code this as

   `..., ... cov(`*name1*`*`*name2 name2*`*`*name3*`)`

   All the above combined can be coded as

   `..., ... var(`*name1 name2 name1*`*`*name2 name2*`*`*name3*`)`

   or as

   `..., ... cov(`*name1 name2 name1*`*`*name2 name2*`*`*name3*`)`

8. All variables except endogenous variables are assumed to have a variance; it is only necessary to code the `var()` option if you wish to place a constraint on the variance or specify an initial value. See items 11, 12, 13, and 16 below. (In `gsem`, the variance and covariances of observed exogenous variables are not estimated and thus `var()` cannot be used with them.)

   Endogenous variables have a variance, of course, but that is the variance implied by the model. If *name* is an endogenous variable, then `var(`*name*`)` is invalid. The error variance of the endogenous variable is `var(e.`*name*`)`.

9. Variables mostly default to being correlated:

   a. All exogenous variables are assumed to be correlated with each other, whether observed or latent.

   b. Endogenous variables are never directly correlated, although their associated error variables can be.

   c. All error variables are assumed to be uncorrelated with each other.

You can override these defaults on a variable-by-variable basis with the `cov()` option.

To assert that two variables are uncorrelated that otherwise would be assumed to be correlated, constrain the covariance to be 0:

`..., ... cov(`*name1*`*`*name2*`@0)`

To allow two variables to be correlated that otherwise would be assumed to be uncorrelated, simply specify the existence of the covariance:

`..., ... cov(`*name1*`*`*name2*`)`

This latter is especially commonly done with errors:

`..., ... cov(e.`*name1*`*e.`*name2*`)`

(In `gsem`, you may not use the `cov()` option with observed exogenous variables. You also may not use `cov()` with error terms associated with family Gaussian, link log.)

10. Means of variables are indicated by the following option:

    `..., ... means(`*name*`)`

    Variables mostly default to having nonzero means:

    a. All observed exogenous variables are assumed to have nonzero means. In `sem`, the means can be constrained using the `means()` option, but only if you are performing `noxconditional` estimation; [SEM] **sem option noxconditional**.

    b. Latent exogenous variables are assumed to have mean 0. Means of latent variables are not estimated by default. If you specify enough normalization constraints to identify the mean of a latent exogenous variable, you can specify `means(`*name*`)` to indicate that the mean should be estimated in either.

    c. Endogenous variables have no separate mean. Their means are those implied by the model. The `means()` option may not be used with endogenous variables.

    d. Error variables have mean 0 and this cannot be modified. The `means()` option may not be used with error variables.

    To constrain the mean to a fixed value, such as 57, code

    `..., ... means(`*name*`@57)`

    Separate `means()` options may be combined:

    `..., ... means(`*name1*`@57 `*name2*`@100)`

11. Fixed-value constraints may be specified for a path, variance, covariance, or mean by using `@` (the "at" symbol). For example,

    `(`*name1* `<-` *name2*`@1)`

    `(`*name1* `<-` *name2*`@1 `*name3*`@1)`

    `..., ... var(`*name*`@100)`

    `..., ... cov(`*name1*`*`*name2*`@223)`

    `..., ... cov(`*name1*`@1 `*name2*`@1 `*name1*`*`*name2*`@.8)`

    `..., ... means(`*name*`@57)`

12. Symbolic constraints may be specified for a path, variance, covariance, or mean by using `@` (the "at" symbol). For example,

    (*name1* <- *name2*`@c1`) (*name3* <- *name4*`@c1`)

    ..., ... `var(`*name1*`@c1` *name2*`@c1)`

    ..., ... `cov(`*name1*`@1` *name2*`@1` *name3*`@1` *name1*`*`*name2*`@c1` *name1*`*`*name3*`@c1)`

    ..., ... `means(`*name1*`@c1` *name2*`@c1)`

    (*name1* <- *name2*`@c1`) ..., `var(`*name3*`@c1) means(`*name4*`@c1)`

    Symbolic names are just names from 1 to 32 characters in length. Symbolic constraints constrain equality. For simplicity, all constraints below will have names `c1`, `c2`, ...

13. Linear combinations of symbolic constraints may be specified for a path, variance, covariance, or mean by using `@` (the "at" symbol). For example,

    (*name1* <- *name2*`@c1`) (*name3* <- *name4*`@(2*c1))`

    ..., ... `var(`*name1*`@c1` *name2*`@(c1/2))`

    ..., ... `cov(`*name1*`@1` *name2*`@1` *name3*`@1` *name1*`*`*name2*`@c1` *name1*`*`*name2*`@(c1/2))`

    ..., ... `means(`*name1*`@c1` *name2*`@(3*c1+10))`

    (*name1* <- *name2*`@(c1/2))` ..., `var(`*name3*`@c1) means(`*name4*`@(2*c1))`

14. All equations in the model are assumed to have an intercept (to include observed exogenous variable `_cons`) unless the `noconstant` option (abbreviation `nocons`) is specified, and then all equations are assumed not to have an intercept (not to include `_cons`). (There are some exceptions to this in `gsem` because some generalized linear models have no intercept or even the concept of an intercept.)

    Regardless of whether `noconstant` is specified, you may explicitly refer to observed exogenous variable `_cons`.

    The following path specifications are ways of writing the same model:

    (*name1* <- *name2*) (*name1* <- *name3*)

    (*name1* <- *name2*) (*name1* <- *name3*) (*name1* <- `_cons`)

    (*name1* <- *name2 name3*)

    (*name1* <- *name2 name3* `_cons`)

    There is no reason to explicitly specify `_cons` unless you have also specified the `noconstant` option and want to include `_cons` in some equations but not others, or regardless of whether you specified the `noconstant` option, you want to place a constraint on its path coefficient. For example,

    (*name1* <- *name2 name3* `_cons@c1`) (*name4* <- *name5* `_cons@c1`)

15. The `noconstant` option may be specified globally or within a path specification. That is,

    (*name1* <- *name2 name3*) (*name4* <- *name5*)`,` `nocons`

    suppresses the intercepts in both equations. Alternatively,

    (*name1* <- *name2 name3*`,` `nocons`) (*name4* <- *name5*)

    suppresses the intercept in the first equation but not the second, whereas

    (*name1* <- *name2 name3*) (*name4* <- *name5*`,` `nocons`)

    suppresses the intercept in the second equation but not the first.

In addition, consider the equation

```
(name1 <- name2 name3, nocons)
```

This can be written equivalently as

```
(name1 <- name2, nocons) (name1 <- name3, nocons)
```

16. Initial values (starting values) may be specified for a path, variance, covariance, or mean by using the `init(#)` suboption:

```
(name1 <- (name2, init(0)))
(name1 <- (name2, init(0)) name3)
(name1 <- (name2, init(0)) (name3, init(5)))
..., ... var((name3, init(1)))
..., ... cov((name4*name5, init(.5)))
..., ... means((name5, init(0)))
```

The initial values may be combined with symbolic constraints:

```
(name1 <- (name2@c1, init(0)))
(name1 <- (name2@c1, init(0)) name3)
(name1 <- (name2@c1, init(0)) (name3@c2, init(5)))
..., ... var((name3@c1, init(1)))
..., ... cov((name4*name5@c1, init(.5)))
..., ... means((name5@c1, init(0)))
```

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **sem path notation extensions** — Command syntax for path diagrams

[SEM] **gsem path notation extensions** — Command syntax for path diagrams

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **Intro 6** — Comparing groups

Title

sem and gsem syntax options — Options affecting interpretation of syntax

Description Syntax Options Remarks and examples Also see

Description

These options affect some minor issues of how `sem` and `gsem` interpret what you type.

Syntax

`sem` *paths* `...,` `...` *syntax_options*

`gsem` *paths* `...,` `...` *syntax_options*

syntax_options	Description
`latent(`*names*`)`	explicitly specify latent variable names
`nocapslatent`	do not treat capitalized *Names* as latent

where *names* is a space-separated list of the names of the latent variables.

Options

`latent(`*names*`)` specifies that *names* is the full set of names of the latent variables. `sem` and `gsem` ordinarily assume that latent variables have the first letter capitalized and observed variables have the first letter lowercased; see [SEM] **sem and gsem path notation**. When you specify `latent(`*names*`)`, `sem` and `gsem` treat the listed variables as the latent variables and all other variables, regardless of capitalization, as observed. `latent()` implies `nocapslatent`.

`nocapslatent` specifies that having the first letter capitalized does not designate a latent variable. This option can be used when fitting models with observed variables only where some observed variables in the dataset have the first letter capitalized.

Remarks and examples

We recommend using the default naming convention. If your dataset contains variables with the first letter capitalized, it is easy to convert the variables to have lowercase names by typing

```
. rename *, lower
```

See [D] **rename group**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

Title

sem estimation options — Options affecting estimation

Description Syntax Options Remarks and examples Also see

Description

These options control how results are obtained.

Syntax

```
sem paths ..., ... estimation_options
```

estimation_options	Description
method(*method*)	*method* may be ml, mlmv, or adf
vce(*vcetype*)	*vcetype* may be oim, eim, opg, sbentler, robust, cluster *clustvar*, bootstrap, or jackknife
nm1	compute sample variance rather than ML variance
noxconditional	compute covariances, etc., of observed exogenous variables
allmissing	for use with method(mlmv)
noivstart	skip calculation of starting values
noestimate	do not fit the model; instead show starting values
maximize_options	control maximization process for specified model; seldom used
satopts(*maximize_options*)	control maximization process for saturated model; seldom used
baseopts(*maximize_options*)	control maximization process for baseline model; seldom used

Options

method() and vce() specify the method used to obtain parameter estimates and the technique used to obtain the variance–covariance matrix of the estimates. See [SEM] **sem option method( )**.

nm1 specifies that the variances and covariances used in the SEM equations be the sample variances (divided by $N - 1$) and not the asymptotic variances (divided by N). This is a minor technical issue of little importance unless you are trying to match results from other software that assumes sample variances. sem assumes asymptotic variances.

noxconditional states that you wish to include the means, variances, and covariances of the observed exogenous variables among the parameters to be estimated by sem. See [SEM] **sem option noxconditional**.

allmissing specifies how missing values should be treated when method(mlmv) is also specified.

Usually, sem omits from the estimation sample observations that contain missing values of any of the observed variables used in the model. method(mlmv), however, can deal with these missing values, and in that case, observations containing missing are not omitted.

Even so, sem, method(mlmv) does omit observations containing .a, .b, ..., .z from the estimation sample. sem assumes you do not want these observations used because the missing value is not missing at random. If you want sem to include these observations in the estimation sample, specify the allmissing option.

noivstart is an arcane option that is of most use to programmers. It specifies that sem is to skip efforts to produce good starting values with instrumental-variable techniques, techniques that require computer time. If you specify this option, you should specify all the starting values. Any starting values not specified will be assumed to be 0 (means, path coefficients, and covariances) or some simple function of the data (variances).

noestimate specifies that the model is not to be fit. Instead, starting values are to be shown and they are to be shown using the coeflegend style of output. An important use of this is to improve starting values when your model is having difficulty converging. You can do the following:

```
. sem ..., ... noestimate
. matrix b = e(b)
. ... (modify elements of b) ...
. sem ..., ... from(b)
```

maximize_options specify the standard and rarely specified options for controlling the maximization process for sem; see [R] **Maximize**. The relevant options for sem are difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, tolerance(#), ltolerance(#), and nrtolerance(#).

satopts(*maximize_options*) is a rarely specified option and is only relevant if you specify the method(mlmv) option. sem reports a test for model versus saturated at the bottom of the output. Thus sem needs to obtain the saturated fit. In the case of method(ml) or method(adf), sem can make a direct calculation. In the other case of method(mlmv), sem must actually fit the saturated model. The maximization options specified inside satopts() control that maximization process. It is rare that you need to specify the satopts() option, even if you find it necessary to specify the overall *maximize_options*.

baseopts(*maximize_options*) is a rarely specified option and an irrelevant one unless you also specify method(mlmv) or method(adf). When fitting the model, sem records information about the baseline model for later use by estat gof, should you use that command. Thus sem needs to obtain the baseline fit. In the case of method(ml), sem can make a direct calculation. In the cases of method(mlmv) and method(adf), sem must actually fit the baseline model. The maximization options specified inside baseopts() control that maximization process. It is rare that you need to specify the baseopts() option even if you find it necessary to specify the overall *maximize_options*.

Remarks and examples

The most commonly specified option among this group is vce(). See [SEM] **sem option method()**, [SEM] **Intro 8**, and [SEM] **Intro 9**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem option method()** — Specifying method and calculation of VCE

[SEM] **sem option noxconditional** — Computing means, etc., of observed exogenous variables

[SEM] **Intro 8** — Robust and clustered standard errors

[SEM] **Intro 9** — Standard errors, the full story

Title

sem group options — Fitting models on different groups

Description Syntax Options Remarks and examples Also see

Description

`sem` can fit combined models across subgroups of the data while allowing some parameters to vary and constraining others to be equal across subgroups. These subgroups could be males and females, age category, and the like.

`sem` performs such estimation when the `group(`*varname*`)` option is specified. The `ginvariant(`*pclassname*`)` option specifies which parameters are to be constrained to be equal across the groups.

Syntax

`sem` *paths* `...,` `...` *group_options*

group_options	Description
`group(`*varname*`)`	fit model for different groups
`ginvariant(`*pclassname*`)`	specify parameters that are equal across groups

pclassname	Description
`scoef`	structural coefficients
`scons`	structural intercepts
`mcoef`	measurement coefficients
`mcons`	measurement intercepts
`serrvar`	covariances of structural errors
`merrvar`	covariances of measurement errors
`smerrcov`	covariances between structural and measurement errors
`meanex`	means of exogenous variables
`covex`	covariances of exogenous variables
`all`	all the above
`none`	none of the above

`ginvariant(mcoef mcons)` is the default if `ginvariant()` is not specified.

`meanex`, `covex`, and `all` exclude the observed exogenous variables (that is, they include only the latent exogenous variables) unless you specify the `noxconditional` option or the `noxconditional` option is otherwise implied; see [SEM] **sem option noxconditional**. This is what you would desire in most cases.

Options

`group(`*varname*`)` specifies that the model be fit as described above. *varname* specifies the name of a numeric variable that records the group to which the observation belongs.

If you are using summary statistics data in place of raw data, *varname* is the name of the group variable as reported by `ssd describe`; see [SEM] **ssd**.

`ginvariant(`*pclassname*`)` specifies which classes of parameters of the model are to be constrained to be equal across groups. The classes are defined above. The default is `ginvariant(mcoef mcons)` if the option is not specified.

Remarks and examples

See [SEM] **Intro 6**, and see [SEM] **Example 20** and [SEM] **Example 23**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Intro 6** — Comparing groups

[SEM] **Example 20** — Two-factor measurement model by group

[SEM] **Example 23** — Specifying parameter constraints across groups

Title

sem model description options — Model description options

Description Syntax Options Remarks and examples Also see

Description

paths and the options above describe the model to be fit by `sem`.

Syntax

`sem` *paths* `...,` `...` *model_description_options*

model_description_options	Description
* `covariance()`	path notation for treatment of covariances; see [SEM] **sem and gsem path notation**
* `variance()`	path notation for treatment of variances; see [SEM] **sem and gsem path notation**
* `means()`	path notation for treatment of means; see [SEM] **sem and gsem path notation**
* `covstructure()`	alternative method to place restrictions on covariances; see [SEM] **sem and gsem option covstructure( )**
`noconstant`	do not fit intercepts
`nomeans`	do not fit means or intercepts
`noanchor`	do not apply default anchoring
`forcenoanchor`	programmer's option
* `reliability()`	reliability of measurement variables; see [SEM] **sem and gsem option reliability( )**
`constraints()`	specify constraints; see [SEM] **sem and gsem option constraints( )**
`from()`	specify starting values; see [SEM] **sem and gsem option from( )**

* Option may be specified more than once.

Options

`covariance()`, `variance()`, and `means()` fully describe the model to be fit. See [SEM] **sem and gsem path notation**.

`covstructure()` provides a convenient way to constrain covariances in your model. Alternatively or in combination, you can place constraints by using the standard path notation. See [SEM] **sem and gsem option covstructure( )**.

`noconstant` specifies that all intercepts be constrained to 0. See [SEM] **sem and gsem path notation**.

`nomeans` specifies that means and intercepts not be fit. The means and intercepts are concentrated out of the function being optimized, which is typically the likelihood function. Results for all other parameters are the same whether or not this option is specified.

This option is seldom specified. sem issues this option to itself when you use summary statistics data that do not include summary statistics for the means.

noanchor specifies that sem is not to check for lack of identification and fill in anchors where needed. sem is instead to issue an error message if anchors would be needed. You specify this option when you believe you have specified the necessary normalization constraints and want to hear about it if you are wrong. See *Identification 2: Normalization constraints (anchoring)* in [SEM] **Intro 4**.

forcenoanchor is similar to noanchor except that rather than issue an error message, sem proceeds to estimation. There is no reason you should specify this option. forcenoanchor is used in testing of sem at StataCorp.

reliability() specifies the fraction of variance not due to measurement error for a variable. See [SEM] **sem and gsem option reliability()**.

constraints() specifies parameter constraints you wish to impose on your model; see [SEM] **sem and gsem option constraints()**. Constraints can also be specified as described in [SEM] **sem and gsem path notation**, and they are usually more conveniently specified using the path notation.

from() specifies the starting values to be used in the optimization process; see [SEM] **sem and gsem option from()**. Starting values can also be specified by using the init() suboption as described in [SEM] **sem and gsem path notation**.

Remarks and examples

To use sem successfully, you need to understand *paths*, covariance(), variance(), and means(); see *Using path diagrams to specify standard linear SEMs* in [SEM] **Intro 2** and [SEM] **sem and gsem path notation**.

covstructure() is often convenient; see [SEM] **sem and gsem option covstructure()**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **sem and gsem option covstructure()** — Specifying covariance restrictions

[SEM] **sem and gsem option reliability()** — Fraction of variance not due to measurement error

[SEM] **sem and gsem option constraints()** — Specifying constraints

[SEM] **sem and gsem option from()** — Specifying starting values

Title

sem option method() — Specifying method and calculation of VCE

Description Syntax Options Remarks and examples Also see

Description

`sem` option `method()` specifies the method used to obtain the estimated parameters.

`sem` option `vce()` specifies the technique used to obtain the variance–covariance matrix of the estimates (VCE), which includes the reported standard errors.

Syntax

`sem ... [ , ... method(`*method*`) vce(`*vcetype*`) ... ]`

method	Description
`ml`	maximum likelihood; the default
`mlmv`	`ml` with missing values
`adf`	asymptotic distribution free

vcetype	Description
`oim`	observed information matrix; the default
`eim`	expected information matrix
`opg`	outer product of gradients
`sbentler`	Satorra–Bentler estimator
`robust`	Huber/White/sandwich estimator
`cluster` *clustvar*	generalized Huber/White/sandwich estimator
`bootstrap [ ,` *bootstrap_options* `]`	bootstrap estimation
`jackknife [ ,` *jackknife_options* `]`	jackknife estimation

`pweights` and `iweights` are not allowed with `sbentler`.

The following combinations of `method()` and `vce()` are allowed:

	`oim`	`eim`	`opg`	`sbentler`	`robust`	`cluster`	`bootstrap`	`jackknife`
`ml`	x	x	x	x	x	x	x	x
`mlmv`	x	x	x		x	x	x	x
`adf`	x	x					x	x

Options

`method(`*method*`)` specifies the method used to obtain parameter estimates. `method(ml)` is the default.

`vce(`*vcetype*`)` specifies the technique used to obtain the VCE. `vce(oim)` is the default.

Remarks and examples

See [SEM] **Intro 4**, [SEM] **Intro 8**, and [SEM] **Intro 9**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Intro 4** — Substantive concepts

[SEM] **Intro 8** — Robust and clustered standard errors

[SEM] **Intro 9** — Standard errors, the full story

[SEM] **Example 26** — Fitting a model with data missing at random

Title

sem option noxconditional — Computing means, etc., of observed exogenous variables

Description Syntax Option Remarks and examples Also see

Description

`sem` has a `noxconditional` option that you may rarely wish to specify. The option is described below.

Syntax

```
sem ... [, ... noxconditional ...]
```

Option

`noxconditional` states that you wish to include the means, variances, and covariances of the observed exogenous variables among the parameters to be estimated by `sem`.

Remarks and examples

Remarks are presented under the following headings:

What is x conditional?
When to specify noxconditional
Option forcexconditional (a technical note)

What is x conditional?

In many cases, `sem` does not include the means, variances, and covariances of observed exogenous variables among the parameters to be estimated. When `sem` omits them, the estimator of the model is said to be x conditional. Rather than estimating the values of the means, variances, and covariances, `sem` uses the separately calculated observed values of those statistics. `sem` does this to save time and memory.

`sem` does not use the x-conditional calculation when it would be inappropriate.

The `noxconditional` option prevents `sem` from using the x-conditional calculation. You specify `noxconditional` on the `sem` command:

```
. sem ..., ... noxconditional
```

Do not confuse the x-conditional calculation with the assumption of conditional normality discussed in [SEM] **Intro 4**. The x-conditional calculation is appropriate even when the assumption of conditional normality is inappropriate.

When to specify noxconditional

It is never inappropriate to specify the noxconditional option. Be aware, however:

1. If you are using the default method(ml), estimated point estimates and standard errors will be the same.
2. If you are using method(adf), estimated point estimates and standard errors will be slightly different, asymptotically equivalent, and there is no reason to prefer one set of estimates over the other.
3. If you are using method(mlmv), the situation is the same as in statement 1.
4. Regardless of the estimation method used, calculation of results will require more computer time and memory. The memory requirements increase quadratically with the total number of estimated parameters in your model. If you have k_1 observed exogenous variables and k_2 latent exogenous variables, the number of added parameters from noxconditional is $k_1 + k_1(k_1 + 1)/2 + k_1k_2$. The resulting total memory requirements can be so great as to require more memory than your computer can provide.

To make statements 1–4 true, there are two cases when sem specifies noxconditional for you:

1. sem defaults to noxconditional whenever you constrain a mean, variance, or covariance of an observed exogenous variable. For example,

```
. sem ..., ... means(x1@m x2@m)
. sem ..., ... var(x1@v x2@v)
. sem ..., ... cov(x1*x2@c x1*x3@c)
. sem ..., ... covstruct(_OEx, diagonal)
```

   See [SEM] **sem and gsem path notation** and [SEM] **sem and gsem option covstructure()**.

2. sem defaults to noxconditional whenever you use method(mlmv) and there are missing values among the observed exogenous variables.

There are only three reasons for you to specify the noxconditional option:

1. Specify noxconditional if you subsequently wish to test means, variances, or covariances of observed exogenous variables with postestimation commands. For example,

```
. sem ..., ... noxconditional
. sem, coeflegend
. test _b[/means(x1)] == _b[/means(x2)]
```

2. Specify noxconditional if you are fitting a model with the group() option.
3. Specify noxconditional if you also specify the ginvariant() option, and you want the ginvariant() classes meanex, covex, or all to include the observed exogenous variables. For example,

```
. sem ..., ... by(agegrp) ginvariant(all) noxconditional
```

You may also wish to specify noxconditional when comparing results with those from other packages. Many packages use the noxconditional approach when using an estimation method other than maximum likelihood (ML). Correspondingly, most packages use the x-conditional calculation when using ML.

Option forcexconditional (a technical note)

In addition to `noxconditional`, `sem` has a `forcexconditional` option:

`sem ... [ , ... forcexconditional ... ]`

This option turns off `sem`'s switching away from the x-conditional calculation when that is required. Do not specify this option unless you are exploring the behavior of x-conditional calculation in cases where it is theoretically inappropriate.

Also see

[SEM] **sem** — Structural equation model estimation command

Title

sem option select( ) — Using sem with summary statistics data

Description Syntax Option Remarks and examples Also see

Description

`sem` may be used with summary statistics data (SSD), data containing only summary statistics such as the means, standard deviations or variances, and correlations and covariances of the underlying, raw data.

You enter SSD with the `ssd` command; see [SEM] **ssd**.

To fit a model with `sem`, there is nothing special you have to do except specify the `select()` option where you would usually specify `if` *exp*.

Syntax

`sem ...[, ... select(#[# ...]) ...]`

Option

`select(#[# ...])` is allowed only when you have SSD in memory. It specifies which groups should be used.

Remarks and examples

See [SEM] **Intro 11**.

`sem` option `select()` is the SSD alternative for `if` *exp* if you only had the underlying, raw data in memory. With the underlying raw data, where you would usually type

```
. sem ... if agegrp==1 | agegrp==3, ...
```

with SSD in memory, you type

```
. sem ..., ... select(1 3)
```

You may select only groups for which you have separate summary statistics recorded in your summary statistics dataset; the `ssd describe` command will list the group variable, if any. See [SEM] **ssd**.

By the way, `select()` may be combined with `sem` option `group()`. Where you might usually type

```
. sem ... if agegrp==1 | agegrp==3, ... group(agegrp)
```

you could type

```
. sem ..., ... select(1 3) group(agegrp)
```

The above restricts `sem` to age groups 1 and 3, so the result will be an estimation of a combined model of age groups 1 and 3 with some coefficients allowed to vary between the groups and other coefficients constrained to be equal across the groups. See [SEM] **sem group options**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Intro 11** — Fitting models with summary statistics data (sem only)

Title

sem path notation extensions — Command syntax for path diagrams

Description Syntax Options Remarks and examples Also see

Description

This entry concerns `sem` only.

The command syntax for describing your SEMs is fully specified by *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()`. How that works is described in [SEM] **sem and gsem path notation** and [SEM] **sem and gsem option covstructure( )**. See those sections before reading this section.

This entry concerns the path features unique to `sem`, and that has to do with the `group()` option for comparing different groups.

Syntax

`sem` *paths* ... [`, covariance() variance() means() group(`*varname*`)`]

`sem` *paths* ... [`, covstructure() means() group(`*varname*`)`]

paths specifies the direct paths between the variables of your model.

The model to be fit is fully described by *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()`.

The syntax of these elements is modified (generalized) when the `group()` option is specified.

Options

`covariance()`, `variance()`, and `means()` are described in [SEM] **sem and gsem path notation**.

`covstructure()` is described in [SEM] **sem and gsem option covstructure( )**.

`group(`*varname*`)` allows models specified with *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()` to be automatically generalized (interacted) with the groups defined by *varname*; see [SEM] **Intro 6**. The syntax of *paths* and the arguments of `covariance()`, `variance()`, `covstructure()`, and `means()` gain an extra syntactical piece when `group()` is specified.

Remarks and examples

The model you wish to fit is fully described by the *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()` that you type. The `group(`*varname*`)` option,

```
. sem ..., ... group(varname)
```

specifies that the model be fit separately for the different values of *varname*. *varname* might be `sex` and then the model would be fit separately for males and females, or *varname* might be something else and perhaps take on more than two values.

Whatever *varname* is, `group(`*varname*`)` defaults to letting some of the path coefficients, covariances, variances, and means of your model vary across the groups and constraining others to be equal. Which parameters vary and which are constrained is described in [SEM] **sem group options**, but that is a minor detail right now.

In what follows, we will assume that *varname* is `mygrp` and takes on three values. Those values are 1, 2, and 3, but they could just as well be 2, 9, and 12.

Consider typing

```
. sem ..., ...
```

and typing

```
. sem ..., ... group(mygrp)
```

Whatever the *paths*, `covariance()`, `variance()`, `covstructure()`, and `means()` are that describe the model, there are now three times as many parameters because each group has its own unique set. In fact, when you give the second command, you are not merely asking for three times the parameters, you are specifying three models, one for each group! In this case, you specified the same model three times without knowing it.

You can vary the model specified across groups.

1. Let's write the model you wish to fit as

   ```
   . sem (a) (b) (c), cov(d) cov(e) var(f)
   ```

   where *a*, *b*, . . . , *f* stand for what you type. In this generic example, we have two `cov()` options just because multiple `cov()` options often occur in real models. When you type

   ```
   . sem (a) (b) (c), cov(d) cov(e) var(f) group(mygrp)
   ```

   results are as if you typed

   ```
   . sem (1: a) (2: a) (3: a)                                  ///
         (1: b) (2: b) (3: b)                                  ///
         (1: c) (2: c) (3: c),                                 ///
                   cov(1: d) cov(2: d) cov(3: d)               ///
                   cov(1: e) cov(2: e) cov(3: e)               ///
                   var(1: f) cov(2: f) cov(3: f)  group(mygrp)
   ```

   The `1:`, `2:`, and `3:` identify the groups for which paths, covariances, or variances are being added, modified, or constrained.

   If `mygrp` contained the unique values 5, 8, and 10 instead of 1, 2, and 3, then `5:` would appear in place of `1:`; `8:` would appear in place of `2:`; and `10:` would appear in place of `3:`.

2. Consider the model

   ```
   . sem (y <- x) (b) (c), cov(d) cov(e) var(f) group(mygrp)
   ```

   If you wanted to constrain the path coefficient `(y <- x)` to be the same across all three groups, you could type

   ```
   . sem (y <- x@c1) (b) (c), cov(d) cov(e) var(f) group(mygrp)
   ```

   See item 12 in [SEM] **sem and gsem path notation** for more examples of specifying constraints. This works because the expansion of `(y <- x@c1)` is

   ```
   (1: y <- x@c1) (2: y <- x@c1) (3: y <- x@c1)
   ```

3. Consider the model

```
. sem (y <- x) (b) (c), cov(d) cov(e) var(f) group(mygrp)
```

If you wanted to constrain the path coefficient `(y <- x)` to be the same in groups 2 and 3, you could type

```
. sem (1: y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),      ///
                                cov(d) cov(e) var(f) group(mygrp)
```

4. Instead of following item 3, you could type

```
. sem (y <- x) (2: y <- x@c1) (3: y <- x@c1) (b) (c),        ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The part `(y <- x) (2: y <- x@c1) (3: y <- x@c1)` expands to

```
(1: y <- x) (2: y <- x) (3: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

and thus the path is defined twice for group 2 and twice for group 3. When a path is defined more than once, the definitions are combined. In this case, the second definition adds more information, so the result is as if you typed

```
(1: y <- x) (2: y <- x@c1) (3: y <- x@c1)
```

5. Instead of following item 3 or item 4, you could type

```
. sem (y <- x@c1) (1: y <- x@c2) (b) (c),      ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The part `(y <- x@c1) (1: y <- x@c2)` expands to

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x@c2)
```

When results are combined from repeated definitions, then definitions that appear later take precedence. In this case, results are as if the expansion read

```
(1: y <- x@c2) (2: y <- x@c1)  (3: y <- x@c1)
```

Thus coefficients for groups 2 and 3 are constrained. The group-1 coefficient is constrained to `c2`. If `c2` appears nowhere else in the model specification, then results are as if the path for group 1 were unconstrained.

6. Instead of following item 3, item 4, or item 5, you could not type

```
. sem (y <- x@c1) (1: y <- x) (b) (c),                ///
                                cov(d) cov(e) var(f) group(mygrp)
```

The expansion of `(y <- x@c1) (1: y <- x)` reads

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)  (1: y <- x)
```

and you might think that `1: y <- x` would replace `1: y <- x@c1`. Information, however, is combined, and even though precedence is given to information appearing later, silence does not count as information. Thus the expanded and reduced specification reads the same as if `1: y <- x` was never specified:

```
(1: y <- x@c1)  (2: y <- x@c1)  (3: y <- x@c1)
```

7. Items 1–6, stated in terms of *paths*, apply equally to what is typed inside the `means()`, `variance()`, `covariance()`, and `covstructure()` options. For instance, if you typed

```
. sem (a) (b) (c), var(e.y@c1) group(mygrp)
```

then you are constraining the variance to be equal across all three groups.

If you wanted to constrain the variance to be equal in groups 2 and 3, you could type

```
. sem (a) (b) (c), var(e.y) var(2: e.y@c1) var(3: e.y@c1), group(mygrp)
```

You could omit typing `var(e.y)` because it is implied. Alternatively, you could type

```
. sem (a) (b) (c), var(e.y@c1) var(1: e.y@c2) group(mygrp)
```

You could not type

```
. sem (a) (b) (c), var(e.y@c1) var(1: e.y) group(mygrp)
```

because silence does not count as information when specifications are combined.

Similarly, if you typed

```
. sem (a) (b) (c), cov(e.y1*e.y2@c1) group(mygrp)
```

then you are constraining the covariance to be equal across all groups. If you wanted to constrain the covariance to be equal in groups 2 and 3, you could type

```
. sem (a) (b) (c), cov(e.y1*e.y2)                                    ///
                   cov(2: e.y1*e.y2@c1) cov(3: e.y1*e.y2@c1) ///
                   group(mygrp)
```

You could not omit `cov(e.y1*e.y2)` because it is not assumed. By default, error variables are assumed to be uncorrelated. Omitting the option would constrain the covariance to be 0 in group 1 and to be equal in groups 2 and 3.

Alternatively, you could type

```
. sem (a) (b) (c), cov(e.y1*e.y2@c1)                    ///
                   cov(1: e.y1*e.y2@c2)                 ///
                   group(mygrp)
```

8. In the examples above, we have referred to the groups with their numeric values, 1, 2, and 3. Had the values been 5, 8, and 10, then we would have used those values.

   If the group variable `mygrp` has a value label, you can use the label to refer to the group. For instance, imagine `mygrp` is labeled as follows:

   ```
   . label define grpvals 1 Male  2 Female  3 "Unknown sex"
   . label values mygrp grpvals
   ```

   We could type

   ```
   . sem (y <- x) (Female: y <- x@c1) (Unknown sex: y <- x@c1) ..., ...
   ```

   or we could type

   ```
   . sem (y <- x) (2: y <- x@c1) (3: y <- x@c1) ..., ...
   ```

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **sem and gsem path notation** — Command syntax for path diagrams

[SEM] **Intro 2** — Learning the language: Path diagrams and command language

[SEM] **Intro 6** — Comparing groups

Title

sem postestimation — Postestimation tools for sem

Postestimation commands margins Remarks and examples Also see

Postestimation commands

The following are the postestimation commands that you can use after estimation by `sem`:

Command	Description
`estat framework`	display results in modeling framework (matrix form)
`estat gof`	overall goodness of fit
`estat ggof`	group-level goodness of fit
`estat eqgof`	equation-level goodness of fit
`estat residuals`	matrices of residuals
`estat ic`	Akaike's and Schwarz's Bayesian information criteria (AIC and BIC)
* `hausman`	Hausman's specification test
`estat mindices`	modification indices (score tests)
`estat scoretests`	score tests
`estat ginvariant`	test of invariance of parameters across groups
`estat eqtest`	equation-level Wald tests
* `lrtest`	likelihood-ratio tests
`test`	Wald tests
`lincom`	linear combination of parameters
`nlcom`	nonlinear combination of parameters
`testnl`	Wald tests of nonlinear hypotheses
`estat stdize:`	test standardized parameters
`estat teffects`	decomposition of effects
`estat stable`	assess stability of nonrecursive systems
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`predict`	factor scores, predicted values, etc.
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`estimates`	cataloging estimation results

* `hausman` and `lrtest` are not appropriate with `svy` estimation results.

For a summary of postestimation features, see [SEM] **Intro 7**.

Postestimation commands such `lincom` and `nlcom` require referencing estimated parameter values, which are accessible via `_b[`*name*`]`. To find out what the names are, type `sem, coeflegend`.

margins

Description for margins

`margins` estimates margins of response for linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [*options*]

statistic	Description
default	linear predictions for all OEn variables
`xb(`*varname*`)`	linear prediction for specified OEn variable
`xb`	not syntactically compatible with `margins`
`xblatent`	not allowed with `margins`
`xblatent(`*varlist*`)`	not allowed with `margins`
`latent`	not allowed with `margins`
`latent(`*varlist*`)`	not allowed with `margins`
`scores`	not allowed with `margins`

Key: OEn = observed endogenous

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

This manual entry concerns `sem`. For information on postestimation features available after `gsem`, see [SEM] **gsem postestimation**.

Also see

[SEM] **sem reporting options** — Options affecting reporting of results

Title

sem reporting options — Options affecting reporting of results

Description Syntax Options Remarks and examples
Reference Also see

Description

These options control how `sem` displays estimation results.

Syntax

`sem` *paths* `...,` `...` *reporting_options*

`sem,` *reporting_options*

reporting_options	Description
`level(#)`	set confidence level; default is `level(95)`
`standardized`	display standardized coefficients and values
`coeflegend`	display coefficient legend
`nocnsreport`	do not display constraints
`nodescribe`	do not display variable classification table
`noheader`	do not display header above parameter table
`nofootnote`	do not display footnotes below parameter table
`notable`	do not display parameter table
`nofvlabel`	display group values rather than value labels
`fvwrap(#)`	allow # lines when wrapping long value labels
`fvwrapon(`*style*`)`	apply *style* for wrapping long value labels; *style* may be `word` or `width`
`byparm`	display results in a single table with rows arranged by parameter
`showginvariant`	report all estimated parameters

Options

`level(#)`; see [R] **Estimation options**.

`standardized` displays standardized values, that is, "beta" values for coefficients, correlations for covariances, and 1s for variances. Standardized values are obtained using model-fitted variances (Bollen 1989, 124–125). We recommend caution in the interpretation of standardized values, especially with multiple groups.

`coeflegend` displays the legend that reveals how to specify estimated coefficients in `_b[]` notation, which you are sometimes required to use when specifying postestimation commands.

`nocnsreport` suppresses the display of the constraints. Fixed-to-zero constraints that are automatically set by `sem` are not shown in the report to keep the output manageable.

`nodescribe` suppresses display of the variable classification table.

`noheader` suppresses the header above the parameter table, the display that reports the final log-likelihood value, number of observations, etc.

`nofootnote` suppresses the footnotes displayed below the parameter table.

`notable` suppresses the parameter table.

`nofvlabel` displays group values rather than value labels.

`fvwrap(#)` specifies how many lines to allow when long value labels must be wrapped. Labels requiring more than # lines are truncated. This option overrides the `fvwrap` setting; see [R] **set showbaselevels**.

`fvwrapon(`*style*`)` specifies whether value labels that wrap will break at word boundaries or break based on available space.

`fvwrapon(word)`, the default, specifies that value labels break at word boundaries.

`fvwrapon(width)` specifies that value labels break based on available space.

This option overrides the `fvwrapon` setting; see [R] **set showbaselevels**.

`byparm` specifies that estimation results with multiple groups be reported in a single table with rows arranged by parameter. The default is to report results in separate tables for each group.

`showginvariant` specifies that each estimated parameter be reported in the parameter table. The default is to report each invariant parameter only once. This option is only effective with the `byparm` option.

Remarks and examples

Any of the above options may be specified when you fit the model or when you redisplay results, which you do by specifying nothing but options after the `sem` command:

```
. sem (...) (...), ...
```
(*original output displayed*)

```
. sem
```
(*output redisplayed*)

```
. sem, standardized
```
(*standardized output displayed*)

```
. sem, coeflegend
```
(*coefficient-name table displayed*)

```
. sem
```
(*output redisplayed*)

Reference

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Example 8** — Testing that coefficients are equal, and constraining them

[SEM] **Example 16** — Correlation

Title

sem ssd options — Options for use with summary statistics data

Description Syntax Options Remarks and examples Also see

Description

Data are sometimes available in summary statistics form only. These summary statistics include means, standard deviations or variances, and correlations or covariances. These summary statistics can be used by `sem` in place of the underlying raw data.

Syntax

`sem` *paths* `...,` `...` *ssd_options*

ssd_options	Description
`select()`	alternative to `if` *exp* for SSD
`forcecorrelations`	allow groups and pooling of SSD correlations

Options

`select()` is an alternative to `if` *exp* when you are using summary statistics data (SSD). Where you might usually type

```
. sem ... if agegrp==1 | agegrp==3 | agegrp==5, ...
```

with SSD in memory, you type

```
. sem ..., ... select(1 3 5)
```

See [SEM] **sem option select( )** and [SEM] **Intro 11**.

`forcecorrelations` tells `sem` that it may make calculations that would usually be considered suspicious with SSD that contain only a subset of means, variances (standard deviations), and covariances (correlations). Do not specify this option unless you appreciate the statistical issues that we are about to discuss. There are two cases where `forcecorrelations` is relevant.

In the first case, `sem` is unwilling to produce `group()` estimates if one or more (usually all) of the groups have correlations only defined. You can override that by specifying `forcecorrelations`, and `sem` will assume unit variances for the group or groups that have correlations only. Doing this is suspect unless you make `ginvariant()` all parameters that are dependent on covariances or unless you truly know that the variances are indeed 1.

In the second case, `sem` is unwilling to pool across groups unless you have provided means and covariances (or means and correlations and standard deviations or variances). Without that information, should the need for pooling arise, `sem` issues an error message. The `forcecorrelations` option specifies that `sem` ignore its rule and pool correlation matrices, treating correlations as if they were covariances when variances are not defined and treating means as if they were 0 when means are not defined. The only justification for making the calculation in this way is that variances truly are 1 and means truly are 0.

Understand that there is nothing wrong with using pure correlation data, or covariance data without the means, so long as you fit models for individual groups. Doing anything across groups basically requires that `sem` have the covariance and mean information.

Remarks and examples

See [SEM] **Intro 11**.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **Intro 11** — Fitting models with summary statistics data (sem only)

[SEM] **ssd** — Making summary statistics data (sem only)

Title

ssd — Making summary statistics data (sem only)

Description Syntax Options Remarks and examples Stored results Also see

Description

`ssd` allows you to enter SSD to fit SEMs and allows you to create SSD from original, raw data to publish or to send to others (and thus preserve participant confidentiality). Data created by `ssd` may be used with `sem` but not `gsem`.

Syntax

To enter summary statistics data (SSD), the commands are

```
ssd init varlist

ssd set [#] observations #

ssd set [#] means vector

ssd set [#] {variances|sd} vector

ssd set [#] {covariances|correlations} matrix

ssd addgroup varname                    (to add second group)

ssd addgroup                            (to add subsequent groups)

ssd unaddgroup #                        (to remove last group)

ssd status [#]                          (to review status)
```

To build SSD from raw data, the command is

```
ssd build varlist [, group(varname) clear]
```

To review the contents of SSD, the commands are

```
ssd status [#]

ssd describe

ssd list [#]
```

In an emergency (ssd will tell you when), you may use

```
ssd repair
```

In the above,

A *vector* can be any of the following:

1. A space-separated list of numbers, for example,

```
. ssd set means 1 2 3
```

2. (stata) *matname*
where *matname* is the name of a Stata $1 \times k$ or $k \times 1$ matrix, for example,

```
. ssd set variances (stata) mymeans
```

3. (mata) *matname*
where *matname* is the name of a Mata $1 \times k$ or $k \times 1$ matrix, for example,

```
. ssd set sd (mata) mymeans
```

A *matrix* can be any of the following:

1. A space-separated list of numbers corresponding to the rows of the matrix, with backslashes (\) between rows. The numbers are either the lower triangle and diagonal or the diagonal and upper triangle of the matrix, for example,

```
. ssd set correlations 1 \ .2 1 \ .3 .5 1
```

or

```
. ssd set correlations 1 .2 .3 \ 1 .5 \ 1
```

2. (ltd) # # ...
which is to say, a space-separated list of numbers corresponding to the lower triangle and diagonal of the matrix, without backslashes between rows, for example,

```
. ssd set correlations (ltd) 1  .2 1  .3 .5 1
```

3. (dut) # # ...
which is to say, a space-separated list of numbers corresponding to the diagonal and upper triangle of the matrix, without backslashes between rows, for example,

```
. ssd set correlations (dut) 1 .2 .3   1 .5   1
```

4. (stata) *matname*
where *matname* is the name of a Stata $k \times k$ symmetric matrix, for example,

```
. ssd set correlations (stata) mymat
```

5. (mata) *matname*
where *matname* is the name of a Mata $k \times k$ symmetric matrix, for example,

```
. ssd set correlations (mata) mymat
```

Options

`group(`*varname*`)` is for use with `ssd build`. It specifies that separate groups of summary statistics be produced for each value of *varname*.

`clear` is for use with `ssd build`. It specifies that it is okay to replace the data in memory with SSD even if the original dataset has not been saved since it was last changed.

Remarks and examples

See

[SEM] **Intro 11** Fitting models with summary statistics data (sem only)

for an introduction, and see

[SEM] **Example 2** Creating a dataset from published covariances
[SEM] **Example 19** Creating multiple-group summary statistics data
[SEM] **Example 25** Creating summary statistics data from raw data

A summary statistics dataset is different from a regular, raw Stata dataset. Be careful not to use standard Stata data-manipulation commands with SSD in memory. The commands include

```
generate
replace
merge
append
drop
set obs
```

to mention a few. You may, however, use `rename` to change the names of the variables.

The other data-manipulation commands can damage your summary statistics dataset. If you make a mistake and do use one of these commands, do not attempt to repair the data yourself. Let `ssd` repair your data by typing

```
. ssd repair
```

`ssd` is usually successful as long as variables or observations have not been dropped.

Every time you use `ssd`, even for something as trivial as describing or listing the data, `ssd` verifies that the data are not corrupted. If `ssd` finds that they are, it suggests you type `ssd repair`:

```
. ssd describe
SSD corrupt
    The summary statistics data should [ssd
    describes the problem].  The data may be fixable;
    type ssd repair.
. ssd repair
  (data repaired)
. ssd describe
 (usual output appears)
```

In critical applications, we also recommend you digitally sign your summary statistics dataset:

```
. datasignature set
5:5(65336):3718404259:2275399871       (data signature set)
```

Then at any future time, you can verify the data are still just as they should be:

```
. datasignature confirm
(data unchanged since 30jun2018 15:32)
```

The data signature is a function of the variable names. If you rename a variable—something that is allowed—then the data signature will change:

```
. rename varname newname
. datasignature confirm
  data have changed since 30jun2018 15:32
r(9);
```

In that case, you can re-sign the data:

```
. datasignature set, reset
5:5(71728):3718404259:2275399871       (data signature set)
```

Before re-signing, however, if you want to convince yourself that the data are unchanged except for the variable name, type `datasignature report`. It is the part of the signature in parentheses that has to do with the variable names. `datasignature report` will tell you what the new signature would be, and you can verify that the other components of the signature match.

See [D] **datasignature**.

Stored results

`ssd describe` stores the following in `r()`:

Scalars

`r(N)`	number of observations (total across groups)
`r(k)`	number of variables (excluding group variable)
`r(G)`	number of groups
`r(complete)`	`1` if complete, `0` otherwise
`r(complete_means)`	`1` if complete means, `0` otherwise
`r(complete_covariances)`	`1` if complete covariances, `0` otherwise

Macros

`r(v#)`	variable names (excluding group variable)
`r(groupvar)`	name of group variable (if there is one)

Also see

[SEM] **Intro 11** — Fitting models with summary statistics data (sem only)

[D] **datasignature** — Determine whether data have changed

[SEM] **Example 2** — Creating a dataset from published covariances

[SEM] **Example 19** — Creating multiple-group summary statistics data

[SEM] **Example 25** — Creating summary statistics data from raw data

Title

test — Wald test of linear hypotheses

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`test` is a postestimation command for use after `sem`, `gsem`, and other Stata estimation commands.

`test` performs the Wald test of the hypothesis or hypotheses that you specify. In the case of `sem` and `gsem`, you must use the `_b[]` coefficient notation.

See [R] **test**. Also documented there is `testparm`. `testparm` cannot be used after `sem` or `gsem` because its syntax hinges on use of shortcuts for referring to coefficients.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Wald tests of linear hypotheses

Syntax

`test` *coeflist*

`test` *exp* `=` *exp* [`= ...`]

`test` [*eqno*] [`:` *coeflist*]

`test` [*eqno* `=` *eqno* [`= ...`]] [`:` *coeflist*]

`test` `(`*spec*`)` [`(`*spec*`) ...`] [`,` *test_options*]

Options

See *Options for test* in [R] **test**.

Remarks and examples

See [SEM] **Example 8** and [SEM] **Example 16**.

`test` works in the metric of SEM, which is to say path coefficients, variances, and covariances. If you want to frame your tests in terms of standardized coefficients and correlations and you fit your model with `sem`, not `gsem`, then prefix `test` with `estat stdize:`; see [SEM] **estat stdize**.

Stored results

See *Stored results* in [R] **test**.

Also see

[SEM] **Example 8** — Testing that coefficients are equal, and constraining them

[SEM] **Example 16** — Correlation

[R] **test** — Test linear hypotheses after estimation

[SEM] **estat stdize** — Test standardized parameters

[SEM] **estat eqtest** — Equation-level tests that all coefficients are zero

[SEM] **lincom** — Linear combinations of parameters

[SEM] **lrtest** — Likelihood-ratio test of linear hypothesis

Title

testnl — Wald test of nonlinear hypotheses

Description Menu Syntax Options
Remarks and examples Stored results Also see

Description

`testnl` is a postestimation command for use after `sem`, `gsem`, and other Stata estimation commands.

`testnl` performs the Wald test of the nonlinear hypothesis or hypotheses. In the case of `sem` and `gsem`, you must use the `_b[]` coefficient notation.

Menu

Statistics > SEM (structural equation modeling) > Testing and CIs > Wald tests of nonlinear hypotheses

Syntax

`testnl` *exp* `=` *exp* [`= ...`] [`,` *options*]

`testnl (`*exp* `=` *exp* [`= ...`]`)` [`(`*exp* `=` *exp* [`= ...`]`) ...`] [`,` *options*]

Options

See *Options* in [R] **testnl**.

Remarks and examples

`testnl` works in the metric of SEM, which is to say path coefficients, variances, and covariances. If you want to frame your tests in terms of standardized coefficients and correlations and you fit the model with `sem`, not `gsem`, then prefix `testnl` with `estat stdize:`; see [SEM] **estat stdize**.

❑ Technical note

`estat stdize:` is unnecessary because, with `testnl`, everywhere you wanted a standardized coefficient or correlation, you could just type the formula. If you did that, you would get the same answer except for numerical precision. In this case, the answer produced with the `estat stdize:` prefix will be a little more accurate because `estat stdize:` is able to substitute an analytic derivative in one part of the calculation where `testnl`, doing the whole thing itself, would be forced to use a numeric derivative.

❑

Stored results

See *Stored results* in [R] **testnl**.

Also see

[R] **testnl** — Test nonlinear hypotheses after estimation

[SEM] **test** — Wald test of linear hypotheses

[SEM] **lrtest** — Likelihood-ratio test of linear hypothesis

[SEM] **estat stdize** — Test standardized parameters

[SEM] **estat eqtest** — Equation-level tests that all coefficients are zero

[SEM] **nlcom** — Nonlinear combinations of parameters

Glossary

ADF, **method(adf)**. ADF stands for asymptotic distribution free and is a method used to obtain fitted parameters for standard linear SEMs. ADF is used by `sem` when option `method(adf)` is specified. Other available methods are ML, QML, and MLMV.

anchoring, **anchor variable**. A variable is said to be the anchor of a latent variable if the path coefficient between the latent variable and the anchor variable is constrained to be 1. `sem` and `gsem` use anchoring as a way of normalizing latent variables and thus identifying the model.

baseline model. A baseline model is a covariance model—a model of fitted means and covariances of observed variables without any other paths—with most of the covariances constrained to be 0. That is, a baseline model is a model of fitted means and variances but typically not all the covariances. Also see *saturated model*. Baseline models apply only to standard linear SEMs.

Bentler–Weeks matrices. The Bentler and Weeks (1980) formulation of standard linear SEMs places the results in a series of matrices organized around how results are calculated. See [SEM] **estat framework**.

bootstrap, **vce(bootstrap)**. The bootstrap is a replication method for obtaining variance estimates. Consider an estimation method E for estimating θ. Let $\widehat{\theta}$ be the result of applying E to dataset D containing N observations. The bootstrap is a way of obtaining variance estimates for $\widehat{\theta}$ from repeated estimates $\widehat{\theta}_1$, $\widehat{\theta}_2$, ..., where each $\widehat{\theta}_i$ is the result of applying E to a dataset of size N drawn with replacement from D. See [SEM] **sem option method( )** and [R] **bootstrap**.

`vce(bootstrap)` is allowed with `sem` but not `gsem`. You can obtain bootstrap results by prefixing the `gsem` command with `bootstrap:`, but remember to specify `bootstrap`'s `cluster()` and `idcluster()` options if you are fitting a multilevel model. See [SEM] **Intro 9**.

Builder. The Builder is Stata's graphical interface for building `sem` and `gsem` models. The Builder is also known as the SEM Builder. See [SEM] **Intro 2**, [SEM] **Builder**, and [SEM] **Builder, generalized**.

categorical latent variable. A categorical latent variable has levels that represent unobserved groups in the population. Latent classes are identified with the levels of the categorical latent variables and may represent healthy and unhealthy individuals, consumers with different buying preferences, or different motivations for delinquent behavior. Categorical latent variables are allowed in `gsem` but not in `sem`. See [SEM] **gsem lclass options** and [SEM] **Intro 2**.

CFA, **CFA models**. CFA stands for confirmatory factor analysis. It is a way of analyzing measurement models. CFA models is a synonym for measurement models.

cluster, **vce(cluster clustvar)**. Cluster is the name we use for the generalized Huber/White/sandwich estimator of the VCE, which is the `robust` technique generalized to relax the assumption that errors are independent across observations to be that they are independent across clusters of observations. Within cluster, errors may be correlated.

Clustered standard errors are reported when `sem` or `gsem` option `vce(cluster` *clustvar*`)` is specified. The other available techniques are OIM, OPG, robust, bootstrap, and jackknife. Also available for `sem` only is EIM.

coefficient of determination. The coefficient of determination is the fraction (or percentage) of variation (variance) explained by an equation of a model. The coefficient of determination is thus like R^2 in linear regression.

command language. Stata's `sem` and `gsem` commands provide a way to specify SEMs. The alternative is to use the Builder to draw path diagrams; see [SEM] **Intro 2**, [SEM] **Builder**, and [SEM] **Builder, generalized**.

complementary log-log regression. Complementary log-log regression is a term for generalized linear response functions that are family Bernoulli, link clog-log. It is used for binary outcome data. Complementary log-log regression is also known in Stata circles as clog-log regression or just clog-log. See *generalized linear response functions*.

conditional normality assumption. See *normality assumption, joint and conditional*.

constraints. See *parameter constraints*.

continuous latent variable. A continuous latent variable is an unobserved variable, such as mathematical ability, with values that are assumed to follow a continuous distribution. Both `sem` and `gsem` allow continuous latent variables that are assumed to follow a Gaussian distribution with a mean and variance that are either estimated or constrained to a specific value for identification. See *identification*.

correlated uniqueness model. A correlated uniqueness model is a kind of measurement model in which the errors of the measurements have a structured correlation. See [SEM] **Intro 5**.

crossed-effects models. See *multilevel models*.

curved path. See *path*.

degree-of-freedom adjustment. In estimates of variances and covariances, a finite-population degree-of-freedom adjustment is sometimes applied to make the estimates unbiased.

Let's write an estimated variance as $\widehat{\sigma}_{ii}$ and write the "standard" formula for the variance as $\widehat{\sigma}_{ii} = S_{ii}/N$. If $\widehat{\sigma}_{ii}$ is the variance of observable variable x_i, it can readily be proven that S_{ii}/N is a biased estimate of the variances in samples of size N and that $S_{ii}/(N-1)$ is an unbiased estimate. It is usual to calculate variances with $S_{ii}/(N-1)$, which is to say the "standard" formula has a multiplicative degree-of-freedom adjustment of $N/(N-1)$ applied to it.

If $\widehat{\sigma}_{ii}$ is the variance of estimated parameter β_i, a similar finite-population degree-of-freedom adjustment can sometimes be derived that will make the estimate unbiased. For instance, if β_i is a coefficient from a linear regression, an unbiased estimate of the variance of regression coefficient β_i is $S_{ii}/(N-p-1)$, where p is the total number of regression coefficients estimated excluding the intercept. In other cases, no such adjustment can be derived. Such estimators have no derivable finite-sample properties, and one is left only with the assurances provided by its provable asymptotic properties. In such cases, the variance of coefficient β_i is calculated as S_{ii}/N, which can be derived on theoretical grounds. SEM is an example of such an estimator.

SEM is a remarkably flexible estimator and can reproduce results that can sometimes be obtained by other estimators. SEM might produce asymptotically equivalent results, or it might produce identical results depending on the estimator. Linear regression is an example in which `sem` and `gsem` produce the same results as `regress`. The reported standard errors, however, will not look identical because the linear-regression estimates have the finite-population degree-of-freedom adjustment applied to them and the SEM estimates do not. To see the equivalence, you must undo the adjustment on the reported linear regression standard errors by multiplying them by $\sqrt{\{(N-p-1)/N\}}$.

direct, **indirect**, and **total effects**. Consider the following system of equations:

$$
\begin{aligned}
y_1 &= b_{10} + b_{11}y_2 + b_{12}x_1 + b_{13}x_3 + e_1 \\
y_2 &= b_{20} + b_{21}y_3 + b_{22}x_1 + b_{23}x_4 + e_2 \\
y_3 &= b_{30} + \qquad\quad b_{32}x_1 + b_{33}x_5 + e_3
\end{aligned}
$$

The total effect of x_1 on y_1 is $b_{12} + b_{11}b_{22} + b_{11}b_{21}b_{32}$. It measures the full change in y_1 based on allowing x_1 to vary throughout the system.

The direct effect of x_1 on y_1 is b_{12}. It measures the change in y_1 caused by a change in x_1 holding other endogenous variables—namely, y_2 and y_3—constant.

The indirect effect of x_1 on y_1 is obtained by subtracting the total and direct effect and is thus $b_{11}b_{22} + b_{11}b_{21}b_{32}$.

EIM, **vce(eim)**. EIM stands for expected information matrix, defined as the inverse of the negative of the expected value of the matrix of second derivatives, usually of the log-likelihood function. The EIM is an estimate of the VCE. EIM standard errors are reported when `sem` option `vce(eim)` is specified. EIM is available only with `sem`. The other available techniques for `sem` are OIM, OPG, robust, cluster, bootstrap, and jackknife.

endogenous variable. A variable, observed or latent, is endogenous (determined by the system) if any path points to it. Also see *exogenous variable*.

error, **error variable**. The error is random disturbance e in a linear equation:

$$y = b_0 + b_1x_1 + b_2x_2 + \cdots + e$$

An error variable is an unobserved exogenous variable in path diagrams corresponding to e. Mathematically, error variables are just another example of latent exogenous variables, but in `sem` and `gsem`, error variables are considered to be in a class by themselves. All (Gaussian) endogenous variables—observed and latent—have a corresponding error variable. Error variables automatically and inalterably have their path coefficients fixed to be 1. Error variables have a fixed naming convention in the software. If a variable is the error for (observed or latent) endogenous variable `y`, then the residual variable's name is `e.y`.

In `sem` and `gsem`, error variables are uncorrelated with each other unless explicitly indicated otherwise. That indication is made in path diagrams by drawing a curved path between the error variables and is indicated in command notation by including `cov(e.`*name1*`*e.`*name2*`)` among the options specified on the `sem` command. In `gsem`, errors for family Gaussian, link log responses are not allowed to be correlated.

estimation method. There are a variety of ways that one can solve for the parameters of an SEM. Different methods make different assumptions about the data-generation process, so it is important that you choose a method appropriate for your model and data; see [SEM] **Intro 4**.

exogenous variable. A variable, observed or latent, is exogenous (determined outside the system) if paths only originate from it or, equivalently, no path points to it. In this manual, we do not distinguish whether exogenous variables are strictly exogenous—that is, uncorrelated with the errors. Also see *endogenous variable*.

family distribution. See *generalized linear response functions*.

fictional data. Fictional data are data that have no basis in reality even though they might look real; they are data that are made up for use in examples.

finite mixture model. A finite mixture model (FMM) is a latent class model in which parameters of a regression model are allowed to vary across classes. The regression model may have a linear or generalized linear response function.

first- and second-order latent variables. If a latent variable is measured by other latent variables only, the latent variable that does the measuring is called first-order latent variable, and the latent variable being measured is called the second-order latent variable.

first-, second-, and higher-level (latent) variables. Consider a multilevel model of patients within doctors within hospitals. First-level variables are variables that vary at the observational (patient) level. Second-level variables vary across doctors but are constant within doctors. Third-level variables vary across hospitals but are constant within hospitals. This jargon is used whether variables are latent or not.

full joint and conditional normality assumption. See *normality assumption, joint and conditional*.

gamma regression. Gamma regression is a term for generalized linear response functions that are family gamma, link log. It is used for continuous, nonnegative, positively skewed data. Gamma regression is also known as log-gamma regression. See *generalized linear response functions*.

Gaussian regression. Gaussian regression is another term for linear regression. It is most often used when referring to generalized linear response functions. In that framework, Gaussian regression is family Gaussian, link identity. See *generalized linear response functions*.

generalized linear response functions. Generalized linear response functions include linear functions and include functions such as probit, logit, multinomial logit, ordered probit, ordered logit, Poisson, and more.

These generalized linear functions are described by a link function $g(\cdot)$ and statistical distribution F. The link function $g(\cdot)$ specifies how the response variable y_i is related to a linear equation of the explanatory variables, $\mathbf{x}_i\boldsymbol{\beta}$, and the family F specifies the distribution of y_i:

$$g\{E(y_i)\} = \mathbf{x}_i\boldsymbol{\beta}, \qquad y_i \sim F$$

If we specify that $g(\cdot)$ is the identity function and F is the Gaussian (normal) distribution, then we have linear regression. If we specify that $g(\cdot)$ is the logit function and F the Bernoulli distribution, then we have logit (logistic) regression.

In this generalized linear structure, the family may be Gaussian, gamma, Bernoulli, binomial, Poisson, negative binomial, ordinal, or multinomial. The link function may be the identity, log, logit, probit, or complementary log-log.

`gsem` fits models with generalized linear response functions.

generalized method of moments. Generalized method of moments (GMM) is a method used to obtain fitted parameters. In this documentation, GMM is referred to as ADF, which stands for asymptotic distribution free and is available for use with `sem`. Other available methods for use with `sem` are ML, QML, ADF, and MLMV.

The SEM moment conditions are cast in terms of second moments, not the first moments used in many other applications associated with GMM.

generalized SEM. Generalized SEM is a term we have coined to mean SEM optionally allowing generalized linear response functions, multilevel models, or categorical latent variables. `gsem` fits generalized SEMs.

GMM. See *generalized method of moments*.

goodness-of-fit statistic. A goodness-of-fit statistic is a value designed to measure how well the model reproduces some aspect of the data the model is intended to fit. SEM reproduces the first- and second-order moments of the data, with an emphasis on the second-order moments, and thus goodness-of-fit statistics appropriate for use after `sem` compare the predicted covariance matrix (and mean vector) with the matrix (and vector) observed in the data.

gsem. `gsem` is the Stata command that fits generalized SEMs. Also see *sem*.

GUI. See *Builder*.

identification. Identification refers to the conceptual constraints on parameters of a model that are required for the model's remaining parameters to have a unique solution. A model is said to be unidentified if these constraints are not supplied. These constraints are of two types: substantive constraints and normalization constraints.

Normalization constraints deal with the problem that one scale works as well as another for each continuous latent variable in the model. One can think, for instance, of propensity to write software as being measured on a scale of 0 to 1, 1 to 100, or any other scale. The normalization constraints are the constraints necessary to choose one particular scale. The normalization constraints are provided automatically by `sem` and `gsem` by anchoring with unit loadings.

Substantive constraints are the constraints you specify about your model so that it has substantive content. Usually, these constraints are 0 constraints implied by the paths omitted, but they can include explicit parameter constraints as well. It is easy to write a model that is not identified for substantive reasons; see [SEM] **Intro 4**.

indicator variables, **indicators**. The term "indicator variable" has two meanings. An indicator variable is a 0/1 variable that contains whether something is true. The other usage is as a synonym for measurement variables.

indirect effects. See *direct, indirect, and total effects*.

initial values. See *starting values*.

intercept. An intercept for the equation of endogenous variable y, observed or latent, is the path coefficient from `_cons` to y. `_cons` is Stata-speak for the built-in variable containing 1 in all observations. In SEM-speak, `_cons` is an observed exogenous variable.

jackknife, **vce(jackknife)**. The jackknife is a replication method for obtaining variance estimates. Consider an estimation method E for estimating θ. Let $\widehat{\theta}$ be the result of applying E to dataset D containing N observations. The jackknife is a way of obtaining variance estimates for $\widehat{\theta}$ from repeated estimates $\widehat{\theta}_1, \widehat{\theta}_2, \ldots, \widehat{\theta}_N$, where each $\widehat{\theta}_i$ is the result of applying E to D with observation i removed. See [SEM] **sem option method()** and [R] **jackknife**.

`vce(jackknife)` is allowed with `sem` but not `gsem`. You can obtain jackknife results by prefixing the `gsem` command with `jackknife:`, but remember to specify `jackknife`'s `cluster()` and `idcluster()` options if you are fitting a multilevel model. See [SEM] **Intro 9**.

joint normality assumption. See *normality assumption, joint and conditional*.

Lagrange multiplier test. Synonym for score test.

latent class analysis. Latent class analysis is useful for identifying and understanding unobserved groups in a population. When performing a latent class analysis, we fit models that include a categorical latent variable with levels called latent classes that correspond to the unobserved groups. In latent class analysis, we can compare models with differing numbers of latent classes and different sets of constraints on parameters to determine the best-fitting model. For a given model, we can compare parameter estimates across classes. We can estimate the proportion of the population in each latent class. And we can predict the probabilities that the individuals in our sample belong to each latent class.

latent class model. Any model with a categorical latent variable that is fit as part of a latent class analysis. In some literature, latent class models are more narrowly defined to include only categorical latent variables and the binary or categorical observed variables that are indicators of class membership, but we do not make such a restriction. See [SEM] **Intro 5**.

latent cluster model. A type of latent class model with continuous observed outcomes.

latent growth model. A latent growth model is a kind of measurement model in which the observed values are collected over time and are allowed to follow a trend. See [SEM] **Intro 5**.

latent profile model. A type of latent class model with continuous observed outcomes.

latent variable. A variable is latent if it is not observed. A variable is latent if it is not in your dataset but you wish it were. You wish you had a variable recording the propensity to commit violent crime, or socioeconomic status, or happiness, or true ability, or even income accurately recorded. Latent variables are sometimes described as imagined variables.

In the software, latent variables are usually indicated by having at least their first letter capitalized.

Also see *continuous latent variable*, *categorical latent variable*, *first- and second-order latent variables*, *first-, second-, and higher-level (latent) variables*, and *observed variables*.

linear regression. Linear regression is a kind of SEM in which there is a single equation. See [SEM] **Intro 5**.

link function. See *generalized linear response functions*.

logit regression. Logit regression is a term for generalized linear response functions that are family Bernoulli, link logit. It is used for binary outcome data. Logit regression is also known as logistic regression and also simply as logit. See *generalized linear response functions*.

manifest variables. Synonym for observed variables.

measure, **measurement**, **x a measurement of X**, **x measures X**. See *measurement variables*.

measurement models, **measurement component**. A measurement model is a particular kind of model that deals with the problem of translating observed values to values suitable for modeling. Measurement models are often combined with structural models and then the measurement model part is referred to as the measurement component. See [SEM] **Intro 5**.

measurement variables, **measure**, **measurement**, **x a measurement of X**, **x measures X**. Observed variable x is a measurement of latent variable X if there is a path connecting $x \leftarrow X$. Measurement variables are modeled by measurement models. Measurement variables are also called indicator variables.

method. Method is just an English word and should be read in context. Nonetheless, method is used here usually to refer to the method used to solve for the fitted parameters of an SEM. Those methods are ML, QML, MLMV, and ADF. Also see *technique*.

MIMIC. See *multiple indicators and multiple causes*.

mixed-effects models. See *multilevel models*.

ML, **method(ml)**. ML stands for maximum likelihood. It is a method to obtain fitted parameters. ML is the default method used by `sem` and `gsem`. Other available methods for `sem` are QML, MLMV, and ADF. Also available for `gsem` is QML.

MLMV, **method(mlmv)**. MLMV stands for maximum likelihood with missing values. It is an ML method used to obtain fitted parameters in the presence of missing values. MLMV is the method used by `sem` when the `method(mlmv)` option is specified; `method(mlmv)` is not available with `gsem`. Other available methods for use with `sem` are ML, QML, and ADF. These other methods omit from the calculation observations that contain missing values.

modification indices. Modification indices are score tests for adding paths where none appear. The paths can be for either coefficients or covariances.

moments (of a distribution). The moments of a distribution are the expected values of powers of a random variable or centralized (demeaned) powers of a random variable. The first moments are

the expected or observed means, and the second moments are the expected or observed variances and covariances.

multilevel models. Multilevel models are models that include unobserved effects (latent variables) for different groups in the data. For instance, in a dataset of students, groups of students might share the same teacher. If the teacher's identity is recorded in the data, then one can introduce a latent variable that is constant within teacher and that varies across teachers. This is called a two-level model.

If teachers could in turn be grouped into schools, and school identities were recorded in the data, then one can introduce another latent variable that is constant within school and varies across schools. This is called a three-level (nested-effects) model.

In the above example, observations (students) are said to be nested within teacher nested within school. Sometimes there is no such subsequent nesting structure. Consider workers nested within occupation and industry. The same occupations appear in various industries and the same industries appear within various occupations. We can still introduce latent variables at the occupation and industry level. In such cases, the model is called a crossed-effects model.

The latent variables that we have discussed are also known as random effects. Any coefficients on observed variables in the model are known as the fixed portion of the model. Models that contain fixed and random portions are known as mixed-effects models.

multinomial logit regression. Multinomial logit regression is a term for generalized linear response functions that are family multinomial, link logit. It is used for categorical-outcome data when the outcomes cannot be ordered. Multinomial logit regression is also known as multinomial logistic regression and as mlogit in Stata circles. See *generalized linear response functions*.

multiple correlation. The multiple correlation is the correlation between endogenous variable y and its linear prediction.

multiple indicators and multiple causes. Multiple indicators and multiple causes is a kind of structural model in which observed causes determine a latent variable, which in turn determines multiple indicators. See [SEM] **Intro 5**.

multivariate regression. Multivariate regression is a kind of structural model in which each member of a set of observed endogenous variables is a function of the same set of observed exogenous variables and a unique random disturbance term. The disturbances are correlated. Multivariate regression is a special case of seemingly unrelated regression.

negative binomial regression. Negative binomial regression is a term for generalized linear response functions that are family negative binomial, link log. It is used for count data that are overdispersed relative to Poisson. Negative binomial regression is also known as nbreg in Stata circles. See *generalized linear response functions*.

nested-effects models. See *multilevel models*.

nonrecursive (structural) model (system), **recursive (structural) model (system)**. A structural model (system) is said to be nonrecursive if there are paths in both directions between one or more pairs of endogenous variables. A system is recursive if it is a system—it has endogenous variables that appear with paths from them—and it is not nonrecursive.

A nonrecursive model may be unstable. Consider, for instance,

$$y_1 = 2y_2 + 1x_1 + e_1$$
$$y_2 = 3y_1 - 2x_2 + e_2$$

This model is unstable. To see this, without loss of generality, treat $x_1 + e_1$ and $2x_2 + e_2$ as if they were both 0. Consider $y_1 = 1$ and $y_2 = 1$. Those values result in new values $y_1 = 2$ and $y_2 = 3$, and those result in new values $y_1 = 6$ and $y_2 = 6$, and those result in new values .... Continue in this manner, and you reach infinity for both endogenous variables. In the jargon of the mathematics used to check for this property, the eigenvalues of the coefficient matrix lie outside the unit circle.

On the other hand, consider these values:

$$\begin{aligned} y_1 &= 0.5y_2 + 1x_1 + e_1 \\ y_2 &= 1.0y_1 - 2x_2 + e_2 \end{aligned}$$

These results are stable in that the resulting values converge to $y_1 = 0$ and $y_2 = 0$. In the jargon of the mathematics used to check for this property, the eigenvalues of the coefficient matrix lie inside the unit circle.

Finally, consider the values

$$\begin{aligned} y_1 &= 0.5y_2 + 1x_1 + e_1 \\ y_2 &= 2.0y_1 - 2x_2 + e_2 \end{aligned}$$

Start with $y_1 = 1$ and $y_2 = 1$. That yields new values $y_1 = 0.5$, and $y_2 = 2$ and that yields new values $y_1 = 1$ and $y_2 = 1$, and that yields new values $y_1 = 0.5$ and $y_2 = 2$, and it will oscillate forever. In the jargon of the mathematics used to check for this property, the eigenvalues of the coefficient matrix lie on the unit circle. These coefficients are also considered to be unstable.

normality assumption, joint and conditional. The derivation of the standard, linear SEM estimator usually assumes the full joint normality of the observed and latent variables. However, full joint normality can replace the assumption of normality conditional on the values of the exogenous variables, and all that is lost is one goodness-of-fit test (the test reported by `sem` on the output) and the justification for use of optional method MLMV for dealing with missing values. This substitution of assumptions is important for researchers who cannot reasonably assume normality of the observed variables. This includes any researcher including, say, variables age and age-squared in his or her model.

Meanwhile, the generalized SEM makes only the conditional normality assumption.

Be aware that even though the full joint normality assumption is not required for the standard linear SEM, `sem` calculates the log-likelihood value under that assumption. This is irrelevant except that log-likelihood values reported by `sem` cannot be compared with log-likelihood values reported by `gsem`, which makes the lesser assumption.

See [SEM] **Intro 4**.

normalization constraints. See *identification*.

normalized residuals. See *standardized residuals*.

observed variables. A variable is observed if it is a variable in your dataset. In this documentation, we often refer to observed variables by using `x1`, `x2`, ..., `y1`, `y2`, and so on; in reality, observed variables have names such as `mpg`, `weight`, `testscore`, and so on.

In the software, observed variables are usually indicated by having names that are all lowercase.

Also see *latent variable*.

OIM, **vce(oim)**. OIM stands for observed information matrix, defined as the inverse of the negative of the matrix of second derivatives, usually of the log likelihood function. The OIM is an estimate of the VCE. OIM is the default VCE that `sem` and `gsem` report. The other available techniques are EIM, OPG, robust, cluster, bootstrap, and jackknife.

OPG, **vce(opg)**. OPG stands for outer product of the gradients, defined as the cross product of the observation-level first derivatives, usually of the log likelihood function. The OPG is an estimate of the VCE. The other available techniques are OIM, EIM, robust, cluster, bootstrap, and jackknife.

ordered complementary log-log regression. Ordered complementary log-log regression is a term for generalized linear response functions that are family ordinal, link clog-log. It is used for ordinal-outcome data. Ordered complementary log-log regression is also known as oclog-log in Stata circles. See *generalized linear response functions*.

ordered logit regression. Ordered logit regression is a term for generalized linear response functions that are family ordinal, link logit. It is used for ordinal outcome data. Ordered logit regression is also known as ordered logistic regression, as just ordered logit, and as ologit in Stata circles. See *generalized linear response functions*.

ordered probit regression. Ordered probit regression is a term for generalized linear response functions that are family ordinal, link probit. It is used for ordinal-outcome data. Ordered probit regression is also known as just ordered probit and known as oprobit in Stata circles. See *generalized linear response functions*.

parameter constraints. Parameter constraints are restrictions placed on the parameters of the model. These constraints are typically in the form of 0 constraints and equality constraints. A 0 constraint is implied, for instance, when no path is drawn connecting x with y. An equality constraint is specified when one path coefficient is forced to be equal to another or one covariance is forced to be equal to another.

Also see *identification*.

parameters, ancillary parameters. The parameters are the to-be-estimated coefficients of a model. These include all path coefficients, means, variances, and covariances. In mathematical notation, the theoretical parameters are often written as $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\alpha}$ is the vector of intercepts, $\boldsymbol{\beta}$ is the vector of path coefficients, $\boldsymbol{\mu}$ is the vector of means, and $\boldsymbol{\Sigma}$ is the matrix of variances and covariances. The resulting parameter estimates are written as $\widehat{\boldsymbol{\theta}}$.

Ancillary parameters are extra parameters beyond the ones just described that concern the distribution. These include the scale parameter of gamma regression, the dispersion parameter for negative binomial regression, and the cutpoints for ordered probit, logit, and clog-log regression, and the like. These parameters are also included in θ.

path. A path, typically shown as an arrow drawn from one variable to another, states that the first variable determines the second variable, at least partially. If $x \rightarrow y$, or equivalently $y \leftarrow x$, then $y_j = \alpha + \cdots + \beta x_j + \cdots + e.y_j$, where β is said to be the $x \rightarrow y$ path coefficient. The ellipses are included to account for paths to y from other variables. α is said to be the intercept and is automatically added when the first path to y is specified.

A curved path is a curved line connecting two variables, and it specifies that the two variables are allowed to be correlated. If there is no curved path between variables, the variables are usually assumed to be uncorrelated. We say usually because correlation is assumed among observed exogenous variables and, in the command language, assumed among latent exogenous variables, and if some of the correlations are not desired, they must be suppressed. Many authors refer to covariances rather than correlations. Strictly speaking, the curved path denotes a nonzero covariance. A correlation is often called a standardized covariance.

A curved path can connect a variable to itself, and in that case, it indicates a variance. In path diagrams in this manual, we typically do not show such variance paths even though variances are assumed.

path coefficient. The path coefficient is associated with a path; see *path*. Also see *intercept*.

path diagram. A path diagram is a graphical representation that shows the relationships among a set of variables using paths. See [SEM] **Intro 2** for a description of path diagrams.

path notation. Path notation is a syntax defined by the authors of Stata's `sem` and `gsem` commands for entering path diagrams in a command language. Models to be fit may be specified in path notation or they may be drawn using path diagrams into the Builder.

Poisson regression. Poisson regression is a term for generalized linear response functions that are family Poisson, link log. It is used for count data. See *generalized linear response functions*.

probit regression. Probit regression is a term for generalized linear response functions that are family Bernoulli, link probit. It is used for binary outcome data. Probit regression is also known simply as probit. See *generalized linear response functions*.

p-value. P-value is another term for the reported significance level associated with a test. Small p-values indicate rejection of the null hypothesis.

QML, **method(ml) vce(robust)**. QML stands for quasimaximum likelihood. It is a method used to obtain fitted parameters and a technique used to obtain the corresponding VCE. QML is used by `sem` and `gsem` when options `method(ml)` and `vce(robust)` are specified. Other available methods are ML, MLMV, and ADF. Other available techniques are OIM, EIM, OPG, cluster, bootstrap, and jackknife.

quadrature. Quadrature is generic method for performing numerical integration. `gsem` uses quadrature in any model including latent variables (excluding error variables). `sem`, being limited to linear models, does not need to perform quadrature.

random-effects models. See *multilevel models*.

regression. A regression is a model in which an endogenous variable is written as a function of other variables, parameters to be estimated, and a random disturbance.

reliability. Reliability is the proportion of the variance of a variable not due to measurement error. A variable without measure error has reliability 1.

residual. In this manual, we reserve the word "residual" for the difference between the observed and fitted moments of an SEM. We use the word "error" for the disturbance associated with a (Gaussian) linear equation; see *error*. Also see *standardized residuals*.

robust, **vce(robust)**. Robust is the name we use here for the Huber/White/sandwich estimator of the VCE. This technique requires fewer assumptions than most other techniques. In particular, it merely assumes that the errors are independently distributed across observations and thus allows the errors to be heteroskedastic. Robust standard errors are reported when the `sem` (`gsem`) option `vce(robust)` is specified. The other available techniques are OIM, EIM, OPG, cluster, bootstrap, and jackknife.

saturated model. A saturated model is a full covariance model—a model of fitted means and covariances of observed variables without any restrictions on the values. Also see *baseline model*. Saturated models apply only to standard linear SEMs.

score test, **Lagrange multiplier test**. A score test is a test based on first derivatives of a likelihood function. Score tests are especially convenient for testing whether constraints on parameters should be relaxed or parameters should be added to a model. Also see *Wald test*.

scores. Scores has two unrelated meanings. First, scores are the observation-by-observation first-derivatives of the (quasi) log-likelihood function. When we use the word "scores", this is what we mean. Second, in the factor-analysis literature, scores (usually in the context of factor scores) refers to the expected value of a latent variable conditional on all the observed variables. We refer to this simply as the predicted value of the latent variable.

second-level latent variable. See *first-, second-, and higher-order latent variables*.

second-order latent variable. See *first- and second-order latent variables*.

seemingly unrelated regression. Seemingly unrelated regression is a kind of structural model in which each member of a set of observed endogenous variables is a function of a set of observed exogenous variables and a unique random disturbance term. The disturbances are correlated and the sets of exogenous variables may overlap. If the sets of exogenous variables are identical, this is referred to as multivariate regression.

SEM. SEM stands for structural equation modeling and for structural equation model. We use SEM in capital letters when writing about theoretical or conceptual issues as opposed to issues of the particular implementation of SEM in Stata with the `sem` or `gsem` commands.

sem. `sem` is the Stata command that fits standard linear SEMs. Also see *gsem*.

SSD, **ssd**. See *summary statistics data*.

standard linear SEM. An SEM without multilevel effects in which all response variables are given by a linear equation. Standard linear SEM is what most people mean when they refer to just SEM. Standard linear SEMs are fit by `sem`, although they can also be fit by `gsem`; see *generalized SEM*.

standardized coefficient. In a linear equation $y = \ldots\ bx + \ldots$, the standardized coefficient β is $(\widehat{\sigma}_y/\widehat{\sigma}_x)b$. Standardized coefficients are scaled to units of standard deviation change in y for a standard deviation change in x.

standardized covariance. A standardized covariance between y and x is equal to the correlation of y and x, that is, it is equal to $\sigma_{xy}/\sigma_x\sigma_y$. The covariance is equal to the correlation when variables are standardized to have variance 1.

standardized residuals, **normalized residuals**. Standardized residuals are residuals adjusted so that they follow a standard normal distribution. The difficulty is that the adjustment is not always possible. Normalized residuals are residuals adjusted according to a different formula that roughly follow a standard normal distribution. Normalized residuals can always be calculated.

starting values. The estimation methods provided by `sem` and `gsem` are iterative. The starting values are values for each of the parameters to be estimated that are used to initialize the estimation process. `sem` and `gsem` provide starting values automatically, but in some cases, these are not good enough and you must both diagnose the problem and provide better starting values. See [SEM] **Intro 12**.

structural equation model. Different authors use the term "structural equation model" in different ways, but all would agree that an SEM sometimes carries the connotation of being a structural model with a measurement component, that is, combined with a measurement model.

structural model. A structural model is a model in which the parameters are not merely a description but are believed to be of a causal nature. Obviously, SEM can fit structural models and thus so can `sem` and `gsem`. Neither SEM, `sem`, nor `gsem` are limited to fitting structural models, however.

Structural models often have multiple equations and dependencies between endogenous variables, although that is not a requirement.

See [SEM] **Intro 5**. Also see *structural equation model*.

structured (correlation or covariance). See *unstructured and structured (correlation or covariance)*.

substantive constraints. See *identification*.

summary statistics data. Data are sometimes available only in summary statistics form, as means and covariances; means, standard deviations or variances, and correlations; covariances; standard deviations or variances and correlations; or correlations. SEM can be used to fit models with such data in place of the underlying raw data. The `ssd` command creates datasets containing summary statistics.

technique. Technique is just an English word and should be read in context. Nonetheless, technique is usually used here to refer to the technique used to calculate the estimated VCE. Those techniques are OIM, EIM, OPG, robust, cluster, bootstrap, and jackknife.

Technique is also used to refer to the available techniques used with `ml`, Stata's optimizer and likelihood maximizer, to find the solution.

total effects. See *direct, indirect, and total effects*.

unstandardized coefficient. A coefficient that is not standardized. If $\texttt{mpg} = -0.006 \times \texttt{weight} + 39.44028$, then -0.006 is an unstandardized coefficient and, as a matter of fact, is measured in mpg-per-pound units.

unstructured and structured (correlation or covariance). A set of variables, typically error variables, is said to have an unstructured correlation or covariance if the covariance matrix has no particular pattern imposed by theory. If a pattern is imposed, the correlation or covariance is said to be structured.

variance–covariance matrix of the estimator. The estimator is the formula used to solve for the fitted parameters, sometimes called the fitted coefficients. The VCE is the estimated variance–covariance matrix of the parameters. The diagonal elements of the VCE are the variances of the parameters or equivalent; the square roots of those elements are the reported standard errors of the parameters.

VCE. See *variance–covariance matrix of the estimator*.

Wald test. A Wald test is a statistical test based on the estimated variance–covariance matrix of the parameters. Wald tests are especially convenient for testing possible constraints to be placed on the estimated parameters of a model. Also see *score test*.

weighted least squares. Weighted least squares (WLS) is a method used to obtain fitted parameters. In this documentation, WLS is referred to as ADF, which stands for asymptotic distribution free. Other available methods are ML, QML, and MLMV. ADF is, in fact, a specific kind of the more generic WLS.

WLS. See *weighted least squares*.

Reference

Bentler, P. M., and D. G. Weeks. 1980. Linear structural equations with latent variables. *Psychometrika* 45: 289–308.

Subject and author index

See the combined subject index and the combined author index in the *Glossary and Index*.