

## intro 7 — Postestimation tests and predictions

[Description](#)[Remarks and examples](#)[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.

Some tests and features after `sem` depend on the assumption of joint normality of the observed variables and others do not. Whenever the joint-normality assumption is the required, we mention it explicitly. If you do not believe your data meets the joint-normality assumption, you will want to avoid those tests and features.

Some tests and features that are available after `sem` are not available after `gsem` because of the joint-normality assumption. Others are not available for other reasons. If we do not mention the joint-normality assumption, then that is not the cause. We usually leave unexplained the other reasons that a test or feature might be inappropriate after `gsem`, because those reasons are not our focus here.

## Remarks and examples

stata.com

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  $Y \leftarrow x_1$  is `_b[Y1:x1]`, and the symbolic name of the coefficient corresponding to the covariance of  $e.Y_1$  and  $e.Y_2$  is `_b[cov(e.Y1,e.Y2):_cons]`.

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

`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](#), 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
(coefficients table omitted)
LR test of model vs. saturated: chi2(2) = 1.78, Prob > chi2 = 0.4111
```

Be warned, this test is based on the assumption of joint normality of the observed variables, so you may want to ignore it. The test is a goodness-of-fit test in badness-of-fit units; a significant result implies that there may be missing paths in the model's specification. 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 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] [example 4](#). Many but not all depend on the joint-normality assumption; see [SEM] [estat gof](#).
2. (sem only.) Command `estat eqgof` reports  $R^2$ -like goodness-of-fit statistics for each equation separately; see [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] [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] [example 10](#).
5. (sem and gsem.) Command `estat ic` reports the Akaike and Bayesian information criteria; see [R] [estat ic](#).

### Performing tests for including omitted paths and relaxing constraints (sem only)

1. (sem only.) Command `estat mindices` reports  $\chi^2$  modification indices and significance values for each omitted path in the model, along with the expected parameter change; see [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] [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] [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] [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] [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] [example 16](#) and [SEM] [estat stdize](#).
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] [example 7](#) and [SEM] [estat teffects](#).

3. (sem only.) Command `estat stable` assesses the stability of nonrecursive structural equation systems; see [SEM] [example 7](#) and [SEM] [estat stable](#).
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](#).

## 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 *stub1*, *stub2*, ... 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 *stub1*, *stub2*, ... 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 *stub1*, *stub2*, ... 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. Two new issues arise: the prediction of generalized linear response variables as opposed to linear response variables, and the optional multilevel nature of 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. Calculations are made using the empirical Bayes predictions of latent variables, which can be computationally intensive.
2. `predict newvar, mu outcome(depvar) fixedonly` also predicts mean responses in natural units, and again it is appropriate to think  $g^{-1}(x_i b)$ . `fixedonly` includes only the fixed part of the model and is equivalent to setting all latent variables to 0, their mean value.
3. Option `pr` is a synonym for `mu` when using family-and-link combinations that produce probabilities, such as logit, probit, etc.
4. `predict newvar, eta outcome(depvar)` calculates the linear prediction  $x_i b$ . Calculations are made using the empirical Bayes predictions of latent variables, which can be numerically intensive.
5. `predict newvar, eta outcome(depvar) fixedonly` also calculates the linear prediction  $x_i b$ , but includes only the fixed part of the model and is equivalent to setting all latent variables to 0, their mean value.
6. Option `outcome()` varies to accommodate the multiple predictions produced by multinomial logit, ordinal probit, ordinal logit, and ordinal cloglog.
  - 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.

7. 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`.
8. 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. Moving on to latent variables, whether endogenous or exogenous:

9. `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 `mode`; see item 12a below.
10. `predict newvar, latent(Lname) se(another_newvar)` will predict both the latent variable and its standard error.
11. 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*)
```

12. There are four options having to do with the iterative procedure for obtaining the predicted values:
  - a. `mode` 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 `mode`, 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.
  - 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.

That leaves us with only multilevel issues to discuss, and there is only one issue, and it is easy. 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.

See [SEM] [example 28g](#), [SEM] [example 29g](#), 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 [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] [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](#).

## Also see

[SEM] [intro 6](#) — Comparing groups (sem only)

[SEM] [intro 8](#) — Robust and clustered standard errors