Title

stata.com

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

stata.com

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