Title

stata.com

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

stata.com

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