Title

stata.com

intro 6 — Comparing groups (sem only)

Description Remarks and examples Reference Also see

Description

`sem` has a unique feature not shared by `gsem`. You can 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` 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.

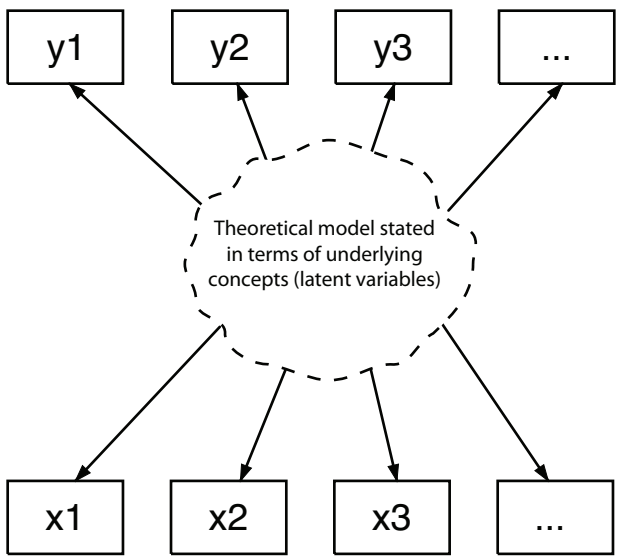
Remarks and examples

stata.com

Remarks are presented under the following headings:

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 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`, ...

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.

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.

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(classes)
```

The classes are 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`	

(*) 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)
```

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.

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)
```

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)
```

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)
```

Relaxing constraints

Just as you can specify

```
..., group(agegrp) ginvariant(classes)
```

and then add constraints, you can also specify

```
..., group(agegrp) ginvariant(classes)
```

and then relax constraints that the 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)
```

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 group options** — Fitting models on different groups

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