Title

stata.com

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.

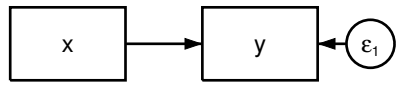
Remarks and examples

stata.com

Remarks are presented under the following headings:

Specifying indicator variables

We wish to fit the following model:

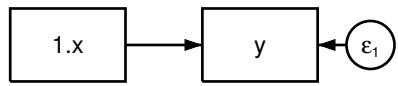


This model corresponds to the equation $y_i = \beta x_i + \epsilon_i$. It can be fit using the Builder with the path diagram above or using the command syntax by typing

```
. 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.

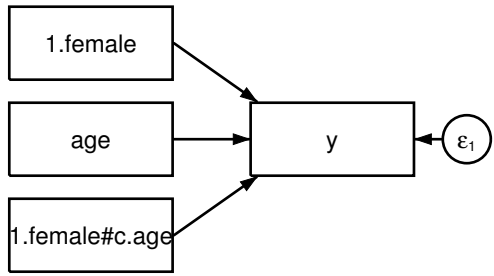
Let's say variable `skill` takes on the values 1, 2, and 3 meaning unskilled, skilled, and highly skilled. Then `3.skill` is an indicator variable for `skill` = 3. We could use `3.skill` in path diagrams or in the command language just as we previously used `1.x`.

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:

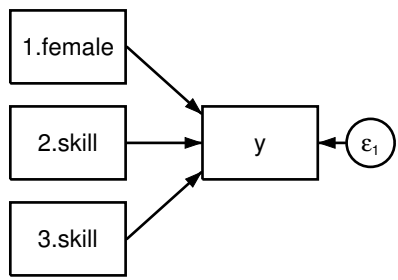
```
. gsem (y<-1.female age 1.female#c.age)
```

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)
```

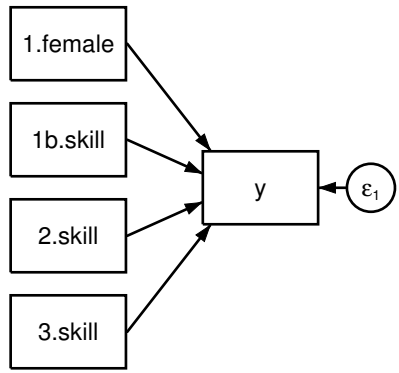
The equation for the model is

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

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.

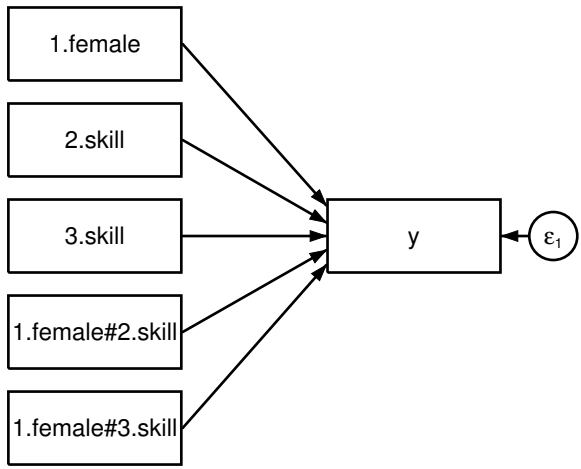
You cannot use the `i.` notation in path diagrams, however. Path diagrams allow only one variable per box. The `i.` notation produces at least two variables, and it usually produces a lot more.

Specifying interactions with categorical variables

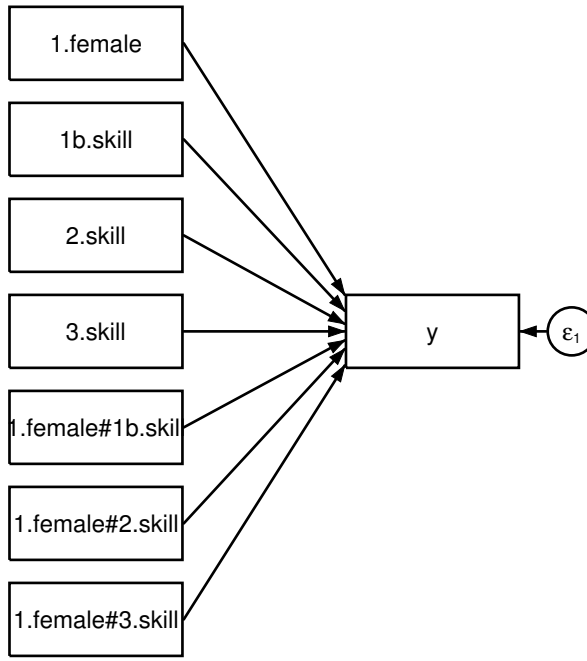
We just discussed the model `(y<-i.female i.skill)`. Obviously, the next step is to include an interaction between female and skill level. To write such a model, we use the same `#` interaction indicator that we have already discussed in the context of indicator variables: `(y<-i.female i.skill i.female#i.skill)`. 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