Title

example 1 — Single-factor measurement model

Description Remarks and examples Reference Also see

Description

The single-factor measurement model is demonstrated using the following data:

```
. use http://www.stata-press.com/data/r13/sem_1fmm
(single-factor measurement model)

. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          x1 |       123    96.28455    14.16444         54        131
          x2 |       123    97.28455    16.14764         64        135
          x3 |       123    97.09756    15.10207         62        138
          x4 |       123    690.9837    77.50737        481        885

. notes

_dta:
  1.  fictional data
  2.  Variables x1, x2, and x3 each contain a test score designed to measure X.
      The test is scored to have mean 100.
  3.  Variable x4 is also designed to measure X, but designed to have mean 700.
```

See *Single-factor measurement models* in [SEM] **intro 5** for background.

Remarks and examples

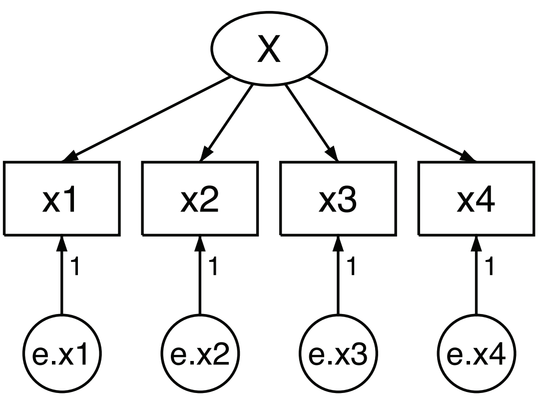
stata.com

Remarks are presented under the following headings:

Single-factor measurement model
Fitting the same model with gsem
Fitting the same model with the Builder
The measurement-error model interpretation

Single-factor measurement model

Below we fit the following model:



```
. sem (x1 x2 x3 x4 <- X)
Endogenous variables
Measurement:  x1 x2 x3 x4
Exogenous variables
Latent:       X
Fitting target model:
Iteration 0:   log likelihood = -2081.0258
Iteration 1:   log likelihood =  -2080.986
Iteration 2:   log likelihood = -2080.9859
Structural equation model                       Number of obs      =       123
Estimation method  = ml
Log likelihood     = -2080.9859
 ( 1)  [x1]X = 1
```

	Coef.	OIM Std. Err.	z	P>|z|	[95% Conf.	Interval]
Measurement						
x1 <-						
X	1	(constrained)				
_cons	96.28455	1.271963	75.70	0.000	93.79155	98.77755
x2 <-						
X	1.172364	.1231777	9.52	0.000	.9309398	1.413788
_cons	97.28455	1.450053	67.09	0.000	94.4425	100.1266
x3 <-						
X	1.034523	.1160558	8.91	0.000	.8070579	1.261988
_cons	97.09756	1.356161	71.60	0.000	94.43953	99.75559
x4 <-						
X	6.886044	.6030898	11.42	0.000	5.704009	8.068078
_cons	690.9837	6.960137	99.28	0.000	677.3421	704.6254
var(e.x1)	80.79361	11.66414			60.88206	107.2172
var(e.x2)	96.15861	13.93945			72.37612	127.7559
var(e.x3)	99.70874	14.33299			75.22708	132.1576
var(e.x4)	353.4711	236.6847			95.14548	1313.166
var(X)	118.2068	23.82631			79.62878	175.4747

```
LR test of model vs. saturated: chi2(2)   =      1.78, Prob > chi2 = 0.4111
```

The equations for this model are

$$x_1 = \alpha_1 + X\beta_1 + e.x_1$$

$$x_2 = \alpha_2 + X\beta_2 + e.x_2$$

$$x_3 = \alpha_3 + X\beta_3 + e.x_3$$

$$x_4 = \alpha_4 + X\beta_4 + e.x_4$$

Notes:

1. Variable X is latent exogenous and thus needs a normalizing constraint. The variable is anchored to the first observed variable, x1, and thus the path coefficient is constrained to be 1. See *Identification 2: Normalization constraints (anchoring)* in [SEM] **intro 4**.

2. The path coefficients for `X->x1`, `X->x2`, and `X->x3` are 1 (constrained), 1.17, and 1.03. Meanwhile, the path coefficient for `X->x4` is 6.89. This is not unexpected; we at StataCorp generated this data, and the true coefficients are 1, 1, 1, and 7.
3. A test for "model versus saturated" is reported at the bottom of the output; the $\chi^2(2)$ statistic is 1.78 and its significance level is 0.4111. We cannot reject the null hypothesis of this test.

   This 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 the 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.

Fitting the same model with gsem

`sem` and `gsem` produce the same results for standard linear SEMs. We are going to demonstrate that just this once.

```
. gsem (x1 x2 x3 x4 <- X)
Fitting fixed-effects model:
Iteration 0:   log likelihood = -2233.3283
Iteration 1:   log likelihood = -2233.3283
Refining starting values:
Grid node 0:   log likelihood = -2081.0303
Fitting full model:
Iteration 0:   log likelihood = -2081.0303
Iteration 1:   log likelihood = -2080.9861
Iteration 2:   log likelihood = -2080.9859
Generalized structural equation model             Number of obs    =        123
Log likelihood = -2080.9859
 ( 1)  [x1]X = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1 <-						
X	1	(constrained)				
_cons	96.28455	1.271962	75.70	0.000	93.79155	98.77755
x2 <-						
X	1.172365	.1231778	9.52	0.000	.9309411	1.413789
_cons	97.28455	1.450052	67.09	0.000	94.4425	100.1266
x3 <-						
X	1.034524	.1160559	8.91	0.000	.8070585	1.261989
_cons	97.09756	1.35616	71.60	0.000	94.43954	99.75559
x4 <-						
X	6.886053	.6030902	11.42	0.000	5.704018	8.068088
_cons	690.9837	6.96013	99.28	0.000	677.3421	704.6253
var(X)	118.2064	23.8262			79.62858	175.474
var(e.x1)	80.79381	11.66416			60.88222	107.2175
var(e.x2)	96.15857	13.93942			72.37613	127.7558
var(e.x3)	99.70883	14.33298			75.22718	132.1577
var(e.x4)	353.4614	236.6835			95.14011	1313.168

Notes:

1. Results are virtually the same. Coefficients differ in the last digit; for instance, `x2<-X` was 1.172364 and now it is 1.172365. The same is true of standard errors, etc. Meanwhile, variance estimates are usually differing in the next-to-last digit; for instance, `var(e.x2)` was 96.15861 and is now 96.15857.

   These are the kind of differences we would expect to see. `gsem` follows a different approach for obtaining results that involves far more numeric machinery, which correspondingly results in slightly less accuracy.

2. The log-likelihood values reported are the same. This model is one of the few models we could have chosen where `sem` and `gsem` would produce the same log-likelihood values. In general, `gsem` log likelihoods are on different metrics from those of `sem`. In the case where the model does not include observed exogenous variables, however, they share the same metric.

3. There is no reason to use `gsem` over `sem` when both can fit the same model. `sem` is slightly more accurate, is quicker, and has more postestimation features.

Fitting the same model with the Builder

Use the diagram above for reference.

1. Open the dataset.

   In the Command window, type

   ```
   . use http://www.stata-press.com/data/r13/sem_1fmm
   ```

2. Open a new Builder diagram.

   Select menu item **Statistics > SEM (structural equation modeling) > Model building and estimation**.

3. Create the measurement component for `X`.

   Select the Add Measurement Component tool, , and then click in the diagram about one-third of the way down from the top and slightly left of the center.

   In the resulting dialog box,

   a. change the *Latent variable name* to `X`;

   b. select `x1`, `x2`, `x3`, and `x4` by using the *Measurement variables* control;

   c. select `Down` in the *Measurement direction* control;

   d. click on **OK**.

   If you wish, move the component by clicking on any variable and dragging it.

   Notice that the constraints of 1 on the paths from the error terms to the observed measures are implied, so we do not need to add these to our diagram.

4. Estimate.

   Click on the **Estimate** button, , in the Standard Toolbar, and then click on **OK** in the resulting *SEM estimation options* dialog box.

You can open a completed diagram in the Builder by typing

```
. webgetsem sem_1fmm
```

The measurement-error model interpretation

As we pointed out in *Using path diagrams to specify standard linear SEMs* in [SEM] **intro 2**, if we rename variable x4 to be y, we can reinterpret this measurement model as a measurement-error model. In this interpretation, X is the unobserved true value. x1, x2, and x3 are each measurements of X, but with error. Meanwhile, y (x4) is really something else entirely. Perhaps y is earnings, and we believe

$$\mathtt{y} = \alpha_4 + \beta_4 \mathtt{X} + \mathtt{e.y}$$

We are interested in β_4, the effect of true X on y.

If we were to go back to the data and type `regress y x1`, we would obtain an estimate of β_4, but we would expect that estimate to be biased toward 0 because of the errors-in-variable problem. The same applies for y on x2 and y on x3. If we do that, we obtain

β_4 based on `regress y x1`	4.09
β_4 based on `regress y x2`	3.71
β_4 based on `regress y x3`	3.70

In the sem output above, we have an estimate of β_4 with the bias washed away:

β_4 based on `sem (y<-X)`	6.89

The number 6.89 is the value reported for (x4<-X) in the sem output.

That β_4 might be 6.89 seems plausible because we expect that the estimate should be larger than the estimates we obtain using the variables measured with error. In fact, we can tell you that the 6.89 estimate is quite good because we at StataCorp know that the true value of β_4 is 7. Here is how we manufactured this fictional dataset:

```
set seed 12347
set obs 123
gen X  = round(rnormal(0,10))
gen x1 = round(100 + X + rnormal(0, 10))
gen x2 = round(100 + X + rnormal(0, 10))
gen x3 = round(100 + X + rnormal(0, 10))
gen x4 = round(700 + 7*X + rnormal(0, 10))
```

The data recorded in sem_1fmm.dta were obviously generated using normality, the same assumption that is most often used to justify the SEM maximum likelihood estimator. In [SEM] **intro 4**, we explained that the normality assumption can be relaxed and conditional normality can usually be substituted in its place.

So let's consider nonnormal data. Let's make X be $\chi^2(2)$, a violently nonnormal distribution, resulting in the data-manufacturing code

```
set seed 12347
set obs 123
gen X  = (rchi2(2)-2)*(10/2)
gen x1 = round(100 + X + rnormal(0, 10))
gen x2 = round(100 + X + rnormal(0, 10))
gen x3 = round(100 + X + rnormal(0, 10))
gen x4 = round(700 + 7*X + rnormal(0, 10))
```

All the `rnormal()` functions remaining in our code have to do with the assumed normality of the errors. The multiplicative and additive constants in the generation of `X` simply rescale the $\chi^2(2)$ variable to have mean 100 and standard deviation 10, which would not be important except for the subsequent `round()` functions, which themselves were unnecessary except that we wanted to produce a pretty dataset when we created the original `sem_1fmm.dta`.

In any case, if we rerun the commands with these data, we obtain

β_4 based on `regress y x1`	3.93
β_4 based on `regress y x2`	4.44
β_4 based on `regress y x3`	3.77
β_4 based on `sem (y<-X)`	6.70

We will not burden you with the details of running simulations to assess coverage; we will just tell you that coverage is excellent: reported test statistics and significance levels can be trusted.

By the way, errors in the variables is something that does not go away with progressively larger sample sizes. Change the code above to produce a 100,000-observation dataset instead of a 123-observation one, and you will obtain

β_4 based on `regress y x1`	3.51
β_4 based on `regress y x2`	3.51
β_4 based on `regress y x3`	3.48
β_4 based on `sem (y<-X)`	7.00

Reference

Acock, A. C. 2013. *Discovering Structural Equation Modeling Using Stata*. Rev. ed. College Station, TX: Stata Press.

Also see

[SEM] **sem** — Structural equation model estimation command

[SEM] **gsem** — Generalized structural equation model estimation command

[SEM] **intro 5** — Tour of models

[SEM] **example 3** — Two-factor measurement model

[SEM] **example 24** — Reliability

[SEM] **example 27g** — Single-factor measurement model (generalized response)