Generalized structural equation models: fitting customized models without programming

Isabel Canette

Senior Statistician

StataCorp LP

2013 Spanish Stata Users Group Meeting
Madrid, October 10, 2012

Introduction

The -gsem- command, introduced in Stata 13, extends generalized linear models to the multivariate/multilevel framework.
We will introduce the different elements of -gsem-:

- family and link
- latent variables
- random effects

Then, we will show how to use these elements as building blocks to perform customized estimations.

Outline

- Quick review of generalized linear models
- Fitting several models simultaneously
- Simple notation for constraints
- Latent variables
- Bivariate Gaussian models
  - standard syntax
  - using latent variables to include the correlation
- Bivariate Gaussian/probit model
- Random effects
- Bivariate Gaussian models with random effects
- Endogeneity: ivprobit
- ivprobit with random effects
- Other relevant features

Quick review of generalized linear models

Generalized linear models are characterized by a family and a link. The family is the distribution; the link is the way the parameter (expected mean) relates to the independent variables. For example, when we fit a linear regression:

```
regress y x1 x2 x3
```

We want to estimate parameters such as:

$$y_i \sim N(\mu_i, \sigma^2) \quad \leftarrow \textbf{family: Gaussian}$$

where $\mu_i = b_0 + b_1 x_{1i} + b_2 x_{2i} + \cdots + b_m x_{mi} = x_i b$
That is,

$$x_i b = f(\mu_i), \text{ with } f = \text{identity function} \quad \leftarrow \textbf{link: identity}$$

Using -glm-, the model could be fit as:

```
glm y x1 x2 x3, family(gaussian) link(identity)
```

We can fit this model with gsem, for example:

```
. sysuse auto, clear
(1978 Automobile Data)

. gsem (mpg <- rep disp turn), link(identity) fam(gaussian) nolog

Generalized structural equation model               Number of obs   =         69
Log likelihood =  -187.4352
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mpg <-						
rep78	0.2645	0.5165	0.51	0.609	-0.7478	1.2768
displacement	-0.0269	0.0082	-3.30	0.001	-0.0430	-0.0109
turn	-0.4831	0.1799	-2.69	0.007	-0.8357	-0.1305
_cons	44.9503	6.8640	6.55	0.000	31.4972	58.4035
var(e.mpg)	13.3970	2.2808			9.5959	18.7036

Note that we use an "arrow" to specify the model; also, see that the variance is estimated.

Families and links available:

	identity	log	logit	probit	cloglog
Gaussian	D	x			
Bernoulli			D	x	x
binomial			D	x	x
multinomial			D		
gamma		D			
negative binomial		D			
ordinal			D	x	x
Poisson		D			

D denotes the default.

For example, if we want to fit the model:

```
poisson y x1 x2 x3
```

we can write:

```
gsem  (y <-x1 x2 x3), family(poisson) link(log)
```

or simply

```
gsem  (y <-x1 x2 x3), poisson
```

Fitting several models simultaneously

`gsem` also allows us to fit several models simultaneously. For example, let's consider the following models, for the weight of boys and girls:

```
webuse childweight, clear
regress  weight age c.age#c.age if girl == 0
regress  weight age c.age#c.age if girl == 1
```

Instead, we could write:

```
. qui generate weightboy = weight if girl == 0
. qui generate weightgirl = weight if girl == 1
. gsem (weightboy <- age c.age#c.age) (weightgirl <- age c.age#c.age), ///
>       nolog vsquish
Generalized structural equation model               Number of obs   =        198
Log likelihood =  -302.2308
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
weightboy <-						
age	7.985022	.6247972	12.78	0.000	6.760442	9.209602
c.age#c.age	-1.74346	.2338615	-7.46	0.000	-2.20182	-1.2851
_cons	3.684363	.3168597	11.63	0.000	3.063329	4.305397
weightg~l <-						
age	7.008066	.5085021	13.78	0.000	6.01142	8.004712
c.age#c.age	-1.450582	.1900543	-7.63	0.000	-1.823081	-1.078082
_cons	3.480933	.2576254	13.51	0.000	2.975997	3.98587
var(e.weig~y)	1.562942	.2210333			1.184581	2.062153
var(e.weig~l)	.978849	.1398356			.7398019	1.295138

Notice that, just this simple feature is conceptually new to Stata: fitting models on different subsets of data simultaneously. We use `coeflegend` to see how to refer to the parameters.

```
. gsem, coeflegend
```

```
Generalized structural equation model           Number of obs   =        198
Log likelihood =  -302.2308
```

	Coef.	Legend
weightboy <-		
age	7.985022	_b[weightboy:age]
c.age#c.age	-1.74346	_b[weightboy:c.age#c.age]
_cons	3.684363	_b[weightboy:_cons]
weightg~l <-		
age	7.008066	_b[weightgirl:age]
c.age#c.age	-1.450582	_b[weightgirl:c.age#c.age]
_cons	3.480933	_b[weightgirl:_cons]
var(e.weig~y)	1.562942	_b[var(e.weightboy):_cons]
var(e.weig~l)	.978849	_b[var(e.weightgirl):_cons]

Now, let's perform a test to see if boys and girls have the same birthweight

```
. test  _b[weightgirl:_cons]=  _b[weightboy:_cons]
 ( 1)  - [weightboy]_cons + [weightgirl]_cons = 0
           chi2(  1) =    0.25
         Prob > chi2 =    0.6184
```

This feature allowed us to perform the test using the original variance (notice that -suest- generates a joint set of results, but it reports robust standard errors).

We don't find evidence that they are different; if we want to impose the constraint that they are equal, we can use the Stata command -constraint-; however, -gsem- also has a convenient notation to set constraints.

Simple notation for constraints

If we want to fit the first equation with the constant term constrained to be equal to 3 and the coefficient for age constrained to be equal to 8, we can write:

```
. gsem (weightboy <- age@8 c.age#c.age _cons@3), nolog

Generalized structural equation model              Number of obs   =        100
Log likelihood = -172.13109

 ( 1)  [weightboy]age = 8
 ( 2)  [weightboy]_cons = 3
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
weightboy <-						
age	8	(constrained)				
c.age#c.age	-1.589425	.0492826	-32.25	0.000	-1.686017	-1.492833
_cons	3	(constrained)				
var(e.weig~y)	1.830784	.258912			1.387584	2.415546

If we want the coefficient for age*age be two times the coefficient for age, we use symbolic constraints:

```
. gsem (weightboy <- age@c1 c.age#c.age@(2*c1) _cons@3), nolog

Generalized structural equation model               Number of obs   =        100
Log likelihood = -265.62392

 ( 1)  - 2*[weightboy]age + [weightboy]c.age#c.age = 0
 ( 2)  [weightboy]_cons = 3
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
weightboy <-						
age	.8886219	.050859	17.47	0.000	.7889401	.9883037
c.age#c.age	1.777244	.101718	17.47	0.000	1.57788	1.976607
_cons	3	(constrained)				
var(e.weig~y)	11.87697	1.679657			9.001763	15.67053

We can use the symbolic notation across equations to constrain birthweights to be equal:

```
. gsem ( weightboy <- age c.age#c.age _cons@a)  ///
>      ( weightgirl <- age c.age#c.age _cons@a), nolog vsquish

Generalized structural equation model             Number of obs   =        198
Log likelihood = -302.35479

 ( 1)  [weightboy]_cons - [weightgirl]_cons = 0
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
weightboy <-						
age	8.191805	.4676429	17.52	0.000	7.275242	9.108369
c.age#c.age	-1.810604	.1913379	-9.46	0.000	-2.18562	-1.435589
_cons	3.56187	.2002524	17.79	0.000	3.169382	3.954357
weightg~l <-						
age	6.870664	.4277058	16.06	0.000	6.032376	7.708952
c.age#c.age	-1.40581	.1676243	-8.39	0.000	-1.734347	-1.077272
_cons	3.56187	.2002524	17.79	0.000	3.169382	3.954357
var(e.weig~y)	1.565277	.2214953			1.186155	2.065575
var(e.weig~l)	.9798347	.1400614			.7404211	1.296662

Latent variables

A latent variable is just an unobserved variable that we include in the model; a simple example is a measurement model:

```
webuse cfa_missing, clear
gsem (test1  test2 test3 test4 <-X)
```

This model assumes that the results for the four observed tests are measurements for an unobserved ability (X).
The underlying model is:

$$testk_i = a_k + b_k X_i + \varepsilon_{ki}$$

Where $X_i \sim N(0, \beta)$ and $\varepsilon_{ki} \sim N(0, \sigma_k^2)$, being all independent from each other.
Naturally, we need to impose some constraints to identify the model.
This model can be fitted more efficiently with -sem-; this is the output from -gsem-:

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
test1 <-						
X	1.0000	(constrained)				
_cons	98.9439	0.6814	145.20	0.000	97.6083	100.2795
test2 <-						
X	1.0700	0.1079	9.91	0.000	0.8584	1.2815
_cons	99.8422	0.6911	144.46	0.000	98.4876	101.1968
test3 <-						
X	0.9489	0.0896	10.59	0.000	0.7733	1.1245
_cons	101.0655	0.6256	161.54	0.000	99.8393	102.2917
test4 <-						
X	1.0216	0.0959	10.65	0.000	0.8337	1.2096
_cons	99.6451	0.6730	148.06	0.000	98.3260	100.9642
var(X)	94.0463	13.9673			70.2951	125.8226
var(e.test1)	101.1131	10.1897			82.9902	123.1935
var(e.test2)	95.4558	10.7948			76.4790	119.1413
var(e.test3)	95.1484	9.0530			78.9611	114.6543
var(e.test4)	101.0942	10.0969			83.1212	122.9535

A similar model can be fitted for non-normal dependent variables. For example, we can fit a measurement model for binary outcomes as follows:

```
. use gsem_1fmm, clear
(single-factor pass/fail measurement model)

. summ x*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	123	.4065041	.4931897	0	1
x2	123	.4065041	.4931897	0	1
x3	123	.4227642	.4960191	0	1
x4	123	.3495935	.4787919	0	1

```
. gsem (x1 x2 x3 x4 <-X), probit
```

Generalized structural equation model Number of obs = 123
Log likelihood = -261.30263

(1) [x1]X = 1

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1 <-						
X	1.0000	(constrained)				
_cons	-0.3667	0.1897	-1.93	0.053	-0.7384	0.0051
x2 <-						
X	1.3329	0.4687	2.84	0.004	0.4143	2.2515
_cons	-0.4470	0.2372	-1.88	0.060	-0.9120	0.0179
x3 <-						
X	0.6040	0.1908	3.17	0.002	0.2300	0.9781
_cons	-0.2277	0.1439	-1.58	0.114	-0.5098	0.0544
x4 <-						
X	9.4533	5.1518	1.83	0.067	-0.6440	19.5507
_cons	-4.8010	2.5180	-1.91	0.057	-9.7363	0.1342
var(X)	2.1735	1.0449			0.8471	5.5765

Correlation among Gaussian equations

By default, in -gsem-, residuals from different equations are independent. However, we can incorporate correlations among the residuals from two Gaussian equations.

The caschool dataset (from the STAR project, analyzed in Stock and Watson's book), contains grades in Math and in writing for 42 students, as well as other variables (average per district of: income, expenditure per student, computers per student)

```
. gsem (math <- income expend comp ) ///
>     (read <- income expend),  ///
>       cov(e.math*e.read) nolog nohead
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
math <-						
income	1.872255	.0949556	19.72	0.000	1.686146	2.058365
expend	-.0021772	.0010898	-2.00	0.046	-.0043133	-.0000412
comp	1.739182	5.623889	0.31	0.757	-9.283438	12.7618
_cons	635.9961	5.498342	115.67	0.000	625.2195	646.7726
read <-						
income	1.943351	.1024576	18.97	0.000	1.742538	2.144164
expend	-.0000537	.0011679	-0.05	0.963	-.0023427	.0022352
_cons	625.4904	5.94331	105.24	0.000	613.8417	637.1391
var(e.math)	177.0705	12.3344			154.4732	202.9735
var(e.read)	206.9465	14.28067			180.7672	236.9172
cov(e.read, e.math)	163.2235	12.3215	13.25	0.000	139.0738	187.3732

```
. est store cov
```

Using latent variables to include correlations

An alternative way to introduce a correlation is via latent variables; this is not very practical in the Gaussian case, but the concept will be used for other models; then, let's see this simple case:

```
. gsem (math <- income expend comp L@1) ///
>     (read <- income expend L@1), nolog nohead nocnsreport
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
math <-						
income	1.872255	.0949556	19.72	0.000	1.686146	2.058365
expend	-.0021772	.0010898	-2.00	0.046	-.0043133	-.0000412
comp	1.739209	5.623883	0.31	0.757	-9.2834	12.76182
L	1	(constrained)				
_cons	635.9961	5.498341	115.67	0.000	625.2195	646.7726
read <-						
income	1.943351	.1024576	18.97	0.000	1.742538	2.144164
expend	-.0000537	.0011679	-0.05	0.963	-.0023427	.0022352
L	1	(constrained)				
_cons	625.4904	5.94331	105.24	0.000	613.8417	637.1391
var(L)	163.2235	12.3215			140.7754	189.2512
var(e.math)	13.84694	5.010724			6.81293	28.14323
var(e.read)	43.72307	5.835795			33.65885	56.79655

```
. est store latent
```

Variable	cov	latent
math		
income	1.8722553	1.8722553
expend	-.00217725	-.00217725
comp	1.7391819	1.7392088
L		1
_cons	635.99608	635.99608
read		
income	1.9433509	1.9433509
expend	-.00005373	-.00005373
L		1
_cons	625.4904	625.4904
var(e.math)		
_cons	177.07048	13.846944
var(e.read)		
_cons	206.94653	43.723065
cov(e.read,e.math)		
_cons	163.22345	
var(L)		
_cons		163.22346

Bivariate Probit/Gaussian distribution

Now, let's assume that instead of the reading score, we have an indicator that tells us if the score was larger than 656. We want to fit a bivariate model where one of the responses is continuous, and the other is binary; we assume that there is an underlying bivariate Gaussian distribution:

$$y_j = x_j\beta + u_{1j} \tag{1}$$

$$t_j^* = z_j\gamma + u_{2j}$$

where $u_1 \sim N(0, \sigma^2)$, $u_2 \sim N(0, 1)$, and $corr(u_1, u_2) = \rho$
And we observe

$$y_j = x_j\beta + u_{1j}$$

$$t_j = t_j^* > 0$$

The gsem syntax we will use to fit the model is:

```
gsem (y <- x1 x2 L@lambda) (t1 <- x1 x3 x4 L@lambda, probit) \\\
     , var(L@1)
```

The model we are fitting is comprised of the following two equations:

$$y_j = x_j b + v_{1j} + \lambda L_j \tag{2}$$

$$t_j = w_j^* > 0$$

where

$$w_j^* = z_j c + v_{2j} + \lambda L_j$$

and $v_1 \sim N(0, s^2)$, $v_2 \sim N(0, 1)$, and $L \sim N(0, 1)$
The second equation consists of a rescaled probit model, with variance $1 + \lambda^2$ instead of one. Therefore,

$$t_j^* = z_j\gamma + u_{2j} = (1/\sqrt{1+\lambda^2})w_j^* = (1/\sqrt{1+\lambda^2})(z_j c + v_{2j} + \lambda L_j)$$

then,

$$\gamma = (1/\sqrt{1+\lambda^2})c$$

Also,

$$y_j = x_j\beta + u_{1j} = x_j b + v_{1j} + \lambda L_j$$

which leads to:

$$\beta = \beta^*$$

$$var(u_1) = var(v_1) + \lambda^2$$

and, from the two equations, we have that:

$$\rho = corr(v_2 + \lambda L, v_1 + \lambda L) = \frac{\lambda^2}{\sqrt{1+\lambda^2} * \sqrt{var(v1) + \lambda^2}}$$

Let's fit the model:

```
gen t = math >   656
gsem (t  <- income  expend comp L@a, probit)
   (read  <- income  expend comp L@a), var(L@1)
```

Below are the results for model 2, (Gaussian/rescaled probit)

```
. gen t = math >   656
. gsem (t  <- income  expend comp  L@a, probit) ///
> (read <- income  expend  L@a) , var(L@1)  nolog  nohead vsquish
 ( 1)  [t]L - [read]L = 0
 ( 2)  [var(L)]_cons = 1
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
t <-						
income	1.755412	.1284457	13.67	0.000	1.503663	2.007161
expend	-.0016773	.0010142	-1.65	0.098	-.0036651	.0003105
comp	-8.208326	6.575307	-1.25	0.212	-21.09569	4.679039
L	12.82859	.6105602	21.01	0.000	11.63191	14.02526
_cons	-20.44742	5.043636	-4.05	0.000	-30.33277	-10.56208
read <-						
income	1.92612	.1008576	19.10	0.000	1.728443	2.123798
expend	.0001651	.0011503	0.14	0.886	-.0020894	.0024196
L	12.82859	.6105602	21.01	0.000	11.63191	14.02526
_cons	624.4523	5.772452	108.18	0.000	613.1385	635.7661
var(L)	1	(constrained)				
var(e.read)	45.04088	7.365645			32.68946	62.05916

We can use -nlcom- to obtain the transformed estimates: below are the results for model 1, (Gaussian/probit)

```
.  nlcom `trans´, noheader
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
t_income	.1364221	.0094049	14.51	0.000	.1179888	.1548554
t_expend	-.0001304	.0000791	-1.65	0.100	-.0002855	.0000248
t_comp	-.6379112	.5109917	-1.25	0.212	-1.639436	.363614
t__cons	-1.589074	.3846328	-4.13	0.000	-2.342941	-.8352079
read_income	1.92612	.1008576	19.10	0.000	1.728443	2.123798
read_expend	.0001651	.0011503	0.14	0.886	-.0020894	.0024196
read__cons	624.4523	5.772452	108.18	0.000	613.1385	635.7661
sig_2	14.47804	.5032311	28.77	0.000	13.49172	15.46435
rho_12	.8833925	.0208869	42.29	0.000	.8424549	.9243302

(the code to apply the transformation is in the appendix)

Random effects

-gsem- allows us to incorporate random effects. For example, let's assume we want to fit the equation for reading with random effects at the county level, that is:

```
mixed read income expend || cnty:
```

this can be done with -gsem- by incorporating a latent variable at the county level:

```
gsem (read <- income expend M[cnty])
```

```
. mixed read income expend || cnty:, nolog nohead var nolr
```

read	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
income	2.086092	.1116347	18.69	0.000	1.867292	2.304892
expend	-.0028924	.001078	-2.68	0.007	-.0050052	-.0007796
_cons	640.1784	5.755908	111.22	0.000	628.897	651.4598

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
cnty: Identity				
var(_cons)	72.89077	20.57412	41.91918	126.7454
var(Residual)	138.2716	10.09433	119.8373	159.5415

```
. gsem (read  <- income expend M[cnty]), nolog nohead nocnsr
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
read <-						
income	2.086091	.1121313	18.60	0.000	1.866318	2.305865
expend	-.0028924	.0010877	-2.66	0.008	-.0050242	-.0007606
M[cnty]	1	(constrained)				
_cons	640.1784	5.80736	110.24	0.000	628.7961	651.5606
var(M[cnty])	72.88823	20.57298			41.91819	126.7396
var(e.read)	138.2716	10.09433			119.8373	159.5415

Bivariate models with random effects

Example: Fitting a bivariate linear model with random effects:

```
gsem (read  <- income  M1[cnty]) ///
     (math  <- income expend M2[cnty]), ///
               cov(e.read*e.math)
```

(this model couldn't be fitted with -mixed-, because it doesn't have the same variables in the two equations)

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
read <-						
income	1.982774	.1065411	18.61	0.000	1.773957	2.191591
M1[cnty]	1	(constrained)				
_cons	626.2353	2.100519	298.13	0.000	622.1183	630.3522
math <-						
income	1.888574	.1000575	18.87	0.000	1.692465	2.084683
expend	-.0014746	.0005723	-2.58	0.010	-.0025962	-.000353
M2[cnty]	1	(constrained)				
_cons	633.2329	3.315404	191.00	0.000	626.7348	639.731
var(M1[cnty])	61.59101	17.69117			35.07691	108.1467
var(M2[cnty])	24.67535	9.882164			11.25558	54.0952
cov(M2[cnty], M1[cnty])	37.59107	12.57116	2.99	0.003	12.95204	62.2301
var(e.read)	142.8323	10.45908			123.736	164.8757
var(e.math)	153.1723	11.24586			132.6433	176.8785
cov(e.math, e.read)	125.2602	10.06473	12.45	0.000	105.5337	144.9868

Endogeneity

Stata has a suite of commands to fit models with endogenous continous regressors. These commands are usually named starting with the letters “iv”.

The command -ivprobit, mle- is a particular case of the bivariate model we already fitted (one continuous and one binary outcome), where all the covariates in the binary equation are included in the linear equation.

```
ivprobit y1 x1 x2 (y2 = x3 x4), mle
```

This model can be also fitted with the following -gsem- syntax:

```
gsem (y1 <- x1 x2 y2 L@a, probit ) ///
     (y2 <- x1 x2 x3 x4  L@a), ///
     var(L@1)
```

These are the results from -ivprobit-:

```
. ivprobit y1 x1 x2 (y2 = x3 x4), nolog
Probit model with endogenous regressors          Number of obs   =       1000
                                                 Wald chi2(3)    =     201.53
Log likelihood = -1610.0197                      Prob > chi2     =     0.0000
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y2	.957426	.0915621	10.46	0.000	.7779675	1.136884
x1	.96111	.0930771	10.33	0.000	.7786821	1.143538
x2	.9967672	.1031633	9.66	0.000	.7945709	1.198964
_cons	-.0283237	.0660784	-0.43	0.668	-.157835	.1011877
/athrho	.5296383	.0845491	6.26	0.000	.3639252	.6953514
/lnsigma	.0032403	.0223607	0.14	0.885	-.0405859	.0470664
rho	.4851046	.0646524			.3486668	.6014088
sigma	1.003246	.0224333			.9602267	1.048192

```
Instrumented:  y2
Instruments:   x1 x2 x3 x4
```

```
Wald test of exogeneity (/athrho = 0): chi2(1) =    39.24 Prob > chi2 = 0.0000
```

These are the results from -gsem- after transformation (that is, parameters for model 1):

```
. nlcom `trans´, noheader
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1_y2	.9574257	.0915622	10.46	0.000	.7779671	1.136884
y1_x1	.96111	.0930771	10.33	0.000	.7786822	1.143538
y1_x2	.9967672	.1031633	9.66	0.000	.7945709	1.198964
y1__cons	-.0283237	.0660784	-0.43	0.668	-.157835	.1011877
y2_x1	.9902859	.0333022	29.74	0.000	.9250147	1.055557
y2_x2	1.035864	.0326621	31.71	0.000	.9718472	1.09988
y2_x3	.9710073	.0319614	30.38	0.000	.9083641	1.033651
y2_x4	.9605932	.0326173	29.45	0.000	.8966644	1.024522
y2__cons	-.0324965	.0319058	-1.02	0.308	-.0950306	.0300377
sigma2	1.003246	.0224333	44.72	0.000	.9592771	1.047214
rho_12	.4851048	.0646527	7.50	0.000	.3583879	.6118218

ivprobit with random effects

We can add random effects to the main equation (building a model similar to the one fitted by -xtivreg, re- for continuous outcome)
Here is an example with simulated data:

```
gsem (y1 <-x1 x2 y2 L@a M[id], probit ) ///
     (y2 <- x1 x2 x3 x4  L@(a)), ///
     var(L@1) from(b)
```

```
. nlcom `trans´, noheader
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1_x1	.9803679	.0344357	28.47	0.000	.9128753	1.047861
y1_x2	.9906779	.03406	29.09	0.000	.9239215	1.057434
y1_y2	.9859219	.0348666	28.28	0.000	.9175845	1.054259
y1__cons	.0040053	.0390496	0.10	0.918	-.0725304	.0805411
y2_x1	.9909343	.0099165	99.93	0.000	.9714982	1.01037
y2_x2	.9890343	.010091	98.01	0.000	.9692563	1.008812
y2_x3	.9804318	.0099311	98.72	0.000	.9609672	.9998965
y2_x4	.9981266	.009831	101.53	0.000	.9788583	1.017395
y2__cons	.0058421	.0100041	0.58	0.559	-.0137656	.0254497
sigma2	1.000268	.007073	141.42	0.000	.986405	1.01413
rho_12	.5021562	.0239045	21.01	0.000	.4553042	.5490083
se_re	.9918293	.0446821	22.20	0.000	.904254	1.079405

values used for the simulation: all the coefficient and variances equal to one, and correlation equal to .5; no constant terms.

This is just the tip of the iceberg; other important features are:

- The Builder
- Setting initial values
- Integration methods
- Futures developements (removing the restriction on the correlation)

Appendix 1: simulation and estimation for a probit model with an endogenous covariate

```
*data simulation and estimation
cscript
set seed 1357
set obs 1000
mat M = [1,.5 \ .5 ,1]
drawnorm u v, cov(M)


forvalues i = 1(1)4{
  gen x‘i’ = rnormal()
}


gen y2 = x1 + x2 + x3 + x4  + v
gen y1star = x1 + x2 + y2 + u
gen y1 = y1star >=0
save ivprobit, replace

ivprobit y1 x1 x2 (y2 = x3 x4), nolog
gsem (y1 <-x1 x2 y2 L@a, probit ) ///
     (y2 <- x1 x2 x3 x4  L@(a)), ///
     var(L@1)
```

```
*back-trasformation
local y1 y1
local y2 y2
local lambda   _b[‘y1’:L]
local s sqrt(1+‘lambda’^2)
local probit_cov y2 x1 x2  _cons
local reg_cov x1 x2 x3 x4 _cons

 foreach v in ‘probit_cov’{
        local trans ‘trans’ (‘y1’_‘v’: _b[‘y1’:‘v’]/‘s’)
}
 foreach v in ‘reg_cov’  {
        local trans ‘trans’ (‘y2’_‘v’: _b[‘y2’:‘v’])
}
 local s2 sqrt( _b[var(e.‘y2’):_cons]  + ‘lambda’^2)
 local trans ‘trans’ (sigma2: ‘s2’)
 local trans ‘trans’ (rho_12:‘lambda’^2/(‘s’*‘s2’))
 nlcom ‘trans’, noheader
```

Appendix 2: simulation and estimation for a probit model with an endogenous covariate and random effects

```
cscript
set more off
set seed 1357
set obs 1000
gen id = _n
gen re = rnormal()
expand 10
mat M = [1,.5 \ .5 ,1]
drawnorm u v, cov(M)
forvalues i = 1(1)4{
  gen x‘i’ = rnormal()
}
gen y2 = x1 + x2 + x3 + x4  + v
gen y1star = x1 + x2 + y2 + u + re
gen y1 = y1star >=0

gsem (y1 <-x1 x2 y2 L@a, probit ) ///
     (y2 <- x1 x2 x3 x4  L@(a)), ///
     var(L@1)
mat b = e(b)
gsem (y1 <-x1 x2 y2 L@a M[id], probit ) ///
     (y2 <- x1 x2 x3 x4  L@(a)), ///
     var(L@1) from(b)
```

```
local y1 y1
local y2 y2
local lambda   _b[‘y1’:L]
local s sqrt(1+‘lambda’^2)
local probit_cov y2 x1 x2 _cons
local reg_cov x1 x2 x3 x4 _cons

 foreach v in ‘probit_cov’{
        local trans ‘trans’ (y1_‘v’: _b[‘y1’:‘v’]/‘s’)
}
 foreach v in ‘reg_cov’  {
        local trans ‘trans’ (‘y2’_‘v’: _b[‘y2’:‘v’])
}
 local s2 sqrt( _b[var(e.‘y2’):_cons]  + ‘lambda’^2)
 local trans ‘trans’ (sigma2: ‘s2’)
 local trans ‘trans’ (rho_12:‘lambda’^2/(‘s’*‘s2’))
 local trans ‘trans’ (se_re: sqrt(_b[var(M[id]):_cons])/‘s’)

 nlcom ‘trans’, noheader
```