Using generalized structural equation models to fit customized models without programming, and taking advantage of new features of -margins-

Isabel Canette

Principal Mathematician and Statistician

StataCorp LP

2015 Stata Users Group Meeting

Lisbon, September 18, 2015

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 models simultaneously
- Latent variables
- Random effects
- Bivariate Gaussian models
  - standard syntax
  - using latent variables to include the correlation
- Bivariate Gaussian/probit model
- Using -margins- to interpret results

Additional material:

- Bivariate Gaussian models with random effects
- Endogeneity: ivprobit
- Bivariate Gaussian/probit model: theoretical framework
- ivprobit with random effects
- Simulating data, estimation and back-transformation for -ivprobit- and -ivprobit- with random effects

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.

Gaussian responses might be right, left or interval-censored.

Additional (Stata 14) distributions for survival analysis: exponential, loglogistic, weibull, lognormal.

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

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 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).

If we use `vce(robust)`, results are the same as from `suest`, and we can use `gsem` to extend the method in `suest` to commands that are not supported by this command (e.g. random effects: see Stata blog for an example)

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.

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

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 is the same as:

```
gsem (test1<-X) (test2<-X) (test3 <-X) (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.
Some constraints will be (automatically) set 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

(similar models can be fit for non-normal dependent variables)

Random effects

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, teachers per student, computers per student, percentage of students that qualify for low price lunch)

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

```
mixed read_scr avginc  expn_stu comp_stu  ///
     calw_pct meal_pct el_pct || county:
```

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

```
gsem (read_scr  <- avginc expn_stu comp_stu ///
           calw_pct meal_pct M[county])
```

```
. mixed read_scr avginc  expn_stu comp_stu || county:, nolog nohead var nolr
```

read_scr	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
avginc	2.019257	.1112244	18.15	0.000	1.801261	2.237253
expn_stu	-.0036192	.0010847	-3.34	0.001	-.0057452	-.0014932
comp_stu	35.32938	9.860425	3.58	0.000	16.0033	54.65546
_cons	640.0145	5.663293	113.01	0.000	628.9147	651.1144

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
county: Identity				
var(_cons)	65.28703	18.62802	37.3214	114.2078
var(Residual)	135.1293	9.855167	117.1306	155.8938

```
. gsem (read_scr  <- avginc expn_stu comp_stu M[county]), nolog nohead nocnsr
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
read_scr <-						
avginc	2.019257	.1116783	18.08	0.000	1.800371	2.238142
expn_stu	-.0036192	.0010913	-3.32	0.001	-.005758	-.0014803
comp_stu	35.32939	9.916355	3.56	0.000	15.89369	54.76509
M[county]	1	(constrained)				
_cons	640.0145	5.716552	111.96	0.000	628.8103	651.2187
var(M[cou~y])	65.28665	18.62785			37.32125	114.207
var(e.read~r)	135.1293	9.855163			117.1305	155.8938

Multivariate Gaussian models: correlations among Gaussian equations

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

```
. gsem (math_scr <- avginc expn_stu comp_stu calw_pct meal_pct) ///
>     (read_scr <- avginc expn_stu comp_stu calw_pct meal_pct), ///
>      cov(e.math_scr*e.read_scr) nohead nolog vsquish
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
math_scr <-						
avginc	.6124285	.0999994	6.12	0.000	.4164333	.8084237
expn_stu	.0007622	.0008784	0.87	0.386	-.0009594	.0024838
comp_stu	18.26066	7.984749	2.29	0.022	2.610839	33.91048
calw_pct	-.0376608	.0643041	-0.59	0.558	-.1636945	.0883729
meal_pct	-.4356281	.0315902	-13.79	0.000	-.4975437	-.3737126
_cons	657.4049	4.346279	151.26	0.000	648.8863	665.9234
read_scr <-						
avginc	.3346619	.08778	3.81	0.000	.1626162	.5067076
expn_stu	.0033704	.0007711	4.37	0.000	.0018591	.0048816
comp_stu	21.39067	7.009058	3.05	0.002	7.653173	35.12818
calw_pct	.1038808	.0564465	1.84	0.066	-.0067523	.2145139
meal_pct	-.6076669	.02773	-21.91	0.000	-.6620168	-.5533171
_cons	654.8222	3.815189	171.64	0.000	647.3446	662.2999
var(e.math~r)	99.13084	6.840677			86.59051	113.4873
var(e.read~r)	76.38455	5.271034			66.72169	87.44682
cov(e.read~r, e.math_scr)	62.86277	5.238091	12.00	0.000	52.5963	73.12924

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_scr <- avginc expn_stu comp_stu calw_pct meal_pct L@1) ///
>      (read_scr <- avginc expn_stu comp_stu calw_pct meal_pct L@1), ///
>       nolog nohead vsquish nocnsreport
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
math_scr <-						
avginc	.6124285	.0999994	6.12	0.000	.4164332	.8084237
expn_stu	.0007622	.0008784	0.87	0.386	-.0009594	.0024838
comp_stu	18.26066	7.98475	2.29	0.022	2.610838	33.91048
calw_pct	-.0376608	.0643041	-0.59	0.558	-.1636945	.0883729
meal_pct	-.4356281	.0315902	-13.79	0.000	-.4975437	-.3737126
L	1	(constrained)				
_cons	657.4049	4.34628	151.26	0.000	648.8863	665.9234
read_scr <-						
avginc	.3346619	.08778	3.81	0.000	.1626162	.5067077
expn_stu	.0033704	.0007711	4.37	0.000	.0018591	.0048816
comp_stu	21.39067	7.009058	3.05	0.002	7.653173	35.12818
calw_pct	.1038808	.0564465	1.84	0.066	-.0067523	.2145139
meal_pct	-.6076669	.02773	-21.91	0.000	-.6620168	-.5533171
L	1	(constrained)				
_cons	654.8222	3.815189	171.64	0.000	647.3446	662.2999
var(L)	62.86278	5.238092			53.39081	74.01515
var(e.math~r)	36.26808	3.857919			29.44288	44.67544
var(e.read~r)	13.52179	3.080668			8.651767	21.1331

math_scr		
avginc	.61242849	.61242849
expn_stu	.00076218	.00076218
comp_stu	18.26066	18.26066
calw_pct	-.03766081	-.03766081
meal_pct	-.43562815	-.43562815
L		1
_cons	657.40488	657.40488
read_scr		
avginc	.33466192	.33466192
expn_stu	.00337035	.00337035
comp_stu	21.390674	21.390674
calw_pct	.10388076	.10388076
meal_pct	-.60766693	-.60766693
L		1
_cons	654.82224	654.82224
var(e.math_scr)		
_cons	99.130844	36.268077
var(e.read_scr)		
_cons	76.384554	13.521785
cov(e.read_scr,e.math_scr)		
_cons	62.86277	
var(L)		
_cons		62.862776

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.

Let's fit the model:

```
gen t = math_scr >   656.297
gsem (t  <- avginc  expn_stu comp_stu calw_pct meal_pct L@a, ///
   probit),
   (read_scr  <-  ///
   avginc  expn_stu comp_stu calw_pct meal_pct el_pct L@a) ///
   var(L@1)
```

When adding a latent variable to the probit model, parameters will be rescaled; we can back transform the parameters to obtain the original probit parameters (as show in the appendix).

Note: this parameterization, tough intuitive, restricts the correlation to be positive (so we can try the model with t1 and 1-t1). For other parameterizations, see example for Heckman model in documentation for gsem

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
t <-						
avginc	.2566474	.1218955	2.11	0.035	.0177367	.4955581
expn_stu	.0006779	.0007682	0.88	0.377	-.0008277	.0021835
comp_stu	3.499589	6.994748	0.50	0.617	-10.20986	17.20904
calw_pct	-.1690675	.0876864	-1.93	0.054	-.3409297	.0027947
meal_pct	-.2484215	.0353662	-7.02	0.000	-.317738	-.1791051
L	6.566976	.417893	15.71	0.000	5.747921	7.386031
_cons	2.937779	4.067807	0.72	0.470	-5.034975	10.91053
read_scr <-						
avginc	.5243488	.082093	6.39	0.000	.3634495	.685248
expn_stu	.0030644	.0007043	4.35	0.000	.001684	.0044448
comp_stu	11.88184	6.514678	1.82	0.068	-.8866891	24.65038
calw_pct	-.0354518	.0546339	-0.65	0.516	-.1425322	.0716287
meal_pct	-.4183846	.0328394	-12.74	0.000	-.4827487	-.3540205
el_pct	-.2597193	.0293696	-8.84	0.000	-.3172826	-.202156
L	6.566976	.417893	15.71	0.000	5.747921	7.386031
_cons	652.2994	3.460149	188.52	0.000	645.5177	659.0812
var(L)	1	(constrained)				
var(e.read~r)	23.37398	3.615527			17.26105	31.65177

But, do we really want to rescale the parameter? (the original parameterization for -probit- is as arbitrary as any other) -margins- (and its new features) can be used to get a more objective measure, in the sense that is independent on the parameterization. Let's see two parameterizations for the probit model:

```
sysuse auto, clear

* classic probit, with variance 1 for the latent variable
probit for  mpg disp
estimates store probit1
margins, dydx(*) post
est store margins1

* rescaled probit, with variance 2 for the latent variable
gsem ( for <-  mpg disp L@1), var(L@1) probit
estimates store probit2
margins, dydx(*) post
estimates store margins2
```

```
. est table probit1 probit2

----------------------------------------
    Variable |   probit1      probit2   
-------------+--------------------------
foreign      |
         mpg | -.11543538   -.16325322  
displacement | -.03743713   -.05294504  
           L |                       1  
       _cons |  7.5827975    10.723891  
-------------+--------------------------
      var(L) |
       _cons |                       1  
----------------------------------------

. display -.16325322/ -.11543538
1.414239

. est table margins1 margins2

----------------------------------------
    Variable |  margins1     margins2   
-------------+--------------------------
         mpg | -.01777433   -.01777358  
displacement | -.00576444   -.00576419  
----------------------------------------
```

I can do this because in Stata 14, -predict- allows for predictions that are marginal on the random effects; therefore, latent variables or random effects for -gsem- are intergrated over their distribution.

Appendix 1: Bivariate models with random effects

Example: Fitting a bivariate linear model with random effects:

```
gsem (read  <- avginc    M1[cty])  ///
     (math <- avginc  expn_stu  M2[cty]), ///
              cov(e.read*e.math) nolog nohead
```

read <-						
avginc	1.982774	.1065411	18.61	0.000	1.773957	2.191591
M1[cty]	1	(constrained)				
_cons	626.2353	2.100519	298.13	0.000	622.1183	630.3522
math <-						
avginc	1.888574	.1000575	18.87	0.000	1.692465	2.084683
expn_stu	-.0014746	.0005723	-2.58	0.010	-.0025962	-.000353
M2[cty]	1	(constrained)				
_cons	633.2329	3.315404	191.00	0.000	626.7348	639.731
var(M1[cty])	61.59101	17.69117			35.07691	108.1467
var(M2[cty])	24.67535	9.882164			11.25558	54.0952
cov(M2[cty], M1[cty])	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

Appendix 2: theoretical framework for the probit/Gaussian model, and transformation to the probit scale.

For the probit/Gaussian bivariate model, we assume that there is an underlying bivariate Gaussian distribution:

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

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

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 comprises the following two equations:

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

$$t_j = w_j^* > 0$$

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, parameters can be transformed to the original probit scale (see Appendix)

Appendix 3: 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):

```
. nlcom `trans´, noheader
```

	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y1_x1	.96111	.0930771	10.33	0.000	.7786822	1.143538
y1_x2	.9967672	.1031633	9.66	0.000	.7945709	1.198964
y1_y2	.9574257	.0915622	10.46	0.000	.7779671	1.136884
y1__cons	-.0283237	.0660784	-0.43	0.668	-.157835	.1011877
x1_y2	.9902859	.0333022	29.74	0.000	.9250147	1.055557
x2_y2	1.035864	.0326621	31.71	0.000	.9718472	1.09988
x3_y2	.9710073	.0319614	30.38	0.000	.9083641	1.033651
x4_y2	.9605932	.0326173	29.45	0.000	.8966644	1.024522
_cons_y2	-.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

Appendix 4: "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
x1_y2	.9909343	.0099165	99.93	0.000	.9714982	1.01037
x2_y2	.9890343	.010091	98.01	0.000	.9692563	1.008812
x3_y2	.9804318	.0099311	98.72	0.000	.9609672	.9998965
x4_y2	.9981266	.009831	101.53	0.000	.9788583	1.017395
_cons_y2	.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.

Appendix 5: 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 x1 x2 y2 _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’ (‘v’_‘y2’: _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 6: 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 x1 x2 y2 _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’ (‘v’_‘y2’: _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
```