Multilevel linear models in Stata: a simulation approach

Isabel Cañette

Senior Statistician

StataCorp LP

2011 Mexican Stata Users Group meeting

May 12, 2011

Simulating data for our models

Simulating data is a powerful tool to understand the model we want to fit, and also to spot identification issues.
Let's start by fitting a linear model on the homework dataset[1]

```
use homework
regress math homework
```

The same coefficients can be obtained by using `xtmixed`

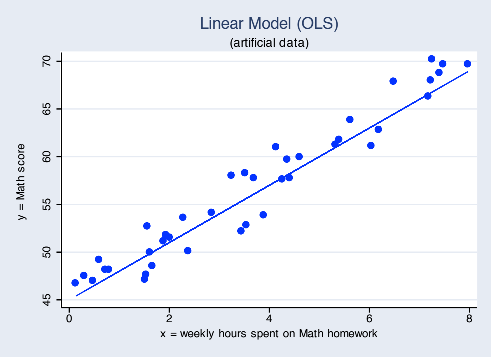
```
. xtmixed math homework, nolog noheader
```

math	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
homework	3.126375	.2860801	10.93	0.000	2.565668	3.687081
_cons	45.56015	.7055719	64.57	0.000	44.17726	46.94305

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
sd(Residual)	9.661575	.2998812	9.09134	10.26758

[1]Kreft, I.G.G and de J. Leeuw. 1998. Introducing Multilevel Modeling. Sage. Rabe-Hesketh, S. and A. Skrondal. 2008. Multilevel and Longitudinal Modeling Using Stata, Second Edition. Stata Press

Simulating data for this model is very simple

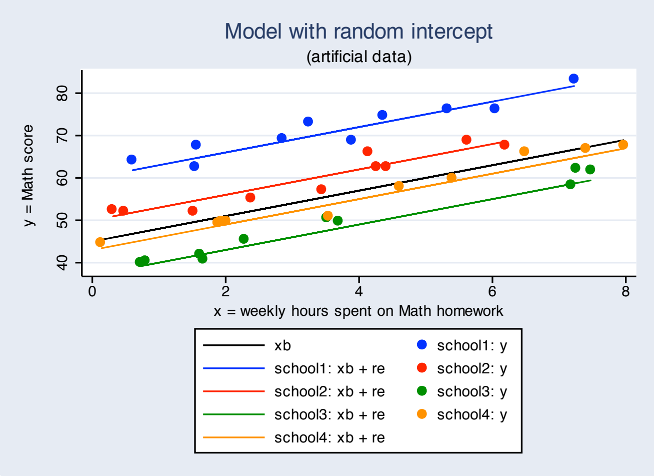



```
. gen x = 8*runiform()
. gen y1 = 3.13*x + 45.56 + 9.66*rnormal()
```

(Notice that I should use the saved results instead of copying them from the screen; I'm just doing this for didactic purposes)

Random-effect models

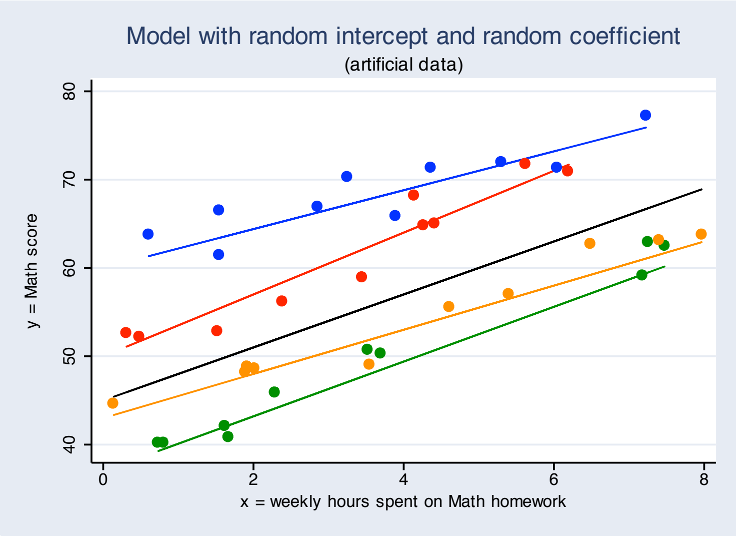
Random intercept only: we are assuming that the intercept varies randomly across schools



The syntax to fit this model would be:

```
xtmixed math homework || schid:
```

Random intercept and random slope: we are assuming that both, intercept and slope, vary randomly across schools)




```
xtmixed math homework || schid: homework
```

```
. xtmixed math homework || schid: homework, nolog  noheader nolrtest
```

math	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
homework	1.974516	.8314652	2.37	0.018	.3448746	3.604158
_cons	46.46441	1.608962	28.88	0.000	43.3109	49.61792

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
schid: Independent				
sd(homework)	3.709275	.6847578	2.58316	5.326314
sd(_cons)	7.12292	1.255007	5.042925	10.06082
sd(Residual)	7.34461	.2419451	6.88539	7.834457

```
. est store original1
```

Simulating data for one-level random-effects models

math	coef
homework	1.974516
_cons	46.46441
schid	Estimate
sd(homework)	3.709275
sd(_cons)	7.12292
sd(Residual)	7.34461

```
set seed 1357
set sortseed 159
set obs 100 // 100 schools
generate schid = _n  // school identifier
generate nu0 = 7.12*rnormal() // random intercept per school
generate nu1 = 3.709*rnormal() // random slope per school
expand 200  // 200 students per school
generate stud_id = _n // student identifier
generate homework = 8*runiform() // indep. variable
generate residual = 7.34*rnormal() // residuals
generate  math = 1.97*homework + 46.46 + nu0 + nu1*homework + residual
xtmixed math homework || schid: homework, nolog noheader nolrtest
est store simulated1
```

```
. estimates table  original1 simulated1
```

Variable	original1	simulated1
math		
homework	1.9745165	1.8530287
_cons	46.464411	46.569009
lns1_1_1		
_cons	1.3108365	1.3818598
lns1_1_2		
_cons	1.9633177	1.8942815
lnsig_e		
_cons	1.9939667	1.9986072

We have assumed that the slope and the intercept are independent. We could have assumed that there was a correlation among them.

```
. xtmixed math homew || schid: homew, cov(unstructured) var nolo nolr nohead
```

math	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
homework	1.980164	.9284486	2.13	0.033	.160438	3.799889
_cons	46.32561	1.758934	26.34	0.000	42.87816	49.77305

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
schid: Unstructured				
var(homework)	17.72652	6.260285	8.871839	35.41875
var(_cons)	62.42455	21.38154	31.90093	122.1539
cov(homework,_cons)	-27.59391	10.56626	-48.3034	-6.884412
var(Residual)	53.29462	3.465962	46.91658	60.53972

```
. est store original2
```

Simulating data for one-level models with correlated random effects

math	coef
homework	1.980164
_cons	46.32561
schid	**Estimate**
var(homework)	17.72652
var(_cons)	62.42455
cov(homework,_cons)	-27.59391
var(Residual)	53.29462

```
clear
set seed 1357
set sortseed 159
set obs 100 // 100 schools
generate schid = _n  // school identifier
matrix a = (17.73, -27.59 \ -27.59, 62.42)
drawnorm nu1 nu0, cov(a) // random slope and intercept
expand 200  // 200 students per school
generate stud_id = _n // student identifier
generate homework = 8*runiform() // indep. variable
generate residual = sqrt(53.29)*rnormal() // residuals
generate  math = 1.98*homework + 46.33 + nu0 + nu1*homework + residual
xtmixed math homework || schid: homework,  ///
         cov(unstructured) var nolog noheader nolrtest
est store original2
```

```
. xtmixed math homework || schid: homework, cov(unstructured) var
```
(output omitted)

```
. est store simulated2
```

```
. est table original2 simulated2
```

Variable	original2	simulated2
math		
homework	1.9801637	2.1013484
_cons	46.325606	45.970628
lns1_1_1		
_cons	1.4375308	1.4200276
lns1_1_2		
_cons	2.0669793	2.0222833
atr1_1_1_2		
_cons	-1.1865765	-1.1093948
lnsig_e		
_cons	1.9879177	1.9931474

Multilevel nested models

Often, researchers tend to model the "natural" nesting structure. For example, schools are naturally nested within regions, because a school can't be in two regions.
xtmixed assumes, by default, that consecutive levels are nested.

```
. xtmixed math homework || region: ||schid:
```

This specification assumes that I have a random intercept for each region, and also one random intercept for each school.

Meaning of "nested"

`xtmixed` assumed that schools on different regions are different, no matter if we repeat the identificators across regions. If we code:

```
region  schid
1    1
1    2
1    3
2    1
2    2
2    3
```

`xtmixed` will interpret that (the effect of) school 1 from region 1 and (the effect of) school 1 from region 2 are different.

Simulating data for nested random-effects models

```
set seed 1357
set sortseed 713
scalar sd_int_region = 5
scalar sd_int_school = 7
scalar sd_res = 1
qui set obs 20  // number of region
gen region = _n // region identifier
gen int_region = sd_int_region*rnormal()
expand 100 // number of schools per region
sort region
gen schoolid = _n // school identifier
gen int_school = sd_int_school*rnormal()
qui expand 100 // number of students per school
gen res = rnormal() // residuals
gen homework = 8*runiform() // indep. variable
gen y = 2*homework +46 + int_region + int_school + res
```

```
. xtmixed y homework || region: ||school:, nolog nolr nohead
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
homework	2.000976	.0009745	2053.38	0.000	1.999067	2.002886
_cons	46.19403	.8541039	54.08	0.000	44.52002	47.86805

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
region: Identity				
sd(_cons)	3.753788	.6304813	2.700866	5.217188
schoolid: Identity				
sd(_cons)	7.060727	.1122247	6.844161	7.284145
sd(Residual)	.998948	.0015874	.9958415	1.002064

Crossed effects

Sometimes we don't want to consider nested-effect models, but crossed-effect models, i.e., models where levels that are not nested. For example, in the pig dataset, we have the dependent variable weight and information on the week and the id.
We may think that each individual pig has some random departure from the line:

```
xtmixed weight week ||id:
```

or instead, that each week determines some departure from this line:

```
xtmixed weigh week || week:
```

What if we want both? We don't want to consider these effects as "nested" How do we simulate data for this model?

Simulating data for crossed-effects models

```
set seed 1357
set sortseed 793
scalar sd_re_week = 1
scalar sd_re_id = 3.5
scalar sd_res = 2
set obs 50 //number of pigs
gen id = _n // pig identifier
gen re_id = sd_re_id*rnormal() // random intercept, pig level
expand 20 // number of weeks
bysort id: gen week = _n // week identifier; these repeat across pigs
gen re_week = sd_re_week*rnormal() // random effect, week
bysort week: replace re_week = re_week[1] // needs to be unique per week
gen res = sd_res*rnormal()
gen weight = 6*week + 19 + re_id + re_week  + res
```

We can estimate the model with the following syntax:

```
. xtmixed weigh week || _all:R.week || id:, nolog nolr nohead
```

weight	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
week	6.003322	.0415515	144.48	0.000	5.921882	6.084761
_cons	19.41274	.6880104	28.22	0.000	18.06426	20.76121

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
_all: Identity				
sd(R.week)	1.033334	.1851922	.7272604	1.468221
id: Identity				
sd(_cons)	3.358588	.3453138	2.745619	4.108404
sd(Residual)	2.004485	.0464529	1.915476	2.097631

Stata tip: always use the R. notation for the level with less categories.

What does exactly, the _all:R.var notation do?

It creates a level "_all" containing all the observations in one category; At this level, a set of covariates is included, consisting of dummies for the categories of var, while constraining the variances to be the same.
That is:

```
xtmixed weight week || _all:R.week
```

Is the same as

```
generate one = 1
tab id, gen(week_dummy)
xtmixed weight week || one: week_dummy*, cov(identity) nocons
```

Which is just an inefficient way to fit the model:

```
xtmixed weight week || week:
```

Naturally-nested vs model-nested models

Let's assume that we have data on return on assets for a set of firms, which belong to different industries and different countries. Industries and countries are naturally crossed. We can model them as they are:

```
. xtmixed asset || _all: R.country ||industry:
```

We might think, instead, that each industry behaves differently for each country, i.e., we can create a "virtual" level, country-industry.

```
. use asset2, clear
. xtmixed asset || country: || industry:
(output omitted)
. estimates store a
```

Application 1: models with crossed and nested effects

Let's assume now that we have repeated measures per firm, and we still have information on industries and countries. We want to model:

- crossed effects on industries and countries
- random effects on firms
- firms nested within both, industries and countries

The first two crossed-levels would be:

```
xtmixed asset || _all: R.country || industry:
```

Now we want firm nested within industry and country. If we write:

```
xtmixed asset || _all: R.country || industry: || firm:
```

Now firms will be nested within industry, which will be nested within _all, and not necessarily within country. What we can do is to generate a variable firm_country, which will be naturally nested within country.

```
gen firm_country = group(firm country)
xtmixed asset || _all: R.country || industry: || firm_country:
```

Application 2: fitting a crossed-effects model with covariates

Let's get back to the crossed-effects model:

```
xtmixed asset || _all: R.country || industry:
```

Now, let's assume that we want to include a covariate with a random coefficient at industry level, let's say company size. This can be done without big modifications on the syntax:

```
xtmixed asset || _all: R.country || industry: size
```

What happens if, in addition, we want to include a covariate with a random coefficient at country level, let's say, amount of taxes per company?

If I write:

```
xtmixed asset || _all: R.country tax || industry: size
```

Then variable tax will be at the "_all" level; this will imply only one realization per coefficient (i.e., a random variable), which will be the same for all the dataset. This is not only not what we want, but also it is a model not identified (Why?).

What we want to do is to create a set of random coefficients for my covariate, with the same variance, independet, and a different "realization" of this random coefficient for each country. This can be done as follows:

```
tab country, gen(id_country)
unab idvar: id_country*
foreach v of local idvar {
  gen tax_`v´ = tax*`v´
}

xtmixed asset || _all: R.country ||_all: tax_*, cov(identity) nocons ///
               || industry: size
```

I am estimating a set of random coefficients for tax, a different realization for each country, and I'm using cov(identity) to establish that these coefficients should be i.i.d.

Final remarks

- `xtmixed` is a versatile command that allows us to fit a variety of models.
- Understanding the mechanics of each piece in the syntax allows us to fit very sophisticated models.
- Simulating data allows us to get a deeper insight on multilevel models, to understand the particular specification we want to use, and eventually spot identification problems.
- `xtmixed` also allows us to specify different structures for the errors, feature not covered in this talk. This feature opens a new array of models, including models with multivariate response.