How can I access the random effects after using the “mixed” command in Stata?

Authored by

stats writer

July 1, 2024

After using the "mixed" command in Stata, the random effects can be accessed by using the "mixed effects" option. This will display the estimated random effects for each level of the grouping variable in the model. The random effects can also be saved and exported for further analysis by using the "save" command. Additionally, the "mixed predict" command can be used to generate predicted values for the random effects. These options allow for further exploration and interpretation of the random effects in the mixed model.

How can I access the random effects after mixed using _diparm? | Stata FAQ

Consider the following mixed model:

```
use https://stats.idre.ucla.edu/stat/data/depression, clear
```

mixed depression i.group visit || sid: visit, cov(un) stddev

Performing EM optimization:

Performing gradient-based optimization:

Iteration 0: log likelihood = -826.15973
Iteration 1: log likelihood = -826.15969

Computing standard errors:

Mixed-effects ML regression Number of obs = 295
Group variable: sid Number of groups = 61

Obs per group:

min = 1

avg = 4.8

max = 6

Wald chi2(2) = 65.86

Log likelihood = -826.15969 Prob > chi2 = 0.0000

--

depression | Coef. Std. Err. z P>|z|

-------------+--

group |

estrogen | -3.795494 1.181205 -3.21 0.001 -6.110614 -1.480374

visit | -1.20499 .1651257 -7.30 0.000 -1.528631 -.8813499

_cons | 17.96487 .9941403 18.07 0.000 16.01639 19.91335

--

--

Random-effects Parameters | Estimate Std. Err.

-----------------------------+--

sid: Unstructured |

sd(visit) | .9108339 .1544413 .6532947 1.269899

sd(_cons) | 5.066354 .6109308 3.999932 6.417094

corr(visit,_cons) | -.5506068 .1358938 -.7622138 -.2326851

---+---

sd(Residual) | 2.892052 .1503486 2.61189 3.202266

LR test vs. linear model: chi2(3) = 173.80 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

The option stddev request the random effects as standard deviation units instead of default variances. Say that you want to use the random effect sd(visit) or cov(visit,_cons) for additional
computations. You can access these values using the undocumented command _diparm
(which stands for display parameter).
When we say undocumented, we mean that it is not in the full manual but it is included
in the online help system (help _diparm).

To use _diparm you have to understand how Stata computes the random effects. Stata

computes the variances as the log of the standard deviation (ln_sigma) and computes
covariances as the arc hyperbolic tangent of the correlation.

You also need to how stmixed names the random effects. The coeflegend option
will not provide these names. The easiest way to get the names of the random effects is to
list of the e(b) matrix, like this.

```
matrix list e(b)
```

e(b)
depression: depression: depression: depression: depression:
lns1_1_1: lns1_1_2: atr1_1_1_2: lnsig_e:
0b. 1.
group group visit _cons _cons _cons _cons _cons
y1 0 -3.7954944 -1.2049904 17.964869 -.09339475
1.6226214 -.61925171 1.0619665

Thus, lns1_1_1 is the name for sd(visit), lns1_1_2 is the name for sd(_cons) and
atr1_1_1_2 is the name for corr(visit,_cons).

**Finally, you need to provide _diparm with three pieces of information: 1)
the name of the parameter, 2) the inverse function (abbr f), and 3)
the derivative (abbr d) of the inverse function.**

**For example, to get the estimate for sd(visit) the name is lns1_1_1. Since mixed
estimates the ln_sigma, the inverse function is exp(). The derivative is easy because the
derivative of exp() is just exp(). Here is _diparm in action.**

```
_diparm lns1_1_1, f(exp(@)) d(exp(@))
```

/lns1_1_1 | .9108339 .1544413 .6532947 1.269899

Note the use of the @ symbol as a place holder for the argument of the function.

All of the information from the _diparm is stored in the return list.

```
return list
```

scalars:

r(cns) = 0

r(crit) = 1.959963984540054

r(df) = .

r(z) = .

r(p) = .

r(i) = 0

r(ub) = 1.269899149716271

r(lb) = .6532946742247581

r(est) = .9108338769019235

r(se) = .1544413384973653

For corr(visit,_cons) the name is atr1_1_1_2, the inverse function is tanh() and the derivative is 1-tanh()^2.

```
_diparm atr1_1_1_2, f(tanh(@)) d(1-tanh(@)^2)
```

/atr1_1_1_2 | -.5506068 .1358938 -.7622138 -.2326851

Next we will rerun the mixed without the stddev option.

```
mixed
```

Mixed-effects ML regression Number of obs = 295

Group variable: sid Number of groups = 61

Obs per group:

min = 1
avg = 4.8
max = 6

Wald chi2(2) = 65.86
Log likelihood = -826.15969 Prob > chi2 = 0.0000

depression	Coef.	Std. Err.	z	P>|z|		
group						
estrogen	-3.795494	1.181205	-3.21	0.001	-6.110614	-1.480374
visit	-1.20499	.1651257	-7.30	0.000	-1.528631	-.8813499
_cons	17.96487	.9941403	18.07	0.000	16.01639	19.91335

Random-effects Parameters	Estimate	Std. Err.		
sid: Unstructured				
var(visit)	.8296184	.2813408	.4267939	1.612644
var(_cons)	25.66794	6.190383	15.99946	41.1791
cov(visit,_cons)	-2.540834	1.122156	-4.740219	-

.3414481

----------+----------

var(Residual) | 8.363968 .8696321 6.82197 10.25451

LR test vs. linear model: chi2(3) = 173.80 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

We could always get the variance by squaring the standard deviation we computed
earlier. But, that wouldn't give us the standard error or confidence intervals.
To get all of these values, we will rerun _diparm with a slightly different
inverse function and derivative.

```
_diparm lns1_1_1, f(exp(@)^2) d(2*exp(@)^2)
```

/ /lns1_1_1 | .8296184 .2813408 .4267939 1.612644

Getting cov(visit,_cons) is a bit more work. Computing a covariance from a correlation is
just a matter of multiplying the correlation by the two

standard deviations.

Doing this with _diparm means that we will need to use three arguments in the command.

We will need to use three place holders for the argument; @1, @2 and @3.

```
_diparm atr1_1_1_2 lns1_1_1 lns1_1_2, f(tanh(@1)*exp(@2+@3)) ///
d((1-tanh(@1)^2)*exp(@2+@3) tanh(@1)*exp(@2+@3) tanh(@1)*exp(@2+@3))
```

/atr1_1_1_2 | -2.540834 1.122156 -4.740219 -.3414481

This time the inverse function had one term with three arguments while the derivative had

three terms, one for each argument. Also note that exp(@2)*exp(@3) was expressed as exp(@2+@3).