14.170: Programming for Economists

1/12/2009-1/16/2009

Melissa Dell
Matt Notowidigdo
Paul Schrimpf

Lecture 3, Maximum Likelihood Estimation in Stata

Introduction to MLE

- Stata has a built-in language to write ML estimators. It uses this language to write many of its built-in commands
  - e.g. probit, tobit, logit, clogit, glm, xtpoisson, etc.

- I find the language very easy-to-use. For simple log-likelihood functions (especially those that are linear in the log-likelihood of each observation), implementation is trivial and the built-in maximization routines are good

- Why should you use Stata ML?
  - Stata will automatically calculate numerical gradients for you during each maximization step
  - Have access to Stata's syntax for dealing with panel data sets (for panel MLE this can result in very easy-to-read code)
  - Can use as a first-pass to quickly evaluate whether numerical gradients/Hessians are going to work, or whether the likelihood surface is too difficult to maximize.

- Why shouldn't you use Stata ML?
  - Maximization options are limited (standard Newton-Raphson and BHHH are included, but more recent algorithms not yet programmed)
  - Tools to guide search over difficult likelihood functions aren't great

ML with linear model and normal errors

Log-likelihood for linear regression model (using normal distribution):

$$
\begin{aligned}
L &= \prod_{i=1}^{N} \left(\frac{1}{\sigma} \phi \left(\frac{y_i - x_i \beta}{\sigma} \right) \right) \\
\log(L) &= \sum_{i=1}^{N} \log \left(\frac{1}{\sigma} \phi \left(\frac{y_i - x_i \beta}{\sigma} \right) \right)
\end{aligned}
$$

NOTE: This log-likelihood function satisfies the "linear form" restriction since the log-likelihood function is the sum of each observations log-likelihood function.

Basic Stata ML

```
program drop _all
program mynormal_lf
 args lnf mu sigma
 qui replace `lnf' = log((1/`sigma')*normalden(($ML_y1 - `mu')/`sigma'))
end

clear
set obs 100
set seed 12345
gen x = invnormal(uniform())
gen y = 2*x + invnormal(uniform())
ml model lf mynormal_lf (y = x) ()
ml maximize
reg y x
```

```
. ml maximize

initial:       log likelihood =     -<inf>  (could not be evaluated)
feasible:      log likelihood = -1087.6507
rescale:       log likelihood = -277.86442
rescale eq:    log likelihood = -223.49417
Iteration 0:   log likelihood = -223.49417  (not concave)
Iteration 1:   log likelihood = -161.99355
Iteration 2:   log likelihood = -145.92343
Iteration 3:   log likelihood = -143.61901
Iteration 4:   log likelihood = -143.61869
Iteration 5:   log likelihood = -143.61869

                                                  Number of obs   =        100
                                                  Wald chi2(1)    =     381.34
Log likelihood = -143.61869                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
           x |   1.995282   .1021763    19.53   0.000      1.79502    2.195544
       _cons |   .0798982   .1020926     0.78   0.434    -.1201996    .2799961
-------------+----------------------------------------------------------------
eq2          |
       _cons |   1.017398   .0719409    14.14   0.000     .8763964      1.1584
------------------------------------------------------------------------------

. reg y x

      Source |       SS       df       MS              Number of obs =     100
-------------+------------------------------           F(  1,    98) =  373.71
       Model |  394.721108     1  394.721108           Prob > F      =  0.0000
    Residual |  103.509873    98  1.05622319           R-squared     =  0.7922
-------------+------------------------------           Adj R-squared =  0.7901
       Total |   498.23098    99  5.03263616           Root MSE      =  1.0277

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.995282   .1032136    19.33   0.000     1.790458    2.200106
       _cons |   .0798982   .1031291     0.77   0.440    -.1247581    .2845546
------------------------------------------------------------------------------
```

ML with linear regression

```
program drop _all
program mynormal_lf
  args lnf mu sigma
  qui replace `lnf' = log((1/`sigma')*normden(($ML_y1-`mu')/`sigma'))
end

clear
set obs 100
set seed 12345
gen x = invnormal(uniform())
gen y = 2*x + x*x*invnormal(uniform())
gen keep = (uniform() > 0.1)
gen weight = uniform()
ml model lf mynormal_lf (y = x) () [aw=weight] if keep == 1, robust
ml maximize
reg y x [aw=weight] if keep == 1, robust
```

```
. ml model lf mynormal_lf (y = x) () [aw=weight] if keep == 1, robust

. ml maximize

initial:       log pseudolikelihood =     -<inf>  (could not be evaluated)
feasible:      log pseudolikelihood = -1202.2519
rescale:       log pseudolikelihood = -267.49701
rescale eq:    log pseudolikelihood = -208.45765
Iteration 0:   log pseudolikelihood = -208.45765  (not concave)
Iteration 1:   log pseudolikelihood = -159.93796
Iteration 2:   log pseudolikelihood = -153.65392
Iteration 3:   log pseudolikelihood = -152.52916
Iteration 4:   log pseudolikelihood = -152.52463
Iteration 5:   log pseudolikelihood = -152.52463

                                                  Number of obs   =         88
                                                  Wald chi2(1)    =      51.93
Log pseudolikelihood = -152.52463                 Prob > chi2     =     0.0000

------------------------------------------------------------------------------
             |               Robust
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
           x |   2.078282   .2883878     7.21   0.000     1.513052    2.643512
       _cons |  -.1211747   .1594751    -0.76   0.447      -.43374    .1913907
-------------+----------------------------------------------------------------
eq2          |
       _cons |   1.369295   .3806752     3.60   0.000     .6231852    2.115405
------------------------------------------------------------------------------

. reg y x [aw=weight] if keep == 1, robust
(sum of wgt is   4.6299e+01)

Regression with robust standard errors                 Number of obs =      88
                                                       F(  1,    86) =   51.34
                                                       Prob > F      =  0.0000
                                                       R-squared     =  0.6901
                                                       Root MSE      =  1.3851

------------------------------------------------------------------------------
             |               Robust
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   2.078282   .2900597     7.17   0.000     1.501662    2.654901
       _cons |  -.1211747   .1603995    -0.76   0.452    -.4400384     .197689
------------------------------------------------------------------------------
```

What's going on in the background?

- We just wrote a 3 (or 5) line program. What does Stata do with it?
- When we call "ml maximize" it does the following steps:
  - Initializes the parameters (the "betas") to all zeroes
  - As long as it has not declared convergence
    - Calculates the gradient at the current parameter value
    - Takes a step
    - Updates parameters
    - Test for convergence (based on either gradient, Hessian, or combination)
  - Displays the parameters as regression output (ereturn!)

How does it calculate gradient?

- Since we did not program a gradient, Stata will calculate gradients numerically. It will calculate a gradient by finding a <u>numerical derivative</u>.

- Review:
  - Analytic derivative is the following:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

  - So that leads to a simple approximation formula for "suitably small but large enough h"; this is a numerical derivative of a function:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

  - Stata knows how to choose a good "h" and in general it gets it right

- Stata updates its parameter guess using the numerical derivatives as follows (i.e. it takes a "Newton" step):

$$\theta_{t+1} = \theta_t - \frac{f'(\theta_t)}{f''(\theta_t)}$$

probit

Log-likelihood function for probit:

$$
\begin{aligned}
\log(L) &= \sum_{i=1}^{N} \log(L_i) \\
\log(L_i) &= \log[(\Phi(X'\beta))^{y_i}(1-\Phi(X'\beta))^{1-y_i}] \\
&= y_i * \log(\Phi(X'\beta)) + (1-y_i) * (1-\Phi(X'\beta)) \\
&= y_i * \log(\Phi(X'\beta)) + (1-y_i) * (\Phi(-X'\beta))
\end{aligned}
$$

Back to Stata ML (myprobit)

```
program drop _all
program myprobit_lf
  args lnf xb
  qui replace `lnf' = ln(norm( `xb'))   if $ML_y1 == 1
  qui replace `lnf' = ln(norm(-1*`xb')) if $ML_y1 == 0
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model lf myprobit_lf (y = x)
ml maximize
probit y x
```

TMTOWTDI!

```
program drop _all
program myprobit_lf
  args lnf xb
  qui replace `lnf' =  ///
      $ML_y1*ln(norm(`xb')) + (1-$ML_y1)*(1 - ln(norm(`xb')))
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model lf myprobit_lf (y = x)
ml maximize
probit y x
```

```
. probit y x

Iteration 0:   log likelihood = -612.54939
Iteration 1:   log likelihood = -542.22446
Iteration 2:   log likelihood = -541.15783
Iteration 3:   log likelihood = -541.15684

Probit estimates                                  Number of obs   =       1000
                                                  LR chi2(1)      =     142.79
                                                  Prob > chi2     =     0.0000
Log likelihood = -541.15684                       Pseudo R2       =     0.1165

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367247   .0475177    11.30   0.000     .4435918    .6298576
       _cons |   .5750247   .0447145    12.86   0.000     .4873859    .6626636
------------------------------------------------------------------------------

. ml model lf myprobit_lf (y = x)

. ml maximize

initial:       log likelihood = -693.14718
alternative:   log likelihood = -612.64995
rescale:       log likelihood = -612.64995
Iteration 0:   log likelihood = -612.64995
Iteration 1:   log likelihood = -541.46569
Iteration 2:   log likelihood = -541.15686
Iteration 3:   log likelihood = -541.15684

                                                  Number of obs   =       1000
                                                  Wald chi2(1)    =     127.58
Log likelihood = -541.15684                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367247   .0475177    11.30   0.000     .4435917    .6298576
       _cons |   .5750247   .0447146    12.86   0.000     .4873858    .6626636
------------------------------------------------------------------------------
```

What happens here?

```
program drop _all
program myprobit_lf
  args lnf xb
  qui replace `lnf' = ln(norm( `xb'))   if $ML_y1 == 1
  qui replace `lnf' = ln(norm(-1*`xb')) if $ML_y1 == 0
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model lf myprobit_lf (y = x) ()
ml maximize
probit y x
```

Difficult likelihood functions?

```
. ml model lf myprobit_lf (y = x) ()

. ml maximize

initial:        log likelihood = -693.14718
alternative:    log likelihood = -612.64995
rescale:        log likelihood = -612.64995
rescale eq:     log likelihood = -612.64995
could not calculate numerical derivatives
flat or discontinuous region encountered
r(430);
```

- Stata will give up if it can’t calculate numerical derivatives. This can be a big pain, especially if it’s a long-running process and happens after a long time. If this is not a bug in your code (like last slide), a lot of errors like this is a sign to leave Stata so that you can get better control of the maximization process.

- A key skill is figuring whether the error above is “bug” in your program or if it is a difficult likelihood function to maximize.

Transforming parameters

```
program drop _all
program mynormal_lf
  args lnf mu ln_sigma
  tempvar sigma
  gen double `sigma' = exp(`ln_sigma')
  qui replace `lnf' = log((1/`sigma')*normden(($ML_y1-`mu')/`sigma'))
end

clear
set obs 100
set seed 12345
gen x = invnormal(uniform())
gen y = 2*x + 0.01*invnormal(uniform())
ml model lf mynormal_lf (y = x) /log_sigma
ml maximize
reg y x
```

$$
\begin{aligned}
\sigma &\in (0,\infty) \Rightarrow \log(\sigma) \in (-\infty,\infty) \\
\rho &\in (-1,1) \Rightarrow \frac{1}{2}\log\left(\frac{1+x}{1-x}\right) \in (-\infty,\infty)
\end{aligned}
$$

From “lf” to “d0”, “d1”, and “d2”

- In some (rare) cases you will want to code the gradient (and possibly) the Hessian by hand. If there are simple analytic formulas for these and/or you need more speed and/or the numerical derivatives are not working out very well, this can be a good thing to do.

- Every ML estimator we have written so far has been of type “lf”. In order to calculate analytic gradients, we need to use a “d1” or a “d2” ML estimator

- But before we can implement the analytic formulas for the gradient and Hessian in CODE, we need to derive the analytic formulas themselves.

gradient and Hessian for probit

Gradient and Hessian functions for probit:

$$
\begin{aligned}
\log(L_i) &= y_i * \log(\Phi(X'\beta)) + (1-y_i) * (\Phi(-X'\beta)) \\
g_j &= \frac{\partial \log(L_j)}{\partial(X'\beta)} = y_i * \phi(X'\beta)/\Phi(X'\beta) - (1-y_i) * (\phi(X'\beta)/\Phi(-X'\beta)) \\
H_j &= \frac{\partial^2 \log(L_j)}{\partial(X'\beta)^2} = \frac{\phi(X'\beta)(X'\beta) * \Phi(X'\beta) - \phi(X'\beta) * \phi(X'\beta)}{[\Phi(X'\beta)]^2} \\
&= -g_j * (g_j + X'\beta)
\end{aligned}
$$

note we will usually program the NEGATIVE Hessian

More probit (d0)

```
program drop _all
program myprobit_d0
  args todo b lnf
  tempvar xb l_j
  mleval `xb' = `b'
  qui {
    gen `l_j' = normalden( `xb')  if $ML_y1 == 1
    replace `l_j' = normalden(-1 * `xb')  if $ML_y1 == 0
    mlsum `lnf' = ln(`l_j')
  }
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model d0 myprobit_d0 (y = x)
ml maximize
probit y x
```

```
. ml model d0 myprobit_d0 (y = x)

. ml maximize

initial:       log likelihood = -693.14718
alternative:   log likelihood = -612.64995
rescale:       log likelihood = -612.64995
Iteration 0:   log likelihood = -612.64995
Iteration 1:   log likelihood = -541.46519  (not concave)
Iteration 2:   log likelihood =  -541.4647  (not concave)
Iteration 3:   log likelihood = -541.27841
Iteration 4:   log likelihood = -541.17061  (not concave)
Iteration 5:   log likelihood = -541.16801  (not concave)
Iteration 6:   log likelihood = -541.15687  (not concave)
Iteration 7:   log likelihood = -541.16635  (not concave)
Iteration 8:   log likelihood = -541.15789
Iteration 9:   log likelihood = -541.15749  (not concave)
Iteration 10:  log likelihood = -541.15691
Iteration 11:  log likelihood = -541.15684
Iteration 12:  log likelihood = -541.15684

                                                  Number of obs   =       1000
                                                  Wald chi2(1)    =     131.65
Log likelihood = -541.15684                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367257    .046778    11.47   0.000     .4450425     .628409
       _cons |   .5750386   .0427927    13.44   0.000     .4911664    .6589108
------------------------------------------------------------------------------

. probit y x

Iteration 0:   log likelihood = -612.54939
Iteration 1:   log likelihood = -542.22446
Iteration 2:   log likelihood = -541.15783
Iteration 3:   log likelihood = -541.15684

Probit estimates                                  Number of obs   =       1000
                                                  LR chi2(1)      =     142.79
                                                  Prob > chi2     =     0.0000
Log likelihood = -541.15684                       Pseudo R2       =     0.1165

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367247   .0475177    11.30   0.000     .4435918    .6298576
       _cons |   .5750247   .0447145    12.86   0.000     .4873859    .6626636
------------------------------------------------------------------------------
```

More probit (d0)

```
program drop _all
program myprobit_d0
  args todo b lnf
  tempvar xb l_j
  mleval `xb' = `b'
  qui {
    gen double `l_j' = norm( `xb')  if $ML_y1 == 1
    replace `l_j' = norm(-1 * `xb')  if $ML_y1 == 0
    mlsum `lnf' = ln(`l_j')
  }
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model d0 myprobit_d0 (y = x)
ml maximize
probit y x
```

Still more probit (d1)

```
program drop _all
program myprobit_d1
  args todo b lnf g
  tempvar xb l_j g1
  mleval `xb' = `b'
  qui {
    gen double `l_j' = norm( `xb')   if $ML_y1 == 1
    replace `l_j' = norm(-1 * `xb')  if $ML_y1 == 0
    mlsum `lnf' = ln(`l_j')

    gen double `g1' = normden(`xb')/`l_j' if $ML_y1 == 1
    replace `g1' = -normden(`xb')/`l_j'   if $ML_y1 == 0
    mlvecsum `lnf' `g' = `g1', eq(1)
  }
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model d1 myprobit_d1 (y = x)
ml maximize
probit y x
```

```
. ml model d1 myprobit_d1 (y = x)

. ml maximize

initial:       log likelihood = -693.14718
alternative:   log likelihood = -612.64995
rescale:       log likelihood = -612.64995
Iteration 0:   log likelihood = -612.64995
Iteration 1:   log likelihood = -541.46581
Iteration 2:   log likelihood = -541.15686
Iteration 3:   log likelihood = -541.15684

                                                  Number of obs   =       1000
                                                  Wald chi2(1)    =     127.58
Log likelihood = -541.15684                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367246   .0475177    11.30   0.000     .4435917    .6298576
       _cons |   .5750247   .0447146    12.86   0.000     .4873858    .6626636
------------------------------------------------------------------------------

. probit y x

Iteration 0:   log likelihood = -612.54939
Iteration 1:   log likelihood = -542.22446
Iteration 2:   log likelihood = -541.15783
Iteration 3:   log likelihood = -541.15684

Probit estimates                                  Number of obs   =       1000
                                                  LR chi2(1)      =     142.79
                                                  Prob > chi2     =     0.0000
Log likelihood = -541.15684                       Pseudo R2       =     0.1165

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367247   .0475177    11.30   0.000     .4435918    .6298576
       _cons |   .5750247   .0447145    12.86   0.000     .4873859    .6626636
------------------------------------------------------------------------------
```

Last probit, I promise (d2)

```
program drop _all
program myprobit_d2
  args todo b lnf g negH
  tempvar xb l_j g1
  mleval `xb' = `b'
  qui {
    gen double `l_j' = norm( `xb')   if $ML_y1 == 1
    replace `l_j' = norm(-1 * `xb')  if $ML_y1 == 0
    mlsum `lnf' = ln(`l_j')

    gen double `g1' = normden(`xb')/`l_j' if $ML_y1 == 1
    replace `g1' = -normden(`xb')/`l_j'   if $ML_y1 == 0
    mlvecsum `lnf' `g' = `g1', eq(1)

    mlmatsum `lnf' `negH' = `g1'*(`g1'+`xb'), eq(1,1)
  }
end

clear
set obs 1000
set seed 12345
gen x = invnormal(uniform())
gen y = (0.5 + 0.5*x  > invnormal(uniform()))
ml model d2 myprobit_d2 (y = x)
ml search
ml maximize
probit y x
```

```
. ml model d2 myprobit_d2 (y = x)

. ml search
initial:       log likelihood = -693.14718
improve:       log likelihood = -693.14718
alternative:   log likelihood = -612.64995
rescale:       log likelihood = -612.64995

. ml maximize

initial:       log likelihood = -612.64995
rescale:       log likelihood = -612.64995
Iteration 0:   log likelihood = -612.64995
Iteration 1:   log likelihood = -541.46581
Iteration 2:   log likelihood = -541.15686
Iteration 3:   log likelihood = -541.15684

                                                  Number of obs   =       1000
                                                  Wald chi2(1)    =     127.58
Log likelihood = -541.15684                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367246   .0475177    11.30   0.000     .4435917    .6298576
       _cons |   .5750247   .0447146    12.86   0.000     .4873858    .6626636
------------------------------------------------------------------------------

. probit y x

Iteration 0:   log likelihood = -612.54939
Iteration 1:   log likelihood = -542.22446
Iteration 2:   log likelihood = -541.15783
Iteration 3:   log likelihood = -541.15684

Probit estimates                                  Number of obs   =       1000
                                                  LR chi2(1)      =     142.79
                                                  Prob > chi2     =     0.0000
Log likelihood = -541.15684                       Pseudo R2       =     0.1165

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .5367247   .0475177    11.30   0.000     .4435918    .6298576
       _cons |   .5750247   .0447145    12.86   0.000     .4873859    .6626636
------------------------------------------------------------------------------
```

Beyond linear-form likelihood fn's

- Many ML estimators I write down do NOT satisfy the linear-form restriction, but OFTEN they have a simple panel structure (e.g. think of any "xt*" command in Stata that is implemented in ML)

- Stata has nice intuitive commands to deal with panels (e.g. "by" command!) that work inside ML programs

- As an example, let's develop a random-effects estimator in Stata ML. This likelihood function does NOT satisfy the linear-form restriction (i.e. the overall log-likelihood function is NOT just the sum of the individual log-likelihood functions)

- This has two purposes:
  - More practice going from MATH to CODE
  - Good example of a panel data ML estimator implementation

Panel model with random effects:

$$\begin{aligned} y_{it} &= x_{it}\beta + u_i + e_{it} \\ u_i &\sim N(0, \sigma_u^2) \\ e_{it} &\sim N(0, \sigma_e^2) \end{aligned}$$

Log-likelihood function is given by the following:

$$\log(L) = \sum_{i=1}^{N} \log(L_i)$$

where i indexes the *group*, not the observation. The log-likelihood function for the group is the following:

$$\begin{aligned} \log(L_i) &= \log\left(\int_{-\infty}^{\infty} f(u_i) \prod_{t=1}^{T} f(y_{it}|u_i) du_i\right) \\ &= \log\left(\int_{-\infty}^{\infty} \frac{1}{\sigma_u\sqrt{2\pi}} \exp\left(-\frac{u_i^2}{2\sigma_u^2}\right) \prod_{t=1}^{T} \left(\frac{1}{\sigma_e\sqrt{2\pi}} \exp\left\{-\frac{(y_{it} - u_i - x_{it}\beta)^2}{2\sigma_e^2}\right\}\right) du_i\right) \\ &= -\frac{1}{2}\left\{\frac{\sum_{t=1}^{T} z_{it}^2 - a_i\left(\sum_{t=1}^{T} z_{it}\right)^2}{\sigma_e^2} + \log(T * \sigma_u^2/\sigma_e^2 + 1) + T * \log(2\pi\sigma_e^2)\right\} \end{aligned}$$

where T is the number of observations for each group, $z_{it} = y_{it} - x_{it}\beta$ and $a_i = \sigma_u^2/(T * \sigma_u^2 + \sigma_e^2)$

Random effects in ML

```
program drop _all
program define myrereg_d0
  args todo b lnf
  tempvar xb z T S_z2 Sz_2 S_temp a first
  tempname sigma_u sigma_e ln_sigma_u ln_sigma_e
  mleval `xb' = `b', eq(1)
  mleval `ln_sigma_u' = `b', eq(2) scalar
  mleval `ln_sigma_e' = `b', eq(3) scalar
  scalar `sigma_u' = exp(`ln_sigma_u')
  scalar `sigma_e' = exp(`ln_sigma_e')

  ** hack!
  sort $panel
  qui {
    gen double `z' = $ML_y1 - `xb'
    by $panel: gen `T' = _N
    gen double `a' = (`sigma_u'^2) / (`T'*(`sigma_u'^2) + `sigma_e'^2)
    by $panel: egen double `S_z2' = sum(`z'^2)
    by $panel: egen double `S_temp' = sum(`z')
    by $panel: gen double `Sz_2'    = `S_temp'^2
    by $panel: gen `first' = (_n == 1)
    mlsum `lnf' = -.5 *                                ///
          (   (`S_z2' - `a'*`Sz_2')/(`sigma_e'^2)  +   ///
              log(`T'*`sigma_u'^2/`sigma_e'^2 + 1) +   ///
              `T'*log(2* _pi * `sigma_e'^2)            ///
          )  if `first' == 1
  }
end
```

Random effects in ML

```
gen double `z' = $ML_y1 - `xb'
by $panel: gen `T' = _N
gen double `a' = (`sigma_u'^2) / (`T'*(`sigma_u'^2) + `sigma_e'^2)
by $panel: egen double `S_z2' = sum(`z'^2)
by $panel: egen double `S_temp' = sum(`z')
by $panel: gen double `Sz_2'    = `S_temp'^2
by $panel: gen `first' = (_n == 1)
mlsum `lnf' = -.5 *                                ///
      (   (`S_z2' - `a'*`Sz_2')/(`sigma_e'^2)  +   ///
          log(`T'*`sigma_u'^2/`sigma_e'^2 + 1) +   ///
          `T'*log(2* _pi * `sigma_e'^2)            ///
      )  if `first' == 1
```

Random effects in ML

where T is the number of observations for each group, $z_{it} = y_{it} - x_{it}\beta$ and $a_i = \sigma_u^2/(T * \sigma_u^2 + \sigma_e^2)$

```
gen double `z' = $ML_y1 - `xb'
by $panel: gen `T' = _N
gen double `a' = (`sigma_u'^2) / (`T'*(`sigma_u'^2) + `sigma_e'^2)
```

Random effects in ML

$$\sum_{t=1}^{T} z_{it}^2 - a_i \left(\sum_{t=1}^{T} z_{it} \right)^2$$

```
by $panel: egen double `S_z2' = sum(`z'^2)
by $panel: egen double `S_temp' = sum(`z')
by $panel: gen double `Sz_2'    = `S_temp'^2
```

Random effects in ML

$$-\frac{1}{2}\left\{\frac{\sum_{t=1}^{T} z_{it}^2 - a_i\left(\sum_{t=1}^{T} z_{it}\right)^2}{\sigma_e^2} + \log(T * \sigma_u^2/\sigma_e^2 + 1) + T * \log(2\pi\sigma_e^2)\right\}$$

```
by $panel: gen `first' = (_n == 1)
mlsum `lnf' = -.5 *                                     ///
      (   (`S_z2' - `a'*`Sz_2')/(`sigma_e'^2)  +   ///
          log(`T'*`sigma_u'^2/`sigma_e'^2 + 1) +   ///
          `T'*log(2* _pi * `sigma_e'^2)            ///
      )  if `first' == 1
```

Random effects in ML

```
program drop _all
program define myrereg_d0
  args todo b lnf
  tempvar xb z T S_z2 Sz_2 S_temp a first
  tempname sigma_u sigma_e ln_sigma_u ln_sigma_e
  mleval `xb' = `b', eq(1)
  mleval `ln_sigma_u' = `b', eq(2) scalar
  mleval `ln_sigma_e' = `b', eq(3) scalar
  scalar `sigma_u' = exp(`ln_sigma_u')
  scalar `sigma_e' = exp(`ln_sigma_e')

  ** hack!
  sort $panel
```

Random effects in ML

```
clear
set obs 100
set seed 12345
gen x = invnormal(uniform())
gen id = 1 + floor((_n - 1)/10)
bys id: gen fe = invnormal(uniform())
bys id: replace fe = fe[1]
gen y = x + fe + invnormal(uniform())
global panel = "id"
ml model d0 myrereg_d0 (y = x) /ln_sigma_u /ln_sigma_e
ml search
ml maximize
xtreg y x, i(id) re
```

“my” MLE RE vs. XTREG, MLE

```
. ml model d0 myrereg_d0 (y = x) /ln_sigma_u /ln_sigma_e

. ml search
initial:       log likelihood = -207.71466
improve:       log likelihood = -207.71466
alternative:   log likelihood = -192.08714
rescale:       log likelihood = -191.69318
rescale eq:    log likelihood = -187.99763

. ml maximize

initial:       log likelihood = -187.99763
rescale:       log likelihood = -187.99763
rescale eq:    log likelihood = -187.99763
Iteration 0:   log likelihood = -187.99763  (not concave)
Iteration 1:   log likelihood = -154.31332
Iteration 2:   log likelihood = -150.62272
Iteration 3:   log likelihood = -149.93738
Iteration 4:   log likelihood = -149.93725
Iteration 5:   log likelihood = -149.93725

                                                  Number of obs   =        100
                                                  Wald chi2(1)    =     116.17
Log likelihood = -149.93725                       Prob > chi2     =     0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
eq1          |
           x |   1.099201   .1019817    10.78   0.000     .8993203    1.299081
       _cons |   .3374044   .2492308     1.35   0.176     -.151079    .8258877
-------------+----------------------------------------------------------------
ln_sigma_u   |
       _cons |  -.3242579   .2658427    -1.22   0.223    -.8453001    .1967842
-------------+----------------------------------------------------------------
ln_sigma_e   |
       _cons |  -.0120325   .0745392    -0.16   0.872    -.1581266    .1340616
------------------------------------------------------------------------------
```

```
. xtreg y x, i(id) mle

Fitting constant-only model:
Iteration 0:   log likelihood = -206.05665
Iteration 1:   log likelihood =  -191.4423
Iteration 2:   log likelihood = -187.64957
Iteration 3:   log likelihood = -187.20916
Iteration 4:   log likelihood = -187.19965
Iteration 5:   log likelihood = -187.19965

Fitting full model:
Iteration 0:   log likelihood = -150.01883
Iteration 1:   log likelihood = -149.93753
Iteration 2:   log likelihood = -149.93725

Random-effects ML regression                    Number of obs      =       100
Group variable: id                              Number of groups   =        10

Random effects u_i ~ Gaussian                   Obs per group: min =        10
                                                               avg =      10.0
                                                               max =        10

                                                LR chi2(1)         =     74.52
Log likelihood  = -149.93725                    Prob > chi2        =    0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.099201   .1019811    10.78   0.000     .8993216     1.29908
       _cons |   .3374044   .2492309     1.35   0.176    -.1510792    .8258879
-------------+----------------------------------------------------------------
    /sigma_u |   .7230637   .1922211                      .4294286    1.217481
    /sigma_e |   .9880396   .0736465                      .8537437    1.143461
         rho |   .3487698    .127123                       .144077    .6121367
------------------------------------------------------------------------------
Likelihood-ratio test of sigma_u=0: chibar2(01)=   24.32 Prob>=chibar2 = 0.000
```

Point estimates identical but standard errors different; why?

Exercises

(A) Implement logit as a simple (i.e. “lf”) ML estimator using Stata’s ML language

(If you have extra time, implement as a d2 estimator, calculating the gradient and Hessian analytically)

(B) Implement conditional logit maximum likelihood (MLE) estimator using Stata’s ML language (NOTE: This is <u>HARD</u>; see hints on-line)

conditional logit

log-likelihood for conditional logit:

$$\log(L_i) = \sum_{t=1}^{T} y_{it} x_{it} \beta - \log \left(\sum_{d_i \in S_i} \exp \left(\sum_{t=1}^{T} d_{it} x_{it} \beta \right) \right)$$

where d_{it} is equal to 0 or 1 such that sum of all d_{it}in the panel equal the sum of all y_{it} in the panel, and S_i is defined to be all combinations of sequences of d_{it} such that sum of all d_{it}in the panel equal the sum of all y_{it} in the panel