14.170: Programming for Economists

1/12/2009-1/16/2009

Melissa Dell
Matt Notowidigdo
Paul Schrimpf
Lecture 5, Large Data Sets in Stata + Numerical Precision
Overview

• This lecture is part wrap-up lecture, part “tips and tricks”
• Focus is on dealing with large data sets and on numerical precision
• Numerical precision
  – Introduction to binary representation
  – Equilibrating matrices
• Large data sets
  – How Stata represents data in memory
  – Speeding up code
  – Tips and tricks for large data sets
Numerical precision

• What the @&*%&&$!^ is going on here?

```lua
local a = 0.7 + 0.1
local b = 0.8
display (`a' == `b')

local a = 0.75 + 0.05
local b = 0.8
display (`a' == `b')
```

```lua
local a = 0.7 + 0.1
local b = 0.8
display (`a' == `b')

local a = 0.75 + 0.05
local b = 0.8
display (`a' == `b')
```
Binary numbers

• Computers store numbers in base 2 ("bits")

\[ 14_{10} = 1110_2 \]
\[ (14 = 2 + 4 + 8) \]

\[ 170_{10} = 10101010_2 \]
\[ (170 = 2 + 8 + 32 + 128) \]

How are decimals stored?
Binary numbers, con’t

\[ 0.875_{10} = 0.111_2 \]
\[ (0.875 = 0.5 + 0.25 + 0.125) \]

\[ 0.80_{10} = 0.1100110011000_2 \]

\[ 0.70_{10} = 0.101100110011_2 \]
\[ 0.10_{10} = 0.000110011001_2 \]
\[ 0.75_{10} = 0.11_2 \]
\[ 0.05_{10} = 0.000011001100_2 \]

QUESTION: Is there a repeating decimal in base 10 that is not repeating in base 2?
Precision issues in Mata

mata
A = (1e10, 2e10 \ 2e-10, 3e-10)
A
rank(A)
luinv(A)
A_inv = (-3e-10, 2e10 \ 2e-10, -1e10)
I = A * A_inv
I
end
Precision issues in Mata

: A
  1 2
  +-------------------------------+
  1 | 1.000000000e+10  2.000000000e+10 |
  2 | 2.000000000e-10  3.000000000e-10 |
  +-------------------------------+

: rank(A)
  1

: luinv(A)
  [symmetric]
  1 2
  +-------+
  1 | 1  + |
  2 | 0  1 |
  +-------+

: A_inv = (-3e-10, 2e10 \ 2e-10, -1e10)
 : I = A * A_inv

: I
  [symmetric]
  1 2
  +-------+
  1 | 1  |
  2 | 0  1 |
  +-------+
Mata

r = c = 0
A = (1e10, 2e10 \ 2e-10, 3e-10)
A
rank(A)
luinv(A, 1e-15)
_epochrc(A, r, c)
A
r
c
rank(A)
luinv(A)
luinv(A)
c':*luinv(A):*r'
end
: luinv(A, 1e-15)

: _equilrc(A, r, c)

: A
  [symmetric]
  1  2
  +-----------+
  1 | .75  |
  2 |  1  1 |
  +-----------+

: r
  1
  +-----------+
  1 | 5.000000e-11 |
  2 | 33333333333 |
  +-----------+

: c
  1  2
  +-----------+
  1 | 1.5  1 |
  +-----------+

: rank(A)
  2

: c'*luinv(A):*r'

: c' * luinv(A) * r'

: _equilrc(A, c, r)
Large data sets in Stata

- Computer architecture overview
  - CPU: executes instructions
  - RAM (also called the “memory”): stores frequently-accessed data
  - DISK (“hard drive”): stores not-as-frequently used data

- RAM is accessed electronically; DISK is accessed mechanically (that’s why you can HEAR it). Thus DISK is several orders of magnitude slower than RAM.

- In Stata, if you ever have to access the disk, you’re pretty much dead. Stata was not written to deal with data sets that are larger than the available RAM. It expects the data set to fit in memory.

- So when you type “set memory XXXm”, make sure that you are not setting the value to be larger than the available RAM (some operating systems won’t even let you, anyway).

- For >20-30 GB of data, Stata is not recommended. Consider Matlab or SAS.
Large data sets in Stata, con’t

• Don’t keep re-creating the same variables over and over again

• “preserve” can really help or really hurt. Know when to use it and when to avoid it

• Don’t estimate parameters you don’t care about

• Lots of “if” and “in” commands could slow things down

• Create “1% sample” to develop and test code (to prevent unanticipated crashes after code has been running for hours)
clear
set seed 12345
set mem 2000m
set matsize 2000
set more off
set obs 5000
gen myn = _n
gen id = 1 + floor((_n - 1)/100)
sort id myn
by id: gen t = 1 + floor((n - 1) / 5)
gen x = invnormal(uniform())
gen fe = invnormal(uniform())
sort id t myn
by id t: replace fe = fe[1]
gen y = 2 + x + fe + 100 * invnormal(uniform())

reg y x
xi i.id*i.t
reg y x _I*

summ t
gen idXt = id * (r(max) + 1) + t
areg y x, absorb(idXt)
Two-way fixed effects

```
. reg y x_Ii

        Source | SS of MS Number of obs = 5000
-------------+-----------------------------------------------
Model | 9217133.0 1000 Prob > F = 0.088
       | 12171350 1000 Rsquared = 0.0163
Residual | 41898622.7 3999 10477.207 Adj Rsquared = -0.0247
          | 41898622.7 3999 Root MSE = 102.36
Total | 51115761.5 4999 10226.197
-------------+-----------------------------------------------

        y | Coef. Std. Err. t P>|t| [95% Conf. Interval]
-------------+-----------------------------------------------
    x |  19006421.1 1.605583  0.60 0.545 -2.197203 4.15729
   _Iid_2 |  15.14362  6.74499  0.23 0.815 -111.7326 142.0799
   _Iid_3 |  49.41053  6.73932  0.76 0.445 -77.51401 176.3339
   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
   _Iid_49 |  44.54136  6.73935  0.68 0.491 -82.38307 171.4659
   _Iid_50 |  25.30376  6.73949  0.39 0.696 -101.6081 152.2439
   _It_2 | -69.70522  6.74047 -10.08 0.282 -196.5326 57.2219
   _It_3 | -29.26425  6.73939 -4.45 0.651 -156.1727 77.6713
   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
   _It_19 | -2.657793  6.74614 -0.02 0.985 -128.1963 125.6807
   _It_20 |  23.69263  6.74172  0.44 0.668  98.23723 156.6225
   _Iidkt_2_2 |  93.22159  91.55252  0.91 0.363  196.29368 262.7077
   _Iidkt_2_3 |  3565748  91.55491  0.00 0.997 -179.1632 179.1632
   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
   _Iidkt_50_19 | -33.42095  91.55229 -0.37 0.715 -212.9165 145.0706
   _Iidkt_50_20 | -50.37366  91.55244 -0.55 0.582 -225.0813 124.1522
   _cons | -5.802056  45.77297 -0.12 0.903 -95.33416 84.12925
-------------+-----------------------------------------------
. areg y x, absorb(Iidkt)

Linear regression, absorbing indicators
Number of obs = 5000
F( 12, 4999) = 0.37
Prob > F = 0.5454
R-squared = 0.0163
Adj Rsquared = -0.0247
Root MSE = 102.36

        y | Coef. Std. Err. t P>|t| [95% Conf. Interval]
-------------+-----------------------------------------------
    x |  19006421.1 1.605583  0.60 0.545 -2.197203 4.15729
   _cons |  2.549234  1.447397  1.76 0.078 -0.2889639 5.387331
-------------+-----------------------------------------------
   Iidkt |  F(999, 3999) = 0.880 0.994 (1000 categories)
```
clear
set seed 12345
set mem 100m
set more off
set obs 500000
gen myn = _n
gen id = 1 + floor((_n - 1)/200)
sort id myn
by id: gen t = _n
gen x = invnormal(uniform())
gen id_fe = invnormal(uniform())
gen t_fe = invnormal(uniform())
by id: replace id_fe = id_fe[1]
sort t id
by t: replace t_fe = t_fe[1]
gen y = 2 + x + id_fe + t_fe + 100 * invnormal(uniform())

xi i.t
xtreg y x _It*, i(id) fe
Fixed Effects with large data sets

```
.xi i.t
i.t _It_1-200 (naturally coded: _It_1 omitted)

.xtreg y x _It*, i(id) fe

Fixed-effects (within) regression
Group variable: id

Number of obs       =      500000
Number of groups    =       2500

R-sq: within         = 0.0008
between             = 0.0009
overall             = 0.0008

Obs per group:
min = 200
avg = 200.0
max = 200

corr(u_i, Xb) = 0.0006

F(200,497300)       = 1.87
Prob > F             = 0.0000

------------------------------------------------------------------------------
       y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+---------------------------------------------------------------
          x |   1.224538   .1416681    8.64  0.000     .9468925    1.502184
_It_2 |   .5291215   2.827906    0.19  0.842    -5.013486    6.071729
_It_3 |   .7475153   2.827904    0.26  0.792    -4.795089    6.29012
_It_4 |   2.120499   2.827907    0.75  0.453    -3.422111    7.663107
_It_5 |   .1249969   2.827904    0.04  0.965    -5.417607    5.667601
_It_6 |  -.5349088   2.827912   -0.19  0.845    -6.077528    5.00771
_It_7 |  -.903490   2.827906   -0.32  0.749    -6.446097    4.639117
_It_8 |  -.0648770   2.977904   -0.24  0.814     .9729989   -6.935989
```
clear
set seed 12345
set mem 100m
set more off
set obs 500000
gen myn = _n
gen id = 1 + floor((_n - 1)/200)
sort id myn
by id: gen t = _n

gen x = invnormal(uniform())
gen id_fe = invnormal(uniform())
gen t_fe = invnormal(uniform())
by id: replace id_fe = id_fe[1]
sort t id
by t: replace t_fe = t_fe[1]
gen y = 2 + x + id_fe + t_fe + 100 * invnormal(uniform())

xtreg y, i(id) fe
predict y_resid, e
xtreg x, i(id) fe
predict x_resid, e
xtreg y_resid x_resid, i(t) fe

~53 seconds
Fixed Effects with large data sets

```
xreg y_resid x_resid \ i(t) \ fe
```

```
Fixed-effects (within) regression
Number of obs = 500000
Group variable: t  Number of groups = 200

R-sq: within = 0.0002
between = 0.0030
overall = 0.0002

Obs per group: min = 2500
avg = 2500.0
max = 2500

F(1,499799) = 75.10
Prob > F = 0.0000

corr(u_i, Xb) = 0.0008
```

|       | Coef. | Std. Err. | t     | P>|t|  | [95% Conf. Interval] |
|-------|-------|-----------|-------|------|---------------------|
| y_resid | x_resid | 1.224538  | 0.141305 | 8.67 | 0.000 | .9475875 - 1.501489 |
|       | _cons  | -2.89e-15 | 0.1410413 | -0.00 | 1.000 | -.2764365 - .2764365 |
| sigma_u | 2.4508088 |          |         |      |        |                      |
| sigma_e | 99.731239 |          |         |      |        |                      |
| rho    | .00060352 |          |         |      |        | (fraction of variance due to u_i) |

F test that all u_i=0:  F(199,499799) = 1.51  Prob > F = 0.0000
Other tips and tricks when you have large number of fixed effects in large data sets

- Use matrix algebra
- Newton steps in parallel
- “zig-zag maximization”
  (Heckman-McCurdy)
clear mata
mata
rseed(14170)
N = 3000

rA = rnormal(5, 5, 0, 1)
rB = rnormal(5, N, 0, 1)
rC = rnormal(N, 5, 0, 1)
d = rnormal(1, N, 0, 1)
V = (rA, rB \ rC, diag(d))

V_inv = luinv(V)
V_inv[1..5,1..5]

~162 seconds
clear mata
mata
rseed(14170)
N = 3000
rA = rnormal(5, 5, 0, 1)
rB = rnormal(5, N, 0, 1)
rC = rnormal(N, 5, 0, 1)
d = rnormal(1, N, 0, 1)
V = (rA, rB \ rC, diag(d))
V_fast = luinv(rA - cross(rB', d ^ -1, rC))
V_fast

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}^{-1} = 
\begin{bmatrix}
(A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\
-D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1}
\end{bmatrix}
\]
Fixed Effects probit

- Finkelstein, Luttmer, Notowidigdo (2008) run Fixed Effects probit as a robustness check
  - What about the incidental parameters problem? (see Hahn and Newey, EMA, 2004)
- But what to do with >11,000 fixed effects!
  - Cannot de-mean within panel as you could with linear probability model
  - Stata/SE and Stata/MP matrix size limit is 11,000
  - Need several computation tricks
Fixed Effects probit

clear
set seed 12345
set matsize 2000
set obs 2000

gen id = 1+floor((-_n - 1)/4)
gen a = invnormal(uniform())
gen fe_raw = 0.5*invnorm(uniform()) + 2*a
bys id: egen fe = mean(fe_raw)
gen x = invnormal(uniform())
gen e = invnormal(uniform())
gen y = (1*x + fe > invnormal(uniform()) + a)

bys id: egen x_mean = mean(x)
gen x_demean = x - x_mean
probit y x
probit y x_demean
sort id y
by id: keep if y[1] != y[_N]
probit y x
xi i.id
probit y x _I*
Fixed Effects probit

```
. probit y x
Iteration 0:  log likelihood =  -1186.2304
Iteration 1:  log likelihood =  -1179.0473
Iteration 2:  log likelihood =  -1169.7587
Iteration 3:  log likelihood =  -1169.7486

Probit regression                     Number of obs   =      2000
                                     LR chi2(1)     =     432.96
                                     Prob > chi2    =     0.0000
                                     Pseudo R2      =     0.1562
Log likelihood =  -1169.7486

------------------------------------------------------------------------------
         |      Coef.     Std. Err.       z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         x  | .6681063     .0350362     19.06   0.000     .5993974    .7368153
_cons    | .01185      .0300633      0.40    0.693    -.0470722    .0700522
------------------------------------------------------------------------------

. probit y x_idmean
Iteration 0:  log likelihood =  -1206.8304
Iteration 1:  log likelihood =  -1222.2689
Iteration 2:  log likelihood =  -1219.6961
Iteration 3:  log likelihood =  -1219.6943

Probit regression                     Number of obs   =      2000
                                     LR chi2(1)     =     332.07
                                     Prob > chi2    =     0.0000
                                     Pseudo R2      =     0.1261
Log likelihood =  -1219.6943

------------------------------------------------------------------------------
         |      Coef.     Std. Err.       z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         x_idmean  | .6597473     .0385849     17.10   0.000     .5841229    .7353729
         _cons     | .0136105     .0295421      0.45    0.645    -.0442938    .0715249
------------------------------------------------------------------------------

. probit y x*_id
Iteration 0:  log likelihood =  -1191.104
Iteration 1:  log likelihood =  -1188.7582
Iteration 2:  log likelihood =  -1185.3479
Iteration 3:  log likelihood =  -1183.2619
Iteration 4:  log likelihood =  -1183.2540
Iteration 5:  log likelihood =  -1183.2540

Probit regression                     Number of obs   =      1704
                                     LR chi2(426)   =     756.70
                                     Prob > chi2    =     0.0000
                                     Pseudo R2      =     0.3139
Log likelihood =  -1183.2540

------------------------------------------------------------------------------
         |      Coef.     Std. Err.       z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         x  |  1.024471     .2964683     3.43    0.000     .4492363    1.599706
     _id_2  |  1.5309761    1.0600311     1.45    0.149     .4630997    2.608848
     _id_3  |  1.2933327    1.0158065     1.28    0.200    -.7399239    3.326584
     _id_4  |  -1.5044056   1.1895403    -1.27    0.200    -.7926388    .8137775
     _id_500  |  -5.475299    1.06699      -5.12    0.000     -8.02488    -3.056723
     _id_501  |  -1.1187791   1.1061461    -1.01    0.315    -.3286554    1.191104
     _cons   |  -1.931788    .7158942    -2.71    0.007    -.3985269    1.21373
------------------------------------------------------------------------------
```

Fixed Effects probit (slow)
clear
set more off
set mem 1000m
set seed 12345
set matsize 3000
set obs 12000
gen id = 1+floor(_n - 1)/4
gen a = invnormal(uniform())
gen fe_raw = 0.5*invnorm(uniform()) + 2*a
bys id: egen fe = mean(fe_raw)
gen x = invnormal(uniform())
gen e = invnormal(uniform())
gen y = (1*x + fe > invnormal(uniform()) + a)

sort id y
by id: keep if y[1] != y[_N]

xi i.id
probit y x _I*
Fixed Effects probit (slow)

\~40 minutes

```
x i.id
  i.id  _Iid_1-3000  (naturally coded: _Iid_1 omitted)
  probit y x _I*
```

Iteration 0:  log likelihood =  -7131.0824
Iteration 1:  log likelihood =  -5185.5403
Iteration 2:  log likelihood =  -5019.9566
Iteration 3:  log likelihood =  -5011.8605
Iteration 4:  log likelihood =  -5011.8245
Iteration 5:  log likelihood =  -5011.8245

Probit regression
Number of obs =  10288
LR chi2(2572) =  4238.52
Prob > chi2 =  0.0000
Pseudo R2 =  0.2972

Log likelihood =  -5011.8245

|          | Coef.  | Std. Err. |      z  |   P>|z|  | [95% Conf. Interval] |
|----------|--------|-----------|--------|------|----------------------|
|          |        |           |        |      |                      |
| x        |  0.9416602 |  0.0213752 |  44.05 |  0.000 |  0.8997655 - 0.9835548 |
| _Iid_2   |  0.3597163 |  0.9682755 |   0.37 |  0.710 | -1.538069 - 2.257501 |
| _Iid_4   | -0.0921325 |  0.9684319 | -0.10 |  0.924 | -1.990224 - 1.805959 |
|          | ...     |  ...      | ...    | ...  | ...                  |
| _Iid_2998|  1.1280680 |  0.9542554 |   1.18 |  0.237 | -0.742238 - 2.998374 |
| _Iid_2999|  0.5476889 |  0.9590439 |   0.57 |  0.568 | -1.332003 - 2.42738  |
| _Iid_3000|  0.4456436 |  0.9182577 |   0.49 |  0.627 | -1.354107 - 2.245394 |
| _cons    | -0.6380591 |  0.6898159 | -0.92 |  0.355 | -1.990073 - 0.7139553 |
clear
set mem 1000m
set seed 12345
set matsize 3000
set obs 12000

gen id = 1+floor((_n - 1)/4)
gen a = invnormal(uniform())
gen fe_raw = 0.5*invnorm(uniform()) + 2*a
bys id: egen fe = mean(fe_raw)
gen x = invnormal(uniform())
gen e = invnormal(uniform())
gen y = (1*x + fe > invnormal(uniform()) + a)
sort id y
by id: keep if y[1] != y[_N]

egen id_new = group(id)
summ id_new
local max = r(max)
gen fe_hat = 0
forvalues iter = 1/20 {
    probit y x, nocons offset(fe_hat)
    capture drop xb*
predict xb, xb nooffset
    forvalues i = 1/`max' {
        qui probit y if id_new == `i', offset(xb)
        qui replace fe_hat = _b[_cons] if id_new == `i'
    }
}
probit y x, noconstant offset(fe_hat)
Fixed Effects probit (faster)  
\sim8 \text{ minutes} 

\texttt{. probit y x, nocons offset\{fe_hat\}}

\begin{verbatim}
Iteration 0:  log likelihood = -6936.1812
Iteration 1:  log likelihood = -5089.0776
Iteration 2:  log likelihood = -5012.1796
Iteration 3:  log likelihood = -5011.8245
Probit regression                         Number of obs  =     10288
Log likelihood = -5011.8245              Wald chi2(1)   =   3073.53
Prob > chi2    =     0.0000

+----------------------------------------+-----------------------------+--------------------------+
|                            | Coef.  | Std. Err. | z     | P>|z| | [95\% Conf. Interval] |
+----------------------------------------+-----------------------------+--------------------------+
| \textit{y}                           | \textit{.9416602}  \textit{.0169854} | 55.44 | 0.000 | \textit{.9083694 -- .9749509} |
| \textit{x}                           | \textit{.9416602}  \textit{.0169854} | 55.44 | 0.000 | \textit{.9083694 -- .9749509} |
| \textit{fe_hat} (offset)             | \textit{.9416602}  \textit{.0169854} | 55.44 | 0.000 | \textit{.9083694 -- .9749509} |
+----------------------------------------+-----------------------------+--------------------------+
\end{verbatim}

\textbf{QUESTION:} Why are standard errors not the same?
Exercises

(A) Speed up fixed effects probit even more by updating fixed effects in parallel
(B) Fix standard errors in FE probit example