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?

```
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')
```

```
. local a = 0.7 + 0.1

. local b = 0.8

. display (`a' == `b')
0

.
. local a = 0.75 + 0.05

. local b = 0.8

. display (`a' == `b')
1
```

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

$0.875_{10} = 0.111_2$

$(0.875 = 0.5 + 0.25 + 0.125)$

$0.80_{10} = 0.11001100\overline{1100}_2$

$0.70_{10} = 0.10110011\overline{0011}_2$

$0.10_{10} = 0.00011001\overline{1001}_2$

$0.75_{10} = 0.11_2$

$0.05_{10} = 0.00001100\overline{1100}_2$

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 |  1.00000e+10   2.00000e+10  |
  2 |  2.00000e-10   3.00000e-10  |
    +-----------------------------+

: rank(A)
  1

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

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

: I = A * A_inv

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

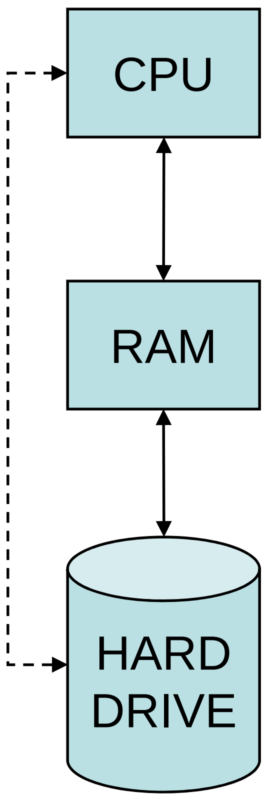
Precision issues in Mata

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

Precision issues in Mata

```
: luinv(A, 1e-15)
                  1               2
    +-------------------------------+
  1 |  -3.00000e-10    2.00000e+10  |
  2 |   2.00000e-10   -1.00000e+10  |
    +-------------------------------+

: _equilrc(A, r, c)

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

: r
                 1
    +---------------+
  1 |  5.00000e-11  |
  2 |   3333333333  |
    +---------------+

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

: rank(A)
  2

: c':*luinv(A):*r'
                  1               2
    +-------------------------------+
  1 |  -3.00000e-10    2.00000e+10  |
  2 |   2.00000e-10   -1.00000e+10  |
    +-------------------------------+
```

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)

Two-way fixed effects

```
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

```
. xi i.id*i.t
i.id              _Iid_1-50           (naturally coded; _Iid_1 omitted)
i.t               _It_1-20            (naturally coded; _It_1 omitted)
i.id*i.t          _IidXt_#_#          (coded as above)

. reg y x _I*

      Source |       SS       df       MS              Number of obs =    5000
-------------+------------------------------           F(1000,  3999) =    0.88
       Model |  9217138.8   1000  9217.1388            Prob > F      =  0.9941
    Residual |  41898622.7  3999  10477.275            R-squared     =  0.1803
-------------+------------------------------           Adj R-squared = -0.0247
       Total |  51115761.5  4999  10225.1973           Root MSE      =  102.36

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .9800434   1.620583     0.60   0.545    -2.197203     4.15729
       _Iid_2 |   15.14362   64.74499     0.23   0.815    -111.7926    142.0799
       _Iid_3 |   49.41093   64.73921     0.76   0.445    -77.51401    176.3359

   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

      _Iid_49 |   44.54136   64.73895     0.69   0.491    -82.38307    171.4658
      _Iid_50 |   25.32036   64.73848     0.39   0.696    -101.6031    152.2439
        _It_2 |  -69.70522   64.74047    -1.08   0.282    -196.6326    57.22219
        _It_3 |  -29.24825   64.73896    -0.45   0.651    -156.1727    97.67618

   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

       _It_19 |  -1.257793   64.74614    -0.02   0.985    -128.1963    125.6807
       _It_20 |   28.69263   64.74172     0.44   0.658    -98.23723    155.6225
   _IidXt_2_2 |   83.21349   91.55262     0.91   0.363    -96.28068    262.7077
   _IidXt_2_3 |   .3550745   91.56491     0.00   0.997    -179.1632    179.8733

   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

█IidXt_50_19 |  -33.42295   91.55229    -0.37   0.715    -212.9165    146.0706
_IidXt_50_20 |  -50.37966   91.55644    -0.55   0.582    -229.8813     129.122
       _cons |  -5.582456   45.77867    -0.12   0.903    -95.33416    84.16925
------------------------------------------------------------------------------

. areg y x, absorb(idXt)

Linear regression, absorbing indicators                Number of obs =    5000
                                                       F(  1,  3999) =    0.37
                                                       Prob > F      =  0.5454
                                                       R-squared     =  0.1803
                                                       Adj R-squared = -0.0247
                                                       Root MSE      =  102.36

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .9800434   1.620583     0.60   0.545    -2.197203     4.15729
       _cons |   2.549234   1.447597     1.76   0.078    -.2888639    5.387331
-------------+----------------------------------------------------------------
        idXt |      F(999, 3999) =      0.880   0.994          (1000 categories)
```

Fixed Effects with large data sets

```
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
```

~674 seconds

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               Number of obs      =    500000
Group variable: id                              Number of groups   =      2500

R-sq:  within  = 0.0008                         Obs per group: min =       200
       between = 0.0009                                        avg =     200.0
       overall = 0.0008                                        max =       200

                                                F(200,497300)      =      1.87
corr(u_i, Xb)  = 0.0006                         Prob > F           =    0.0000

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.224538   .1416581     8.64   0.000     .9468925    1.502184
       _It_2 |   .5291215   2.827906     0.19   0.852    -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     -3.42211    7.663107
       _It_5 |   .1249969   2.827904     0.04   0.965    -5.417607    5.667601
       _It_6 |  -.5349088   2.827912    -0.19   0.850    -6.077528     5.00771
       _It_7 |    -.90349   2.827906    -0.32   0.749    -6.446097    4.639117
       _It_8 |  -.9549779   2.827904    -0.34   0.736    -6.497542    4.587666
```

Fixed Effects with large data sets

```
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

```
. xtreg y_resid x_resid, i(t) fe
warning: existing panel variable is not t

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

R-sq:  within  = 0.0002                         Obs per group: min =      2500
       between = 0.0030                                        avg =    2500.0
       overall = 0.0002                                        max =      2500

                                                F(1,499799)        =     75.10
corr(u_i, Xb)  = 0.0008                         Prob > F           =    0.0000

------------------------------------------------------------------------------
     y_resid |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     x_resid |   1.224538   .1413035     8.67   0.000     .9475875    1.501489
       _cons |  -2.89e-15   .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)

Matrix algebra

```
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

Matrix algebra

```
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
```

<1 second

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}$$

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 = -1386.2304
Iteration 1:   log likelihood = -1175.0473
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
Log likelihood = -1169.7486                       Pseudo R2       =     0.1562

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .6681063   .0350562    19.06   0.000     .5993974    .7368153
       _cons |     .01189   .0300833     0.40   0.693    -.0470722    .0708522
------------------------------------------------------------------------------

. probit y x_demean

Iteration 0:   log likelihood = -1386.2304
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)      =     333.07
                                                  Prob > chi2     =     0.0000
Log likelihood = -1219.6943                       Pseudo R2       =     0.1201

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    x_demean |   .6597479   .0385849    17.10   0.000     .5841229    .7353729
       _cons |   .0136155   .0295461     0.46   0.645    -.0442938    .0715249
------------------------------------------------------------------------------
```

```
. probit y x

Iteration 0:   log likelihood =  -1181.104
Iteration 1:   log likelihood = -996.25035
Iteration 2:   log likelihood = -991.28997
Iteration 3:   log likelihood = -991.27891

Probit regression                                 Number of obs   =       1704
                                                  LR chi2(1)      =     379.65
                                                  Prob > chi2     =     0.0000
Log likelihood = -991.27891                       Pseudo R2       =     0.1607

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .6765638   .0380693    17.77   0.000     .6019494    .7511782
       _cons |   .0089522   .0326835     0.27   0.784    -.0551062    .0730106
------------------------------------------------------------------------------

. xi i.id
i.id              _Iid_1-500          (naturally coded; _Iid_1 omitted)

. probit y x _I*

Iteration 0:   log likelihood =  -1181.104
Iteration 1:   log likelihood = -838.75262
Iteration 2:   log likelihood = -805.34791
Iteration 3:   log likelihood =   -803.269
Iteration 4:   log likelihood = -803.25405
Iteration 5:   log likelihood = -803.25404

Probit regression                                 Number of obs   =       1704
                                                  LR chi2(426)    =     755.70
                                                  Prob > chi2     =     0.0000
Log likelihood = -803.25404                       Pseudo R2       =     0.3199

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.024471   .0564683    18.14   0.000     .9137949    1.135146
      _Iid_2 |  -.5300751   1.028931    -0.52   0.606    -2.546743    1.486593
      _Iid_3 |   1.293332   1.018006     1.27   0.204    -.7019239    3.288587
      _Iid_4 |  -1.504805   1.184503    -1.27   0.204    -3.826388    .8167775

    ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

    _Iid_499 |  -.5475299     1.0869    -0.50   0.614    -2.677815    1.582755
    _Iid_500 |  -.1185591    1.10614    -0.11   0.915    -2.286554    2.049435
       _cons |  -.1931788   .7153942    -0.27   0.787    -1.595326    1.208968
------------------------------------------------------------------------------
```

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

```
.
. xi 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
Log likelihood = -5011.8245                       Pseudo R2       =     0.2972

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   .9416602   .0213752    44.05   0.000     .8997655    .9835548
      _Iid_2 |   .3597163   .9682755     0.37   0.710    -1.538069    2.257501
      _Iid_4 |  -.0921325   .9684319    -0.10   0.924    -1.990224    1.805959

   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...

   _Iid_2998 |   1.128068   .9542554     1.18   0.237     -.742238    2.998374
   _Iid_2999 |   .5476889   .9590439     0.57   0.568    -1.332003     2.42738
   _Iid_3000 |   .4456436    .918257     0.49   0.627    -1.354107    2.245394
       _cons |  -.6380591   .6898159    -0.92   0.355    -1.990073    .7139553
------------------------------------------------------------------------------
```

Fixed Effects probit (faster)

```
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)

~8 minutes

```
. probit y x, nocons offset(fe_hat)

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
                                                  Wald chi2(1)    =    3073.53
Log likelihood = -5011.8245                       Prob > chi2     =     0.0000

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

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