14.170: Programming for Economists

1/12/2009-1/16/2009

Melissa Dell
Matt Notowidigdo
Paul Schrimpf
Lecture 2, Intermediate Stata
Warm-up and review

• Before going to new material, let’s talk about the exercises …

  – exercise1a.do (privatization DD)
    • TMTOWTDI!
    • You had to get all of the different exp* variables into one variable. You had many different solutions, many of them very creative.
      – Replace missing values with 0, and then you added up all the exp* variables
      – You used the rsum() command in egen (this treats missing values as zeroes which is strange, but it works)
      – You “hard-coded” everything
Intermediate Stata overview slide

• Quick tour of other built-in commands: non-parametric estimation, quantile regression, etc.
  – If you’re not sure it’s in there, ask someone. And then consult reference manuals. And (maybe) e-mail around. Don’t re-invent the wheel! If it’s not too hard to do, it’s likely that someone has already done it.
  – Examples:
    • Proportional hazard models (streg, stcox)
    • Generalized linear models (glm)
    • Kernel density (kdensity)
    • Conditional fixed-effects poisson (xtpoisson)
    • Arellano-Bond dynamic panel estimation (xtabond)

• But sometimes newer commands don’t (yet) have exactly what you want, and you will need to implement it yourself
  – e.g. xtpoisson doesn’t have clustered standard errors

• Monte carlo simulations in Stata
  – You should be able to do this based on what you learned last lecture (you know how to set variables and use control structures). Just need some matrix syntax.

• More with Stata matrix syntax
  – Precursor to Mata, Stata matrix language has many useful built-in matrix algebra functions
“Intermediate” Stata commands

• Hazard models (streg, stcox)
• Generalized linear models (glm)
• Non-parametric estimation (kdensity)
• Quantile regression (qreg)
• Conditional fixed-effects poisson (xtpoisson)
• Arellano-Bond dynamic panel estimation (xtabond)

I have found these commands easy to use, but the econometrics behind them is not always simple. Make sure to understand what you are doing when you are running them. It’s easy to get results, but with many of these commands, the results are sometimes hard to interpret.

But first, a quick review and an easy warm-up …
Quick review, FE and IV

clear
set obs 10000
gen id = floor( (_n - 1) / 2000)
bys id: gen fe = invnorm(uniform()) if _n == 1
by id: replace fe = fe[1]

.gen spunk = invnorm(uniform())
gen z = invnorm(uniform())
gen schooling = invnorm(uniform()) + z + spunk + fe
.gen ability = invnorm(uniform()) + spunk
gen e = invnorm(uniform())
gen y = schooling + ability + e + 5*fe

reg y schooling

xtreg y schooling , i(id) fe
xi: reg y schooling i.id
xi i.id
reg y schooling _I*
areg y schooling, absorb(id)

ivreg y (schooling = z) _I*
xтivreg y (schooling = z), i(id)
xтивreg y (schooling = z), i(id) fe
Data check

<table>
<thead>
<tr>
<th>id</th>
<th>Freq.</th>
<th>Percent</th>
<th>Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2,000</td>
<td>20.00</td>
<td>20.00</td>
</tr>
<tr>
<td>1</td>
<td>2,000</td>
<td>20.00</td>
<td>40.00</td>
</tr>
<tr>
<td>2</td>
<td>2,000</td>
<td>20.00</td>
<td>60.00</td>
</tr>
<tr>
<td>3</td>
<td>2,000</td>
<td>20.00</td>
<td>80.00</td>
</tr>
<tr>
<td>4</td>
<td>2,000</td>
<td>20.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Total</td>
<td>10,000</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>

list in 1/20

<table>
<thead>
<tr>
<th>id</th>
<th>fe</th>
<th>spunk</th>
<th>z</th>
<th>schooling</th>
<th>ability</th>
<th>e</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>0.5801376</td>
<td>-2.406844</td>
<td>-1.099788</td>
<td>-1.469551</td>
<td>-1.660281</td>
<td>-3.460338</td>
<td>-5.751783</td>
</tr>
<tr>
<td>2.</td>
<td>0.5801376</td>
<td>1.726103</td>
<td>0.050004</td>
<td>1.268643</td>
<td>-0.5901318</td>
<td>0.1685174</td>
<td>3.747717</td>
</tr>
<tr>
<td>3.</td>
<td>0.5801376</td>
<td>-2.909676</td>
<td>0.5172438</td>
<td>1.350124</td>
<td>0.4649253</td>
<td>0.3624368</td>
<td>0.5078174</td>
</tr>
<tr>
<td>4.</td>
<td>0.5801376</td>
<td>-1.353611</td>
<td>1.395514</td>
<td>-1.227456</td>
<td>-0.9913431</td>
<td>-2.209503</td>
<td>-1.527614</td>
</tr>
<tr>
<td>5.</td>
<td>0.5801376</td>
<td>-0.3666069</td>
<td>1.210201</td>
<td>1.267264</td>
<td>-0.4916896</td>
<td>0.9330731</td>
<td>4.609318</td>
</tr>
<tr>
<td>6.</td>
<td>0.5801376</td>
<td>1.412722</td>
<td>-0.9439728</td>
<td>1.550789</td>
<td>0.8241584</td>
<td>0.6179897</td>
<td>4.657646</td>
</tr>
<tr>
<td>7.</td>
<td>0.5801376</td>
<td>0.6583463</td>
<td>-2.457964</td>
<td>0.5150581</td>
<td>1.745385</td>
<td>-1.313493</td>
<td>3.848238</td>
</tr>
<tr>
<td>8.</td>
<td>0.5801376</td>
<td>-0.4458856</td>
<td>-0.946522</td>
<td>-0.027943</td>
<td>0.3363754</td>
<td>-0.9354327</td>
<td>2.598836</td>
</tr>
<tr>
<td>9.</td>
<td>0.5801376</td>
<td>1.510083</td>
<td>1.064582</td>
<td>4.218287</td>
<td>1.393152</td>
<td>-0.2734299</td>
<td>8.228698</td>
</tr>
<tr>
<td>10.</td>
<td>0.5801376</td>
<td>-0.433168</td>
<td>0.8806828</td>
<td>-0.5063311</td>
<td>1.581337</td>
<td>1.035773</td>
<td>5.012467</td>
</tr>
<tr>
<td>11.</td>
<td>0.5801376</td>
<td>-1.169954</td>
<td>-0.686674</td>
<td>1.268839</td>
<td>2.474484</td>
<td>-0.0743157</td>
<td>6.569959</td>
</tr>
<tr>
<td>12.</td>
<td>0.5801376</td>
<td>0.2398426</td>
<td>0.2654099</td>
<td>1.94654</td>
<td>0.961118</td>
<td>0.4235876</td>
<td>5.384752</td>
</tr>
<tr>
<td>13.</td>
<td>0.5801376</td>
<td>0.5992975</td>
<td>-0.7429279</td>
<td>0.407327</td>
<td>1.009442</td>
<td>-1.1929502</td>
<td>3.309983</td>
</tr>
<tr>
<td>14.</td>
<td>0.5801376</td>
<td>-1.192293</td>
<td>0.5730755</td>
<td>-0.2901712</td>
<td>-1.393483</td>
<td>-2.0018129</td>
<td>0.2638426</td>
</tr>
<tr>
<td>15.</td>
<td>0.5801376</td>
<td>-0.7581176</td>
<td>0.4379372</td>
<td>1.211469</td>
<td>-1.1717598</td>
<td>0.3249905</td>
<td>3.615406</td>
</tr>
<tr>
<td>16.</td>
<td>0.5801376</td>
<td>-1.283779</td>
<td>0.3321107</td>
<td>-0.2236305</td>
<td>-1.573899</td>
<td>-1.32857</td>
<td>-0.2254121</td>
</tr>
<tr>
<td>17.</td>
<td>0.5801376</td>
<td>-0.506322</td>
<td>-0.1371481</td>
<td>0.590051</td>
<td>0.1497437</td>
<td>0.4564812</td>
<td>5.108964</td>
</tr>
<tr>
<td>18.</td>
<td>0.5801376</td>
<td>-2.214334</td>
<td>2.878901</td>
<td>3.168465</td>
<td>-2.988493</td>
<td>0.0033352</td>
<td>3.082548</td>
</tr>
<tr>
<td>19.</td>
<td>0.5801376</td>
<td>-1.386452</td>
<td>-0.6438373</td>
<td>-3.562516</td>
<td>-2.277211</td>
<td>-0.79568</td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td>0.5801376</td>
<td>-0.5060894</td>
<td>-1.114979</td>
<td>-1.021156</td>
<td>-0.5424608</td>
<td>-1.334018</td>
<td>0.795975</td>
</tr>
</tbody>
</table>
Results

```
: reg y schooling

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>157778.58</td>
<td>1</td>
<td>157778.58</td>
<td>F( 1, 9998) = 12539.25</td>
</tr>
<tr>
<td>Residual</td>
<td>125802.59</td>
<td>9998</td>
<td>12.5827761</td>
<td>Prob &gt; F = 0.0000</td>
</tr>
<tr>
<td>Total</td>
<td>283581.17</td>
<td>9999</td>
<td>28.3409537</td>
<td>R-squared = 0.5564</td>
</tr>
</tbody>
</table>

|        | Coef.    | Std. Err. | t     | P>|t|  | [95% Conf. Interval] |
|--------|----------|-----------|-------|-----|---------------------|
| y      | 2.121985 | 0.0189499 | 111.98 | 0.000 | 2.064839 - 2.15913 |
| _cons  | 0.435372 | 0.0355205 | 12.26  | 0.000 | 0.3657449 - 0.5049994 |

: xtregr y schooling , i(id) fe

Fixed-effects (within) regression
Group variable: id
Number of obs = 10000
Number of groups = 5
R-sq:  within = 0.7538
        between = 0.9997
        overall = 0.5564
Obs per group: min = 2000
              avg = 20000
              max = 2000
corr(u_i, Xb) = 0.4070
F(1, 9994) = 30599.79
Prob > F = 0.0000

|        | Coef.     | Std. Err.  | t     | P>|t|  | [95% Conf. Interval] |
|--------|-----------|------------|-------|-----|---------------------|
| y      | 1.337186  | 0.0076442  | 174.93 | 0.000 | 1.322201 - 1.35217 |
| _cons  | 0.5120157 | 0.0130914  | 39.11  | 0.000 | 0.486354 - 0.5376774 |

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma_u</td>
<td>4.0359677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigma_e</td>
<td>1.3070063</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rho</td>
<td>0.9050817</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F test that all u_i=0:  F(4, 9994) = 15912.37  Prob > F = 0.0000
```
### Results, con’t

```stata
.xi: reg y schooling i.id
.i.id  _Iid_0-4  (naturally coded: _Iid_0 omitted)

Source       |   SS    |   df   |   MS   | Number of obs = 10000
-------------|--------|--------|--------|--------------------------
Model        | 266508.771  |   5  | 53301.7541  | F(  5, 9994) = 31202.27
Residual     | 17072.4053  | 9994 | 1.70826549  | Prob > F = 0.0000
-------------|--------|--------|--------|--------------------------
Total        | 283581.176  | 9999 | 28.3609537  | Adj R-squared = 0.9398
             |        |        |        | Root MSE = 1.307

|                |    Coef. |  Std. Err. |       t     |      P>|t|   [95% Conf. Interval] |
|----------------|----------|------------|-------------|--------|------------------------|
| schooling      | 1.337186  | .0076442   | 174.93      | 0.000  | 1.322201 - 1.35217    |
|   _Iid_1       | 1.943564  | .0414423   | 46.90       | 0.000  | 1.862329 - 2.024799  |
|   _Iid_2       | 3.240114  | .0416526   | 77.79       | 0.000  | 3.158467 - 3.321761  |
|   _Iid_3       | 3.337536  | .0416526   | 80.16       | 0.000  | 3.255294 - 3.419152  |
|   _Iid_4       | 10.63719  | .0447286   | 237.82      | 0.000  | 10.54951 - 10.72486  |
|   _cons        | -3.319665 | .0297053   | -111.75     | 0.000  | -3.377893 - -3.261437 |
```

```stata
.xi i.id
.i.id  _Iid_0-4  (naturally coded: _Iid_0 omitted)

, reg y schooling _I*

Source       |   SS    |   df   |   MS   | Number of obs = 10000
-------------|--------|--------|--------|--------------------------
Model        | 266508.771  |   5  | 53301.7541  | F(  5, 9994) = 31202.27
Residual     | 17072.4053  | 9994 | 1.70826549  | Prob > F = 0.0000
-------------|--------|--------|--------|--------------------------
Total        | 283581.176  | 9999 | 28.3609537  | Adj R-squared = 0.9398
             |        |        |        | Root MSE = 1.307

|                |    Coef. |  Std. Err. |       t     |      P>|t|   [95% Conf. Interval] |
|----------------|----------|------------|-------------|--------|------------------------|
| schooling      | 1.337186  | .0076442   | 174.93      | 0.000  | 1.322201 - 1.35217    |
|   _Iid_1       | 1.943564  | .0414423   | 46.90       | 0.000  | 1.862329 - 2.024799  |
|   _Iid_2       | 3.240114  | .0416526   | 77.79       | 0.000  | 3.158467 - 3.321761  |
|   _Iid_3       | 3.337536  | .0416526   | 80.16       | 0.000  | 3.255294 - 3.419152  |
|   _Iid_4       | 10.63719  | .0447286   | 237.82      | 0.000  | 10.54951 - 10.72486  |
|   _cons        | -3.319665 | .0297053   | -111.75     | 0.000  | -3.377893 - -3.261437 |
```
Results, con’t

. areg y schooling, absorb(id)

Linear regression, absorbing indicators

|          | Coef. | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
|----------|-------|-----------|-------|-----|----------------------|
| schooling| 1.337186 | .0076442 | 174.93 | 0.000 | 1.322201 1.35217 |
| _cons    | .5120157 | .0130914 | 39.11 | 0.000 | .486354 .5376774 |

id | F(4, 9994) = 15912.367 0.000 (5 categories)

; ivreg y (schooling = z) _I*

Instrumental variables (2SLS) regression

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 10000</th>
<th>F(5, 9994) = 21342.82</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>262696.221</td>
<td>5</td>
<td>52539.2442</td>
<td>Prob &gt; F = 0.0000</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>20884.9547</td>
<td>9994</td>
<td>2.08974932</td>
<td>R-squared = 0.9264</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>283581.176</td>
<td>9999</td>
<td>28.3609537</td>
<td>Adj R-squared = 0.9263</td>
<td></td>
</tr>
</tbody>
</table>

|          | Coef. | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
|----------|-------|-----------|-------|-----|----------------------|
| schooling| .9760564 | .0150672 | 64.78 | 0.000 | .9465216 1.006591 |
| _Iid_1   | 2.086821 | .0461029 | 45.26 | 0.000 | 1.996451 2.177192 |
| _Iid_2   | 3.48405 | .0468333 | 74.39 | 0.000 | 3.392248 3.575852 |
| _Iid_3   | 3.574886 | .0467741 | 76.43 | 0.000 | 3.4832 3.666673 |
| _Iid_4   | 11.44499 | .056795 | 201.51 | 0.000 | 11.33366 11.55632 |
| _cons    | -3.570865 | .0339812 | -105.08 | 0.000 | -3.637475 -3.504255 |

Instrumented: schooling

Instruments: _Iid_1 _Iid_2 _Iid_3 _Iid_4 z
. xtivreg y (schooling = z), i(id)

G2SLS random-effects IV regression
Group variable: id
Number of obs = 10000
Number of groups = 5

R-sq: within = 0.7538
between = 0.9997
overall = 0.5564

Obs per group: min = 2000
avg = 2000.0
max = 2000

Wald chi2(1) = 2446.34
Prob > chi2 = 0.0000

corr(u_i, X) = 0 (assumed)

-------------------------------------------------------------
y | Coef.   Std. Err.     z  P>|z|     [95% Conf. Interval]
-------------+--------------------------------------------------------
schooling | .9703856   .0196194  49.46   0.000    .9319323    1.008839
cons | .5478375   .0610966   8.97   0.000    .4280902    .6675847
-------------------------------------------------------------
sigma_u | .0998834
sigma_e | 1.4455965
rho | .00475143  (fraction of variance due to u_i)
-------------------------------------------------------------
Instrumented: schooling
Instruments:  z

. xtivreg y (schooling = z), i(id) fe

Fixed-effects (within) IV regression
Group variable: id
Number of obs = 10000
Number of groups = 5

R-sq: within = 0.6988
between = 0.9997
overall = 0.5564

Obs per group: min = 2000
avg = 2000.0
max = 2000

corr(u_i, Xb) = 0.4070

Wald chi2(1) = 6172.49
Prob > chi2 = 0.0000

-------------------------------------------------------------
y | Coef.   Std. Err.     z  P>|z|     [95% Conf. Interval]
-------------+--------------------------------------------------------
schooling | .9760564   .0150672  64.78   0.000    .9465252    1.005588
cons | .5472836   .0145307  37.66   0.000    .5188041    .5757632
-------------------------------------------------------------
sigma_u | 4.3435356
sigma_e | 1.4455965
rho | .90027943  (fraction of variance due to u_i)

F test that all u_i=0:  F(4,9994) = 11075.25  Prob > F = 0.0000

Instrumented: schooling
Instruments:  z
### Results, con’t

**G2SLS random-effects IV regression**

- **Group variable**: id
- **Number of obs**: 10000
- **Number of groups**: 5
- **R-sq**: within = 0.7538
- **Obs per group**: min = 2000
- **between = 0.9997**
- **avg = 2000.0**
- **overall = 0.5564**
- **max = 2000**

- **corr(u_i, X)** = 0 (assumed)
- **Wald chi2(1)** = 2446.34
- **Prob > chi2** = 0.0000

| y | Coef. | Std. Err. | z | P>|z| | [95% Conf. Interval] |
|---|-------|-----------|---|------|----------------------------|
| schooling | 0.9703856 | 0.0196194 | 49.46 | 0.000 | 0.9319323 - 1.008839 |
| cons | 0.5478375 | 0.0610966 | 8.97 | 0.000 | 0.4820902 - 0.6175847 |

**sigma_u** | 0.0998834 |
**sigma_e** | 1.4455965 |
**rho** | 0.00475143 | (fraction of variance due to u_i)

**Instrumented**: schooling
**Instruments**: z

---

**G2SLS fixed-effects IV regression**

- **Group variable**: id
- **Number of obs**: 10000
- **Number of groups**: 5
- **R-sq**: within = 0.6988
- **Obs per group**: min = 2000
- **between = 0.9997**
- **avg = 2000.0**
- **overall = 0.5564**
- **max = 2000**

- **corr(u_i, X)** = 0.4070
- **Wald chi2(1)** = 6172.49
- **Prob > chi2** = 0.0000

| y | Coef. | Std. Err. | z | P>|z| | [95% Conf. Interval] |
|---|-------|-----------|---|------|----------------------------|
| schooling | 0.9760564 | 0.0150672 | 64.78 | 0.000 | 0.9456252 - 1.006588 |
| cons | 0.5472836 | 0.0165307 | 37.76 | 0.000 | 0.5188041 - 0.5757632 |

**sigma_u** | 4.3435356 |
**sigma_e** | 1.4455965 |
**rho** | 0.90027943 | (fraction of variance due to u_i)

**F test that all u_i=0**: F(4,9994) = 11075.25
**Prob > F** = 0.0000

**Instrumented**: schooling
**Instruments**: z
Fixed effects in Stata

• Many ways to do fixed effects in Stata. Which is best?
  – “xi: regress y x i.id” is almost always inefficient
  – “xi i.id” creates the fixed effects as variables (as “_lid0”, “_lid1”, etc.), so assuming you have the space this lets you re-use them for other commands (e.g. further estimation, tabulation, etc.)
  – “areg” is great for large data sets; it avoids creating the fixed effect variables because it demean the data by group (i.e. it is purely a “within” estimator). But it is not straightforward to get the fixed effect estimates themselves (“help areg postestimation”)
  – “xtreg” is an improved version of areg. It should probably be used instead (although requires panel id variable to be integer, can’t have a string)
  – What if you want state-by-year FE in a large data set?
Generalized linear models (glm)

- $E[y] = g(X^*B) + e$
- $g()$ is called the “link function”. Stata’s “glm” command supports log, logit, probit, log-log, power, and negative binomial link functions.
- Can also make distribution of “e” non-Gaussian and make a different parametric assumption on the error term (Bernoulli, binomial, Poisson, negative binomial, gamma are supported).
- Note that not all combinations make sense (i.e. can’t have Gaussian errors in a probit link function).
- This is implemented in Stata’s ML language (more on this next lecture).
- If link function or error distribution you want isn’t in there, it is very easy to write in Stata’s ML language (again, we will see this more next lecture).
- See Finkelstein (QJE 2007) for an example and discussion of this technique.
Manning (1998) …

“In many analyses of expenditures on health care, the expenditures for users are subject to a log transform to reduce, if not eliminate, the skewness inherent in health expenditure data… In such cases, estimates based on logged models are often much more precise and robust than direct analysis of the unlogged original dependent variable. Although such estimates may be more precise and robust, no one is interested in log model results on the log scale per se. Congress does not appropriate log dollars. First Bank will not cash a check for log dollars. Instead, the log scale results must be retransformed to the original scale so that one can comment on the average or total response to a covariate x. There is a very real danger that the log scale results may provide a very misleading, incomplete, and biased estimate of the impact of covariates on the untransformed scale, which is usually the scale of ultimate interest.”
clear
set obs 100

gen x = invnormal(uniform())
gen e = invnormal(uniform())
gen y = exp(x) + e

gen log_y = log(y)

reg y x
reg log_y  x, robust

glm y x, link(log) family(gaussian)
glm, con’t

- Regression in levels produces coefficient that is too large, while regression in logs produces coefficient that is too low (which we expect since distribution of y is skewed)
Non-parametric estimation

- Stata has built-in support for kernel densities. Often a useful descriptive tool to display “smoothed” distributions of data.
- Can also non-parametrically estimate probability density functions of interest.
- **Example:** Guerre, Perrigne & Vuong (EMA, 2000) estimation of first-price auctions with risk-neutral bidders and iid private values:
  - Estimate distribution of bids non-parametrically
  - Use observed bids and this estimated distribution to construct distribution of values
  - Assume values are distributed according to following CDF:
    \[ F(v) = 1 - e^{-v} \]
  - Then you can derive the following bidding function for \( N=3 \) bidders
    \[ b = \frac{(v+0.5)e^{-2v} - 2(1+v)e^{-v} + 1.5}{1 - 2e^{-v} + e^{-2v}} \]
  - QUESTION: Do bidders “shade” their bids for all values?
GPV with kdensity

clear
set mem 100m
set seed 14170
set obs 5000

local N = 3
gen value = -log(1-uniform())
gen bid = ( (value+0.5)*exp(-2*value)-2*(1+value)*exp(-value)+1.5 ) ///
    / (1-2*exp(-value)+exp(-2*value))
sort bid
gen cdf_G = _n / _N
kdensity bid, width(0.2) generate(b pdf_g) at(bid)

** pseudo-value backed-out from bid distribution
gen pseudo_v = bid + (1/(`N'-1))*cdf_G/pdf_g

twoway (kdensity value, width(0.2) ) (kdensity pseudo_v, width(0.2) ), ///
title("Kernel densities of actual values and pseudo-values") ///
scheme(s2mono) ylabel(, nogrid) graphregion(fcolor(white)) ///
legend(region(style(none))) ///
legend(label(1 "Actual values")) ///
legend(label(2 "Pseudo-values")) ///
legend(cols(1)) ///
xtitle("valuation")
graph export gpv.eps, replace
GPV with \texttt{kdensity}

Kernel densities of actual values and pseudo-values
Quantile regression (qreg)

```stata
qreg log_wage age female edhsg edclg black other _I*, quantile(.1)
matrix temp_betas = e(b)
matrix betas = (nullmat(betas) \ temp_betas)

qreg log_wage age female edhsg edclg black other _I*, quantile(.5)
matrix temp_betas = e(b)
matrix betas = (nullmat(betas) \ temp_betas)

qreg log_wage age female edhsg edclg black other _I*, quantile(.9)
matrix temp_betas = e(b)
matrix betas = (nullmat(betas) \ temp_betas)
```

**QUESTIONS:**
- What does it mean if the coefficient on “edclg” differs by quantile?
- What are we learning when the coefficients are different? (HINT: What does it tell us if the coefficient is nearly the same in every regression)
- What can you do if education is endogenous?
Non-linear least squares (NLLS)

```stata
clear
set obs 50
global alpha = 0.65
gen k=exp(invnormal(uniform()))
gen l=exp(invnormal(uniform()))
gen e=invnormal(uniform())
gen y=2.0*(k^{\alpha}*l^{1-\alpha})+e
```

```
: nl (y = {b0} * l^{b2} * k^{b3})
(obs = 50)
Iteration 0:  residual SS = 350.2076
Iteration 1:  residual SS = 126.9884
Iteration 2:  residual SS = 63.38518
Iteration 3:  residual SS = 59.83952
Iteration 4:  residual SS = 59.83428
Iteration 5:  residual SS = 59.83427

Source |      SS        df    MS----------+-------------------
Model  | 290.37331      3 96.7911034        Number of obs = 50
Residual | 59.834276      47 1.273069265    F( 3,  47) = 76.03
----------+-------------------
Total    | 350.207584      50 7.00415168        Prob > F = 0.0000
         |                   R-squared = 0.8291
         | Adj R-squared = 0.8182
         | Root MSE = 1.128304
         | Res. dev. = 150.8716

          y |      Coef.  Std. Err.      t    P>|t|    [95% Conf. Interval]
----------+------------------------------+-----------+----------+--------------------------------------
    b0  | 2.021732    0.1762746   11.47   0.000     1.667113     2.37635
    b2  | 0.3486468    0.0621272    5.61   0.000     0.2236629     0.4736306
    b3  | 0.6559907    0.1146375    5.72   0.000     0.4253699     0.8866116

(Stata output)
```

(SEs, P values, CIs, and correlations are asymptotic approximations)
Non-linear least squares (NLLS)

• NLLS: minimize the sum of squared residuals to get parameter estimates
  – QUESTION: How are the standard errors calculated?

• The “nl” method provides built-in support for NLLS minimization, and also provides robust and clustered standard errors.

• The syntax allows for many types of nonlinear functions of data and parameters

• Will see examples of NLLS in Stata ML later
clear
clear
set obs 100
set seed 14170

global d = 0.6
global n = 4.0
global A = 2.0

gen k = exp(invnormal(uniform()))
gen l = exp(invnormal(uniform()))
gen e = 0.1 * invnormal(uniform())

** CES production function

gen y = ///
   $A*( (1-$d)*k^($n) + $d*l^($n) )^(1/$n) + e

nl (y = {b0}*( (1-{b1})*k^({b2}) + {b1}*l^({b2}) )^(1/{b2}) ), ///
   init(b0 1 b1 0.5 b2 1.5) robust
More NLLS

** CES production function **

```
gen y = ///
> *A*( (1-d)^k*(n) + d*b1*(n)^(-1/n) + e
```

```
nl (y = b0*( (1-b1)^k*(b2) ) + ///
> (b1)*1^((b2)^(1/123))), ///
> init(b0 1 b1 0.5 b2 1.5) robust
```

(obs = 100)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>residual SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1138.476</td>
</tr>
<tr>
<td>1</td>
<td>3220384</td>
</tr>
<tr>
<td>2</td>
<td>1023647</td>
</tr>
<tr>
<td>3</td>
<td>1017296</td>
</tr>
<tr>
<td>4</td>
<td>1017295</td>
</tr>
</tbody>
</table>

Nonlinear regression with robust standard errors

- Number of obs = 100
- F(3, 97) = 180629.75
- Prob > F = 0.0000
- R-squared = 0.9997
- Root MSE = 0.1024099
- Res. dev. = -175.0146

|     | Coef.  | Robust Std. Err. | t     | P>|t| | [95% Conf. Interval] |
|-----|--------|------------------|-------|-----|-----------------------|
| b0  | 1.99775| 0.0189951        | 105.17| 0.000| 1.96005 - 2.03545     |
| b1  | 0.595155| 0.0063743       | 93.37 | 0.000| 0.5825038 - 0.6078061 |
| b2  | 3.955974| 0.2414426       | 16.38 | 0.000| 3.476777 - 4.43517    |

(SEs, P values, CIs, and correlations are asymptotic approximations)
Conditional FE Poisson (xtpoisson)

• Useful for strongly skewed count data (e.g. days absent), especially when there are a lot of zeroes (since otherwise a log transformation would probably be fine in practice)

• “xtpoisson” provides support for fixed and random effects

```
xtpoisson days_absent gender math reading, i(id) fe
```

• See Acemoglu-Linn (QJE 2004) for a use of this technique (using number of approved drugs as the “count” dependent variable)

• Note that they report clustered standard errors, which are NOT built into Stata

• NOTE: this command is implemented in Stata’s ML language
Arellano-Bond estimator (xtabond)

- Dynamic panel data estimator using GMM
- Specification is lagged dependent variable and use excluded lags as instruments for the other lags
- Example of a GMM implementation in Stata
- Syntax:
  ```
  tsset state year
  xtabond log_payroll miningXoilprice _I*, lags(2)
  ```

- “tsset” is standard command to tell Stata you have a time-series data set (the panel variable is optional for some commands, but for xtabond it is required)
Other important commands

• The following commands are commonly used and you should be aware of them (since they are all ML estimators, we will see some of them tomorrow)
  – probit
  – tobit
  – logit
  – clogit
  – ivprobit
  – ivtobit

• I will also not be discussing these useful commands:
  – heckman
  – cnsreg
  – mlogit
  – mprobit
  – ologit
  – oprobit

• You should look these up on your own, especially after Lecture 3
Stata matrix language

• Before Mata (Lecture 4), Stata had built-in matrix language. Still useful even with Mata because Mata syntax is somewhat cumbersome.

• When to use Stata matrix language:
  – Adding standard errors to existing estimators
  – Writing new estimators from scratch (when such estimators are naturally implemented using matrix algebra)
  – Storing bootstrapping and Monte Carlo results (simulations)
Monte Carlo in Stata

- There is a “simulate” command that is supposed to make your life easier. I don’t think it does, but you should decide for yourself. (“help simulate”)

- Monte Carlo simulations can clarify intuition when the math isn’t obvious.

- EXAMPLE: We will use a simulation to demonstrate the importance of using a robust variance-covariance matrix in the presence of heteroskedasticity.
clear
clear set more off
set mem 100m
set matsize 1000
local B = 1000
matrix Bvals = J(`B', 1, 0)
matrix pvals = J(`B', 2, 0)
forvalues b = 1/`B' {
    drop _all
    quietly set obs 200
    gen cons = 1
    gen x = invnormal(uniform())
    gen e = x*x*invnormal(uniform())
    gen y = 0*x + e

    qui regress y x cons, nocons
    matrix betas = e(b)
    matrix Bvals[`b',1] = betas[1,1]
    qui testparm x
    matrix pvals[`b',1] = r(p)

    qui regress y x cons, robust nocons
    qui testparm x
    matrix pvals[`b',2] = r(p)
}
drop _all
svmat Bvals
svmat pvals
summ *, det
Monte Carlo in Stata, con’t
Monte Carlo in Stata, con’t

```
set more off
On UNIX, this will keep the buffer from “locking”

set matsize 1000
Sets default matrix size

matrix Bvals = J(`B', 1, 0)
 Creates `B’-by-1 matrix

drop _all
Unlike “clear”, this only drops the data (NOT matrices!)

quietly set obs 200
Suppresses output

qui regress y x cons, nocons
“qui” is abbreviation; nocons means constant not included

matrix betas = e(b)
e() stores the return values from regression; e(b) is betas

matrix Bvals[`b',1] = betas[1,1]
syntax to set matrix values

qui testparm x
performs a Wald test to see if “x” is statistically significant

qui regress y x cons, robust nocons
uses “robust” standard errors

svmat Bvals
writes out matrix as a data column
```
Monte Carlo in Stata, con’t

```
.summ *, det

Bvals1
Perentiles                  Smallest
1%    -.7108901              -.867761
5%    -.4510751              -.8387734
10%   -.3646669              -.8244087  Obs    1000
25%   -.1722884              -.7698361  Sum of Wgt. 1000
50%   .0038993                      Mean    -.0046056
75%   .1797461              .6604921  Largest Std. Dev. .2693396
90%   .3240661              .7583284  Variance  .0725433
95%   .4148443              .816509  Skewness  -.1505003
99%   .605374               .9324661  Kurtosis  3.210481

pvals1
Perentiles                  Smallest
1%    6.92e-07               2.28e-11
5%    .0000421              9.73e-10
10%   .0003121              1.40e-08  Obs    1000
25%   .0123931              1.59e-08  Sum of Wgt. 1000
50%   .1238899                      Mean    .2603048
75%   .4540875              .9949121  Largest Std. Dev. .2998066
90%   .7795787              .9956773  Variance  .089884
95%   .9047533              .9977634  Skewness  1.036197
99%   .9803216              .9999624  Kurtosis  2.771922

pvals2
Perentiles                  Smallest
1%    .0083116               .0009476
5%    .0529822              .0011581
10%   .0896799              .002251  Obs    1000
25%   .2348141              .0027212  Sum of Wgt. 1000
50%   .4535128                      Mean    .4742613
75%   .7167521              .9971678  Largest Std. Dev. .2866344
90%   .8921935              .9980425  Variance  .082274
95%   .9535336              .9989994  Skewness  .1459861
99%   .9904989              .9999799  Kurtosis  1.861137
```
clear
set obs 10
set seed 14170
gen x1 = invnorm(uniform())
gen x2 = invnorm(uniform())
gen y = 1 + x1 + x2 + 0.1 * invnorm(uniform())

gen cons = 1
mkmat x1 x2 cons, matrix(X)
mkmat y, matrix(y)
matrix list X
matrix list y

matrix beta_ols = invsym(X'*X) * (X'*y)
matrix e_hat = y - X * beta_ols
matrix V = (e_hat' * e_hat) * invsym(X'*X) / (rowsof(X) - colsof(X))
matrix beta_se = (vecdiag(V))'
local rows = rowsof(V)
forvalues i = 1/`rows' {
    matrix beta_se[`i',1] = sqrt(beta_se[`i',1])
}
matrix ols_results = [beta_ols, beta_se]
matrix list ols_results
reg y x1 x2

\[ \beta = (X'X)^{-1}X'y \]
. matrix list X

X[10,3]

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>-1.5950224</td>
<td>.20092869</td>
<td>1</td>
</tr>
<tr>
<td>r2</td>
<td>-.64034598</td>
<td>-.60358792</td>
<td>1</td>
</tr>
<tr>
<td>r3</td>
<td>-.40134595</td>
<td>2.359099</td>
<td>1</td>
</tr>
<tr>
<td>r4</td>
<td>-.60476729</td>
<td>.11293815</td>
<td>1</td>
</tr>
<tr>
<td>r5</td>
<td>-.26287592</td>
<td>-.71784865</td>
<td>1</td>
</tr>
<tr>
<td>r6</td>
<td>-.36271503</td>
<td>-1.9359163</td>
<td>1</td>
</tr>
<tr>
<td>r7</td>
<td>1.7407799</td>
<td>1.1414781</td>
<td>1</td>
</tr>
<tr>
<td>r8</td>
<td>-.03460691</td>
<td>2.2267994</td>
<td>1</td>
</tr>
<tr>
<td>r9</td>
<td>1.4960149</td>
<td>1.4628167</td>
<td>1</td>
</tr>
<tr>
<td>r10</td>
<td>.48152901</td>
<td>-1.2280046</td>
<td></td>
</tr>
</tbody>
</table>

. matrix list y

y[10,1]

<table>
<thead>
<tr>
<th></th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>-.51445416</td>
</tr>
<tr>
<td>r2</td>
<td>-.36395637</td>
</tr>
<tr>
<td>r3</td>
<td>2.8763379</td>
</tr>
<tr>
<td>r4</td>
<td>.46112738</td>
</tr>
<tr>
<td>r5</td>
<td>-.05183486</td>
</tr>
<tr>
<td>r6</td>
<td>-1.1868566</td>
</tr>
<tr>
<td>r7</td>
<td>3.9082622</td>
</tr>
<tr>
<td>r8</td>
<td>3.0423635</td>
</tr>
<tr>
<td>r9</td>
<td>4.094496</td>
</tr>
<tr>
<td>r10</td>
<td>.25329242</td>
</tr>
</tbody>
</table>
matrix ols_results = [beta_ols, beta_se]

matrix list ols_results

ols_results[3,2]
    y         r1
    x1 1.0719562 .02491212
    x2 .97005487 .01720588
    cons .97870197 .02363347

reg y x1 x2

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>35.9448584</td>
<td>2</td>
<td>17.9724292</td>
<td>F( 2, 7) = 3390.36</td>
</tr>
<tr>
<td>Residual</td>
<td>.03710729</td>
<td>7</td>
<td>.005301041</td>
<td>Prob &gt; F = 0.0000</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>9</td>
<td>3.99799619</td>
<td>R-squared = 0.9990</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Adj R-squared = 0.9987</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Root MSE = .07281</td>
</tr>
</tbody>
</table>

| y      | Coef.   | Std. Err. | t     | P>|t|  | 95% Conf. Interval          |
|--------|---------|-----------|-------|-----|-----------------------------|
| x1     | 1.071956 | .02491212 | 43.03 | 0.000 | 1.013048 1.130864           |
| x2     | .9700549 | .01720588 | 56.38 | 0.000 | .9293694 1.01074            |
| _cons  | .978702  | .0236335  | 41.41 | 0.000 | .9228177 1.034586           |
clear
set obs 100000
set seed 14170
gen x1 = invnorm(uniform())
gen x2 = invnorm(uniform())
gen y = 1 + x1 + x2 + 0.1 * invnorm(uniform())
gen cons = 1
mkmat x1 x2 cons, matrix(X)
mkmat y, matrix(y)
matrix list X
matrix list y

matrix beta_ols = invsym(X'*X) * (X'*y)
matrix e_hat = y - X * beta_ols
matrix V = (e_hat' * e_hat) * invsym(X'*X) /
rowsof(X)^- colsof(X))
matrix beta_se = (vecdiag(V))'
local rows = rowsof(V)
forvalues i = 1/`rows' {
    matrix beta_se[`i',1] = sqrt(beta_se[`i',1])
}
matrix ols_results = [beta_ols, beta_se]
matrix list ols_results
reg y x1 x2
. clear

. set obs 100000
obs was 0, now 100000

. set seed 14170

. gen x1 = invnorm(uniform())

. gen x2 = invnorm(uniform())

. gen y = 1 + x1 + x2 + 0.1 * invnorm(uniform())

. gen cons = 1

. mkmat x1 x2 cons, matrix(X)
matsize too small to create a [100000,3] matrix
r(908);

end of do-file

r(908);

("help set matsize"; maximum matrix size is
11,000 on Stata/SE and Stata/MP)
clear
clear
global xlist = "x1 x2"
global xlist = "x1 x2"

set obs 100000
set obs 100000
set seed 14170
set seed 14170
gen x1 = invnorm(uniform())
gen x1 = invnorm(uniform())
gen x2 = invnorm(uniform())
gen x2 = invnorm(uniform())
gen y = 1 + x1 + x2 + 100 * invnorm(uniform())
gen y = 1 + x1 + x2 + 100 * invnorm(uniform())
global xlist = "x1 x2"
global xlist = "x1 x2"
matrix accum XpX = $xlist
matrix accum XpX = $xlist
matrix vecaccum Xpy = y $xlist
matrix vecaccum Xpy = y $xlist
matrix beta_ols = invsym(XpX) * Xpy'
matrix beta_ols = invsym(XpX) * Xpy'
matrix list beta_ols
matrix list beta_ols
gen e_hat = y
gen e_hat = y
local i = 1
local i = 1
foreach var of varlist $xlist {
foreach var of varlist $xlist {
  replace e_hat = e_hat - beta_ols[`i',1] * `var'
  replace e_hat = e_hat - beta_ols[`i',1] * `var'
  local i = `i' + 1
  local i = `i' + 1
}
}
** constant term!
** constant term!
replace e_hat = e_hat - beta_ols[`i',1]
replace e_hat = e_hat - beta_ols[`i',1]
matrix accum e2 = e_hat, noconstant
matrix accum e2 = e_hat, noconstant
matrix V = invsym(XpX) * e2[1,1] / (_N - colsof(XpX))
matrix V = invsym(XpX) * e2[1,1] / (_N - colsof(XpX))
matrix beta_se = (vecdiag(V))'
matrix beta_se = (vecdiag(V))'
local rows = rowsof(V)
local rows = rowsof(V)
forvalues i = 1/`rows' {
forvalues i = 1/`rows' {
  matrix beta_se[`i',1] = sqrt(beta_se[`i',1])
  matrix beta_se[`i',1] = sqrt(beta_se[`i',1])
}
}
matrix ols_results = [beta_ols, beta_se]
matrix ols_results = [beta_ols, beta_se]
matrix list ols_results
matrix list ols_results
reg y x1 x2
reg y x1 x2
```
reg y x1 x2

Source | SS       | df  | MS          |         | Number of obs = 100000
-------+----------+-----+-------------+---------+---------------------
Model  | 213407,047  | 2   | 106703,523  |         | F(  2, 99997) = 10.72 |
Residual | 995296663  | 99997 | 9953,26523  |         | Prob > F = 0.00000 |
Total   | 995510070  | 99999 | 9955,20025  |         | R-squared = 0.0002  |
        |          |     |             |         | Adj R-squared = 0.0002 |
        |          |     |             |         | Root MSE = 99.766 |

| y | Coef.     | Std. Err. | t    | P>|t| | [95% Conf. Interval] |
|---|-----------|-----------|------|------|----------------------|
x1 | 1.0141917 | .31550833 | 3.21 | 0.001 | .3957093 1.632504 |
x2 | 1.0507246 | .31459431 | 3.34 | 0.001 | .4344236 1.667326 |
_cons | .8685545 | .31548979 | 2.75 | 0.006 | .2501979 1.48691 |
```
clear
set obs 1000
program drop _all
program add_stat, eclass
    ereturn scalar `1' = `2'
end

gen z = invnorm(uniform())
gen v = invnorm(uniform())
gen x = .1*invnorm(uniform()) + 2.0*z + 10.0*v

gen y = 3.0*x + (10.0*v + .1*invnorm(uniform()))
reg y x
estimates store ols
reg x z
test z
return list
add_stat "F_stat" r(F)
estimates store fs
reg y z
estimates store rf
ivreg y (x = z)
estimates store iv
estout * using baseline.txt, drop(_cons) ///
    stats(F_stat r2 N, fmt(%9.3f %9.3f %9.0f)) modelwidth(15) ///
    cells(b(fmt(%9.3f)) se(par fmt(%9.3f)) p(par([ ] fmt(%9.3f))) ) ///
    style(tab) replace notype mlabels(, numbers )
“helper” programs

```
.*reg x z
.

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs =  1000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Model</td>
<td>4328.58096</td>
<td>1</td>
<td>4328.58096</td>
<td>F(  1,    998) =  43.91</td>
</tr>
<tr>
<td>Residual</td>
<td>98386.7101</td>
<td>998</td>
<td>98.5858819</td>
<td>Prob &gt; F =  0.0000</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Total</td>
<td>102717.291</td>
<td>999</td>
<td>102.820111</td>
<td>R-squared =  0.0421</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------</td>
</tr>
</tbody>
</table>

| x | Coef.  | Std. Err. | t     | P>|t|  | [95% Conf. Interval] |
|---|--------|-----------|-------|------|-----------------|------------------|
| z | 2.077355 | .3135057 | 6.63  | 0.000 | 1.462149     | 2.692561         |
| _cons | .1735656 | .3139839 | 0.55  | 0.581 | -.4425789    | .78971           |

.* test z
( 1)  z = 0

F(  1,    998) =  43.91
Prob > F =  0.0000

.* return list

scalars;
r(drop) = 0
r(df_r) = 998
r(F) = 43.90670219895473
r(df) = 1
r(p) = 5.62473556326e-11

.* add_stat "F_stat" r(F)

.* estimates store fs


```
$ more baseline.txt

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ols</td>
<td>fs</td>
<td>rf</td>
<td>IV</td>
</tr>
<tr>
<td>b/se/p</td>
<td>3.959</td>
<td>3.034</td>
<td>6.303</td>
<td>43.907</td>
</tr>
<tr>
<td>x</td>
<td>0.006</td>
<td>0.146</td>
<td>1.254</td>
<td>0.000</td>
</tr>
<tr>
<td>z</td>
<td>2.077</td>
<td>0.998</td>
<td>0.042</td>
<td>0.025</td>
</tr>
<tr>
<td>N</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>
```

"helper" programs
```
**
** Monte Carlo to investigate heteroskedasticity-robust s.e.'s
**
clear
set more off
set seed 14170
local count = 0

global B = 1000
forvalues i = 1(1)$B {
    quietly {
        clear
        set obs 2000
        gen x = invnorm(uniform())
        gen y = 0*x + abs(0.1*x)*invnorm(uniform())
        regress y x
        test x
        if (r(p) < 0.05) {
            local count = `count' + 1
        }
    }
}
local rate = `count' / $B
di "Rejection rate (at alpha=0.05): `rate'"
```

0.236 🙁
**
** Monte Carlo to investigate heteroskedasticity-robust s.e.'s
**
clear
set more off
set seed 14170
local count = 0

global B = 1000
forvalues i = 1(1)$B {
quietly {
  clear
  set obs 2000
  gen x = invnorm(uniform())
  gen y = 0*x + abs(0.1*x)*invnorm(uniform())
  regress y x, robust
  test x
  if (r(p) < 0.05) {
    local count = `count' + 1
  }
}
}
local rate = `count' / $B
di "Rejection rate (at alpha=0.05): `rate'"
0.048 😊
(robust_regress.ado file)

program define robust_regress, eclass
    syntax varlist
    gettoken depvar varlist: varlist

    quietly regress `depvar' `varlist'
predict resid, residuals
    gen esample = e(sample)
    local obs = e(N)
    matrix betas = e(b)

    matrix accum XpX = `varlist'
gen all = _n
    sort all
    matrix opaccum W = `varlist', opvar(resid) group(all)
    matrix V = invsym(XpX) * W * invsym(XpX)

    ereturn post betas V, dep(`depvar') o(`obs') esample(esample)
    ereturn display
end

\[
V_{\text{robust}} = (XX)^{-1} \left( \sum_{i=1}^{N} (\hat{e}_i x_i)'(\hat{e}_i x_i) \right) * (XX)^{-1}
\]
ADO files in Stata

**
** Monte Carlo to investigate heteroskedasticity-robust s.e.'s
**
clear
set more off
set seed 14170
local count = 0

global B = 1000
forvalues i = 1(1)$B {
quietly {
    clear
    set obs 2000
    gen x = invnorm(uniform())
    gen y = 0*x + abs(0.1*x)*invnorm(uniform())
    robust_regress y x
    test x
    if (r(p) < 0.05) {
        local count = `count' + 1
    }
}
local rate = `count' / $B
di "Rejection rate (at alpha=0.05): `rate" 0.049 😊


ADO files in Stata

clear
set obs 2000
gen x = invnorm(uniform())
gen y = 0*x + abs(0.1*x)*invnorm(uniform())
robust_regress y x
regress y x, robust

```

. robust_regress y x
(obs=2000)

|     | Coef.  | Std. Err. |      z  |  P>|z|  |   [95% Conf. Interval] |
|-----|--------|-----------|---------|------|------------------------|
| y   |        |           |         |      |                        |
| x   | -.0001554 | .0035848  | -0.04   | 0.965 | -.0071815 - .0068707   |
| _cons | -.0013051 | .002211   | -0.59   | 0.555 | -.0056385 - .0030283   |

. regress y x, robust

Linear regression  Number of obs = 2000
F( 1, 1998) = 0.00
Prob > F = 0.9655
R-squared = 0.0000
Root MSE = .09897

|     | Coef.  | Std. Err. | t     |  P>|t|  |   [95% Conf. Interval] |
|-----|--------|-----------|-------|------|------------------------|
| y   |        |           |       |      |                        |
| x   | -.0001554 | .0035866  | -0.04 | 0.965 | -.0071892 - .0068735   |
| _cons | -.0013051 | .0022121  | -0.59 | 0.555 | -.0056433 - .0030331   |
```
ADO files in Stata

(program define robust_regress, eclass
    syntax varlist
    gettoken depvar varlist: varlist

    quietly reg `depvar' `varlist', robust
    predict resid, residuals
    gen esample = e(sample)
    local obs = e(N)
    matrix betas = e(b)

    matrix accum XpX = `varlist'
    gen all = _n
    sort all
    matrix opaccum W = `varlist', opvar(resid) group(all)
    matrix V = (_N/(_N-colsof(XpX))) * invsym(XpX) * W * invsym(XpX)

    ereturn post betas V, dep(`depvar') o(`obs') esample(esample)
    ereturn display
end)

\[ V_{\text{robust}} = \frac{N}{N-K} \cdot (X'X)^{-1} \cdot \left( \sum_{i=1}^{N} (\hat{e}_i x_i)'(\hat{e}_i x_i) \right) \cdot (X'X)^{-1} \]
“Reflections on Trusting Trust”

• Ken Thompson Turing Award speech:
  (http://www.ece.cmu.edu/~ganger/712.fall02/papers/p761-thompson.pdf)

  “You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me).”
(an aside) More on “trusting trust”

clear
set obs 2000
set seed 14170
gen x = invnorm(uniform())
gen id = 1 + floor((_n - 1) / 50)
gen y = x + id + abs(x) * invnorm(uniform()) + id
areg y x, \///
    cluster(id) absorb(id)

STATA v9.1
STATA v10.0
Exercises

• Go to following URL:

• Download each DO file
  – No DTA files! All data files loaded from the web (see “help webuse”)

• 1 exercise (increasing difficulty)
  A. Monte carlo test of OLS/GLS with serially correlated data
  B. Heckman two-step with bootstrapped standard errors
  C. Correcting for measurement error of known form