Introduction to Stata - Session 3

Simon, Hong

ECON 3150/4150, UiO

January 15, 2018

Before we start

1. Download caschool.dta from course webpage
2. Start STATA from the Start menu
   2.1 Alternatively, from kiosk.uio.no using internet explorer: Log on, navigate to analyse, open Stata

Outline

1. Stata workflow
2. Working with do-files
   - text files with Stata code, why better?
   - formatting: make your file readable
   - comments: include description of the code
3. Regression, prediction, testing
4. Basic graphs
   - scatter plots
   - line plots
   - overlying graphs
   - basic formatting
   - exporting

Stata workflow

Always better to separate changing & analysing the data:

1. First prepare your data for analysis
   - copy data from disk to memory
   - describe data
   - change date
   - save modified data to disk under new name
2. Then analyse the modified data
   - copy analysis data into memory
   - start logging results to file
   - perform analysis
   - close log file

Advice: one directory per project & start session in project directory

DO files

Until now we have mostly used the command line:

- great to develop but not to reproduce your analysis
- ALWAYS organize your work in Stata scripts

Stata scripts are called do-files after their extension (.do)
Use do-files (with informative names) to organize your work:

- create dataset
  crincome.do makes data file income.dta
- analysis
  andescr.do calculates my descriptive statistics
  anreg.do performs my regression analysis
- making graphs
  grwageplot.do makes the graph wageplot.eps

Make a do-file

Use caschool.dta, then open a do file and try to do the following things

1. Read the data into Stata
2. Keep only `read_scr` , `math_scr` , `enrl_tot` , `teachers` and `el_pct`
3. Make new variable `score` equal to mean of reading and math score
4. Make new var `str` equal the student-to-teacher ratio
5. Label the variables:
   `enrl_tot` Enrollment
   `teachers` Teachers
   `el_pct` Percent english-learners
   `score` Mean test score
   `str` Student/Teachers
6. Make a new variable `zscore` as standardized score, i.e.

$$zscore = \frac{score - mean(score)}{SD(score)}$$

In case something went wrong

```
cd "PATH"
use "PATH/caschool.dta", clear
desc
keep read_scr math_scr enrl_tot teachers el_pct
egen score = rowmean(read_scr math_scr)
gen str = enrl_tot/teachers
label var score    "Test score"
label var enrl_tot "Enrollment"
label var teachers "Teachers"
label var str      "Student/Teacher"
label var el_pct   "Percent english-learners"
egen meanscore = mean(score)
egen sdscore = sd(score)
gen zscore = (score - meanscore) / sdscore
```

Digression: Stata keeps one (1) table in memory at a time

```
. sum el_pct
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      el_pct |       420    15.76816    18.28593          0   85.53972

. sum score
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       score |       420    654.1565    19.05335      605.55      706.75

. return list
scalars:
                  r(N) =  420
              r(sum_w) =  420
               r(mean) =  654.156548055013
                r(Var) =  363.0300564285935
                 r(sd) =  19.05334764361879
                r(min) =  605.550048828125
                r(max) =  706.75
                r(sum) =  274745.7501831055

. display r(mean)
654.15655
```

zscore : alternatives

```
egen meanscore = mean(score)
egen sdscore = sd(score)
gen zscore = (score - meanscore) / sdscore

/*
   mean and standard deviation of score are stored
   temporarily under r(mean) and r(sd)
*/

sum score
gen zscore1 = (score - r(mean)) / r(sd)

/*
  std creates the standardized values of exp. The options specify
  the desired mean and standard deviation.
  The default is mean(0) and std(1), producing a
  variable with mean 0 and standard deviation 1.
*/
egen zscore2 = std(score)
```

Documenting - Comments

Use comments in your do-files when the code needs explaining or is better readable with a comment

- Single line comments:
  ```
  // comment here
  ```
- Multi line comments:
  ```
  /*
  [commented out]
  */
  ```
- Break lines:
  ```
  list pop /// the rest of the line is commented out
          if country=="NOR"
  ```

Stata syntax - Regression

We are interested in the relationship btw the standardized score and student-to-teacher ratio.

- OLS fits a line through this observation cloud

```
twoway scatter zscore str
```

- Specifically, we estimate α and β in the equation

$$stdscore_i = \alpha + \beta ratio_i + \epsilon_i \tag{1}$$

Make a do-file - cont.

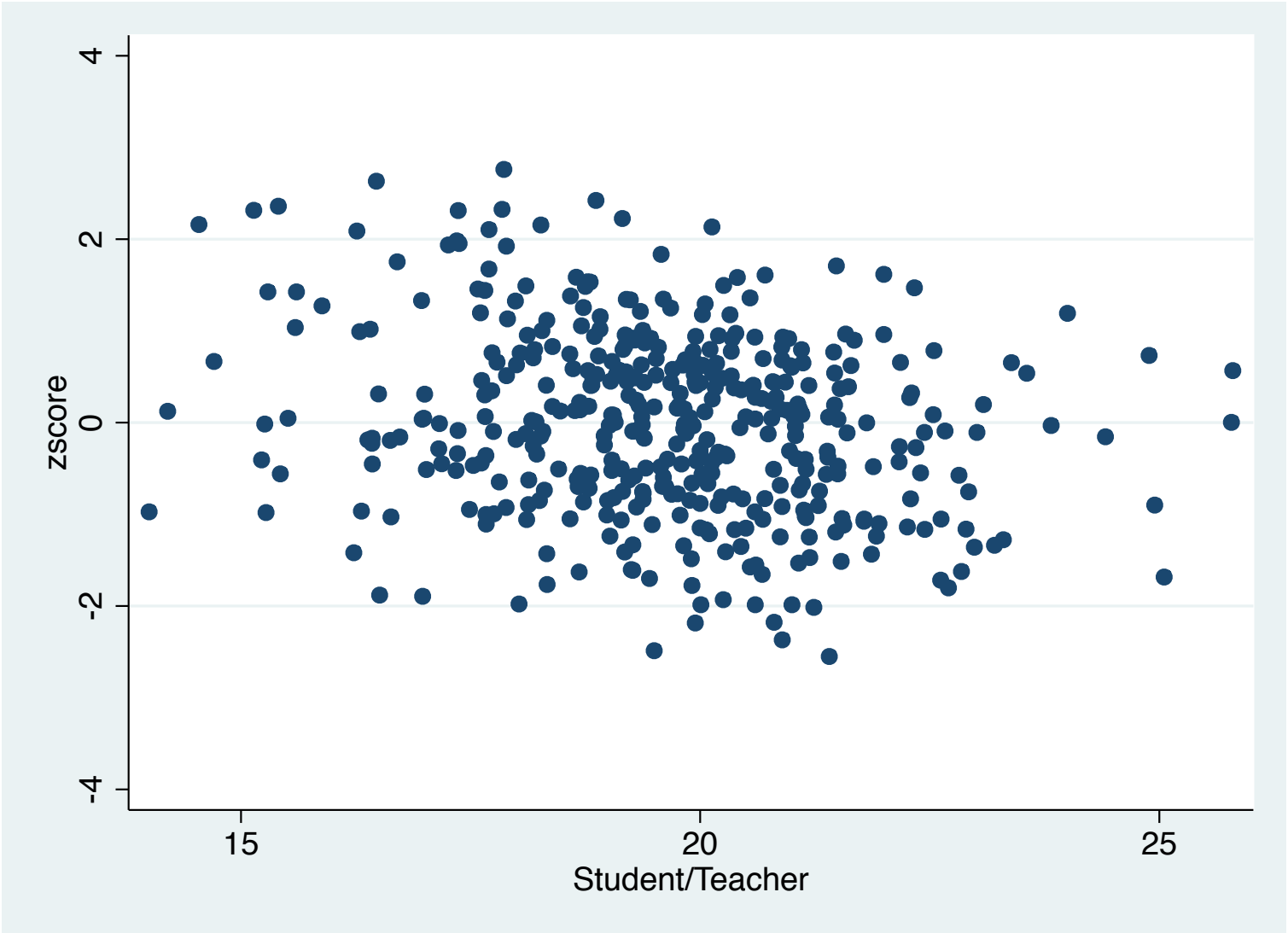
1. save the data under new name (e.g. `caschool_ready.dta`)
2. Summarize all variables
3. Draw a scatter of `zscore` against `str`
4. Regress `zscore` on `str`
5. Make new var `zscorehat` as the prediction from the regression
   - *hint*: use `-predict-`
6. Draw a scatter of zscore against str, including the predicted regression line
   - *hint*: use `-twoway (scatter y x) (line z x)-`

In case something went wrong

```
save caschool_ready.dta
summ
two scatter zscore str
regress zscore str, robust
predict zscorehat
two (scatter zscore str) ///
    (line zscorehat str) ///
    , scheme(s1color) legend(pos(7) ring(0))
```

In case something went wrong

```
two scatter zscore str
```

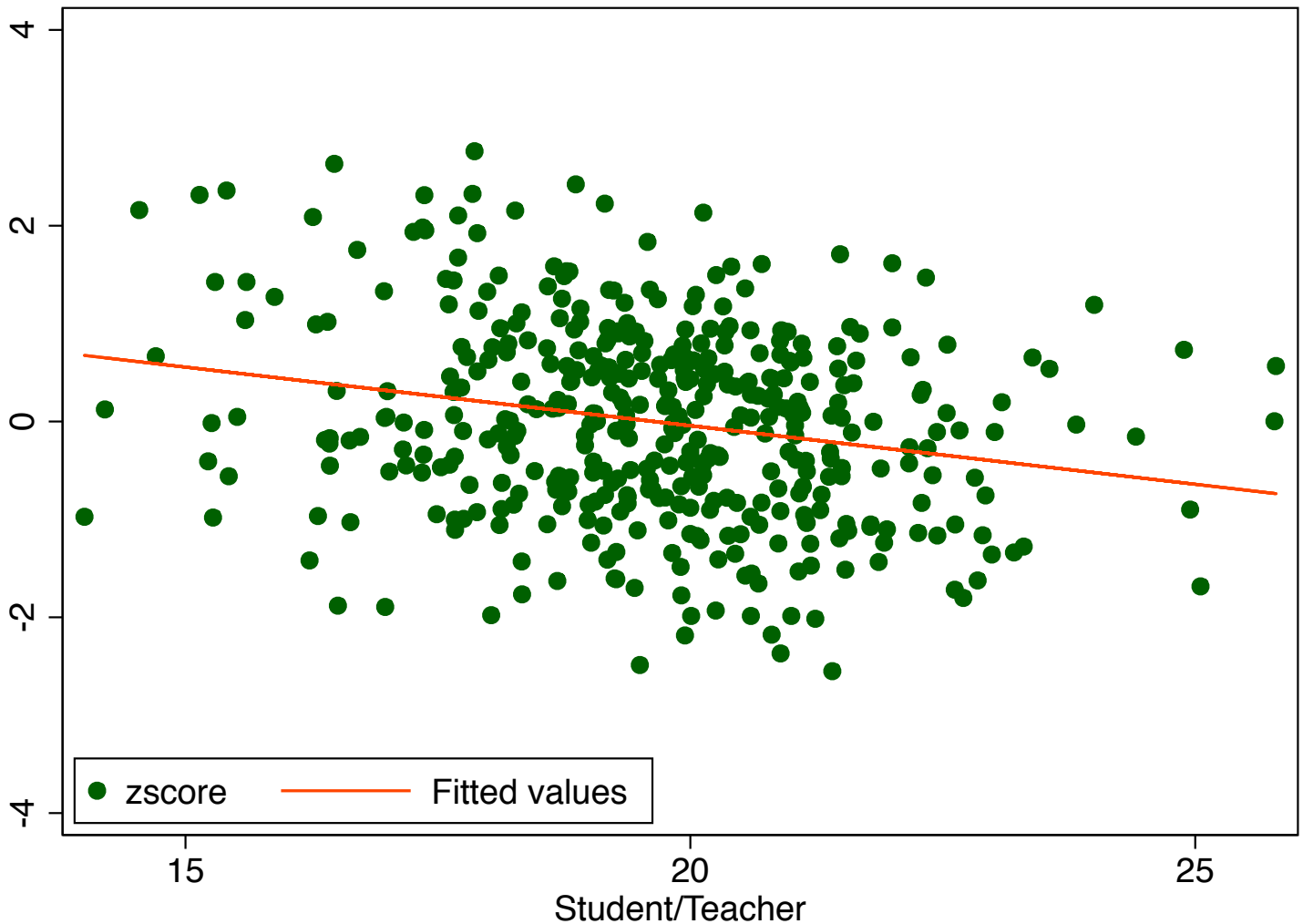


In case something went wrong

```
. regress zscore str, robust

Linear regression                               Number of obs =     420
                                                F(  1,   418) =   19.26
                                                Prob > F      =  0.0000
                                                R-squared     =  0.0512
                                                Root MSE      =  .97521

------------------------------------------------------------------------------
             |               Robust
      zscore |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         str |  -.1196539    .027265    -4.39   0.000    -.1732475   -.0660604
       _cons |   2.350054   .5439653     4.32   0.000     1.280806    3.419302
------------------------------------------------------------------------------

. predict zscorehat
(option xb assumed; fitted values)
```

In case something went wrong

```
two (scatter zscore str) ///
    (line zscorehat str) ///
    , scheme(s1color) legend(pos(7) ring(0))
```



Making tables from regression results

Estimation commands such as `-regress-` store results (press `ereturn list` after regression to show what is stored)

- These can be used to make tables using Stata's `-estimates-` `-help estimates-`
- To store estimates in memory: `-estimates store-`
- To activate previously stored estimates: `-estimates restore-`
- To table estimates: `-estimates table [estnames]-`

Making tables from regression results

```
. quietly reg zscore str , robust
. est store ratiost

. quietly reg zscore el_pct , robust
. est store elperc

. quietly reg zscore str el_pct , robust
. est store ratioel

. est table ratiost elperc ratioel

----------------------------------------------------
    Variable |  ratiost       elperc      ratioel
-------------+--------------------------------------
         str | -.11965394                -.05780065
      el_pct |              -.0352251   -.03410302
       _cons |  2.3500539    .55543454    1.6729707
----------------------------------------------------
```

The estout-package

`estout` is a user contributed add-on with many options

- you should install such add-ons in a dedicated directory (named e.g. ado or stata)
- this is a little cumbersome when you are working from the server
  - see course web page for how to install programs (add-ons) that you find online or using Stata's `-findit-` or `-net search-`
  - this is very useful in practice

  Now type `findit estout` , scroll down and click through to install

```
esttab *, se
```

Make a do-file, cont.

1. Make a new var `elhigh` equal to 1 if `el_pct` $>$ `mean(el_pct)`
2. Table means of `score` and `zscore` for the two groups
3. Regress `zscore` on `str` controlling for `elhigh`
4. Table results from this and the previous regression together
5. Make new var `zscorehat_elhigh` as the prediction
6. Draw a scatter of `zscore` against `str`, including the predicted regression line, where both scatter and line are separate for the two groups

In case something went wrong

```
sum el_pct
gen elhigh = el_pct > r(mean) if el_pct != .
label var elhigh "Many english-learners"
label define elhighlbl 1 "el_pct > `r(mean)'"
label define elhighlbl 0 "el_pct <= `r(mean)'", add
label values elhigh elhighlbl
tabstat score zscore str, by(elhigh)
regress zscore str elhigh, robust
estimates store elhigh
predict zscorehat_elhigh
two (scatter zscore str if elhigh == 1, mcolor(blue) msymbol(Oh)) ///
  (scatter zscore str if elhigh == 0, mcolor(red) msymbol(X)) ///
  (line zscorehat_elhigh str if elhigh == 1, lwidth(medthick) lpattern(solid) lcolor(blue)) ///
  (line zscorehat_elhigh str if elhigh == 0, lwidth(medthick) lpattern(solid) lcolor(red)) ///
  , scheme(s1color) legend(pos(7) ring(0) label(1 "Many english-learners") label(2 "Few english-learners"))
```

Saving your results (logging)

You can save your results to file using -log-

```
log using anauto
```

1. the log file exists

```
log using anauto, replace
```

2. the log file is already open

```
close log
```

3. when there is no open log, final solution:

```
capture close log
```

Plain text log file:

```
log using anauto, replace text
```

Try to use the same name as the do file!

A typical do file (anreg.do)

```
clear
cd "M://My Documents/statacourse"
capture log close
log using anreg , replace
set more off

// do analysis here

// sometimes:
quietly log close
      // do something that you don't need to log here
      quietly log using anreg , append
       // do further analysis here

log close
// always leave one empty line at the end
```

Make a do-file, cont.

1. Make new var `strelhigh` as the interaction of `str` and `elhigh`
2. Repeat the previous regression, incl. `strelhigh`
3. Form the prediction `zscorehat_elhighint`
4. Table results from all the regressions together
5. Draw a scatter of `zscore` against `str`, including the predicted regression line, where both scatter and line are separate for the two groups

```
gen strelhigh = str * elhigh
regress zscore str strelhigh elhigh, robust
test strelhigh elhigh
estimates store elhighint
predict zscorehat_elhighint
two (scatter zscore str if elhigh == 1, mcolor(blue) msymbol(Oh)) ///
  (scatter zscore str if elhigh == 0, mcolor(red) msymbol(X)) ///
  (line zscorehat_elhighint str if elhigh == 1, lwidth(medthick) lpattern(solid) lcolor(blue)) ///
  (line zscorehat_elhighint str if elhigh == 0, lwidth(medthick) lpattern(solid) lcolor(red)) ///
  , scheme(s1color) legend(pos(7) ring(0) label(1 "Many english-learners") label(2 "Few english-learners"))
```

Hypothesis testing

To do hypothesis testing, use -test-

- test one variable $\beta_1 = 0$

```
test var1
```

- test one variable $\beta_1 = 0$. $\beta_2 = 1$

```
test var1=0 var2=0
```

- test combined hypotheses $\beta_1 = \beta_2$

```
test var1 = var2
```

Hypothesis testing

```
. quietly regress zscore str strelhigh elhigh , robust

. test strelhigh
 ( 1)  strelhigh = 0

       F(  1,   416) =    0.32
            Prob > F =    0.5717

. test strelhigh elhigh
 ( 1)  strelhigh = 0
 ( 2)  elhigh = 0

       F(  2,   416) =  107.95
            Prob > F =    0.0000

. gen strellow = str * (1 - elhigh )
. quietly regress zscore strellow strelhigh elhigh , robust

. test strellow = strelhigh

 ( 1)  strellow - strelhigh = 0

       F(  1,   416) =    0.32
            Prob > F =    0.5717
```

Hypothesis testing

You can also do tests of e.g. means

```
. ttest zscore, by(elhigh)

Two-sample t test with equal variances
------------------------------------------------------------------------------
   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
el_pct < |     276    .4122849    .0507242    .8426944    .3124278    .5121421
el_pct > |     144   -.7902127    .0649548    .7794575   -.9186084   -.6618171
---------+--------------------------------------------------------------------
combined |     420   -1.62e-09     .048795           1   -.0959135    .0959135
---------+--------------------------------------------------------------------
    diff |            1.202498    .0844605                1.036477    1.368518
------------------------------------------------------------------------------
    diff = mean(el_pct <) - mean(el_pct >)                        t =  14.2374
Ho: diff = 0                                     degrees of freedom =      418

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 1.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 0.0000
```

Saving Graphs

- You can save your last graph (graph window has to be still open) in Stata to disk using

```
graph export filename
```

- The extension determines the format, e.g.

```
graph export zscore-str.eps
```

- if the file exists, use option -replace-, to overwrite it with the new graph.

```
graph export zscore-str.eps, replace
```

Note

1. Best quality: Vector based formats (ps, eps, pdf, wmf/emf (Win only))
2. Most portable: Pixel-based formats (png)

What you should have learned

1. Stata workflow
2. Working with do-files
   - text files with Stata code
   - formatting: make your file readable!
   - comments: include description of the code
3. Regression, prediction, testing
4. Basic graphs
   - scatter plots
   - line plots
   - overlaying graphs
   - basic formatting
   - exporting