Econometrics: Introduction to Stata

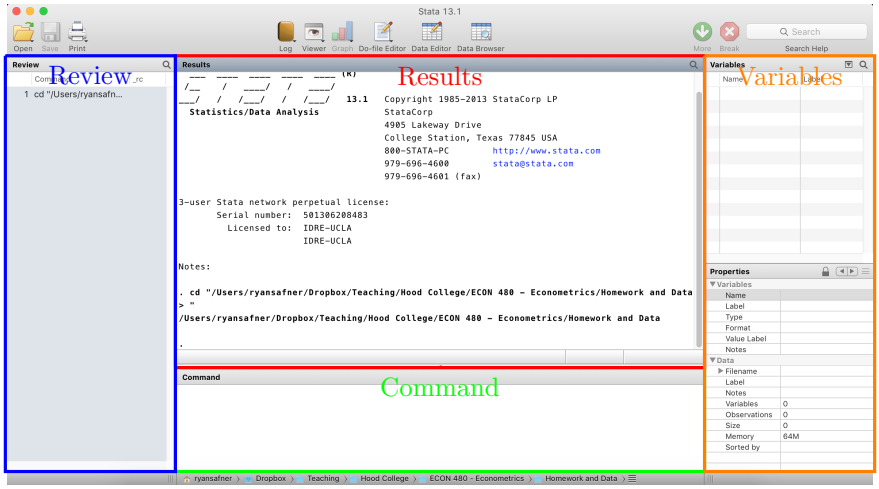
Ryan Safner

Fall 2017

Stata is an extremely powerful software package that can work with large datasets & spreadsheets, run many high-powered statistical tests, and create detailed graphics. Its main equivalents are SPSS, SAS, and R, all of which can perform more advanced statistics than can a program you might be familiar with, like Microsoft Excel.

All of these more advanced software packages will require you to write and execute code to varying degrees. Stata is a good choice because its coding language is very succinct, precise, and intuitive, as well as the fact that nearly all commands can be executed by clicking through toolbars (though this is much less efficient!).

The main Stata screen is reproduced below, and is divided up into several main parts:



Stata Graphical User Interface

- Command: input commands here
- Results: Shows the results of your commands
- Review: Lists the history of all commands run
- Variables: Lists and describes the variables in dataset
- There are two ways of running Stata
  - Interactive mode: Enter individual commands and get a response one at a time
  - Batch mode: Write (and run) a `.do` file where Stata will execute all commands and report results in order from the file
- You can write and edit a `.do` file in any text editor (e.g. Notepad)
- An example:

```
clear
*this line defines the semicolon as the line delimiter;
# delimit ;
*set memory for 10 MB;
set memory 10m;
*set working directory*;
cd "/Users/ryansafner/Dropbox/Teaching/Hood College/ECON 480 - Econometrics"
*write results to a log file;
log using example1.log, replace;
*read in raw data;
use "cps87.dat";
label var age "age in years";
label var rac "1=white, non-hisp, 2=place, n.h, 3=hisp";
```

  - Asterisks (*) beginning a line indicate a comment
    - Stata will ignore everything on the line
    - This is for you to comment and explain what your commands are doing

1 Getting Started & Basic Commands

- The first thing you may want to do is `set memory #`
  - `#` represents a max MB (m) or GB (g)
    - e.g. `100m` for 100 MB, `2g` for 2 GB
  - The default maximum memory size (default 1 MB)
  - Useful for large datasets
- If you have existing data open in Stata, you should `clear` to open a new dataset
- Commands can be at most one line long
- For multiple commands (or in particular for `.do` files), you need a delimiter: `#delimit ;`
  - Stata will break up your input into a separate command each time it reaches a `;`
  - If you do this, you must end each line with a `;`!
- On startup, Stata assumes your preferred working directory is `C:/data`, it rarely is though!
  - To change this, use the `cd "/directory/"` (with your actual directory within the quotes) command or go through `File` → `Change Working Directory`
  - All files opened and saved will come from this directory unless otherwise specified
  - For example (Mac):

```
cd "/Users/ryansafner/Dropbox/Teaching/Hood College/ECON 480 - Econometrics"
```

- Get in the habit of logging your work with `log using "filename.log",replace;`
  - This creates a `.log` file that you can examine in any text editor (like Notepad)
  - A `.log` file lists both the commands you input as well as Stata's output for each command
  - `replace` tells Stata to overwrite anything that may be previously on the `.log` file
  - At the end of your session (or `.do` file), you must include `log close`!

- To import an existing dataset
  - Stata data files are `.dat`, open with `use "filename.dta"`
  - For another file type (e.g. an Excel `.csv` or `.xls` spreadsheet) use `insheet using "filename.csv"`
    - Note: Stata will assume that the first row contains the variable names
  - Alternatively, go through `File` → `Open` for `.dat` files or `File` → `Import` for everything else
- To execute an existing `.do` file (that properly specifies the file data is to be drawn from), use `do "filename.do"`or `File` → `Do`
- With an existing dataset loaded in the memory, there are a number of useful operations before we even get to testing anything
- Label variables with `label variable` *`varname`* `"label"`
  - e.g. for an existing variable `age`, `label variable age "Age in Years"`

1.1 Looking at the Data & Summary Statistics

- `describe` *`var1 var2`* (or `desc` for short) will describe the variable(s), telling you the type of data (integer, character, etc) and the label that (hopefully, in a well documented dataset) describes what each variable measures
- `tabulate` *`var1`* (or `tab` for short) will list each value in the variable's distribution
  - Be careful, variables with many observations take a long time to tabulate!
- `summarize` *`var1 var2`* (or `summ` for short) provides summary statistics (number of observations, mean, standard deviation, min, and max) for all variables listed
- Rename a variable with `rename oldname newname`
  - e.g. `realgdp`, `rename rgdp realgdp` renames variable "rgdp" to "realgdp"
- `correlate` *`var1 var2`* (or `corr` for short) will create a table relating the correlation between 2 or more variables
  - e.g. `corr wage educ exper tenure`

1.2 Transforming Variables

- To create a new variable from an existing variable(s), use the `generate` *`newvar formula`* (or `gen` for short) new variables by transforming existing variables
  - *'newvar'* is the name you give to the created variable
  - *'formula'* is the formula for creating the new variable, using existing variables
- Major operators for making new variables on different conditions e.g. `gen var2=5 if var1==2 | var3>=7` makes a new variable var2 if both conditions (var1 is 2 and var 3 is at least 7) are met
  - `if`: make a new variable that equals some value *IF* a condition is met
  - `&`: make a new variable if one condition is met *AND* another condition is met
  - `|`: make a new variable if one condition is met *OR* another condition is met
  - `>`: 'is greater than'
  - `<`: 'is less than'
  - `==`: 'is equal to' (note the two equals signs!)
  - `>=`: 'is greater than or equal to

  - `<=`: 'is less than or equal to'

- Examples:
  - e.g. `gen wagecents=wage*100` multiples wage (in dollars) by 100 (to get in cents)
  - e.g. `gen lngdp=ln(gdp)` takes the natural log of gdp
  - e.g. `gen age2=age*age` squares age
  - e.g. `gen latino=race==3` creates a dummy variable (=1 if latino, else =0) from race, which has multiple categories (e.g. 1 if white, 2 if black, 3 if latino, 4 if asian, etc)
  - e.g. `gen nonwhite=((race==2)|(race==3))` creates a dummy variable from multiple categories of race
  - e.g. `gen femaleowner=((gender==0)&(owner==1))` creates a dummy variable from the union of two categorical variables (gender and owner)
  - e.g. `gen young=(age<=20)` creates a dummy variable that = 1 for those people who's age is less than or equal to 20

- Sometimes you want to drop a variable, use `drop` *`varname`*

- Sometimes you may want to redefine and overwrite a variable that's already defined, use `replace` *`var1`*

1.3 Graphics

- Stata can make graphs by command, or through the `Graphics` menu, allowing you to customize all sorts of things about the graph through either approach
  - e.g. `histogram` *`var`* creates a histogram of the variable *var*
  - e.g. `scatter` *`var1 var 2`* creates a scatterplot of the two variables *var1* and *var2*
    - Scatterplots can include a regression line by appending `|| line` *`predictedvar indepvar`* to your command (where "*predictedvar*" is the variable storing the predicted values (see regression below) and "*indepvar*" is the independent variable
    - Scatterplots can include any ordinary line for reference by its equation. This is particularly useful for residual plots, by including a horizontal line through the origin, by appending e.g. `, yline(0)` to the command

1.4 Regression

- The command for regression is `regress` (`reg` for short), followed by the dependent variables and then any/all independent variable(s)
  - e.g. `reg wage educ` regresses wage (Y) on educ (X)
  - e.g. `reg wage educ exper` regresses wage (Y) on educ (X_1) and exper (X_2)
  - Recall we say that we "regress the dependent variable on the independent variable(s)"

- Stata can create a new variable that gives you the predicted values ($\hat{Y}$) of the dependent variable for each observed value of the independent variable using the `predict` command, followed by the name you wish for the predicted variable, and then `xb` (to tell it to use a linear regression)
  - This must be done *after* you have ran a regression! It will only work for the most recent regression!
  - e.g. `predict wage_hat, xb` stores the predicted values in the new variable "wage_hat" from the recent regression
  - You can also create a variable storing the residuals in the same way, by telling Stata to use "`residuals` instead of `xb` after the comma

    - e.g. `predict res_wage, residuals` stores the residuals in the new variable "res_wage"

- Ensuring you use heteroskedasticity-robust standard errors, simply append `, robust` (or `r` for short) to the end of each regression command
  - e.g. `reg wage educ, r`
- You can make fancy regression tables for word processors (MS Word, LaTeX) using `outreg2` after every regression you run
  - This first requires setting a document to output them all, using the command `using ''filename.extension''`, where you name the file and file extension
  - Run each regression, followed by `outreg2 using ''filename.extension'', append`, which tells Stata to add a new column for each regression to the same document
  - Then open the file and voila!
  - e.g.

```
*Install outreg2 if you don't have it*
ssc install outreg2

*Save in Downloads folder*
cd ~/Downloads/

*Run regression 1*
reg wage educ, r

*Outreg to document called example.doc in Downloads folder*
outreg2 using "example.doc"

*Run regression 2*
reg wage educ exper, r

*Outreg it, be sure to append it*
outreg2 using "example.doc", append

*Repeat for all regressions*
reg wage educ exper tenure, r
outreg2 using "example.doc", append
```

1.5 Statistical Tests

While most test statistics (and confidence intervals) are automatically reported with regression outputs, sometimes it may be useful to do them manually

- **Confidence Intervals**: `ci var, level(C)` constructs a confidence interval of level C for the variable *var*
  - e.g. `ci wage, level(95)` constructs a 95% confidence interval for the variable cars
- **T-test for Sample Means**: `ttest var=H0` runs a t-test for the null hypothesis that a variable equals some value "H0"
  - e.g. `ttest wage=20` tests against $H_0 : wage = 0$
  - Note Stata automatically reports the results of three possible alternative hypotheses: $H_a : mean > H_0$, $H_a : mean \neq H_0$, and $H_a : mean < H_0$!

- **T-test for Sample Proportions**: `prtesti N phat p0` runs a Z-test for the null hypothesis that a population proportion equals some value "p0" for sample size N
  - e.g. `prtesti 500 0.47 .50` tests our sample finding $\hat{p} = 0.47$ against $H_0 : p = 0.55$ with 500 observations
  - Note Stata automatically reports the results of three possible alternative hypotheses: $H_a : p > p_0$, $H_a : p \neq p_0$, and $H_a : p < p_0$!
- **T-test for Difference in Sample Means**: For a two sample t-test, we need `ttest var, by(type) unequal`, where type is a categorical variable that splits the data into different subgroups
  - e.g. `ttest salary, by(league)` tests whether there is a difference in average salary of players between the NFL or AFL (league)
- **F-test for nonlinearity**: `test` *var1 var2* runs an F-test on *'var1'* and *'var2'*
  - `test educ exper` runs an F-test on educ and exper

1.6 An Example `.do` file

The best way to learn Stata is by doing. Run each command and figure out what Stata is doing. Then change the command slightly (or look at different data) to see what changes. For a while, you will find yourself copy-pasting code and changing it to suit your needs. Eventually, you will have enough experience to remember the main commands on your own without having to always double check. This is how you learn code–it is learning a new language.

This code uses the WAGE1.DTA dataset you can find on Blackboard.

```
clear

set mem 10M

*Change directory -- I set this as Downloads, you can set it wherever you wish*
cd ~/Downloads/

*Create a log file, I'll call it 'doexample.do'*
log using doexample.log

*Load in Data -- if it not in the same folder, you may have to do this manually
*Or it may make sense to change your working directory (above) using 'cd' to
*the folder where the data is stored
use WAGE1.DTA

*What is the variable wages? (desc for short)*
describe wage

*Summary Statistics for wages (summ for short)*
summarize wage

*Get more detail*
summarize wage, detail

*You can export summary statistics in a nice neat table in MS Word
*This is a process of 3 commands:
*1. estpost (to collect the data
estpost sum wage educ exper
*notice it shows a lot of different summary statistics
```

```
*2. estesto (to store this information and give it a name)
*call it 'wagedescrip'
eststo descrip
*3. estout (to publish it)
*publish to a word doc called 'est'
*and pick which summary statistics to include in the cells() command
estout using est.doc, cells("count mean sd min max") label
*Note you can export it to a different folder in quotes, such as
*estout using "/statistics/est.doc"
*Note also you can just look at it in Stata by removing the 'using est.doc' portion

*Graph a box plot of wages*
Graph box wage

*Create separate boxplots by years of education*
Graph box wage, over(educ)

*Plot a histogram of wages (hist for short), showing percentages instead of counts*
histogram wage, percent

*Look at distribution of education (tab for short)*
tabulate educ

*Plot a histogram of educ, showing percentages instead of counts*
histogram educ, percent

*Plot a histogram of educ*
histogram educ, percent

*Find correlation between wages and education*
corr wage educ

*Create scatterplot of wage and educ*
scatter wage educ

*Regress wages on education (reg for short)
regress wage educ

*Create a variable for the predicted values*
predict wage_hat, xb

*Create a variable for the residuals of model*
predict res_wage, residuals

*Look at OLS regression line on scatterplot
*Note || means "also do this"*
scatter wage educ || line wage_hat educ

*Look at residual plot
*also plot a horizontal line at y=0 to compare residuals to
scatter res_wage educ, yline(0)

*Use Robust Standard errors in our regression (r for short)*
reg wage educ, robust
```

```
84
85 *Export regression output to a MS Word document
86 *I'm calling it 'example'
87 *Once we are done with our regs in Stata, open this document in MS Word*
88 outreg2 using example.doc
89
90 *Change Wage from Dollars to Cents by generating (gen for short) a new variable*
91 generate wagecents = wage*100
92
93 *Label this new variable 'wage in cents'
94 label variable wagecents "wage in cents"
95
96 *Double check it worked
97 desc wagecents
98
99 *Regress Wage in Cents on Education with robust SEs*
100 reg wagecents educ,r
101
102 *Add this regression to our Word output
103 *append tells Stata to add another column
104 outreg2 using example.doc, append
105
106 *Let's look at a regression of wage and experience and add it to our regression output
107 reg wage exper, r
108 outreg2 using example.doc, append
109
110 *Create a dummy variable for college grads (=1 if educ is greater than or equal to 16, else=0)
111 gen Collegegrad=(educ>=16)
112 label variable Collegegrad "=1 if Graduated College"
113
114 *Let's go overboard and generate new dummy variables for every level of schooling*
115 tab educ, gen(educ_)
116
117 *Create a dummy variable for new hires (=1 if tenure is 0 (note the double ==!, else=0)*
118 gen newhire=(tenure==0)
119 label variable newhire "=1 if New Hire"
120
121 *Be sure to close the log when you're done!*
122 log close
```