AFRE 835: Applied Econometrics

Appendix 2A: Stata

Spring 2017

Outline

Introduction

- Stata is a statistical package providing
  - Excellent data management capabilities
  - An extensive set of built-in statistical techniques
  - Excellent graphical capabilities
- While Stata also has an accompanying matrix programming capability (Mata), its comparative advantage lies in its command drive features.
- The purpose of this appendix is to provide a *brief* overview of Stata.
- It is not a full fledged tutorial.
- The best way to learn Stata is to jump in and use Stata in conjunction with one of the many available on-line tutorials or sample data sets.

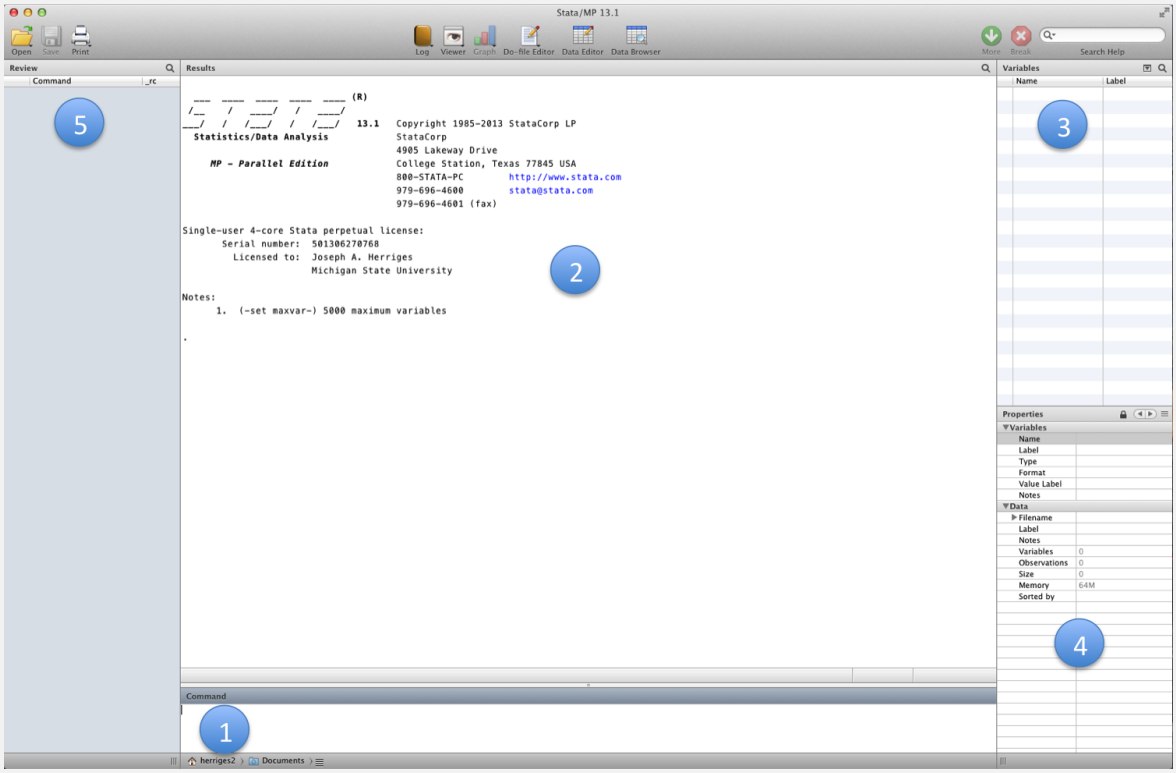
Resources

- Professor Songqin Jin provides a useful walkthrough the various features of Stata.
- There are numerous tutorials on the web for Stata, including
  - UCLA's webpage: http://www.ats.ucla.edu/stat/stata/
  - Princeton's webpage (Germán Rodríguez): http://data.princeton.edu/stata/
  - YouTube videos
- Documentation provided by Stata, including
  - Video tutorials in YouTube, available at http://www.stata.com/links/video-tutorials/
  - Manuals (particularly "Getting Started with Stata for..." series) available at: http://www.stata.com/support/documentation/
- Stata has an extensive interactive help feature.
- Wikibook for stata at: http://en.wikibooks.org/wiki/Stata

Accessing Stata

- Stata-14 is installed in the AFRE graduate student computer lab;
- Stata12-SE is available in all Windows 7 student computer labs on campus;
- Stata-13 is available on the HPCC system
- Student versions (6-month, 1 year, and perpetual license) are available (at a price) from Stata.

Stata 13 Interface (Stata 14 is the same)



The Interface Components correspond to:

1. The Command Window, where interactive commands are issued;
2. The Results Window, where output from issued commands appear;
3. The Variables Window, which lists the variables currently in memory;
4. The Properties Window, which lists the characteristics of a currently highlighted variable;
5. The Review Window, which provides a history of commands issued during the current session.

Modes of Operation

- There are three basic ways to use Stata
  1. Typing Commands (interactive)
  2. Menu option (interactive)
  3. Do files (batch)
- The menu options can be a useful way to learn syntax, but ...
- In the long run you will want to create ".do" files to store your programs.

Data in Stata

- Data in Stata is organized much like an Excel spreadsheet, with
  - *observations* corresponding to rows
  - *variables* corresponding to columns
- Stata (including variable names) is case-sensitive.
- Variables can take on one of 6 types

type	Minimum	Maximum	bytes
byte	-127	100	1
int	-32,767	32,740	2
long	-2,147,483,647	2,147,483,620	4
float	$-1.70141173319 10^{38}$	$1.70141173319 10^{38}$	4
double	$-8.9884656743*10^{307}$	$8.9884656743*10^{307}$	8
string	NA	NA	NA

Missing Data

- Missing data for a variable is denoted by "." for numeric data and by "" for string variables.
- Missing data are numerically equivalent to $+\infty$ (This can be important when forming variables from existing variables);
- The command *missing(x)* returns value of 1 (true) for observations for which *x* is missing; and a value of 0 (false) otherwise.

Loading Data

- There are a number of ways to load data into Stata, including
  1. Using a Stata dataset (.dta): *use*
  2. Using a delimited data set (.cvs, .xls): *insheet*
  3. Using a fixed width data set (.txt): *infile*
  4. Copying and pasting from an Excel spreadsheet
  5. Directly loading using Data Editor
- Stata comes with a number of example datasets; e.g.,
  1. auto
  2. lifeexp

  These are accessed using *sysuse* or *File/Example Datasets...*
- There are also a series of datasets associated with the manuals (accessible using menu: *File/Example Datasets...*).
- Data can be viewed using either the DataEditor or DataBrowser windows.
  - As a general rule, you should avoid using DataEditor to avoid inadvertently changing your data.
  - Stata has a number of safeguards against changing your data.

Command Syntax

- The most common command syntax takes the form:

  command [*varlist*] [*if*] [*in*], [*option*]

  where
  - *command* is the name of the command;
  - *varlist* is the optional list of variables the command will use;
  - *if* is the optional list of restrictions to the sample over which the command will operate;
  - *in* is the optional restriction of records over which the sample will operate;
  - *option* is the optional list of options used with the specific command in question.
- Note: The brackets are not used, but simply indicate the item is typically optional.
- There are additional syntax items for specific commands; e.g., [*weight*] providing for the weighting of the data.

Syntax Examples

- *list*
  simply lists (in the Output window) the entire current sample;
- *list in 1/10*
  lists records 1 through 10;
- *list make price if price>10000*
  lists the variables *make* and *price* for those records with *price* greater than 10000.
- *summarize mpg*
  provides summary statistics for the variable mpg;
- *summarize mpg, detail*
  provides detailed summary statistics for the variable mpg;
- *summarize mpg in 1/10 if price>10000*
  provides summary statistics for the variable mpg for records 1 through 10 that have a price $>$ 10000;

Key Commands

General Type	Commands
Getting help	help
Operating system interface	pwd, cd, sysdir, dir
Using and saving data from disk	use, sysuse, save, append, merge, compress, clear, preserve, restore
Inputting data into Stata	input, edit, infile, infix, insheet
Basic data reporting	describe, codebook, list, browse, count, summarize, table, tabulate, correlate
Data manipulation	generate, replace, egen, rename, drop, keep, sort, order, by, reshape, collapse, encode, decode, missing
Formatting	format, label
Keeping track of your work	log, notes
Convenience	display, quietly

Operator and Expressions

- Operators: +,-,*,/,^
- Logical Operators:
  - & and
  - | or
  - ! not
  - > greater than
  - < less than
  - >= greater than or equal to
  - <= less than or equal to
  - == equal to
  - ! == not equal to
- Functions: *abs(x)*, *ln(x)*,*sqrt(x)*,*floor(x)*,*round(x)*, *int(x)*, *ceil(x)*

Do Files

- Do files consist of a list of commands (saved as *filename.do*);
- The advantage of .do files is that they allow you to replicate your results;
- .do files can call other .do files;
- .do files are created by clicking on the pen/paper icon on the top bar of Stata;
- .do files are executed either by
  - Clicking on the "Do" icon in the top right of the editor screen
  - Or using the keyboard shortcut (ctrl-D in Windows or Command + Shift + D on a Mac)
- If you highlight a portion of your code prior to execution, only that portion is executed.
- During debugging, you can also create a breakpoint in your code by adding an *exit* command.

Comments

- You should get in the habit of adding comments to your code;
- The following types of comments are allowed:
  - To comment out a line:
    ** comment*
  - To comment out everything in between:
    / comment */*
  - To comment out the rest of a line
    //

Delimiter

- The default delimiter in Stata is the carriage return
- With this delimiter, long lines can be broken up into multiple lines using *///*.
- Alternatively, you can change the delimiter to a semi-colon using *#delimit ;*

Log File

- The log file stores a printable copy of the Results.
- To open a log file, type log using <file name.log>, replace (or append) in the program window.
  - Replace replaces the previous log file with the same name.
  - Append adds the new results to the previous log file with the same name. An affix ".log" is needed otherwise the log file will be saved with an affix ".smcl" which is only readable by STATA while the ".log" file can be opened in Microsoft word or any text editor programs.
  - Adding capture prior to the log file command will avoid the program stopping in case the log file does not already exist.
- If you want to view the log window while in Stata, type view <log file name> in the command window.
- To close the log file, type log close in the command window.

Useful Starting Code

```
*******************************************************************************
*                                                                             *
*       Starting Code for Stata Overview Session                              *
*       Writen by: Joe Herriges                                               *
*       Last Update: 9/9/2016                                                 *
*                                                                             *
*******************************************************************************
#delimit ;
set     more off;
set     matsize 800;
capture log close;
clear   all;
global  startsrc
        = "/Users/herriges/Dropbox/classes/EC 823/Lectures Fall 2016/Stata/";
global  csource = "$startsrc/code";
global  dsource = "$startsrc/data";
global  osource = "$startsrc/output";
cd      "$csource";
log     using Starting_code.log, replace;
local   in_data = "$dsource/cattaneo2";
local   out_data = "$osource/Starting_Data_out";
use     "`in_data'";
exit;
*******************************************************************************
*                                                                             *
*       Add code here                                                         *
*                                                                             *
*******************************************************************************;
```

Creating and Saving Variables

- The basic command for creating a new variable is *generate*; e.g., *generate x = z + 4;*
- Once a variable has been created (loaded into memory or generate), you cannot simply change it using the generate command;
- Instead, you must replace it; e.g., *replace x = y + 4;*
- You can also do replaces conditionally; e.g., *replace x = y + 4 in 1/20 if y=2;*
- Note: Any changes you make are to the version of the file in memory unless you explicitly tell Stata to *save* the file in memory.

Generating Lags and Leads

- To generate lags values of a variable, one would type

```
sort time
gen laginc=income[_n-1]
```

- The sort command sorts the data by whatever time variable you have for your data, while the second generates lag values.
- To generate lagged variable by group (identified by groupid), one would type

```
sort groupid time
by groupid: gen laginc=income[_n-1]
```

- Leads can also be created.

The egen Command

- egen stands for extended generation and is a useful tool for a lot of specialized situations.
- Examples:
  - To generate the mean of a variable (vname) and store the mean for all observations one would enter:
    egen mvar=mean(varname).
  - To generate the mean of a variable (vname) by group (grpid) and store these means for each group one would enter:
    egen grpmvar=mean(varname), by (grpid).
  - To generate the standard deviation of a variable (vname) and store the standard deviation for all observations one would enter:
    egen sdvar=sd(varname).

Simple Tests and Correlations

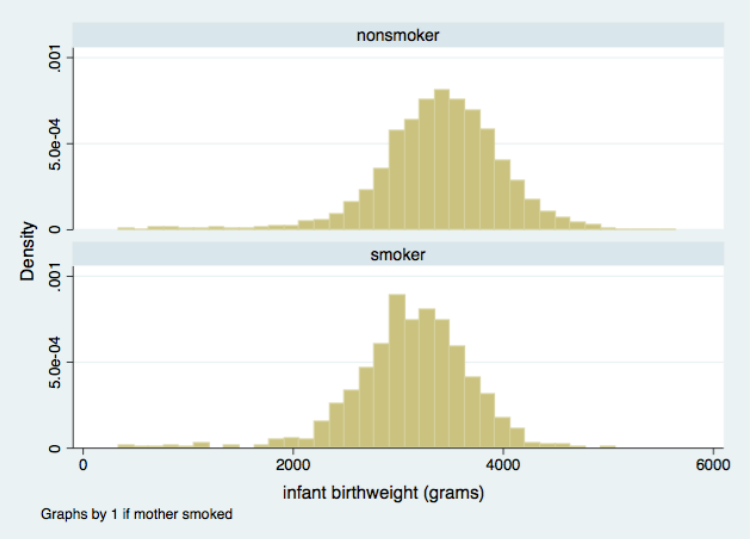
- Test for sample means (by default, 95% confidence interval is
  - One-sample mean-comparison test, type
    ttest varname==value, level(#)
    where # is between 10 and 99.99 inclusive. For example, type:
    ttest ynet==9750, level(99)
  - Two-sample mean-comparison test, type
    ttest var1==var2
    For example, to test whether two variables have the same mean:
    ttest ynetpc==xconspc
  - Two-group mean-comparison test, type
    ttest varname, by(group ID)
    For example, to test whether net income differs by gender type:
    ttest netinc, by(gender)
- To calculate the correlation coefficients between variables type
  pwcorr varlist

Data Managements

- Stata has excellent data documentation capabilities;
- Stata allows you to label variables and to include notes regarding your data sources;
- One can also label values of a categorical variable;
- These labels (e.g., "1=Yes"; "2=No") do not change the value of the variable, but simple make it easier read tables, graphs, etc. that use these variables.
- Missing values can be further categorized to distinguish different types or sources of "missingness."

Graphics

- Stata also excellent graphic capabilities;
- Particularly useful is the simple histogram command;
- You can also stack histograms using, for example, histogram bweight, by(mbsmoke, col(1))



Loops

- Loops are very easy to implement in Stata
- There are basically three types of loops:
  1. forvalues;
  2. foreach;
  3. while

Forvalues

- Used to loop over numeric values
- Example:
  forvalues x=1/10 {
  generate var'x' = sqrt('x');
  }
- Numeric values can be specified in a variety of ways
  - *1/5* = 1,2,3,4,5;
  - *10(5)25* = 10, 15, 20, 25;

Foreach

- A more general looping structure;
- Example
  foreach x of varlist var1-var4 {
  summarize 'x');
  }
- Or
  foreach ii in "var1" "var2" {
  qui replace 'ii' = . if 'ii'==-9;
  tab "ii";
  }

While Loops

- Keeps looping until a specific condition is met;
- Example:
  local x = 1;
  while x < 10 {
  local x = x+2;
  display 'x';
  }