SOC 7717 Event History Analysis and Sequence Analysis

Week 1 Lecture 2: Introduction to Stata

Wen Fan

Spring 2019

Outline

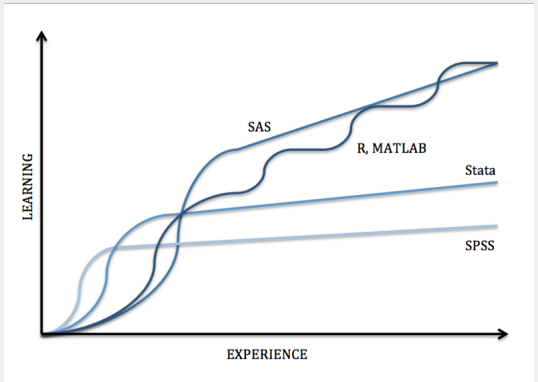
- Stata in perspective
- Lay of the land
- Stata language
- Opening a data set
- Some basic (but important) commands
- OLS regressions and model presentations
- Getting help if you're stuck
- Exercise

Stata in perspective

What's the alternative?

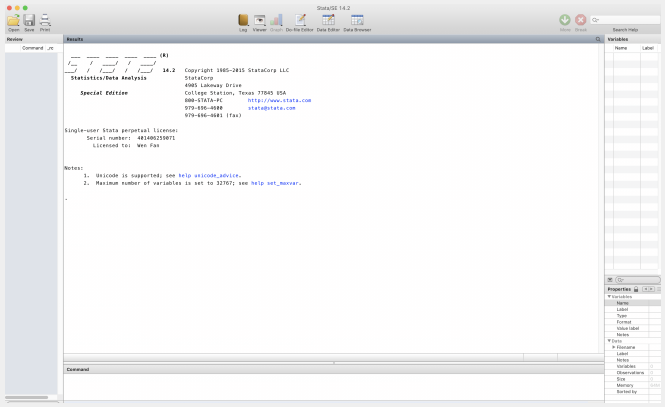
	Stata	SPSS	SAS	R
Learning Curve	Steep/gradual	Gradual/flat	Pretty steep	Pretty steep
User interface	Programming/point-and-click	Mostly point-and-click	Programming	Programming
Data manipulation	Very strong	Moderate	Very strong	Very strong
Data analysis	Powerful	Powerful	Powerful/versatile	Powerful/versatile
Graphics	Very good	Very good	Good	Excellent
Cost	Affordable (perpetual licenses, renew only when upgrade)	Expensive (but not need to renew until upgrade, long term licenses)	Expensive (yearly renewal)	Open source

Learning curve

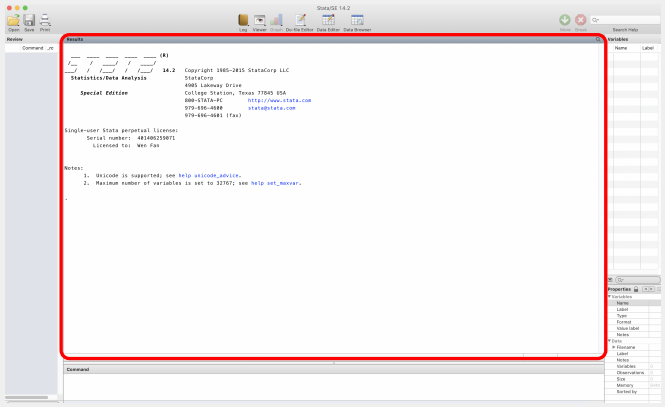


Lay of the land

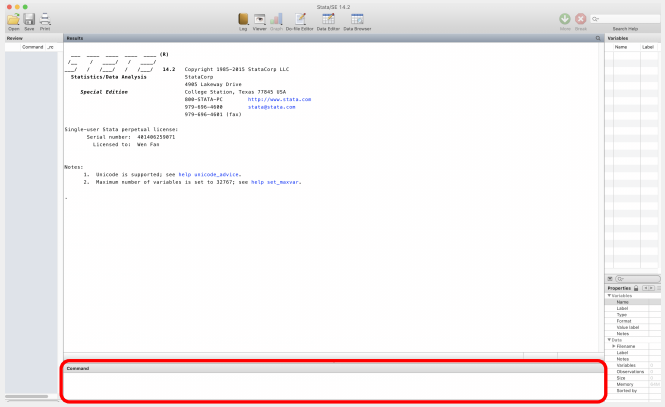
Lay of the land



Results window

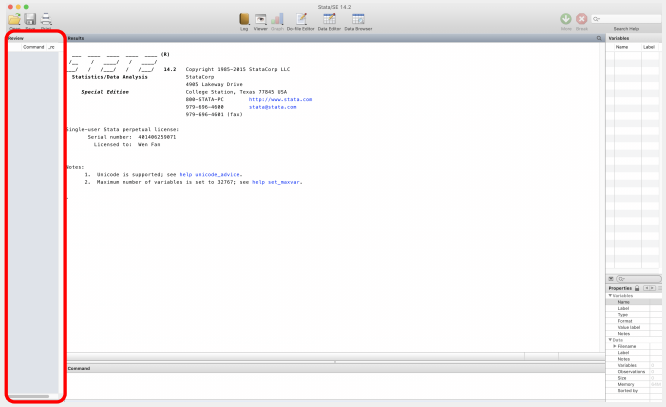


Command window



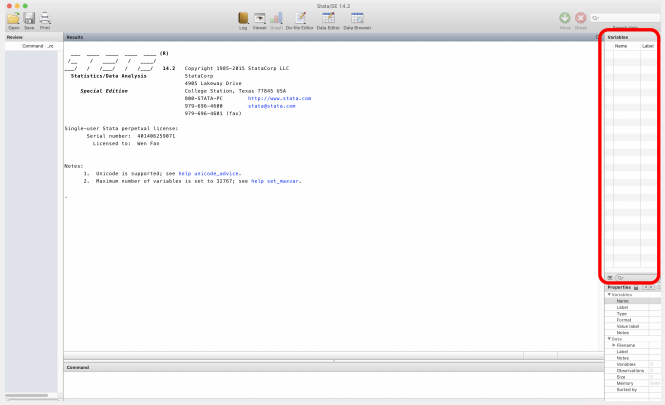
7

Review window



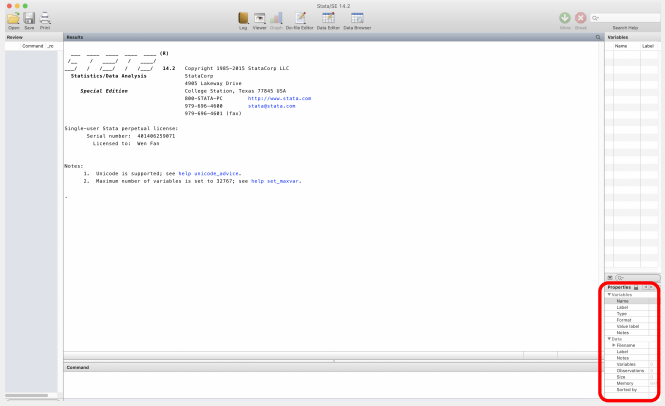
8

Variables window



9

Properties window

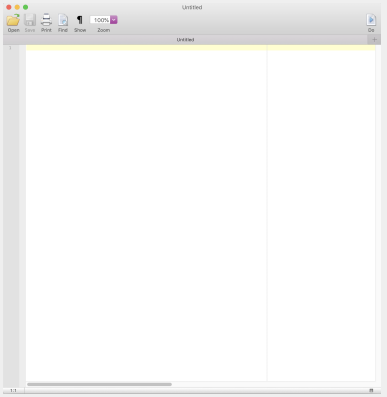


10

Three ways to execute commands

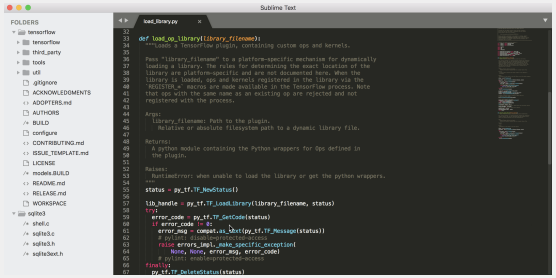
- Drop down
  - Helpful for complex commands such as creating graphs
- Command line
  - Useful for trying new things and exploring data
- Do-files
  - Text files that contain your commands
  - You can send one command at a time, or you can run a do-file and Stata will run down through the list in order, doing each thing you've asked and ignoring `*` and `//` lines
  - Advantages: You have a record of the commands you ran; you can use the features of the text editor

.do files (from Stata's own editor)



.do files

- You can also integrate Stata with an external editor (`http://fmwww.bc.edu/repec/bocode/t/textEditors.html`)
- E.g., Sublime Text



Stata language

General notes

- Usually in Stata, there are multiple ways of doing the same thing
- I'm showing you one way
- You might know another way, discover a shortcut, write your own code, or need to use another way

Stata language

- Case sensitive
- Missing
  - `.` means "missing"
  - You can also specify `.a` through `.z` to make different kinds of missing values (e.g., refused, not applicable). Note that `.` < `.a` < `.b` < ... < `.z`
  - The `.`-series is always the largest value in the dataset

Stata language

Type ...	When you want to say ...
`==`	Equals
`!=`	Does not equal
`&`	And
`|`	Or
`<`	Less than
`>`	Greater than

Stata language

- Stata command syntax:
  - Action (Verb)
  - On which variables or files (Subject)
  - Qualifiers (if and in)
  - Special options (Adjectives/Adverbs)

Opening a data set

Reading data into Stata

```
38  *** 1. open data
39
40  cd "N:\ENTER FULL PATH NAME HERE"
41
42  /* The 'cd' command tells Stata what directory you are working in.  The name of
43     the current working directory is listed beneath the variable window.  You can
44     display the current working directory by entering the 'pwd' command. */
45
46  use GSS2014.dta, clear
47
48  /* 'use' tells Stata what file to open.  Notice that if we had not named a
49     working directory earlier, we'd have to specify the full path name */
```

Specify the directory where your data are housed.
To get the current working directory, type `pwd`.

Reading data into Stata

```
38  *** 1. open data
39
40  cd "N:\ENTER FULL PATH NAME HERE"
41
42  /* The 'cd' command tells Stata what directory you are working in.  The name of
43     the current working directory is listed beneath the variable window.  You can
44     display the current working directory by entering the 'pwd' command. */
45
46  use GSS2014.dta, clear
47
48  /* 'use' tells Stata what file to open.  Notice that if we had not named a
49     working directory earlier, we'd have to specify the full path name */
```

Give Stata the name of the data file that you want to read into memory.

Reading data into Stata

```
38  *** 1. open data
39
40  cd "N:\ENTER FULL PATH NAME HERE"
41
42  /* The 'cd' command tells Stata what directory you are working in.  The name of
43     the current working directory is listed beneath the variable window.  You can
44     display the current working directory by entering the 'pwd' command. */
45
46  use GSS2014.dta, clear
47
48  /* 'use' tells Stata what file to open.  Notice that if we had not named a
49     working directory earlier, we'd have to specify the full path name */
```

Leave yourself (lots and lots and lots of) notes.
Do so by enclosing your text between `/*` and `*/`.

Comments

- Begin the line with `*`
- Begin the comment with `//`
- Place the comment between `/*` and `*/` delimiters
- Anything from `///` to the end of the line is treated as a comment $\longrightarrow$ a good way to deal with long lines (i.e., as a line break tool)
- Use those lines for comments or other annotation to improve the record

Recommended do-file structure

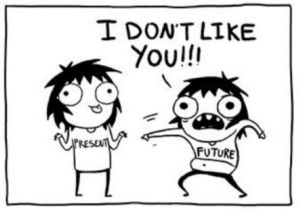
```
// program: _name_.do
// task:
// project:
// author: _who_ / _date_

*** 0. program and data setup
version 14.1
clear all
set linesize 80
macro drop _all

cd "ENTER WORKING DIRECTORY HERE"
capture log close
log using _name_, replace text
use "DATA.dta", clear

*** 1
// Description of task 1
*** 2
// Description of task 2

log close
exit
```

Always begin with a note to yourself about what's in the .do file, etc.

Tell Stata to keep a running log of what you do.

Log files

- A log file is a text file that documents the results window of your Stata session
- That way, when future-you can't remember what Spring-2019-you did, future-you can check the record, instead of ...



Other ways to read in data...

- `infile`: reads in a data set that is not in Stata format. First it reads a dictionary that describes the format of the data, then it inputs the raw data
  - » Example: `infile using dfile`
  - » Example: `infile using dfile, using(data)`
- `insheet`: reads in ASCII (text) data created by a spreadsheet
  - » Example: `insheet using filename`

Some basic (but important) commands

Getting familiar with your data

- `describe`: provides a summary of the data that are currently in memory
  - » `describe, short`
- `codebook`: produces a codebook that describes the contents of the data or a particular variable
- `notes`: attaches a note to the data file
- `summarize`: produces basic summary statistics
- `tabulate`: creates one-way frequency tables and two-way cross-classifications

Getting familiar with your data

- `histogram`: creates histograms
- `kdensity`: generates kernel density estimates
- `graph box`: produces box plots

Variable manipulation

- `generate`: generates a new variable
- `recode`: alters the values of a variable according to the rules you specify
  - Make sure not to overwrite the original variable!
- `label variable`: provides a label for the new variable
- `label define` and `label values`: assign value labels to the new variable
- `tabulate varname, generate(prefix)`: converts a categorical variable into dummy variables

Correlations and *t*-test

- Correlactions
  - `correlate varlist`: two continuous variables
  - `spearman varlist`: two ordinal variables
  - `tabulate var1 var2, chi2`: two nominal variables
- `ttest onevar, by(othervar)`: *t*-test

egen

- The `egen` command offers a number of useful extensions, e.g.,
  - `egen avgpinc = mean(pinc)`
- `egenmore`
  - `ssc install egenmore`
  - `findit egenmore`

OLS regressions and model presentations

OLS regressions

- Let's first run a simple OLS regression: `regress pinc age`
- Save the estimates as model 1:
  `estimates store m1, title(Model 1)`
- Now let's add a control for race: `regress pinc age i.race`
- Save the estimates as model 2:
  `estimates store m2, title(Model 2)`

Model presentations

Using the `esttab` command, we can compare results from our two models:

- `esttab m1 m2, b(%10.4f) se scalars(N r2 F) mtitles`

Or, if you want tables presented in spreadsheets:

- CSV file: `esttab m1 m2 using "NAME.csv", b(%10.4f) se scalars(N r2 F ll) mtitles`
- XML file: `outreg2 using "NAME", excel bdec(3) alpha(0.001, 0.01, 0.05) symbol(***, **, *)`

Getting help if you're stuck

What if you're stuck?

- `help`
  - Gives you the basic format of the command
  - Gives you a list and description of available options
  - Provides example code
  - Gives you a list of other commands that you may (or may not) be interested in
- Stata's internal pdf documentation

What if you're still stuck?

- Google
- UCLA Academic Technology Services (`https://stats.idre.ucla.edu/stata/`)
- The Statalist (`https://www.statalist.org/forums/`)
- Ask me!

Exercise

Exercise

Use the marriage duration data set, `duration.dta`, to do the following tasks:

- How many observations are there?
- How many men are there in the data? (# and % of all persons)
- How many women are married? (# and % of all women)
- What does the observed distribution of marriage durations look like? E.g. min, max, median, mean; general shape, etc.
- Create a new (fictional) variable `age`. For each person, we'll set it equal to their observation number (first sort observations in ascending order by `time`) plus 30. For the sake of practice, we'll assume that any ages greater than 50 are missing.

Exercise (Cont.)

- How many cases are missing a value for `age`?
- Create a new age variable with values equal to the square of each person's age.
- Create a new age variable with values equal to the natural logarithm of each person's age.
- Create a new categorical variable summarizing age in 5-year bands (30–34, 35–39, 40–44, 45–49, 50+). Give the new variable a useful variable name and label its values.