Stata Brush Up Course

Session 1

Sergi Quintana

IDEA

September 2nd, 2024

Plan for the course

We will see:

- Stata syntaxt and functioning.
- Data manipulation and visualization.
- Estimation and post-estimation.
- Programming with Stata.

At the end of the course there will be an exam and also a small project.

Project Instructions

By Friday 6th you will have to do a small project in which using the data set of your choice:

- 1. Produce summary statistics of the data.
- 2. Identify interesting facts of your data and describe them. Here you can go as far as you want, but I suggest you present the results with graphs and regression outputs.

You will have to send me by email a pdf file with the answers to both questions and the do file and data set used to achieve the results.

Plan for Today

- What is Stata?
- Resources.
- Elements and file extensions.
- Memory, maxvar and break.
- Syntaxt.
- Variables and data types.
- Managing packages.
- Initial commands.
- Calling do-files and errors.

What is Stata?

- Stata is a statistical package for managing, analyzing, and graphing data.
- Stata may be used either as a point-and-click application or as a command-driven package.
- Stata is fast. That speed is due partly to careful programming, and partly because Stata keeps the data in memory. Stata's file model is that of a word processor: a dataset may exist on disk, but the dataset in memory is a copy.
- Having the data in memory means that the dataset size is limited by the amount of computer memory. Stata stores the data in memory in an efficient format—you will be surprised how much data can fit.

Why Stata?

Features	SPSS	SAS	Stata	JMP (SAS)	R	Python (Pandas)
Learning curve	Gradual	Pretty steep	Gradual	Gradual	Pretty steep	Steep
User interface	Point-and-click	Programming	Programming/ point-and-click	Point-and-click	Programming	Programming
Data manipulation	Strong	Very strong	Strong	Strong	Very strong	Strong
Data analysis	Very strong	Very strong	Very strong	Strong	Very strong	Strong
Graphics	Good	Good	Very good	Very good	Excellent	Good
Cost	Expensive (perpetual, cost only with new version). Student disc.	Expensive (yearly renewal) Free student version, 2014	Affordable (perpetual, cost only with new version). Student disc.	Expensive (yearly renewal) Student disc.	Open source (free)	Open source (free)
Released	1968	1972	1985	1989	1995	2008

Why Stata?

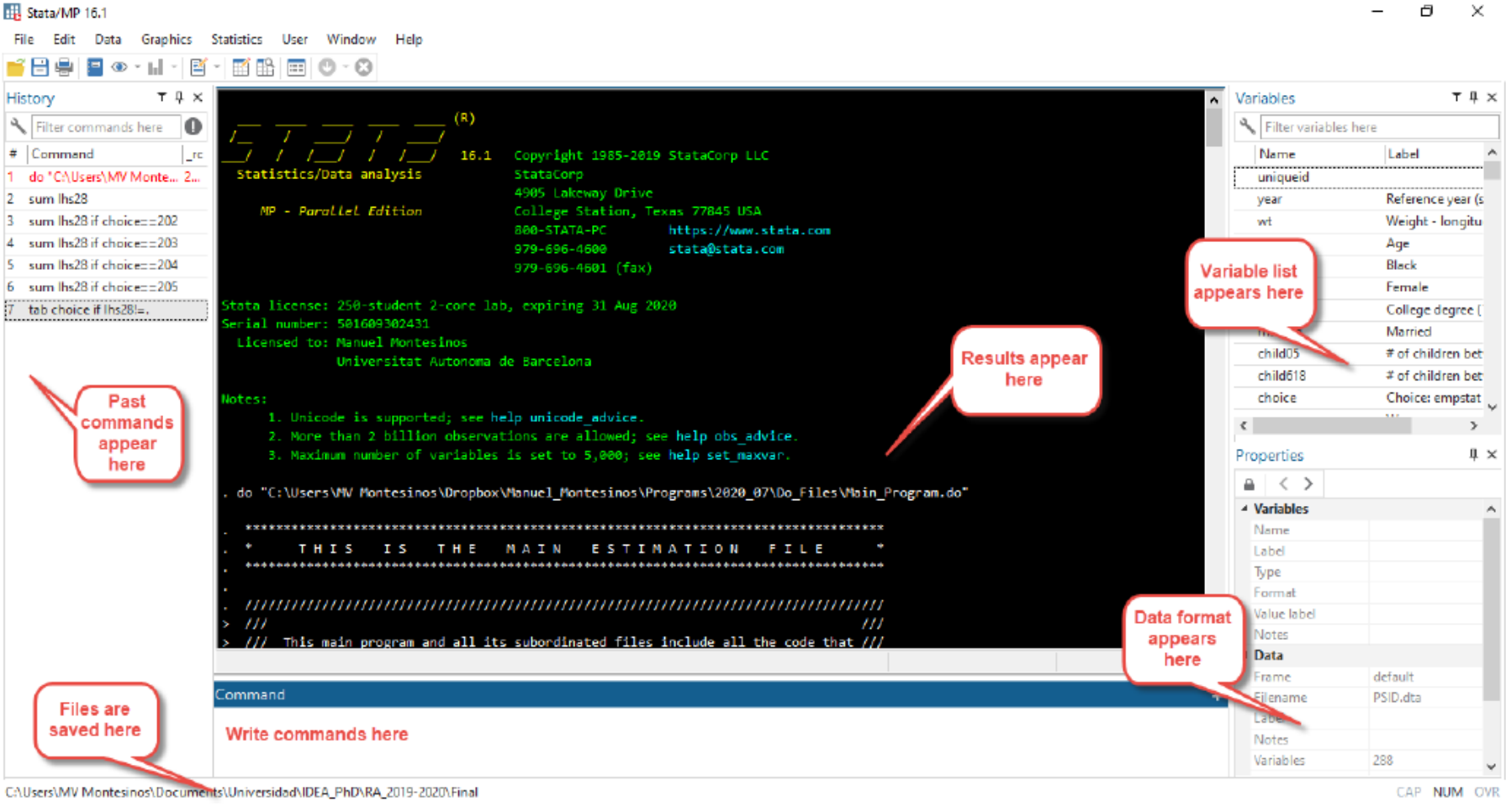
Product features	Stata/IC	Stata/SE	Stata/MP		
			2-core	4-core	6+
Maximum number of variables					
Up to 2,048 variables	✓	✓	✓	✓	✓
Up to 32,767 variables	-	✓	✓	✓	✓
Up to 120,000 variables	-	-	✓	✓	✓
Maximum number of observations					
Up to 2.14 billion	✓	✓	✓	✓	✓
Up to 20 billion	-	-	✓	✓	✓
Runs most estimation commands...					
Fast	✓	✓	✓	✓	✓
Twice as fast as Fast	-	-	✓	✓	✓
Almost four times as fast as Fast	-	-	-	✓	✓
Even faster	-	-	-	-	✓

Resources

There are many helpful materials for Stata user's:

- Stata Manual. https://www.stata.com/manuals/u.pdf
- Help command.
- Stata forum. https://www.statalist.org/forums/
- https://www.stata.com/links/resources-for-learning-stata/

Elements of Stata



Elements of Stata

Stata is a command-line driven package. You can enter commands in either of 3 ways:

- Interactively: click through the menu on top of the screen.
- Manually: type the first command in the command window and execute it, then the next, and so on.
- Do-file: type up a list of commands in a do-file, an ASCII file that collects commands to be executed sequentially, and run it.

We will be always using do-files.

File Extensions

Relevant file extensions are:

- .dta: data file
- .do: do file
- .log: Stata output
- .gph: graphic file

Managing memory size

Stata stores data in memory.

The **compress** command reduces the amount of memory requiredto store the data without loss of precision or any other disadvantages.

Compress works by examining the values you have stored and changing the data types of variables when that can be done without loss of precision.

Typing compress every so often is a good idea.

Managing Memory

The memory command will show you the major components of Stata's memory footprint.

```
. use https://www.stata-press.com/data/r18/regsmpl
(NLS women 14-26 in 1968)
. memory
Memory usage
```

	Used	Allocated
Data	856,020	67,108,864
strLs	0	0
Data & strLs	856,020	67,108,864
Data & strLs	856,020	67,108,864
Variable names, %fmts, ...	4,644	191,927
Overhead	1,081,344	1,081,744
Stata matrices	0	0
ado-files	34,505	34,505
Stored results	0	0
Mata matrices	0	0
Mata functions	0	0
set maxvar usage	5,281,738	5,281,738
Other	2,841	2,841
Total	7,251,976	73,701,619

Maxvar and the More Condition

Those are two important settings.

- Stata has a default maximum amount of variables.
- If you need more you can use maxvar.
- The allowed maximum will depend on your Stata version.

By default, Stata does not pause its output.

- Use set more off or on to pause or not the output.

The Break Key

When you want to make Stata stop what it is doing and return to the Stata dot prompt, you click on Break

Stata for Windows:	click on the **Break** button (it is the button with the big red X), or press *Ctrl+Pause/Break*
Stata for Mac:	click on the **Break** button or press *Command+.* (period)
Stata for Unix(GUI):	click on the **Break** button or press *Ctrl+k*
Stata for Unix(console):	press *Ctrl+c* or press *q*

Syntaxt

With few exceptions, the basic Stata language syntax is

[by *varlist*:] *command* [*varlist*] [=*exp*] [if *exp*] [in *range*] [*weight*] [, *options*]

- square brackets distinguish optional qualifiers and options from required ones.
- varlist denotes a list of variable names
- command denotes a Stata command
- exp denotes an algebraic
- range denotes an observation range
- weight denotes a weighting expression
- options denotes a list of options

Syntaxt

Some features are:

- Most commands that take a subsequent varlist do not require that you explicitly type one.
- If no varlist appears, these commands assume a varlist of all.
- In commands that alter or destroy data, Stata requires that the varlist be specified explicitly.

Some commands take a varname, rather than a varlist. A varname refers to exactly one variable.

Syntaxt

by varlist:

- The by varlist: prefix causes Stata to repeat a command for each subset of the data for which the values of the variables in varlist are equal.
- The result of the command will be the same as if you had formed separate datasets for each group of observations

Syntaxt

if exp

- The if exp qualifier restricts the scope of a command to those observations for which the value of the expression is true.
- An example is:

  summarize marriage_rate divorce_rate if region==”West”
- The double equal sign in region==”West” is not an error.

  Stata uses a double equal sign to denote equality testing and one equal sign to denote assignment

Syntaxt

in range

- The in range qualifier restricts the scope of the command to a specific observation range.
- A range specification takes the form #1 /#2, where #1 and #2 are positive or negative integers.
- Negative integers are understood to mean “from the end of the data”, with -1 referring to the last observation.
- The implied first observation must be less than or equal to the implied last observation.

Syntaxt

options

- Many commands take command-specific options.
- These are described along with each command in the Reference manuals.
- Options are indicated by typing a comma at the end of the command, followed by the options you want to use.

options	Description
Main	
`detail`	display additional statistics
`meanonly`	suppress the display; calculate only the mean; programmer's option
`format`	use variable's display format
`separator(#)`	draw separator line after every # variables; default is `separator(5)`

Other information

Some important features of stata commands are the abbreviation rules and the quietly prefix.

Abbreviation rules: Stata allows abbreviations of commands and variables.

options	Description
Main	
`detail`	display additional statistics
`meanonly`	suppress the display; calculate only the mean; programmer's option
`format`	use variable's display format
`separator(#)`	draw separator line after every # variables; default is `separator(5)`

Quietly prefix: We will use it when we want the output of a command not to be displayed in the output window.

Variables

List of existing variables

- Variable names can be repeated.

  If you type list state mrgrate dvcrate state, the variable state will be listed twice, once in the leftmost column and again in the rightmost column of the list.
- if you suffix * to a partial variable name (for example, sta*), you are referring to all variable names that start with that letter combination.
- If you prefix * to a letter combination (for example, *rate), you are referring to all variables that end in that letter combination.
- If you put * in the middle (for example, m*rate), you are referring to all variables that begin and end with the specified letters.

Variables

Factor variables

Factor variables are extensions of varlists of existing variables. When a command allows factor variables, in addition to typing variable names from your data, you can type factor variables, which might look like

`i.`*varname*

`i.`*varname*`#i.`*varname*

`i.`*varname*`#i.`*varname*`#i.`*varname*

`i.`*varname*`##i.`*varname*

`i.`*varname*`##i.`*varname*`##i.`*varname*

Variables

Factor variables

Consider a variable named group that takes on the values 1, 2, and 3. Stata command list allows factor variables, so we can see how factor variables are expanded by typing

```
. list group i.group in 1/5
```

	group	1.group	2.group	3.group
1.	1	1	0	0
2.	1	1	0	0
3.	2	0	1	0
4.	2	0	1	0
5.	3	0	0	1

Variables

Factor-variables operators

There are five factor-variable operators:

Operator	Description
`i.`	unary operator to specify indicators
`c.`	unary operator to treat as continuous
`o.`	unary operator to omit a variable or indicator
`#`	binary operator to specify interactions
`##`	binary operator to specify full-factorial interactions

We will come back to this once we perform regressions and other models.

Data Types

Numbers:

Numbers can be stored in one of five variable types:

Storage type	Minimum	Maximum	without being 0	Bytes
`byte`	-127	100	± 1	1
`int`	$-32{,}767$	32,740	± 1	2
`long`	$-2{,}147{,}483{,}647$	2,147,483,620	± 1	4
`float`	$-1.70141173319 \times 10^{38}$	$1.70141173319 \times 10^{38}$	$\pm 10^{-38}$	4
`double`	$-8.9884656743 \times 10^{307}$	$+8.9884656743 \times 10^{307}$	$\pm 10^{-323}$	8

Do not confuse the term integer, which is a characteristic of a number, with int, which is a storage type. For instance, the number 5 is an integer, no matter how it is stored; thus, if you read that an argument must be an integer, that does not mean that it must be stored as an int.

Data Types

Missing values

A number may also take on the special value missing, denoted by a period (.). You specify a missing value anywhere that you may specify a number. Missing values differ from ordinary numbers in one respect: any arithmetic operation on a missing value yields a missing value.
Important! In Stata missing values are coded as the highest possible value of that storage type. Take that in mind when doing logical operations!

Important reference is :
https://www.stata.com/manuals/u12.pdfu12.2.1Missingvalues

Data Types

Stata has nine date, time, and date-and-time numeric encodings known collectively as %t variables or values. They are:

%tC	calendar date and time, adjusted for leap seconds
%tc	calendar date and time, ignoring leap seconds
%td	calendar date
%tw	week
%tm	calendar month
%tq	financial quarter
%th	financial half-year
%ty	calendar year
%tb	business calendars

Data Types

Date and time variables are best read as strings. You then use one of the string-to-numeric conversion functions to convert the string to an appropriate numeric value:

Format	String-to-numeric conversion function
%tc	clock(*string*, *mask*)
%tC	Clock(*string*, *mask*)
%td	date(*string*, *mask*)
%tw	weekly(*string*, *mask*)
%tm	monthly(*string*, *mask*)
%tq	quarterly(*string*, *mask*)
%th	halfyearly(*string*, *mask*)
%ty	yearly(*string*, *mask*)

Installing Packages

Stata has a very large repository of commands. However, sometimes you might want to use commands that are not installed by default with stata. We can install packages from:

- Boston College Statistical Software Components (SSC) archive.
- Packages obtained from internet.

Saving and printing output- log files

Stata can record your session into a file called a log file but does not start a log automatically; you must tell Stata to record your session:

To start a log:	`. log using` *filename*
Your session is now being recorded in file *filename*`.smcl`.	
To temporarily stop logging:	
Temporarily stop:	`. log off`
Resume:	`. log on`
To stop logging and close the file:	`. log close`
You can now print *filename*`.smcl` or type:	`. translate` *filename*`.smcl` *filename*`.log`
to create *filename*`.log` that you can load into your word processor.	
You can also create a PDF of *filename*`.smcl` on Windows or Mac:	`. translate` *filename*`.smcl` *filename*`.pdf`

Starting Commands

The initial commands are those related with the directory:

- pwd: will display your current working directory.
- dir: checks which files are inside the current working directory.
- cd: allows to change the working directory to a different path.
- mkdir: creates a new directory.

Basic Data Commands

These are some commands to understand our dataset:

- summarize
- list
- describe
- codebook
- browse
- count
- inspect
- table
- tabulate

Exercices - Basic Data Commands

- Set your working directory and create a new folder "folder 1".
- set the maxvar to the maximum possible.

Using the auto data:

- Can we save some memory space? What is the amount of variables and observations?
- Display the first 5 values of the variables price and mpg.
- Report the unique values of the variable rep78. Does it have missings? Show its frequency.
- Report for each value of headroom the mean and the standard deviation of the price. Now repeat for each possible value of rep78.

Basic Data Operations

The most basic data operations are:

- Sort
- Order
- Rename
- Label
- Drop
- Keep

Exercices - Basic Data Operations

Using "data1.dta" :

- Change the order of the variables so that age and female are the first ones.
- Rename the variable female not_male.
- Sort the variables according to the variable age.
- Keep only the asvab variables.
- Label asvab_mean as "proxy for ability".

Exercice Brush Up

Create a pdf log file with the answer.

1. Load the data and inspect the data set. How many variables and observations? Which variable is a string? Can we save some space in the memory?
2. List the first 20 observations of a dummy variable when year_b is 1961. Compute the fraction of individuals born each year by gender
3. Display the amount of individuals that are born each year, and do it by ethnicity.
4. Label the variable female as gender. In the variable sex, label 0 as male and 1 as female.
5. Rename all the variables that start by asvab as starting by ability. Also all the variables that finish with _b as _born.
6. Generate a table where for each value of female we get the mean and the standard deviation of earnings, for each value of pov78.

Calling Do-Files From Others

Suppose we have a do-file called "myfile.do".

In order to call it from another do-file, we need:

- Make sure the current directory on our second do-file is where "myfile.do" is stored.
- type : do myfile

Errors in Stata

When an error occurs, Stata produces an error message and a *return code*. For instance,

```
. list myvar
no variables defined
r(111);
```

We ask Stata to list the variable named `myvar`. Because we have no data in memory, Stata responds with the message "no variables defined" and a line that reads "`r(111)`".

The "no variables defined" is called the error message.

The 111 is called the return code. You can click on blue return codes to get a detailed explanation of the error.

- We can type help errors to try to find the solution to our problem.
- Errors will break the code.

Dealing with Errors

Sometimes the user can anticipate that errors will occur. There are ways to deal and ingore errors when the user wants to do so.

- capture command: capture executes a command and supresses all its output, even the errors.
- nonstop option.