Computing Camp - Stata Lecture 1

Expedited Introduction to Stata

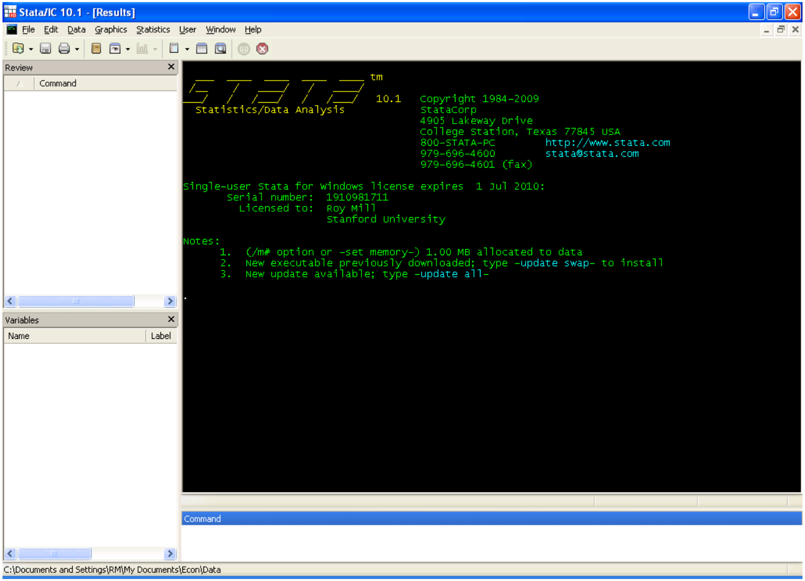
Roy Mill

September, 2010

Plan A

- Class 1 - Introduction
  - Graphical user interface
  - Datasets, command syntax and do-files
- Class 2 - Programming basics
  - Macros and post-estimation variables
  - Loops and control flow
- Class 3 - Two main examples
  - Posting results to a table
  - Creating a summary statistics table
  - Monte Carlo simulation

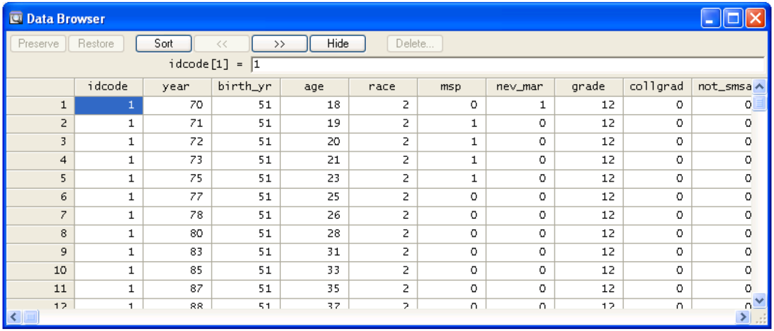
Stata - love at first sight?


Stata/IC 10.1 - [Results]
File Edit Data Graphics Statistics User Window Help
Review
Command
Variables
Name
Label
Command
C:\Documents and Settings\RM\My Documents\Econ\Data

```
  ___  ____  ____  ____  ____ tm
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   10.1   Copyright 1984-2009
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
                                      College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

Single-user Stata for Windows license expires  1 Jul 2010:
       Serial number:  1910981711
         Licensed to:  Roy Mill
                       Stanford University

Notes:
      1.  (/m# option or -set memory-) 1.00 MB allocated to data
      2.  New executable previously downloaded; type -update swap- to install
      3.  New update available; type -update all-

.
```

Datasets

Datasets are the objects of statistical analysis. They contain a matrix of which rows represent different observations (draws of random variables) and the columns are the variables.
Each cell contains the value of the variable for the observation in question:

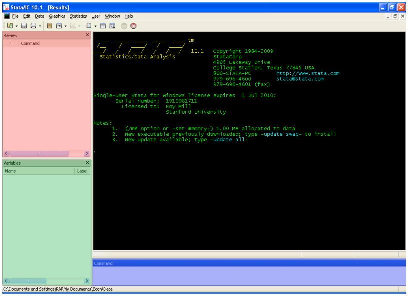


idcode[1] = 1

	idcode	year	birth_yr	age	race	msp	nev_mar	grade	collgrad	not_smsa
1	1	70	51	18	2	0	1	12	0	0
2	1	71	51	19	2	1	0	12	0	0
3	1	72	51	20	2	1	0	12	0	0
4	1	73	51	21	2	1	0	12	0	0
5	1	75	51	23	2	1	0	12	0	0
6	1	77	51	25	2	0	0	12	0	0
7	1	78	51	26	2	0	0	12	0	0
8	1	80	51	28	2	0	0	12	0	0
9	1	83	51	31	2	0	0	12	0	0
10	1	85	51	33	2	0	0	12	0	0
11	1	87	51	35	2	0	0	12	0	0
12	1	88	51	37	2	0	0	12	0	0

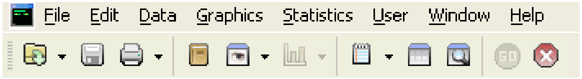
Main windows

- Results (black) - text output of the commands you run
- Command (blue) - allows to enter commands to run
- Review (red) - shows previously run commands
- Variables (green) - shows the variables in the loaded dataset


Stata/IC 10.1 - [Results]
File Edit Data Graphics Statistics User Window Help
Review
Command
Variables
Name
Label
Command
C:\Documents and Settings\RM\My Documents\Econ\Data

```
                                   tm
  ___  ____  ____  ____  ____
 /__    /   ____/   /   ____/      10.1   Copyright 1984-2009
___/   /   /___/   /   /___/              StataCorp
  Statistics/Data Analysis                4905 Lakeway Drive
                                          College Station, Texas 77845 USA
                                          800-STATA-PC        http://www.stata.com
                                          979-696-4600        stata@stata.com
                                          979-696-4601 (fax)

Single-user Stata for Windows license expires  1 Jul 2010:
       Serial number:  1910981711
         Licensed to:  Roy Mill
                       Stanford University

Notes:
      1.  (/m# option or -set memory-) 1.00 MB allocated to data
      2.  New executable previously downloaded; type -update swap- to install
      3.  New update available; type -update all-

.
```

Menu and Upper bar



- - Open a *data* file
- - Save dataset in memory to a file on disk
- - Open data editor (left), or data browser (right), for dataset in memory
- - Start a new do-file in a new do-file editor window
- - Stop execution of a command

From the menu you can easily call forms that will run some commands for you. For example: “Data → Describe Data → Summary Statistics” will open a form and then run the `summarize` command accordingly.

Commands and Syntax Conventions

Commands in Stata usually take the following form:

```
<command name> [... something ...] [if] [in] [, options]
```

A few conventions:

- Angle brackets - ⟨⟩ - mean that you *must* put something in their place. In other words, they are *mandatory*
- Square brackets - [] - mean that you *may* put something in their place. In other words, they are *optional*
- General syntax will be in blue color, specific examples in green

For example:

```
use mydataset.dta, clear
drop if male==1
save mydataset_females.dta, replace
```

Use and Save

Stata works with a single dataset in memory. It can work with multiple files, but not at the same time. You will need to load the dataset to the internal memory from the disk and you can save it back to the disk after you are done.

To load a dataset into the memory we run the `use` command:

```
use <file path> [, clear]
```

- file path - required path to the dta file you want to load.
- clear - Ignore existing dataset in memory, even if unsaved.

Use and Save

Now, after messing with the file, we might want to save it on file for later use.

```
save <file path> [, replace]
```

- file path - required path to the dta file you want to save.
- replace - If a filename by that name in this folder exists, replace it with the dataset in memory.

Notes:

- The `clear` and `replace` options will appear in the future and will have the same use: `clear` will overwrite the current dataset in memory and `replace` will overwrite the file on disk.
- If you will use the icons in the upper bar for loading and saving datasets, Stata will actually run the the `use` and `save` commands.

Use and Save

Examples:

```
use "C:\Documents and Settings\RM\Papers\RawData.dta", clear


// Alternatively...
cd "C:\Documents and Settings\RM\Papers"
use RawData, clear



// (... do some stuff ...)

// Now save it to the "data" subfolder
save data\GoodData, replace
```

Sniffing Around - What's in the Data

So we loaded a dataset and we want to learn more about what's there. Here are a few commands worthy of notice:

- <u>describe</u> - Lists the variables names, labels and formats
- <u>list</u> - Lists the observations' matrix: each observation with its values for each of the variables.
- <u>browse</u> - Like list, but opens up a window and is more convenient
- <u>tabulate</u> - Reports a histogram of a variable or joint histogram of two variables.
- <u>summarize</u> - For each variable requested, reports the number of observations with non-missing values, the mean, standard deviation, and other summary statistics.

Examples and their Output

```
describe idcode year birth_yr
```

```
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------
idcode          int    %8.0g                  NLS ID
year            byte   %8.0g                  interview year
birth_yr        byte   %8.0g                  birth year
```

```
tab mothers_brothers fathers_brothers
```

```
mothers_br |                       fathers_brothers
    others |         0          1          2          3          4 |     Total
-----------+-------------------------------------------------------+----------
         0 |       132        138         74         24          7 |       375 
         1 |       113        131         69         26          9 |       348 
         2 |        72         87         37          7          2 |       205 
         3 |        21         18         12          4          1 |        56 
         4 |         7          5          2          2          0 |        16 
-----------+-------------------------------------------------------+----------
     Total |       345        379        194         63         19 |     1,000 
```

Examples and their Output

```
su south race age
```

```
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       south |     28526    .4095562    .4917605          0          1
        race |     28534    1.303392    .4822773          1          3
         age |     28510    29.04511    6.700584         14         46
```

```
su tenure, detail
```

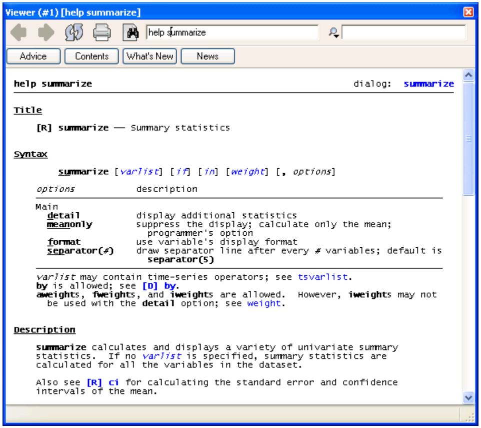
```
                    job tenure, in years
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%     .0833333              0
10%     .1666667              0       Obs               28101
25%           .5              0       Sum of Wgt.       28101

50%     1.666667                      Mean           3.123836
                        Largest       Std. Dev.      3.751409
75%     4.166667       23.33333
90%     8.416667           24.5       Variance       14.07307
95%     11.41667          24.75       Skewness       1.939685
99%     16.91667       25.91667       Kurtosis       6.901501
```

Note: su is short for summarize

Commands' Help Files

Each command should come with an accompanying help file. To learn more about additional options, other features, or to troubleshoot, the first place to look at is the help file:

```
help <command-name>
```



```
help summarize                                          dialog:  summarize

Title

    [R] summarize — Summary statistics

Syntax

        summarize [varlist] [if] [in] [weight] [, options]

    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)
    ----------------------------------------------------------------------
    varlist may contain time-series operators; see tsvarlist.
    by is allowed; see [D] by.
    aweights, fweights, and iweights are allowed.  However, iweights may not
      be used with the detail option; see weight.

Description

    summarize calculates and displays a variety of univariate summary
    statistics.  If no varlist is specified, summary statistics are
    calculated for all the variables in the dataset.

    Also see [R] ci for calculating the standard error and confidence
    intervals of the mean.
```

Commands' Help Files

Main parts of a usual help file (by the order I usually read them):

- Syntax - How to specify your command
- Description - Tells you what the command does generally
- Examples (at the bottom) - Shows you specific examples of how to run the command. Sometimes with an explanation.
- Options - The same command with different options can do totally different things. Skim through the options and look for the good ones.

Tips:

- Don't be afraid to experiment. Your data is saved on file, so you can always load it back if you made a mistake.
- Error messages looks scary, but don't let them fail you. READ them and try to understand them.
  - Remember: Errors don't mean that you are stupid, they mean that Stata is stupid.

Basic Data Manipulation - generate

To add a new variable we use the `gen` (short for `generate`) command:

```
gen <new-variable-name> = <expression> [if] [in]
```

For example:

```
gen four = 4
```

will create a variable (=column) that will contain the number 4 for all observations (=rows).

```
gen age_sq = age^2
```

will create a variable that will contain the square of the value in the age variable for the same row.

Conditions in Stata

Sometimes we want to apply a command only to some observations, not all.
We need to tell Stata what distinguishes those observations, so we construct a logical condition:

```
male == 1          age >= 21            4 > 60
```

- Stata evaluates the condition and turns it to 1, if the statement is true, or 0 if the statement is false.
- For example, since $(4 > 60)$ is not true, Stata will treat the expression $(4 > 60)$ as if it was 0.
- The other conditions involve variable names. They will be invoked as part of a command. Stata will apply the command only to observations for which the values inside the specified variables make the statement true.
    - For example, this is how we ask Stata to run `summarize` on females only:

```
su income if male == 0
```

Conditions in Stata

Note that some of the observations - those for which male == 1 - will not be processed by the `summarize` command.

We can combine multiple conditions with AND, OR and NOT operators:

```
(male == 1 & age >= 21) | (male == 0 & age >= 50)
```

will be true for males aged 21 and above or females aged 50 and above.
Adding the ! operator before a condition will negate it:

```
!(age >= 21)
```

will be true for people strictly younger than 21.

Conditions in Stata

Lastly, do not forget operators precedence:

$$6+5\times2 \neq (6+5)\times2$$

The same goes for | and & (like + and × respectively):

```
male == 1 & age > 21 | age < 10
```

will be true for males older than 21 and *all* children under 10.

```
male == 1 & (age >= 21 | age <= 10)
```

will be true for males only that are either older than 21 or younger than 10.

Note: missing values (.) are bigger than any value:
(`.  > 400000` is true)

For a complete list of logical as well as other operators, see
`help operator`

Back to Data Manipulation

`gen` can also take a condition. Observations for which the condition is false will have a missing value in the new variable:

```
gen age_sq_males = age^2 if male == 1
```

`age_sq_males` will contain the square of age for males and a missing value for females.

Note the difference between the assignment = and the comparison ==

One can also use the evaluation of a condition as the value to put into the new variable:

```
gen really_old = age > 22
```

Remember, the expression `age > 22` will be translated to either 1 or 0 according to whether age is bigger than 22 or not.

Back to Data Manipulation

Question: Which of the next three commands is best?

```
gen dropout = 1 if schooling < 12

gen dropout = schooling < 12

gen dropout = schooling < 12 if schooling != .
```

All commands will make those with `schooling < 12` have the value 1 in `dropout`. But what about the other ones?

- First line will assign missing values to all other observations
- Second line will assign zeroes to all other observations
- Third line will assign missing values to all observations that have a missing value in `schooling` and zeroes to the rest

So you will probably want to use the third line rather than the first two.

Meet `replace`, `gen`'s sister

Just like gen, but for existing variables instead of new ones, use replace (other statistical packages such as SAS don't even have this distinction between generating and replacing):

```
replace <variable-name> = <expression> [if] [in]
```

For example:

```
replace four = 5
```

will change the values in the variable `four` to now be 5.

```
gen     actual_price = discount_price if discount == 1
replace actual_price = full_price if discount == 0
```

will first create a variable that will contain the value from `discount_price` for all observations in which `discount` contains 1, then replace puts the value of `full_price` into `actual_price` if `discount` contains 0.

A quick note on `in`

This slide is not very important

You can use `in` instead of `if` when you want to apply a command to observations according to their observation-number.

Examples:

- To list variables in the first 10 observations, run:

```
list name eyecolor height in 1/10
```

- To set the year from the 1000th observation to the last to be 2009, run:

```
replace year = 2009 in 1000/l
```

Do-files

Until now we used the command window to type in commands and run them one-by-one. This was working interactively. What if you have many commands to run?

A .do file is a text file that contains a batch of commands each written as a separate line. This way you can save your commands for:

- later review, improvement and additional work
- collaborating with your colleagues - they can continue what you started

Comments

In a do file you can also explain what you are doing by adding comments. Here are a few ways to write comments:

```
/* Multi-line comments can be written easily like this
   I can continue babbling on and on
   until I have nothing more to say about this program */

* One line comments that start at the beginning of the line
* can be written by putting a * at the beginning of the line

replace a = b    // One line comments that don't necessarily
                 // begin at the beginning of a line can be
                 // written by those double-slashes. Everything to
                 // the right of a double-slash is a comment.
```

Long lines in do-files

As you noticed, each Stata command takes one line. Once you hit the return, or enter, key, Stata runs the command. This is also true for do-files.
But what if you have a really long command?

```
su income mot_educ fat_educ school age agesq south bigcity tenure comm
```

You can break the line with `///`:

```
su income mot_educ fat_educ school age agesq south bigcity ///
 tenure commute kids_u5 kids_18 tot_kids siblings ///
 if male & professional
```

Another way is to write `#delimit;` at the beginning of the do file and then end each command with a semicolon (;):

```
#delimit;
su income mot_educ fat_educ school age agesq south bigcity
 tenure commute kids_u5 kids_18 tot_kids siblings
 if male & professional;
tab age;
```

Log files

One last file you can save your work to is your log file. Unlike the do file, a log file will also save the text output resulted by your commands.

Whatever appeared in the results (big black) window, from when a log file was opened until it was closed, will be saved to the requested log file.

A log file is used to see what your program have done. Unlike the do file that will be edited and improved by you, the log file is automatically created by your program.

```
// Open a log file
log using <log-file-name> [, append replace text]

/* ... */

// Close current log file
log close
```

Log files

- replace - overwrite the file on disk if it already exists
- append - add the output to the end of the existing file if one exists - otherwise open a new one
- text - save the output in text format. In some cases Stata's default is to save it in a text-like format of its own called SMCL.

Here's a tip:
If a log is already open (usually after the last run ended tragically with an error), opening a log will create yet another error

To solve that, add the following line right before the `log using` line:

```
cap log close
```

File Types Summary

	data files	do files	log files
What's in it?	Observations and variables	A batch of commands	Text output from your commands
How do you open it?	`use` command mainly	Any text file editor	`log` commands
What is it good for?	Saving your data	Saving sequence of actions on the data	Recording commands you ran and their output
File extensions	`.dta`	`.do`	`.log`

Setting system parameters

To configure Stata, there are some parameters that you can change. Here are some examples:

- To change the amount of memory Stata takes from the operating system to 10 MB, run:

```
set memory 10m
```

- To change the default log format from SMCL to text (instead of specifying `text` in the options of the `log` command):

```
set logtype text
```

In most cases, you can change the parameter permanently. This way future Stata sessions will start with the new value for the parameter.
You can do it by specifying the `permanently` option (remember the comma...)

For more information about how to change Stata's system parameters, see `help set`.