# A QUICK INTRODUCTION TO STATA

This module provides a quick introduction to STATA. After completing this module you will be able to input data, save data, transform data, create basic tables, create basic graphs, describe data, and perform regression analysis. You will not be a STATA expert. You will however understand enough of STATA so that you can independently work with STATA using the help facility, online support, and manuals.

## 1. WHAT IS STATA AND WHAT DOES IT DO?

STATA is highly sophisticated software package for data management, data analysis, and graphing.

STATA contains a multitude of readily accessible prepared programs ("canned packages") for both data management and analysis. For example, STATA has commands for sorting, combining, deleting, adding, and reshaping data. STATA has programs for the most widely used data analyses, e.g., ordinary least squares, probit and logit, analysis of variance, descriptive statistics, etc.

STATA also has programming capacity. Users may write their own programs for data management and analysis.

STATA has internet capacity. Users can use the internet to update STATA and to find ado-files to perform cutting edge analysis. An ado-file is a user developed program that STATA will integrate into its software programs.

## 2. ELEMENTS OF STATA'S PERSONALITY.

STATA is case sensitive.

STATA is both command driven and menu driven.

STATA may be used interactively or from commands stored in a do-file.

## 3. STARTING AND EXITING STATA

A. Starting STATA

1. Selecting STATA from the Windows **Start** menu.

2. Double-click on a STATA dataset.

   STATA datasets have a **.dta** extension, e.g., **small dataset.dta**.

3. Double-click on a STATA command file, that is, a do-file.

   Do-files haves a **.do** extension, for example, **small dataset.do**.

4. Double-click on a STATA graph file.

   STATA graph files haves a **.gph** extension, for example, **prettygraph.gph**.

B. Save your work before exiting from STATA.

After starting STATA, working interactively or using a do-file, completing all your tasks of some portion of your tasks, you may wish to save your work. The save command creates STATA a datafile.

To save your work, type the following command.

**save E:\Florida Education Fund\small dataset**

Use this command if there is no existing dataset named **small dataset.dta.** If there is an existing dataset, use the following command.

**save E:\Florida Education Fund\small dataset, replace**

The **replace** option tells STATA to overwrite the existing dataset.

The menu operations are as follows.

**File > save > select file name**

**File > save as > select file name**

**Select File button and follow instructions.**

C. Exiting STATA

1. Type **exit** on STATA's command line.

2. Click the exit button.

3. Click **File > Exit** from STATA's command menu.

STATA will not allow you to exit if there are data in memory. If data are in STATA memory you must **clear** the data from memory. Be careful with the **clear** command! This command clears from memory all work going back to the last **save**. Typing **exit, clear** will allow you to exit but it also clears without saving everything in memory.

## 4. IMMEDIATE TASKS AFTER STARTING STATA

```
set memory 250m
set more off
#delimit ;

log using E:\Florida Education Fund\small dataset, replace ;

/* STATA will give this file a .log extentsion. */

use E:\Florida Education Fund\small dataset ;

/* STATA assumes this file has a .dta extentsion. */
```

The 5 lines above are extremely important. These lines, with appropriate individual changes should be typed at the beginning of each session (if working interactively) or included in STATA command files.

**set memory 250m** tells stata to set memory to 250 megabits. This command tells stata how much memory you will need. With small files you'll need less memory. With large files you'll need more memory. If you do not have enough memory STATA will tell you via an error message. The commands will not be executed.

**set more off** is an optional command. It controls what see on the screen. In this case, the material on the screen will continue until the output of the last command. Without this command, output to the screen will have "-More" at the bottom of the screen, indicating that you must press the "enter" or "return" key on your keyboard to see the remainder of the output.

**#delimit ;** is an optional command. It tells STATA that a semicolon ";" identifies the end of an operational command. STATA does need this, the user might need this. It clarifies where one command ends and another begins. You don't have to use a semicolon, but I choose to use a semicolon because it's similar to SAS. LIMDEP uses $.

**log using E:\Florida Education Fund\small dataset, replace;** is semi-optional. This command creates a STATA log file. The file's name will be Regress.smcl and it will be located in E:\PSID\Output directory.

**replace** is an option. It tells STATA to replace a previously existing log-file with this name in this specific directory.

STATA puts the **.smcl** at the end of the file. This stands for STATA machine control language, that is, STATA's own programming language. At the end of the command file, the very last two lines, you can convert this to a text file and tell STATA to close the log file. Henceforth, all output and commands will be recorded in the log file.

**smcl** = STATA machine control language. Only STATA can read this log file. If you want a log file that can be read by any word processor, using the following command.

**log using E:\Florida Education Fund\small dataset, text replace ;**

Alternatively, you may use

**set logtype text, permanently ;**

The **permanently** option specifies that the current and all future log files will be text files.

use **E:\Florida Education Fund\small dataset ;** is a mandatory command. It tells STATA the location of the data you are using. **use** tells STATA to "go and get this data at this particular location." In this case, the data is located at **E:\Florida Education Fund\small dataset**. Obviously, **small dataset** is the name of the raw data file. This is not an ascii file. **small dataset** is a STATA dataset. It's full name is **small dataset.dta**. STATA is smart; it recognizes its own datasets with or without the .dta extension. When the .dta extension is not included, STATA assumes that it's a STATA dataset. Note the semicolon at the end of the command.

File > open > **E:\Florida Education Fund\small dataset** is the menu approach for this command.

Suppose we have **small dataset.txt** (an ascii dataset) instead of **small dataset.dta**. STATA will not open **small dataset.txt**. How would we get STATA to open this file? We have to import **small dataset.txt into STATA.**

## 5. LOG FILES

Log-files permit users to record their activities for each session of STATA, that is, each time you start and exit STATA.

Suppose you are working interactively and accidently **clear** STATA before saving your results. Hakuna matata! You can use your log-file to immediately redo your entire session. If you are working interactively, you can use the log-file to quickly construct a do-file.

STATA log-files have **.smcl extension**, for example, **Regress.smcl.**

scml is STATA machine code language. You can read this file in the STATA viewer.

Use the menu commands as follows

**File > view > browse (to select file)**

Text editors, e.g., Microsoft Word, WordPad, and Notepad, cannot read STATA log-files with the **.smcl** extension. You may find this irritating. If so, a text file copy of the STATA log-file may be obtained with the following command.

translate "E:\Regress.smcl" "E:\Regress.log" ;

Regress.log is a text file that can be read by Microsoft Word or any other text editor. In the case, the quotes (") at the beginning and ending of the file name are redundant, but probably a good habit to develop. The quotes are necessary only if there are blanks in the file name.

There is a way to avoid having to create a text copy of the smcl file. After the **#delimit ;** type the following command.

**set logtype text ;**

If the above command is used prior to the **#delimit ;** command, the **;** is unnecessary. This command tells STATA that the log-file should be a text file. In this case, the log command will tell STATA to create E:\Regress.log.

If you want all of your log-files to be text files and not just the current log-file, type the following command.

**set logtype text, permanently ;**

An individual log file is closed with the following command.

**log close ;**

This command does not close STATA. It does not clear STATA's memory. It only closes the log file. You can open a new log file immediately thereafter. Or, if your session is at an end, **save** and/or **clear** then **exit.**

## 6. HELP!

There are multiple sources of obtaining help with STATA.

A. Use the STATA manuals. There are detailed descriptions of every STATA command or program. The more often you use the manuals, the more you become comfortable with using the manuals.

B. Click the **Help** command. The manuals are not always readily available. The **Help** facility is an electronic version of the material in the base manuals. Further, **Help** will direct you to reliable internet sites that contain additional information on your problem.

C. Google "name of issue STATA". More often times than not, a large number of web locations will be found that address the issue you are concerned with. Remember, there are STATA users all over the world.

D. Join the STATA listserv. The members of this listserv are STATA geeks and they live to solve STATA problems. The more complex, the better.

E.Use **Help > STATA web site > user-support** and **Help > STATA web site > frequently asked questions**

F. Send an e-mail to STATA user-support. Carefully explain your issue. Attach your log file and, if necessary, your data-file or at least a reasonable sample of your data-file. STATA will assign your problem to an expert. The expert will get back in contact with fairly quickly. For simple problems the expert may contact you within an hour. Complex problems may require several hours.

## 7. SAMPLE SESSION: INTERACTIVE, MENU

In this session, we will use buttons and menu commands to work with

E:\Florida Education Fund\small dataset.xls and

E:\Florida Education Fund\small dataset.txt.

Menu steps will have the following pattern.

Step1 > Step2 > …

Often after the final menu step, the dialog box will require additional information. If so, after the final menu step we write out the information required in the dialog box. This may seem a bit convoluted, but this will be straightforward as we proceed.

Also, each time we provide the menu steps we will also show how to accomplish the same task via a STATA command. Commands are entered at the *command window*. In our examples the **commands will be entered in bold letters**.

Regardless of whether we type a command in the command window or accomplish the task using the file menus, STATA records a command in the *review window*.

This is immensely convenient. Suppose you forget the command for carrying out a task. No problem. Use the files menus to carry out the task, then copy the command from the *review window* into your do-file.

## A. Start STATA session

## B. Enter preliminary commands

```
set memory 100m
set more off
#delimit ;

/* Create log file. This file records all commands and all output. */

File > log > begin >
                    File name: small dataset
                    Save as type: Log(*.log)
```

## C. Input data into STATA: at least four options.

i. Import raw data

```
File > Import > ASCII data created by spreadsheet > E:\Florida Education
Fund\small dataset.txt
```

ii. Cut and paste raw data

```
Data > Data editor  cut and past E:\Florida Education Fund\small dataset.xls
```

iii. Use command to input raw data

```
insheet using "E:\Florida Education Fund\small dataset.txt" ;
```

iv. Input data by hand using data editor

```
 input v. edit

 list v. browse

File > Import > ASCII data created by spreadsheet > E:\Florida Education
Fund\small dataset.txt

/* Obtain description of data */

Data > describe data > describe data in memory
```

## D. Save your data

There's no need to input your data multiple times. Once you have inputted and observed the data, save the data as a STATA dataset.

```
File > save >
             File name: small dataset
             Save as type: *.dta
```

**save E:\Florida Education Fund\small dataset, replace ;**

## E. Data Management Commands

/* Provide a label for dataset */

Data > Labels > Label dataset

**label data "Labor Market Survey"  ;**

Data > describe data > describe data in memory ;

**describe ;**

/* Change generic variable names to mnemonic codes */

i. Click on editor. Place cursor in desired column. Click right hand side of mouse.

   Variables > Properties > change name as desired.

   Repeat for each column.

ii. Data > variable utilities > rename variables

```
rename v1 indviden ;
rename v2 age ;
rename v3 sex ;
rename v4 educ ;
rename v5 wage ;
rename v6 sector ;
rename v7 county_and_city ;
rename v8 state ;
```

Data > describe data > describe data in memory ;

/* Create variable labels */

i. Click on editor. Place cursor in desired column. Click right hand side of mouse.

   Variables > Properties > change variable label as desired.

   Repeat for each column.

ii. Data > Labels > Label variable

```
label variable indviden "Individual identification number" ;
label variable age "Age of head of household" ;
label variable sex "Sex of survey respondent" ;
label variable educ "Years of education of head" ;
label variable wage "Mean weekly wage of head last year" ;
label variable sector "Sector of employment for primary job" ;
label variable county_and_city "County and city of employment" ;
label variable state "State of employment" ;
```

```
Data > describe data > describe data in memory ;

/* sex is a string variable; hence, we cannot perform numerical operations
with this variable. There are several ways to use sex to create a numeric
variable. */

/* The encode command maps a string variable into a numeric variable. In this
case, a new variable ‘gender’ will be created that has a value of 1 for males
and 2 for females. Males are assigned 1 because the first observation is a
male.

NB: the decode command can be used to transform numeric variables to string
variables.  */

Data > Create or change data > Other variable transformation commands >
Encode value labels from string variable ;

encode sex, generate(gender) ;

/* create a binary variable ‘woman’ which takes on a value of 0 for males and
1 for females */

Data > Create or change data > Create new variable
           Variable name: woman
           Contents of new variable: (expression): sex == "female"

generate woman = sex == "female" ;

/* Create value labels for sector variable */

Data > Labels > Label values > Define or modify values > Define or modify
value lables > Define
              Value: 0
              Text: private
              Value: 1
              Text: public

Data > Labels > Label values > Assign value labels to variable > attach a
value label to a variable
               Variable: sector
               Value Label: sector

label define sector 0 "private" 1 "public" ;
label values sector sector ;

Data > describe data > describe data in memory

describe ;

/* List all data by observation number. */

Data > describe data > List data > submit

list ;

/* List observations 2 through 5 for educ and wage */
```

Data > describe data > List data >
(main tab) Variables: educ wage
(by/if/in tab) check use a range of observations From: 2 To: 5
Submit

**list educ wage in 2/5 ;**

/* Sort data by mean weekly wage */

Data > sort > ascending sort
Variables: wage

Click on the data browser button to examine the impact of your sort. Of course, you can also use the data editor to sort.

**sort wage ;**

/* For years of education and weekly wages, list last 3 observations with lowest wages. -4 is the fourth from the last observation, while -1 is the last observation */

Data > describe data > List data >
(main tab) Variables: educ wage
(by/if/in tab) check use a range of observations list 9 12
Submit

**list educ wage in -4/-1 ;**

/* Create your own data codebook. The **codebook** command lists the following information for every variable in the dataset: type of variable (numeric or string), variable label, range, units, unique values, missing, tabulation. */

Data > describe data > describe data contents (codebook) ;

**describe ;**

/* The basic descriptive statistics include the number of observations for each variable, mean, standard deviation, minimum value, and maximum value. */

Data > describe data > summary statistics > (main tab) click the standard display option > submit

**summarize ;**

/* In addition to the basic statistics, you can also obtain percentiles, skewness, kurtosis, and variance. */

Data > describe data > summary statistics > (main tab) click the additional statistics option > submit

**summarize, detail ;**

/* Sometimes we may wish to know descriptive statistics for particular subgroups of workers: persons with at least a high school diploma (educ >= 12, governmental workers (sector = 1), private sector workers (sector = 0) */

```
Data > Describe data > Summary statistics > (main tab) variables: wage
                          (by/if/in): educ >= 12
                          submit

Data > Describe data > Summary statistics > (main tab) variables: wage
                          (by/if/in): sector == 1
                           submit

Data > Describe data > Summary statistics > (main tab) variables: wage
                          (by/if/in): sector == 0
                          submit
```

```
summarize wage if educ >= 12 ;
summarize wage if sector == 1 ; /* public sector */
summarize wage if sector == 0 ; /* private sector */
```

The last two summarize commands can be combined into one command.

```
bysort sector: summarize wage ;
```

The bysort modifier does two things. First, it tells STATA to sort the data by sector. Then, it tells STATA to execute the summarize command by order of the sorted variable.

```
/* STATA creates a multitude of tables. It is straightforward to create a
oneway table for years of education and for weekly wage rate. */

Statistics > summaries, tables, & tests > Tables > oneway tables > (main tab)
categorical variable: educ

Statistics > summaries, tables, & tests > Tables > oneway tables > (main tab)
categorical variable: wage
```

```
tabulate educ ;
tabulate wage ;
```

```
/* correlation matrix with descriptive statistics */

Statistics > summaries, tables, & tests > Summary statistics > Correlations
and covariances > (main tab) variables: age educ gender wage sector
      display tab): click display correlation/covariance matrix for variables
                    click display mean, std. dev., min, and max with matrix
```

```
correlate age educ gender wage sector, means ;
```

```
/* Create twoway of mean weekly wages by employment sector table with row
totals, column totals, chi-square test for difference in means. */

Statistics > summaries, tables, & tests > Tables > Twoway tables with
measures of association > (main tab) row variable: wage
                                     column variable: sector
                                     submit
```

```
tabulate wage sector ;
```

```
/* Add Pearson's Chi-squared statistic for difference in means by sector */
```

Statistics > summaries, tables, & tests > Tables > Twoway tables with measures of association > (main tab) row variable: wage
column variable: sector
Test statistics: Pearson's Chi-squared
Submit

**tabulate wage sector, chi2 ;**

/* Add Pearson's Chi-squared statistic for difference in means by sector. Add row and column frequencies. */

Statistics > summaries, tables, & tests > Tables > Twoway tables with measures of association > (main tab) row variable: wage
column variable: sector
Test statistics: Pearson's Chi-squared
Cell contents: within-column frequencies
Within-row frequencies
Submit

**tabulate wage sector, chi2 row col ;**

/* show scatter plot of wages against education */

Graphics > Easy graphs > scatter plots > (main tab) X variables: educ
Y variables: wage

**graph twoway scatter wage educ ;**

**graph twoway (scatter wage educ) (qfit wage educ) ;**

**graph twoway (scatter wage educ) (qfit wage educ), by(gender) ;**

## 8. SUBMIT DO-FILE E:\FLORIDA EDUCATION FUND\SMALL DATASET.DO.

A do-file is a file containing the set of commands for a STATA session. The advantage of a writing your commands in a do-file is that it gives you a permanent record of how you derived your results. If there is an error in your do-file, STATA stops at the command that is not written properly and provides an error message. Sometimes this message is helpful, sometimes it is not.

The do-file "small dataset.do" contains quickly reproduces all that we have covered in the interactive sample session.

## 9. CREATING NEW VARIABLES

Basic structure of command to create new variable.

*generate newvariable = expression*

educ, wage, age

generate educ_square = educ^2 ; /* create educ_squared */
generate educpluswage = educ + wage ; /* add educ and wage */
generate wageminusage = wage - age ; /* subtract age from educ */

```
other mathematical relations
* multiplication
/ division
> greater than
< less than
>= greater than or equal to
<= less than or equal to
== equal to
!= not equal to

logical relations
! not
| or
& and

/* generate creates a missing value if condition is not fulfilled. */

generate yearsofcolleduc = educ - 12 if educ >= 13 ;
replace  yearsofcolleduc = 0 if yearsofcolleduc == . ;


generate tailwage = wage if (25 >= age | age >= 55) ;

generate primewage = wage if (25 <= age & age <= 55) ;

generate malewage = wage if sex != "female" ;


/* "county_and_city" and "state" are string variables. We are going to do two
things with these variables. First, we are going to combine county and city
and state into one variable with space between city and state. (There's
already space between county and city). Next, we are going to separate city
from county and city. */


generate countycitystate = county_and_city + " " + state ;

list countycitystate ;

/* In order to remove city from county_and_city we need to make use of two
string functions, strpos(s1, s2) and substr(s, n1, n2).

strpos(county_and_city, " ") says the following: go to the county_and_city
variable, locate the first position where a space occurs, assign an integer
equal to the number of places of that position to the end of the name of the
county_and_city variable; if the condition is not met strpos() assigns a
value of 0.

 Highlands Sebring           strpos assigns a value of 10
 Travis Austin               strpos assigns a value of 7
 Washtenaw Ann Arbor         strpos assigns a value of 10
 Kings Brooklyn              strpos assigns a value of 6
 Macon Tuskegee              strpos assigns a value of 6
 Genesee Flint               strpos assigns a value of 8
 Riverside Riverside         strpos assigns a value of 10
 Wayne Detroit               strpos assigns a value of 6
 SaintJoseph South Bend      strpos assigns a value of 12
```

```
Leon Tallahassee            strpos assigns a value of 5
Dade Miami                  strpos assigns a value of 5
Hillsborough Tampa          strpos assigns a value of 13
```

strpos()+1 adds 1 to each of the assign position values

substr(county_and_city, n1, n2) says the following: go to the county_and_city variable, locate the position number 1, include everything from position 1 to position n2. If n2 is assigned "." then STATA reads to the end of the variable name.

```
/* Now, let's put all of this together to substract the city from county_and
city. */

generate city = substr(county_and_city, strpos(county_and_city, " ")+1, .) ;

list city ;
```

## 9. Regression Analysis & Additional Data Management

```
regress wage educ ; /* ordinary least squares */

probit woman educ ; /* probit equation */

mfx ;          /* obtain marginal effects after probit */

logit woman educ ; /* logit equation */

mfx ;            /* marginal effects after logit */

logistic woman educ ; /* odds ratio from logit regression */


/* drop, keep, append, joinby, merge

estimation (regress, probit (and mfx), logit (and mfx), logistic

hypothesis testing

information criteria

time series analysis

Working with Panel Study on Income Dynamics, Current Population Survey, and
other large complex datasets.             */
```