STATA Tutorial

to Accompany

Introduction to Econometrics

by James H. Stock and Mark W. Watson

STATA Tutorial to accompany Stock/Watson *Introduction to Econometrics*

STATA is a powerful statistical and econometric software package. This tutorial is an introduction to some of the most commonly used features in STATA. These features were used for the statistical analysis reported in Chapters 3-11 of *Introduction to Econometrics*. This tutorial provides the necessary background to reproduce the results in Chapters 3-11 and to carry out related exercises.

1. GETTING STARTED

STATA is available for computer systems such as Windows, Mac, and Unix. Your professor will tell you how to access STATA at your university. Most of you will be using STATA on Windows computers. To access STATA on these computers, you need to double-click on the STATA icon. This icon will probably be labeled ***Intercooled Stata X***, where ***X*** is a number indicating the version of STATA available on your system. (All of the results in Chapters 3-11 of *Introduction to Econometrics* were computed using ***Intercooled STATA 7***.)

Once you have started STATA, you will see a large window containing several smaller windows. At this point you can load the dataset and begin the statistical analysis. STATA can be operated "interactively" or in "batch mode." When you use STATA interactively, you type a STATA command in the STATA Command window and hit the Return/Enter key on your keyboard. STATA executes the command and the results are displayed in the STATA Results window. Then you enter the next command, STATA executes it, and so forth, until the analysis is complete. Even the simplest statistical analysis will involve several STATA commands.

When STATA is used in batch mode, all of the commands for the analysis are listed in a file, and STATA is told to read the file and execute all of the commands. These files are called **do files** by STATA and are saved using a **.do** suffix. For example, all of the STATA commands for the analysis in Chapter 4 of the text are contained in a file called **ch4.do** available on the Web site. When STATA executes this file, all of the empirical results for Chapter 4 are produced.

Using STATA in batch mode has two important advantages over using STATA interactively. First, the **do** file provides an audit trail for your work. The file provides an exact record of each STATA command. Second, even the best computer programmers will make typing or other errors when using STATA. When a command contains an error, it won't be executed by STATA, or worse, it will be executed but produce the wrong result. Following an error, it often necessary to start the analysis from the beginning. If you are using STATA interactively, you must retype all of the commands. If you are using a **do** file, then you only need to correct the command containing the error and rerun the file. For these reasons, you are strongly encouraged to use **do** files, and this tutorial will discuss STATA used in this way.

2. DO-FILE INGREDIENTS AND KEY COMMANDS

A STATA **do** file has four different kinds of commands or ingredients:

1. Administrative commands that tell STATA where to save results, how to manage computer memory, and so forth

2. Commands that tell STATA to read and manage datasets

3. Commands that tell STATA to modify existing variables or to create new variables

4. Commands that tell STATA to carry out the statistical analysis

There are many commands within each category. A list of some of the most useful commands is given at the end of this tutorial.

This section and the next will work through five examples of **do** files. After you have worked through these examples, you will have a good basic understanding of STATA.

To begin, you need to download the **do** files. These files are called files **stata1.do**, **stata2.do**, … , **stata5.do** and are available on the Web site. Create a folder (or directory) on your computer called **statafiles** and download the **do** files to this folder. You also need to download the file **caschool.dta** to this folder.

The **do** files are "text" or "ascii" files. They can be read with any text editor. For example, if you are using Windows, you can use Notepad to edit these files. If you use a word processor such as MS Word, make sure that you save the files using the **Save as type** "Text Only" option, and save them using the suffix **.do**.

The file **stata1.do** contains seven lines of text

```
log using \statafiles\stata1.log, replace
use \statafiles\caschool.dta
describe
generate income = avginc*1000.
summarize income
log close
exit
```

The first line is an administrative command that tells STATA where to write the results of the analysis. STATA output files are called **log** files, and the first line tells STATA to open a **log** file called **stata1.log** in the folder (or directory) **statafiles**. If there is already a file with the same name in the folder, STATA is instructed to replace it.

The second and third lines concern the dataset. Datasets in STATA are called **dta** files. The dataset used in this example is **caschool.dta,** which you downloaded to the folder **statafiles**. The second line tells STATA the location and name of the dataset to be used

for the analysis. The third line tells STATA to "describe" the dataset. This command produces a list of the variable names and any variable descriptions stored in the dataset.

The fourth line tells STATA to create a new variable called **income**. The new variable is constructed by multiplying the variable **avginc** by 1000. The variable **avginc** is contained in the dataset and is the average household income in a school district expressed in thousands of dollars. The new variable **income** will be the average household income expressed in dollars instead of thousands of dollars.

The fifth line of the program tells STATA to compute some summary statistics (mean, standard deviations, and so forth) for income. The STATA command for this is **`summarize`**.

The last two lines close STATA. The command **`log close`** closes the file **stata1.log** that contains the output. The command **`exit`** tells STATA that the program has ended.

To execute this **do** file, first open STATA on your computer. (Double-click on the ***Intercooled STATA*** icon on a Windows computer. Follow your professor's instructions for opening STATA on other computers.) Click on the **File** menu, then **Do**, and then select the file **\statafiles\stata1.do**. This will run the **do** file. You can see the program being executed in the STATA Results window. You might see that the program execution pauses and that **`-- more --`** is displayed at the bottom of the Results window. If this happens, push any key on the keyboard and execution will continue. (We will get rid of this annoyance with a STATA command in the next example.) You can exit STATA by clicking on the **File** menu and then **Exit**. STATA will ask you if you really want to exit. Respond "Yes". Your output will be in **\statafiles\stata1.log**. Open this file using a text editor, and you will see the statistical output.

This example has used six STATA commands:

```
log using
use
describe
generate
log close
exit
```

These commands are summarized at the end of this tutorial.

The next section introduces several more STATA commands in four other **do** files. After you work through these examples, you will understand the elements of STATA.

3. MORE EXAMPLES

stata2.do

Here are the commands in **stata2.do**

```
# delimit ;
************************************************;
* Administrative Commands;
************************************************;
set more off;
clear;
log using \statafiles\stata2.log,replace;
************************************************;
* Read in the Dataset;
************************************************;
use \statafiles\caschool.dta;
describe;
************************************************;
* Transform data and Create New Variables;
************************************************;
**** Construct average district income in $'s;
generate income = avginc*1000;
************************************************;
* Carry Out Statistical Analysis;
************************************************;
***** Summary Statistics for Income;
summarize
          income;
************************************************;
* End the Program ;
************************************************;
log close;
exit;
```

The file **stata2.do** carries out exactly the same calculations as **stata1.do**; however it uses four features of STATA for more complicated analyses. The first new command is **`# delimit ;`**. This command tells STATA that each STATA command ends with a semicolon. If STATA does not see a semicolon at the end of the line, then it assumes that the command carries over to following line. This is useful because complicated commands in STATA are often too long to fit on a single line. **stata2.do** contains an example of a STATA command written on two lines: near the bottom of the file you see the command **`summarize income`** written on two lines. STATA combines these two lines into one command because the first line does not end with a semicolon. While two lines are not necessary for this command, some STATA commands can get long, so it is good to get used to using this feature.

A word of warning: if you use the **`# delimit ;`** command, it is critical that you end each command with a semicolon. Forgetting the semicolon on even a single line means that the **do** file will not run properly.

The second new feature in **stata2.do** is that many of the lines begin with an asterisk. STATA ignores the text that comes after *****, so that these lines can be used to describe what the **do** filing is doing. Notice that each of these lines ends with a semicolon. If the

semicolon had not been included, then STATA would have included the next line as part of the text description.

The third new feature in the file is the command **`set more off`**. This command eliminates the need to type a key whenever STATA fills the Results window, so that **`– more –`** will no longer be displayed in the Results window.

Finally, the last new command is **`clear`**. This command erases any data already in STATA's memory. It is a good idea to use the **`clear`** command before starting a new analysis.

stata3.do

Here are the commands in **stata3.do**

```
# delimit ;
************************************************;
* Administrative Commands;
************************************************;
set more off;
clear;
log using \statafiles\stata3.log,replace;
************************************************;
* Read in the Dataset;
************************************************;
use \statafiles\caschool.dta;
describe;
************************************************;
* Transform data and Create New Variables;
************************************************;
generate testscr_lo = testscr if (str<20);
generate testscr_hi = testscr if (str>=20);
************************************************;
* Carry Out Statistical Analysis;
************************************************;
* Compute statistics for test scores;
summarize testscr;
ttest testscr=0;
ttest testscr_lo=0;
ttest testscr_hi=0;
ttest testscr_lo=testscr_hi, unequal unpaired;
************************************************;
* Repeat the analysis using STR = 19;
************************************************;
replace testscr_lo = testscr if (str<19);
replace testscr_hi = testscr if (str>=19);
************************************************;
* Carry Out Statistical Analysis;
************************************************;
* Compute statistics for test scores;
ttest testscr_lo=testscr_hi, unequal unpaired;
************************************************;
* End the Program ;
************************************************;
log close;
exit;
```

This file introduces three new features. First, it creates new variables using only a portion of the dataset. Two of the variables in the dataset are **testscr** (the average test score in a school district) and **str** (the district's average class size or student-teacher ratio). The STATA command **generate testscr_lo = testscr if (str<20)** creates a new variable **testscr_lo** that is equal to **testscr**, but this variable is only defined for districts that have an average class size of less than twenty students (that is, for which **str<20**). Similary **testscr_hi** is a new variable that is equal to class size for districts with average class sizes greater than or equal to twenty students (**str>=20**).

The statement **str<20** is an example of a "relational operation." STATA uses several relational operators:

```
<     less than
>     greater than
<=    less than or equal to
>=    greater than or equal to
==    equal to
~=    not equal to
```

The second new STATA feature in this file is the command **ttest**. This command constructs tests and confidence intervals for the mean of a population or for the difference between two means. This command is used two different ways in the program. The first use is **ttest testscr=0**. This command computes the sample mean and standard deviation of the variable **testscr,** computes a *t*-test that the population mean is equal to zero, and computes a 95% confidence interval for the population mean. (In this example the *t*-test that the population mean of test scores is equal to zero is not interesting, but the confidence interval for the mean is.) This same command is used for **testscr_lo** and **testscr_hi**. The second form of the command is **ttest testscr_lo=testscr_hi, unequal unpaired**, which is used to test the hypothesis that **testscr_lo** and **testscr_hi** come from populations with the same mean. That is, this command computes the *t*-statistic for the null hypothesis that the mean of test scores for districts with class sizes less than twenty students is the same as the mean of test scores for districts with class sizes greater than twenty students. This command uses two "options" that are listed after the comma in the command. These options are **unequal** and **unpaired**. The option **unequal** tells STATA that the variances in the two populations may not be the same. The option **unpaired** tells STATA that the observations are for different districts, that is, these are not panel data representing the same entity at two different time periods. As written, this command carries out the comparison of means from different populations described in Section 3.4 of the text.

The third new feature in this file is the command **replace**. This appears near the bottom of the file. Here, the analysis is to be carried out again, but using 19 as the cutoff for small classes. Since the variables **testscr_lo** and **testscr_hi** already exist (they were defined by the **generate** command earlier in the program), STATA cannot "generate" variables with the same name. Instead, the command **replace** is used to replace the existing series with new series.

stata4.do

Here are the commands in **stata4.do**

```
# delimit ;
************************************************;
* Administrative Commands;
************************************************;
set more off;
clear;
log using \statafiles\stata4.log,replace;
************************************************;
* Read in the Dataset;
************************************************;
use \statafiles\caschool.dta;
describe;
*************************************************************;
***** Table 4.1 ******;
*************************************************************;
summarize str testscr, detail;
************************************************;
**** Figure 4.2;
************************************************;
graph testscr str;
*************************************************************;
***** Correlation ******;
*************************************************************;
correlate str testscr;
*************************************************************;
***** Equation 4.7 and 4.26 ******;
*************************************************************;
regress testscr str, robust;
*************************************************************;
***** Equation 4.33 ******;
*************************************************************;
generate d = (str<20);
regress testscr d, robust;
log close;
clear;
exit;
```

This file reproduces the empirical results shown in Chapter 4 of the text. It contains several features of STATA that have not been used in the previous examples. First, the **`summarize`** command is now written using the option **`detail`**, which provides more detailed summary statistics. The command is written as **`summarize str testscr, detail`**. This command tells STATA to compute summary statistics for the two variables **`str testscr`**. The option **`detail`** produces detailed summary statistics that include, for example, the percentiles that are reported in Table 4.1.

The command **`graph testscr str`** constructs a scatterplot of **`testscr`** versus **`str`**. This scatterplot is shown in Figure 4.2 in the text.

The command **`correlate str testscr`** tells STATA to compute the correlation between **`str testscr`**.

The command **`regress testscr str, robust`** tells STATA to run an OLS regression with **`testscr`** as the dependent variable and **`str`** as the regressor. The option **`robust`** tells STATA to use heteroskedasticity-robust formulas for the standard errors of the regression coefficient estimators.,

The section of the **do** file that produces the results in Equation (4.33) contains two new features. First, it creates the binary (or indicator, or dummy) variable **`d`** using the STATA command **`generate d = (str<20)`**. The variable **`d`** is equal to 1 if the expression in parentheses is true (that is, when **`str<20`**) and is equal to 0 if the expression is false (that is, **`str•20`**). STATA allows you to use any of the relational operators listed above (**`<`**, **`>`**, **`<=`**, etc.).

stata5.do

Here are the commands in **stata5.do**

```
# delimit ;
************************************************;
* Administrative Commands;
************************************************;
set more off;
clear;
log using \statafiles\stata5.log,replace;
************************************************;
* Read in the Dataset;
************************************************;
use \statafiles\caschool.dta;
desc;
************************************************************;
***** Table 5.1 ******
************************************************************;
gen str_20 = (str<20);
gen ts_lostr = testscr if str_20==1;
gen ts_histr = testscr if str_20==0;
gen elq1 = (el_pct<1.94);
gen elq2 = (el_pct>=1.94)*(el_pct<8.78);
gen elq3 = (el_pct>=8.78)*(el_pct<23.01);
gen elq4 = (el_pct>=23.01);
ttest ts_lostr=ts_histr, unp une;
ttest ts_lostr=ts_histr if elq1==1, unp une;
ttest ts_lostr=ts_histr if elq2==1, unp une;
ttest ts_lostr=ts_histr if elq3==1, unp une;
ttest ts_lostr=ts_histr if elq4==1, unp une;
************************************************************;
***** Eq 5.9 ******
************************************************************;
reg testscr str, r;
************************************************************;
***** Eq 5.10 ******
************************************************************;
reg testscr str el_pct, r;
************************************************************;
***** Eq 5.18 ******
************************************************************;
replace expn_stu = expn_stu/1000;
reg testscr str expn_stu el_pct, r;
* Diplay Covariance Matrix;
vce;
```

```
* F-test reported in text;
test str expn_stu;
*************************************************************;
******* Correlations reported in text;
*************************************************************;
cor testscr str expn_stu el_pct meal_pct calw_pct;
*************************************************************;
******* Table 5.2;
*************************************************************;
* Column (1);
reg testscr str, r;
display "Adjusted Rsquared = " _result(8);
* Column (2);
reg testscr str el_pct, r;
dis "Adjusted Rsquared = " _result(8);
* Column (3);
reg testscr str el_pct meal_pct, r;
dis "Adjusted Rsquared = " _result(8);
* Column (4);
reg testscr str el_pct calw_pct, r;
dis "Adjusted Rsquared = " _result(8);
* Column (5);
reg testscr str el_pct meal_pct calw_pct, r;
dis "Adjusted Rsquared = " _result(8);
*;
*************************************************************;
******* Appendix -- rule of thumb f-statistic;
*************************************************************;
reg testscr str expn el_pct;
test str expn;
reg testscr el_pct;
log close;
clear;
exit;
```

This file produces all of the empirical results from Chapter 5 of the text. STATA allows you to abbreviate commands when there is no possibility of confusion, and this file uses abbreviations for STATA commands throughout (**`generate`** is abbreviated **`gen`**, **`regress`** is abbreviated **`reg`**, and so forth). It contains four new commands.

The first two new commands are in the analysis of Equation 5.18. The command **`reg testscr str expn_stu el_pct, r`** computes the regression reported in Equation 5.18. The command **`vce`** asks STATA to print out the estimated variances and covariances of the estimated regression coefficients. The command **`test str expn_stu`** asks STATA to carry out the joint test that the coefficients on **`str`** and **`expn_stu`** are both equal to zero.

The third new command is in the analysis of Table 5.2. When STATA computes an OLS regression, it computes the adjusted *R*-Squared ($\bar{R}^2$) as described in Section 5.10 of the text. However, it does not automatically print the adjusted *R*-squared in the log file. The command **`display "Adjusted Rsquared = " _result(8)`** tells STATA to display (or print out) the adjusted *R*-Squared.

Other Examples

You will find other examples of **do** files on the text Web site. You can download STATA **do** files to reproduce all of the analysis in Chapters 3-11. You will also find STATA **do** files for Chapters 12-14. Chapters 12-14 concern time series data, and STATA programming for time series is somewhat more complicated than for cross-sectional or panel data. EVIEWS and RATS are econometric programs specifically designed for time series data, and the Web site contains EVIEWS and RATS programs for Chapters 12-14.

4. MISCELLANEOUS TOPICS

Examining Data in STATA Datasets

It is often useful to examine the data in a STATA dataset. For example, the dataset **caschool.dta** contains the California test score data analyzed in Chapters 4-7, and you might want to see the test scores or class size for a particular school district. There are two ways to examine the data in STATA. To examine the data on the screen you can use the **Data Editor** or **Data Browser**. The Data Editor displays all of the variables in a dataset in spreadsheet format; that is, in a format like Excel. The Data Editor can also be used to add more data to a dataset or to correct errors in a data set. Data can be entered into the Data Editor's spreadsheet using the keyboard or by cutting and pasting. The Data Browser is just like the Data Editor, except that it does not allow you to modify the existing data set. If you want to look at the dataset, it is safer to use the Data Browser because that way it is impossible to change the data inadvertently.

To use to Data Editor or Browser, first load the dataset into STATA. This can be done by typing the command **`use` *`dataset.dta`*** in STATA's Command Window, where ***`dataset.dta`*** contains the location and name of the dataset that you want to examine. For example, to examine the California test score dataset you would type **`use \statafiles\caschool.dta`**. The Editor or Browser can be opened by clicking on the appropriate icon at the top of the main STATA window. The Editor icon looks like a spreadsheet, and the Browser icon looks like a spreadsheet with a magnifying glass on top of it. You can also open the Editor by clicking on the main **Window** menu and then **Data Editor**.

The data can also be printed in the **log** file. To do this, put the command **`list` *`variable name list`*** in the **do** file, where ***`variable name list`*** is a list of the names of the variables to be printed. For example, to print the data on **`testscr`** and **`str`** from **`caschool.dta`**, insert the command **`list testscr str`** into any of the **do** files in the last section. A word of caution: if you use the command **`list`** without a variable list, STATA will print all of the variables in the **log** file. If the dataset is large, this will produce a very large **log** file.

Creating STATA Datasets

All of the data used in the text are available in STATA datasets. These data can read into STATA with the **use** command. When you study other datasets, they need to be converted into STATA data files (**.dta** files). There are several ways to do this conversion. Two are discussed here.

The computer program StatTransfer converts Excel, Lotus, Access, and other popular formats into STATA data files. If StatTransfer is available at your university, use it.

If StatTransfer is not available, then use the following procedure:

(1) Put your data into an Excel spreadsheet.
(2) Save the Excel spreadsheet as a **csv** (comma delimited) file.
(3) Use the STATA command **insheet** to read the data into STATA. Then save the data as a STATA data file.

Here is an example:

(1) Create an Excel file with this format.

str	testscr
17.89	690.80
21.52	661.20
18.70	643.60
17.36	647.70
18.67	640.85
21.41	605.55
19.50	606.75

This file contains 7 observations on the variables **str** and **testscr**. The Excel file must have this special format: the first row must contain the variable names, then the data must be listed in the other rows with no blanks.

(2) In Excel save this file using **Save as Type CSV (Comma delimited)** using the name **mydata.csv**. If you look at the file **mydata.csv** using a text editor, you will see

```
str,testscr
17.89,690.80
21.52,661.20
18.70,643.60
17.36,647.70
18.67,640.85
21.41,605.55
19.50,606.75
```

These are data from the Excel file, but in a different format.

(3) Run the following STATA do file.

```
# delimit ;
clear;
log using convert.log,replace;
insheet using mydata.csv;
save mydata.dta, replace;
log close;
exit;
```

The command **`insheet using mydata.csv`** reads in the data file created in step 2. The command **`save mydata.dta, replace`** saves the data as a STATA dataset with the name **`mydata.dta`** replacing any other dataset with that name. The data set can now be read into STATA using the command **`use mydata.dta`**.

This example used **`mydata`** as the name of the dataset. You can replace this with the name of your choosing.

Increasing Memory

On many computers STATA is set up so that it uses a limited amount of the computer's memory. If your STATA analysis requires more memory than is allocated, you will get a message that says something like **`no room to add more observations.`**

When this happens you need to increase the amount of memory available to STATA. To do this, follow these two steps. First, find out how large your dataset is. For example, suppose that you have a large dataset stored in the file **\statafiles\bigdata.dta**. If you type

```
describe using \statafiles\bigdata.dta
```

STATA will print a description of the dataset **`bigdata.dta`** in the STATA Results window. For this example, STATA prints the results

```
Contains data
  obs:           2075
 vars:            380
 size:      6,308,000
```

in the STATA Results window. This says the dataset contains 380 variables, each with 2075 observations, and requires 6.308 megabytes of memory. To make sure STATA has enough memory to read this dataset and carry out the statistical analysis, put the command

```
set memory 8m
```

at the top of your **do** file. This tells STATA to set aside 8 megabytes of memory for the data. (Only 6.308m is needed to read the dataset, but additional memory may be required when new variables are generated.)

On most computers, STATA is configured so that it can run OLS regressions that contain up to forty regressors. This is sufficient for nearly all applications. However, if you have an application that requires more than forty regressors, you will need to use the STATA command **`set mat #`**, which is described in Section 6.

5. GETTING MORE INFORMATION ABOUT STATA

STATA has detailed help files available for all STATA commands. You can access these by clicking on the **Help** menu in STATA.

STATA commands are described in detail in the STATA *User's Guide* and *Reference Manual*. You should be able to find these in the computer lab or at your computer center.

STATA provides a comprehensive Web site at www.stata.com.

Finally, you can find several good STATA tutorials on the Web. An easy way to find a list is to do a Google search for **Stata tutorial**.

6. A SUMMARY OF SELECTED STATA COMMANDS

This section lists several of the most useful STATA commands. Many of these commands have options. For example, the command **`summary`** has the option **`detail`** and the command **`regress`** has the option **`robust`**. In the descriptions below, options are shown in square brackests [ ]. Many of these commands have several options and can be used in many different ways. The descriptions below show how these commands are commonly used. Other uses and options can be found in STATA's **Help** menu and in the other resources listed in Section 5.

The list of commands provided here is a small fraction of the commands in STATA, but these are the important commands that you will need to get started. They will allow you to do all of the analysis contained in Chapters 3-7 of the text. Chapters 8-14 use specialized commands for panel data, probit and logit regression, instrumental variables regression, and time series analysis. These specialized commands can be found in the **do** files for these chapters. A detailed description of these specialized commands can found in STATA's **Help** menu and in the other resources listed in Section 5.

Administrative Commands

`# delimit`

delimit sets the character that marks the end of a command. For example, the command **`# delimit ;`** tells STATA that all commands will end with an asterisk. This command is used in **do** files.

`clear`

This command deletes/erases all variables from the current STATA session.

`exit`

In a **do** file, **`exit`** tells STATA that the **do** file has ended. If you type **`exit`** in STATA's Command window, then STATA will close.

`log`

This command controls STATA **log** files, which is where STATA writes output. There are two common uses of this command. The first is

`log using` ***`filename`*** **`[,append replace]`**

This opens the file given by ***`filename`*** as a **log** file for STATA output. The options **`append`** and **`replace`** are used when there is already a file with the same name. With **`append`**, STATA will append the output to the bottom of the existing file. With **`replace`**, STATA will replace the existing file with the new output file.

`log close`

This closes the current **log** file.

set mat #

This sets the maximum number of variables that can be used in a regession. The default maximum is 40. If you have many, many observations and want to run a regression with 45 variables, then you will need to use the command **set mat #** where **#** is a number greater than 45.

set memory #m

This is used in Windows and Unix versions of STATA to set the amount of memory used by the program. See the discussion in Section 4 above.

set more off

This tells STATA not to pause and display the **–more-** message in the Results window.

Data Management

describe

This describes the contents of data in memory or on disk. A related command is **describe using *filename***, which describes the dataset in ***filename***.

drop *list of variables*

This deletes/erases the variables in ***list of variables*** from the current STATA session. For example, **drop str testscr** will delete the two variables **str** and **testscr**.

keep *list of variables*

This deletes/erases all of the variables from the current STATA session except those in ***list of variables***. Alternatively, it keeps the variables in the list and drops everything else. For example, **keep str testscr** will keep the two variables **str** and **testscr** and delete all of the other variables in the current STATA session.

list *list of variables*

This command tells STATA to print all of the observations for the variables listed in ***list of variables***.

save *filename* [, replace]

This tells STATA to save the dataset that is currently in memory as a file with name ***filename***. The option **replace** tells STATA that it may replace any other file with the name ***filename***. See the example in Section 4 of this tutorial.

use *filename*

This tells STATA to load a dataset from the file ***filename***.

Transforming and Creating New Variables

New variables are created using the command **generate**, and existing variables are modified using the command **replace.**

Examples:

```
generate newts = testscr/100
```

creates a new variable called **newts** that is constructed as the variable **testscr** divided by 100.

```
replace testscr = testscr/100
```

changes the variable **testscr** so that all observations are divided by 100.

You can use the standard arithmetic operations of addition (+), subtraction (–), multiplication (*), division (/), and exponentiation (^) in generate/replace commands. For example,

```
generate ts_squared = testscr*testscr
```

creates a new variable **ts_squared** as the square of **testscr**. (This could also have been accomplished using the command **generate ts_squared = testscr^2 .**)

You can also use relational operators to construct binary variables. For example, **stata4.do** included the command

```
generate d = (str<20);
```

which created the binary variable **d** that was equal to 1 when **str<20** and was equal to 0 otherwise.

Standard functions can also be used. Three of the most useful are:

abs(x)	computes the absolute value of *x*
exp(x)	provides the exponentiation of *x*
ln(x)	computes the natural logarithm of *x*

For example, the command

```
generate ln_ts = ln(testscr)
```

creates the variable **ln_ts**, which is equal to the logarithm of the variable **testscr**.

Finally, logical operators can also be used. For example,

```
generate testscr_lo = testscr if (str<20);
```

creates a variable **testscr_lo** that is equal to **testscr**, but only for those observations for which **str<20**.

Statistical Operations

correlate *list of variables*

This command tells STATA to compute the correlation between each of the variables in ***list of variables***.

graph

Tells STATA to produce a graph. STATA can produce several kinds of graphs, including histograms, pie charts, box plots, and scatterplots. The command to produce a scatterplot is **graph *varname1 varname2***, where ***varname1*** and ***varname2*** are the names of the series to be plotted. The STATA commands for other types of graphs can become complicated. These are explained in the resources listed in Section 5.

predict *newvarname* [, residuals]

When this command follows the **regress** command, the OLS predicted values or residuals are calculated and saved under the name ***newvarname***. When the option **residuals** is used, the residuals are computed; otherwise the predicted values are computed.

Example:

```
regress testscr str expn_stu el_pct, robust
predict tshat
predict uhat, residuals
```

In this example **testscr** is regressed on **str, expn_stu, el_pct** (first command); the fitted values are saved under the name **tshat** (second command), and the residuals are saved under the name **uhat** (third command).

regress depvar *list of variables* [if expression] [,robust noconstant]

Carries out an OLS regression of the variable **depvar** on ***list of variables***. When **if expression** is used, then the regression is estimated using observations for which **expression** is true. The option **robust** tells STATA to use the heteroskedasticity-robust formula for the standard errors of the coefficient estimators. The option **noconstant** tells STATA not to include a constant (intercept) in the regression.

Examples:

```
regress testscr str, robust

regress testscr str expn_stu el_pct, robust
```

summarize [*list of variables*] [, details]

Computes summary statistics. If the command is used without a list of variables, then summary statistics are computed for all of the variables in the dataset. If the command is used with a list of variables, then summary statistics are computed for all variables in the list. If the option **details** is used, more detailed summary statistics (including percentiles) are computed.

Examples:

summarize testscr str
computes summary statistics for the variables **testscr** and **str**.

summarize testscr str, detail
computes detailed summary statistics for the variables **testscr** and **str**.

test

This command is used to test hypotheses about regression coefficients. It can be used to test many types of hypotheses. The most common use of this command is to carry out a joint test that several coefficients are equal to zero. Used this way, the form of the command is **test *list of variables*** where the test is to be carried out on the coefficients corresponding to the variables given in ***list of variables.***

Example:

```
regress testscr str expn_stu el_pct, robust
test str expn_stu
```

In this example **testscr** is regressed on **str**, **expn_stu**, and **el_pct** (first command), and a joint test of the hypothesis that the coefficient on **str** and **expn_stu** are jointly equal to zero is carried out (second command).

ttest

This command is used to test a hypothesis about the mean or the difference between two means. The command has several forms. Here are a few:

ttest *varname* = # [if expression] [, level(#)]
Tests the null hypothesis that the population mean of the series ***varname*** is equal to #. When **if expression** is used, then the test is computed using observations for which **expression** is true. The option **level(#)** is the desired level of the confidence interval. If this option is not used, then a confidence level of 95% is used.

Examples:

`ttest testscr=0;`

Tests the null hypothesis that the population mean of **`testscr`** is equal to 0 and computes a 95% confidence interval.

`ttest testscr=0, level(90);`

Tests the null hypothesis that the population mean of **`testscr`** is equal to 0 and computes a 90% confidence interval.

`ttest testscr=0 if (str<20);`

Tests the null hypothesis that the population mean of **`testscr`** is equal to 0 and computes a 95% confidence interval only using observations for which **`str<20`**.

`ttest` ***`varname1`*** **`=`** ***`varname2`*** **`[if expression] [, level(#) unpaired unequal ]`**

Tests the the null hypothesis that the population mean of series ***`varname1`*** is equal to the population mean of series ***`varname2`***. When **`if expression`** is used, then the test is computed using observations for which **`expression`** is true. The option **`level(#)`** is the desired level of the confidence interval. If this option is not used, then a confidence level of 95% is used. The option **`unpaired`** means that the observations are not paired (they are not panel data), and the option **`unequal`** means that the population variances may not be the same. (Section 3.4 of the text describes the equality of means test under the **`unpaired`** and **`unequal`** assumption.)

Examples:

`ttest testscr_lo=testscr_hi, unequal unpaired`

Tests the null hypothesis that the population mean of **`testscr_lo`** is equal to the population mean of **`testscr_hi`** and computes a 95% confidence interval. Calculations are performed using the unequal variance and unpaired formula of Section 3.4.

`ttest ts_lostr=ts_histr if elq1==1, unpaired unequal`

Tests the null hypothesis that the population mean of population is equal to the population mean of **`ts_histr`** and computes a 95% confidence interval. Calculations are performed for those observations for which **`elq1`** is equal to 1. Calculations are performed using the unequal variance and unpaired formula of Section 3.4.