STATA 10 Tutorial

by Manfred W. Keil

to Accompany

Introduction to Econometrics

by James H. Stock and Mark W. Watson

----

1. STATA: INTRODUCTION

This tutorial will introduce you to a statistical and econometric software package called STATA. The tutorial is an introduction to some of the most commonly used features in STATA. These features were used by the authors of your textbook to generate the statistical analysis report in Chapters 3-9 (Stock and Watson, 2011). The tutorial provides the necessary background to reproduce the results of Chapters 3-9 and to carry out related exercises. It does not cover panel data (Chapter 10), binary dependent variables (Chapter 11), instrumental variable analysis (Chapter 12), or time-series analysis (Chapters 14-16).

 The most current professional version is STATA 11. Both STATA 10 and 11 are sufficiently similar so that those who have access to STATA 11 can use this tutorial for the more advanced version. As with many statistical packages, newer versions allow you to use more advanced and recently developed techniques that you, as a first time user, most likely will not encounter in a first course of econometrics. There are several versions of STATA 10 and 11, such as STATA/IC, STATA/SE, and STATA/MP. The difference is basically in terms of the number of variables STATA can handle and the speed at which information is processed. Most users will probably work with the "Intercooled" (IC) version.

STATA runs on the Windows (2000, 2003, XP, Vista, Server 2008, or Windows 7), Mac, and Unix computers platform. I assume most of you will be using STATA on Windows computers. It is produced by StataCorp in College Station, TX. You can read about various product information at the firm's Web site, www.stata.com . There are 19 manuals that can be purchased with STATA 11, although subsets can be bought separately. Perhaps the most useful of these are the *User's Guide* and three volumes of the *Base Reference Manuals* ($210 together). You can order STATA by calling (800) 782-8272 or writing to service@stata.com. In addition, if you purchase the Student Version ("GradPlans"), you can acquire STATA at a steep discount. Prices vary, but you could get a "perpetual license" for STATA/IC for $179, or a six-month license for as low as $65.

Econometrics deals with three types of data: cross-sectional data, time series data, and panel (longitudinal) data (see Chapter 1 of the Stock and Watson (2011)). In a *cross-section* you analyze data from multiple entities at a single point in time. In a *time series* you observe the behavior of a single entity over multiple time periods. This can range from high frequency data such as financial data (hours, days); to data observed at somewhat lower (monthly) frequencies, such as industrial production, inflation, and unemployment rates; to quarterly data (GDP) or annual (historical) data. One big difference between cross-sectional and time series analysis is that the order of the observation numbers does not matter in cross-sections. With time series, you would lose some of the most interesting features of the data if you shuffled the observations. Finally, *panel data* can be viewed as a combination of cross-sectional and time series data, since multiple entities are observed at multiple time periods. STATA allows you to work with all three types of data.

STATA is most commonly used for cross-sectional and panel data in academics, business, and government, but you can work with it relatively easily when you analyze time-series data.

STATA allows you to store results within a program and to "retrieve" these results for further calculations later. Remember how you calculated confidence intervals in statistics say for a population mean? Basically you needed the sample mean, the standard deviation, and some value from a statistical table. In STATA, you can calculate the mean and standard deviation of a sample and then temporarily "store" these. You then work with these numbers in a standard formula for confidence intervals. In addition, STATA provides the required numbers from the relevant distribution (normal, χ^2, F, etc.).

While STATA is truly "interactive," you can also run a program as a "batch" mode

- *Interactive use*: you type a STATA command in the STATA *Command Window* (see below) 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 typically will involve several STATA commands.
- *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* and are saved using a *.do* suffix.

In the good old days the equivalent of writing a *Do-File* was to submit a "batch" of cards, each card containing a single command (now line), to a technician, who would use a card reader to enter these into the computer. The computer would then execute the sequence of statements. (You stored this batch of cards typically in a filing cabinet, and the deck was referred to as a "file.") While you will work at first in interactive mode by clicking on buttons or writing single line commands, you will very soon discover the advantage of running your regressions in batch mode. This method allows you to see the history of commands, and you can also analyze where exactly things went wrong if there are problems ("errors") with any of your commands. This tutorial will initially explain the interactive use of STATA since it is more intuitive. However, we will switch as soon as it makes sense into the batch mode and you should seriously try to do your research/class work using this mode ("*Do-Files*").

STATA produces highly professionally looking graphs and charts. However, it requires some practice to generate these. A separate manual (Graphics) is devoted to the topic only. Since STATA works in a Windows format, it allows you to cut and paste the data into other Windows-based program, such as Word or WordPerfect.

Finally, there is a warning about the limitations of this tutorial. The purpose is to help you gain an initial understanding of how to work with STATA. I hope that the tutorial looks less daunting than the manuals. However, it cannot replace the accompanying manuals, which you will have to consult for more detailed questions (alternatively use "**Help**" within the program). Feel free to provide me with feedback of how the tutorial can be improved for future generations of students (mkeil@cmc.edu). Colleagues of mine and I have decided to set up a "Wiki" run by students but supervised by faculty at my academic institution. We have found that the "wisdom of crowds" often produces valuable information for those who follow. This

is, of course, just a suggestion. Finally you may want to think about working with statistical software as learning a new language: practicing it routinely will result in improvement. If you set it aside for too long, you will only remember the most important lines but will forget the important details. Another danger of tutorials like this is that you simply follow the instructions and when you are done, you do not remember the commands. It is therefore a good idea to keep a separate sheet and to write down commands and examples of them if you think you will use them later. I will give you short exercises so that you can practice the commands on your own.

2. CROSS-SECTIONAL DATA

Interactive Use: Data Input and Simple Data Analysis

Let's get started. Click on the STATA icon to begin your session, or choose STATA 10 from your START window. Once you have started STATA, you will see a large window containing several smaller windows. At this point you can load a data set or enter data (described below) and begin the statistical analysis.
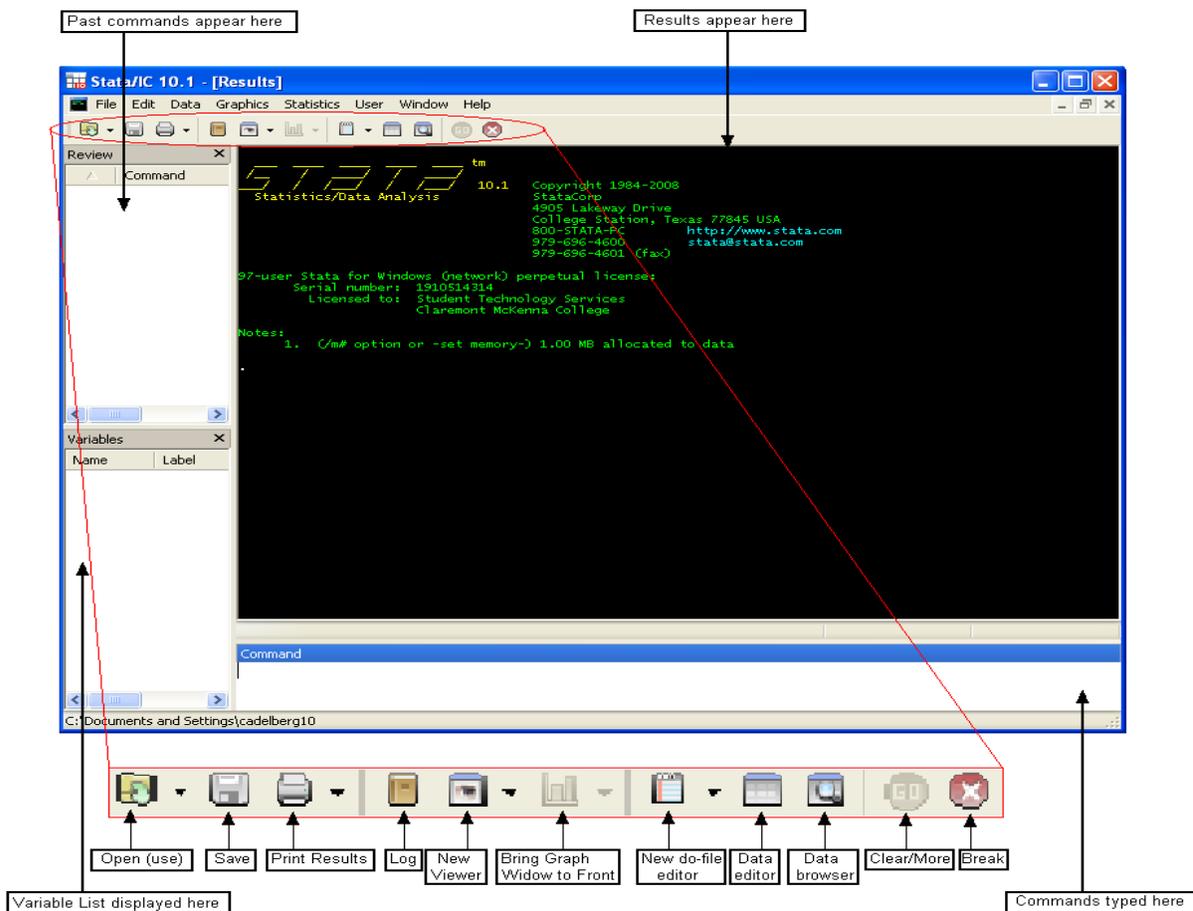
The results of your various operations will be displayed in the so-called *Results Window*. On the bottom left, there is a *Variables Window*, which shows the names of variables currently active in the *datafile*. Above it is the *Review Window*, which lets you view previously used STATA commands. In interactive use, STATA allows you to execute commands either by clicking on command buttons or by typing the equivalent command into the *Command Window*.

In this tutorial, we will work with two data applications: two cross-sectional (California Test Score Data Set used in chapters 4-9; and the Current Population Survey Data Set used in Chapters 3 and 8) as an exercise.

a) The Easy and *Tedious Way: Manual Data Entry*

In Chapters 4 to 9 you will work with the California Test Score Data Set. These are cross-sectional data. There are 420 observations from K-6 and K-8 school districts for the years 1998 and 1999. You will not want to enter a large amount of data manually, since it is tedious and leaves room for human error. As a result, it is generally not a recommended method of inputting data. However, there are occasions when you have collected data by yourself (something that economists are doing more and more). The alternative is to enter the data into a spreadsheet (Excel) and then to cut and paste the data (see below).

Entering data manually is used here for pedagogical purposes since it gives you an initial understanding of how to work with data in STATA. In other words, it will be useful that you become aware of entering, and editing, data in the program. Here I will use a sub-sample of 10 observations from the California Test Score Data Set.

To start, click on the **Data Editor** button on the toolbar, or type the command *edit* into the Command Window. This will open the following screen:
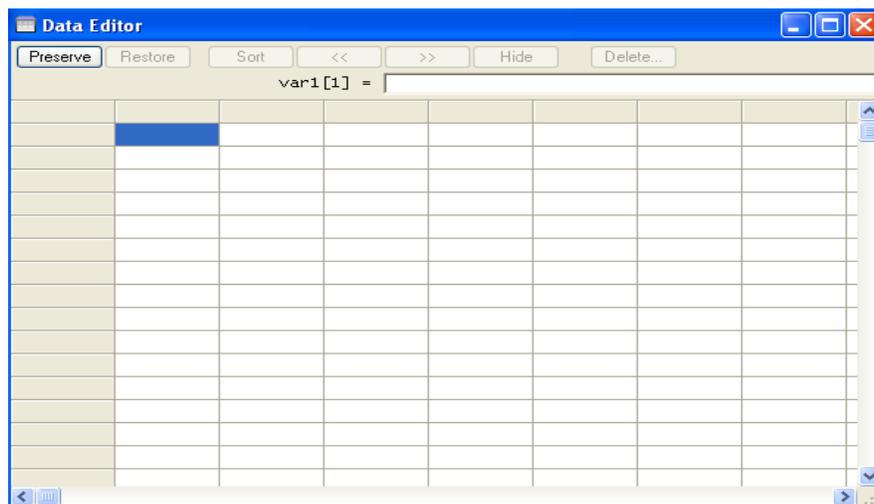
To enter data manually, start typing in the observations (you will name the variables subsequently). Here I have chosen 10 observations of test scores (*testscr*) and the student-teacher ratio (*str*) from the data set you will use in Chapter 4 of the textbook.

obs	testscr	str
1	606.8	19.5
2	631.1	20.1
3	631.4	21.5
4	631.8	20.1
5	631.9	20.4
6	632.0	22.4
7	632.0	22.9
8	638.5	19.1
9	638.7	20.2
10	639.3	19.7

After entering the data, double-click the grey box at the top of the first column (the box directly above the blue one in the above picture). This will result in the following box to appear:
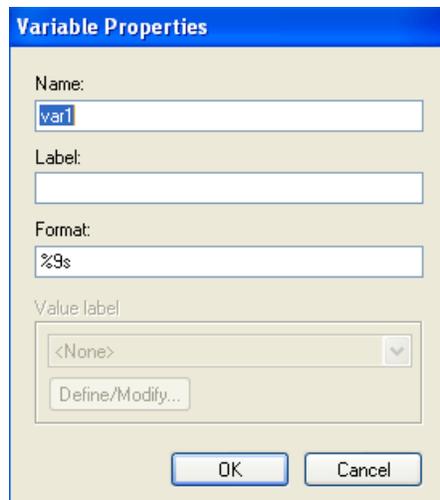
In the *Name* box, replace *var1* with the name of the first column variable, here *testscr*. In the *Label* box, you may want to enter information that that helps you remember how the data was created originally or as information for others who may subsequently work with your data. I suggest you enter here

Avg test score (=(read_scr+math_scr)/2)

Similarly you could enter for the second variable *str*

Student teacher ratio (teachers/enrl_tot)

After completing this task, the *Data Editor* screen should look as follows:
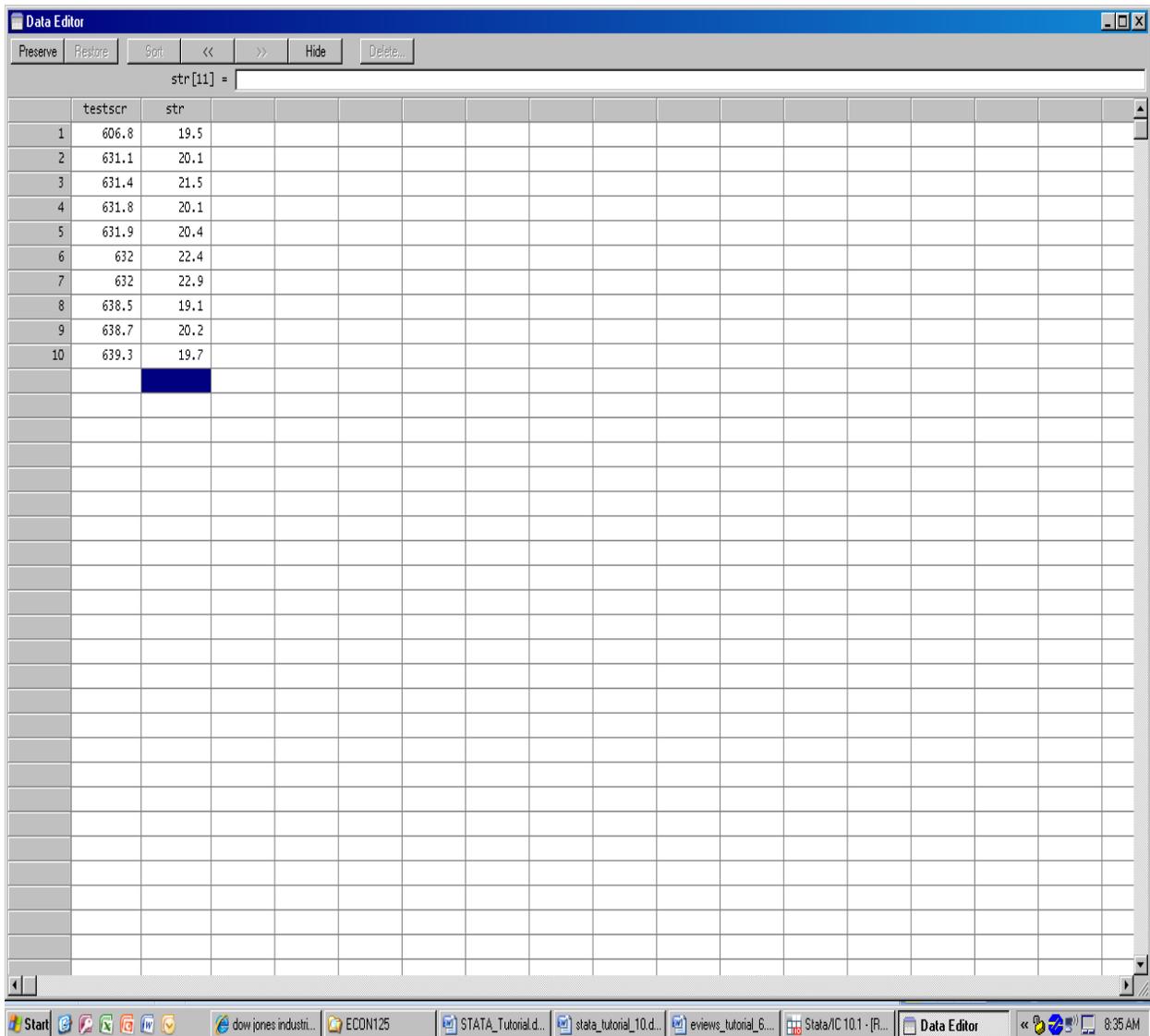
Next hit the *Preserve button* in the upper left hand corner of the *Data Editor*, and then close the box. Note that your commands to edit and preserve the data now appear in the *Results Box*, your command to edit is listed in the *Command Box*, and your newly created variables are shown in the variable list on the lower left-hand side:
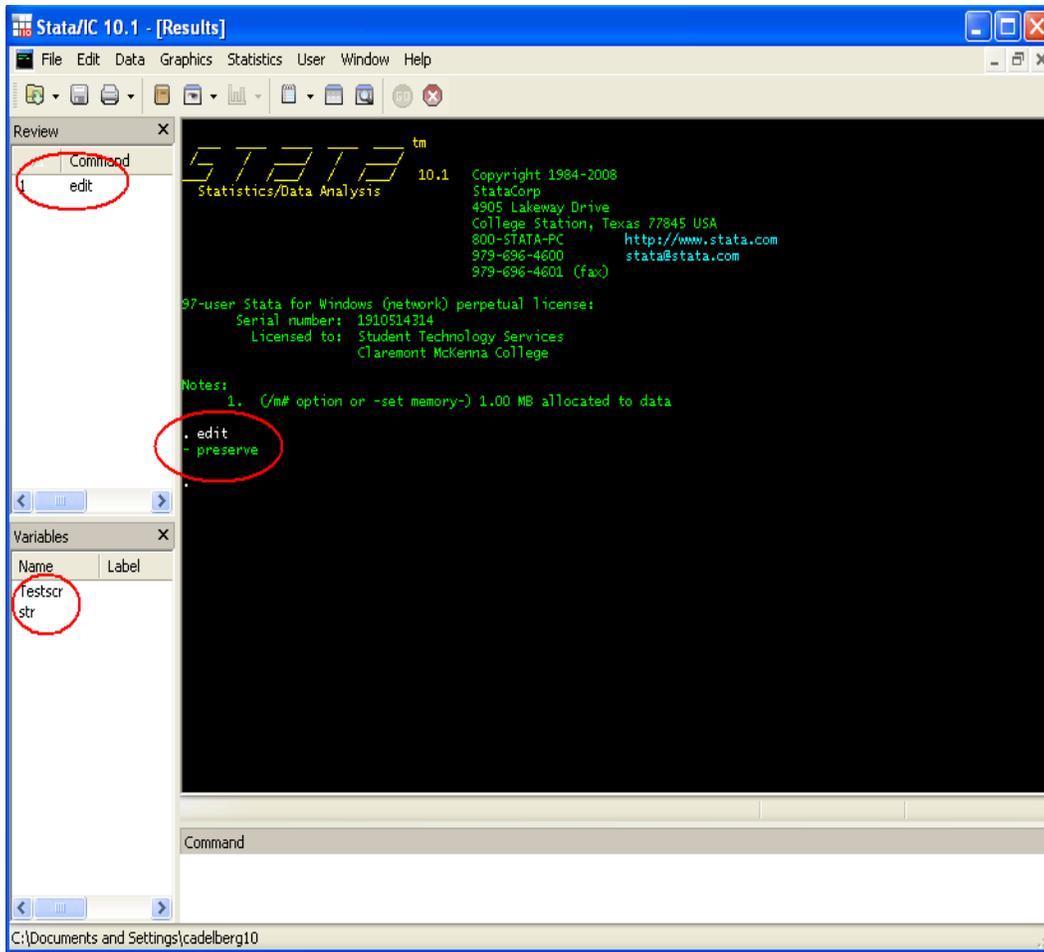
Entering data in this way is very tedious, and you will make data input errors frequently. You will see below how to enter data directly from a spreadsheet or an ASCII file, which are the most common forms of data you will receive in the future.

In general, you can look at variables that already exist by typing in the command

list varname1, varname2, ...

where *varnamei* refers to a variable that exists in your workfile. Try it here by typing

list testscr str

This command will list, one screen at a time, the data on the variables for every observation in the data set. (Missing values are denoted by a period or "." in STATA.) Later on, you will work with large data set, and you will probably not want to see all observations. You can imagine how long this may take with 5,000 observations or more. Failing to look at the data observation by observation of course takes away the ability to spot errors in the data set,

8

perhaps generated by others during data entry. However, there are other methods to spot such problems such as summarizing the data.

You can always stop the listing by hitting the *break* button on the toolbar (it looks like a red pentagon with a white "x" in the middle). This button can be used to stop the execution of any demand in STATA.

You should see the following:
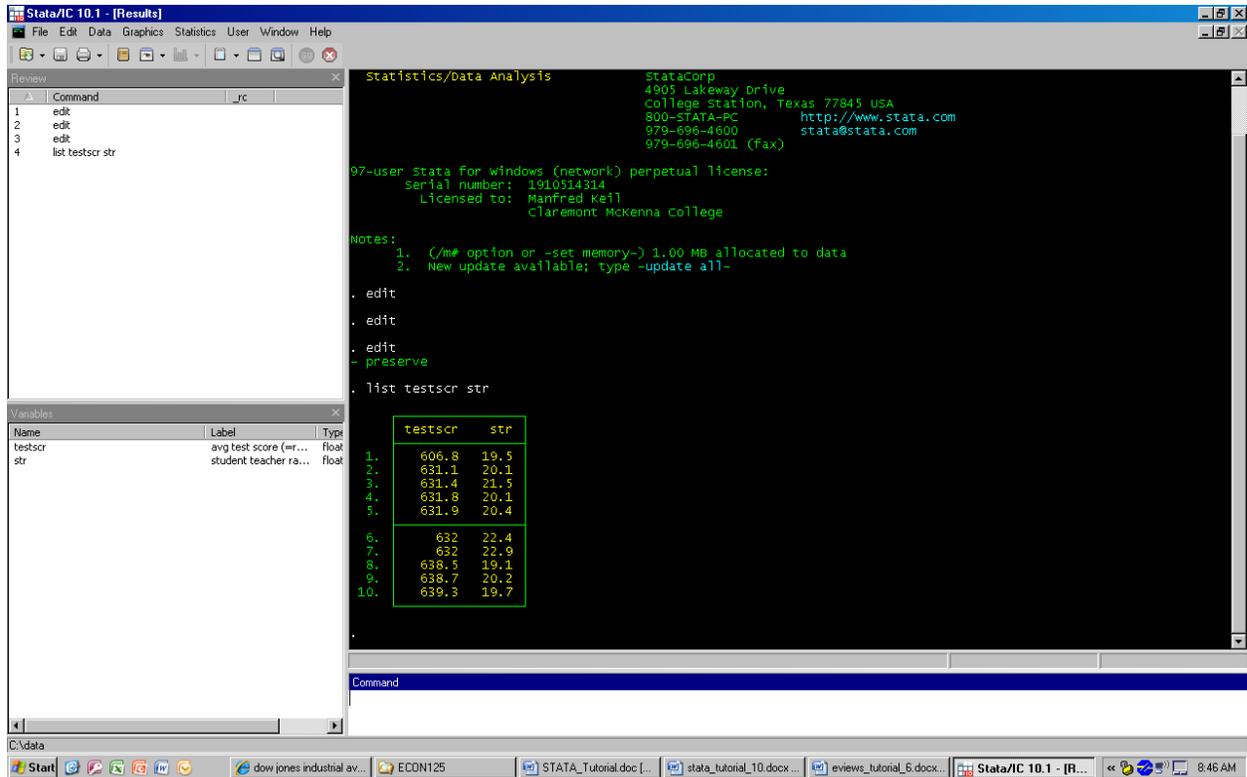
b) Summary Statistics

For the moment, let's just see if we are working with the same data set. Type in the following command

sum testscr str, detail

sum stands for "summarize" and the option *detail* gives you a more extensive list of summary statistics for each of the variables you have entered. These include the median and certain percentiles of the frequency distribution. You will learn later that you can also obtain summary statistics for a subset of your data by adding an *if* or *in* command following the variable name.

```
. sum testscr str, detail

                  avg test score (=read_scr+math_scr)/2)
       ─────────────────────────────────────────────────────────────
           Percentiles      Smallest
     1%        606.8          606.8
     5%        606.8          631.1
    10%       618.95          631.4       Obs                    10
    25%        631.4          631.8       Sum of Wgt.            10

    50%       631.95                      Mean               631.35
                               Largest    Std. Dev.        9.264422
    75%        638.5            632
    90%          639          638.5       Variance         85.82951
    95%        639.3          638.7       Skewness        -1.992948
    99%        639.3          639.3       Kurtosis         6.247294

                  student teacher ratio (teachers/enrl_tot)
       ─────────────────────────────────────────────────────────────
           Percentiles      Smallest
     1%         19.1           19.1
     5%         19.1           19.5
    10%         19.3           19.7       Obs                    10
    25%         19.7           20.1       Sum of Wgt.            10

    50%        20.15                      Mean                20.59
                               Largest    Std. Dev.        1.260908
    75%         21.5           20.4
    90%        22.65           21.5       Variance         1.589888
    95%         22.9           22.4       Skewness         .7828885
    99%         22.9           22.9       Kurtosis         2.295517
```

The summary statistics are explained in Chapter 2 of your textbook (for example, *Kurtosis* is defined in equation (2.15) on page 25 in Stock and Watson (2011).

If your summary statistics differ, then check the data again. To return to the data observations, *edit* the data using the *Data Editor*. Once you have located the data problem, click on the observation and change it. After correcting the problem, press the preserve button again.

Once you have entered the data, there are various things you can do with it. You may want to keep a hard copy of what you just entered. If so, click on the **Print** button. This will print the entire output of what you have produced so far.

In general, it is a good idea to save the data and your work frequently in some form. Many of us have learned through painful experiences how easy it is to lose hours of work by not backing up data/results in some fashion. To save the data set you created, either press the **Save** button or click on **File** and then **Save As**. Follow the usual Windows format for saving files (drives, directories, file type, etc.). If you save datasets in STATA readable format, then you should use the extension ".dta." Once you have saved your work, you can call it up the next time you intend to use it by clicking on **File** and then **Open**. Try these operations by saving the current workfile under the name "*SW10smpl.dta*."

c) Graphical Presentations

Most often it is a good idea to generate graphs ("pictures") to get some "feel" for the data. You will be able to detect outliers which may be the result of data entry errors or you will be able to see if the data "makes sense." Although STATA offers many graphing options, we will only go

through a few commonly used ones here.[1] There are two graphs that you will use most often: line graphs, where one or more variables are plotted across entities (these will become more important in time series analysis when you are plotting variables over time), and scatterplots (crossplots), where one variable is graphed against another.

To create a line graph in a cross section, you can add a third variable in your data set which takes on the number of the observation (here: 1, 2, 3, …, 10). Name it "*obs*" and label it "School *District No.*" Let's plot the student-teacher ratio for the first 10 observations using the *scatter* command. The command is followed by the two variables you would like to see plotted, where the first one appears on the *Y* axis and the second on the *X* axis.

scatter varname1 varname2

plots *variable 1* against *variable 2*. Try this with the student-teacher ratio and the just created variable *obs*.

The resulting graph just gives you the data points here. There are two ways to make this more informative, one is to connect the points by using the *line* command followed by the two variable names. Alternatively you can use the *twoway connected* command to have both the points and the lines displayed. Try both here:

line str obs
twoway connected str obs

After the graph appears, you can edit it using the *Graph Editor* (either use **File** and then **Start Graph Editor** or push the *Graph Editor* button). Alter the graph until it looks like the one below. Some of the alternations can be made in the resulting dialog boxes.

Graph 1
Student-Teacher Ratio Across 10 School Districts

---
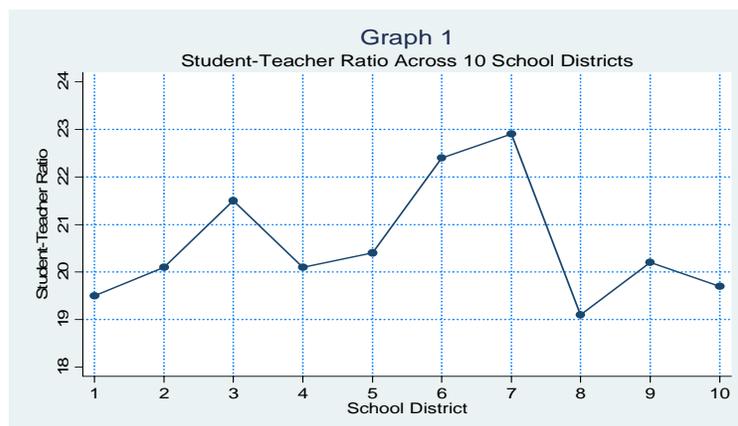
[1] I found the following STATA site particularly useful for graphs:
http://www.stata.com/support/faqs/graphics/gph/statagraphs.html

Frequently you will be interested either in causal relationships between variables or in the ability of one variable to forecast another. As a result, it is a good idea to plot two variables in the same graph.

The first way to look for a relationship is to plot the observations of both variables. This can be done by generalizing the command *twoway connected* to include more than two variable names (one for the *Y* axis and one for the *X* axis). Try this here with

<p style="text-align: center;">*twoway connected str testscr obs*</p>

The resulting graph is pretty uninformative, since test scores and student-teacher ratios are on a different scale. You can allow for two (or more) scales by entering the following command:

<p style="text-align: center;">*twoway (scatter str obs, c(1) yaxis(1)) (scatter testscr obs, c(1) yaxis(2))*</p>

This command instructs STATA to use two *Y* axis, one for the student-teacher ratio on the left side of the graph, and the other for test scores on the right side of the graph. You may want to "beautify" the resulting graph by using the graph editor. See if you can produce something like the graph below:
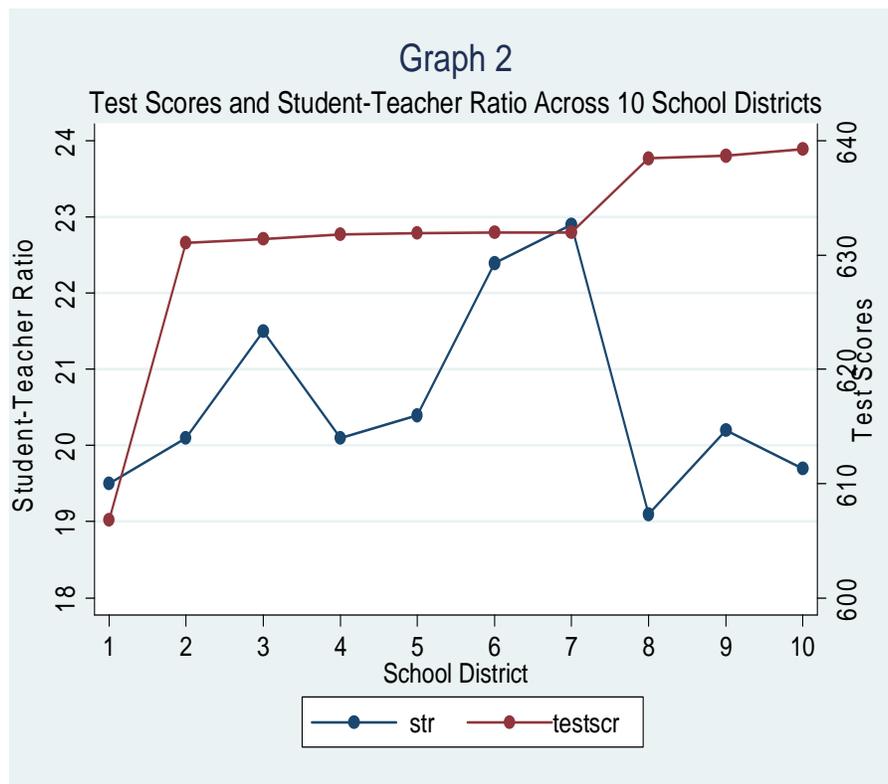
To get an even better idea about the relationship, you can display a two-dimensional relationship in a scatterplot (see page 92 of your Stock and Watson (2011) textbook). Given our discussion above, you could simply use the command *scatter testscr str*. However, you may want to see what a fitted line through that scatter plot would look like, in which case you have to modify the command slightly:

<p align="center">*twoway scatter testscr str || lfit testscr str*</p>

This will result in the following graph (after beautification):

Graph 3

Scatterplot of Test Scores vs. Student-Teacher Ratio
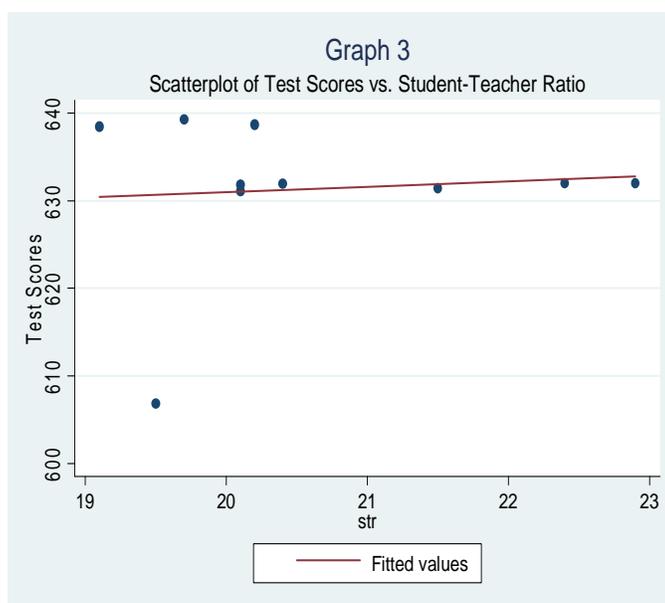
(Not to worry about the positive slope here. Remember, this is a sample, and a very small one at that. After all, you may get 10 heads in 10 flips of a coin.)

d) Simple Regression

There is a commonly held belief among many parents that lower student-teacher ratios will result in better student performance. Consequently, in California, for example, all K-3 classes were reduced to a maximum student-teacher ratio of 20 ("Class Size Reduction Act" – CSR) in the late '90s. This comes at a cost, of course. Initially, it was $1.8 billion a year. At such a high cost, the natural question arises whether or not it is worth it. That is why you are analyzing the effect of reducing student-teacher ratios in Chapters 4-9 of the Stock and Watson textbook.

For the 10 school districts in our sample, we seem to have found a *positive* relationship between larger classes and poor student performance. Not to worry – we will soon work with

all 420 observations from the California School Data Set, and we will then find the negative relationship you have seen in the textbook – for now, we are more concerned about learning techniques in STATA.

In the previous section, we included a regression line in the scatterplot, something that you should have encountered towards the end of your statistics course. However, the graph of the regression line does not allow you to make quantitative statements about the relationship; you want to know the exact values of the slope and the intercept. For example, in general applications, you may want to predict the effect of an increase by one in the explanatory variable (here the student-teacher ratio) on the dependent variable (here the test scores).

To answer the questions relating to the more precise nature of the relationship between class size and student performance, you need to estimate the regression intercept and slope. A regression line is little else than fitting a line through the observations in the scatterplot according to some principle. You could, for example, draw a line from the test score for the lowest student-teacher ratio to the test score for the highest student-teacher ratio, ignoring all the observations in between. Or you could sort the data by student-teacher ratio and split the sample in half so that the observations with the lowest ten student-teacher ratios are in one set, and the observations with the highest ten student-teacher ratios are in the other set. For each of the two sets you could calculate the average student-teacher ratio and the corresponding average test score, and then connect the two resulting points. Or you could just eyeball the relationship. Some of these principles have better properties than others to infer the true underlying (population) relationship from the given sample. The principle of ordinary least squares (OLS), for example, will give you desirable properties under certain restrictive assumptions that are discussed in Chapter 4 of the Stock/Watson textbook.

Back to computing. If the dependent variable, Y, is only determined by a single explanatory variable X in a linear fashion of the type

$$Y_i = \beta_0 + \beta_1 X_i + u_i \qquad\qquad i=1,2, ..., N$$

with "u" representing the error, or random disturbance, not accounted for by the linear equation, then the task is to find a value for β_0 and β_1 . If you had values for these coefficients, then β_1 describes the effect of a unit increase in X on Y. Often a regression line is a linear approximation to an underlying relationship and the intercept β_0 only has a useful meaning if observations around $X=0$ occur in the data. As we have seen in the scatterplot above, there are no observations around the student-teacher ratio of zero, and it is therefore better not to interpret the numerical value of the intercept at all. Your professor most likely will give you a serious penalty in the exam for interpreting the intercept here because with no students present, there is no score to record. (What would be the function of the teacher in that case?)

There are various ways to estimate the regression line. The command for regressing a variable Y on a constant (intercept) and another variable X is:

$$reg\ Y\ X$$

where "*reg*" stands for least squares regression. For the current application, type

$$reg\ testscr\ str,\ r$$

where the "*r*" following the comma indicates that you are using heteroskedasticity-robust standard errors (even though you have not requested an intercept to be included, STATA will automatically do so. There is an option for you to suppress the intercept, but you will most likely never use it).

The output appears as follows:

```
reg testscr str, r

Linear regression                              Number of obs =       10
                                               F(  1,      8) =     0.07
                                               Prob > F      =   0.8013
                                               R-squared     =   0.0068
                                               Root MSE      =   9.7928


                         Robust
    testscr      Coef.   Std. Err.       t    P>|t|     [95% Conf. Interval]

        str   .6069657   2.333494     0.26    0.801    -4.77408    5.988012
      _cons   618.8526   51.06079    12.12    0.000    501.1062     736.599
```

According to these results, lowering the student-teacher ratio by one student per class results in an decrease of 0.6 points, on average, in the districtwide test score. Using the notation of your textbook, you should display the results as follows:

$$\widehat{TestScore} = 618.9.1 + 0.61 \times STR,\ R^2 = 0.007,\ SER = 9.8$$
$$(51.1)\qquad(2.33)$$

Note that the result for the 10 chosen school districts is quite different from the sample of all 420 school districts. However, this is a rather small sample and the regression R^2 is quite low. As a matter of fact, in Chapter 5 of your textbook, you will learn that the above slope coefficient is not statistically significant.

e) Entering Data from a Spreadsheet

So far you entered data manually. Most often you will work with larger data sets that are *external* to the STATA program, i.e., they will not be included in, or be part of, the program itself. This makes sense as data sets either become very large or are generated by another program, such as a spreadsheet.

Stock and Watson present the California Test Score Data Set in Chapter 4 of the textbook.

Locate the corresponding Excel file *caschool.xls* on the accompanying web site (where you found this tutorial) and open it. Next, following the procedures discussed previously, start STATA and open the Data Editor. Return to the Excel file and mark F1:R421. Next, using the "*copy*" and "*paste*" commands common to Windows programs, move the data block to STATA. You are presumably familiar with this procedure. Make sure to select the grey box to the immediate right of "1" before pasting. Note that STATA has conveniently included the name of the variables in the Data Editor.

This is what you should see in STATA:
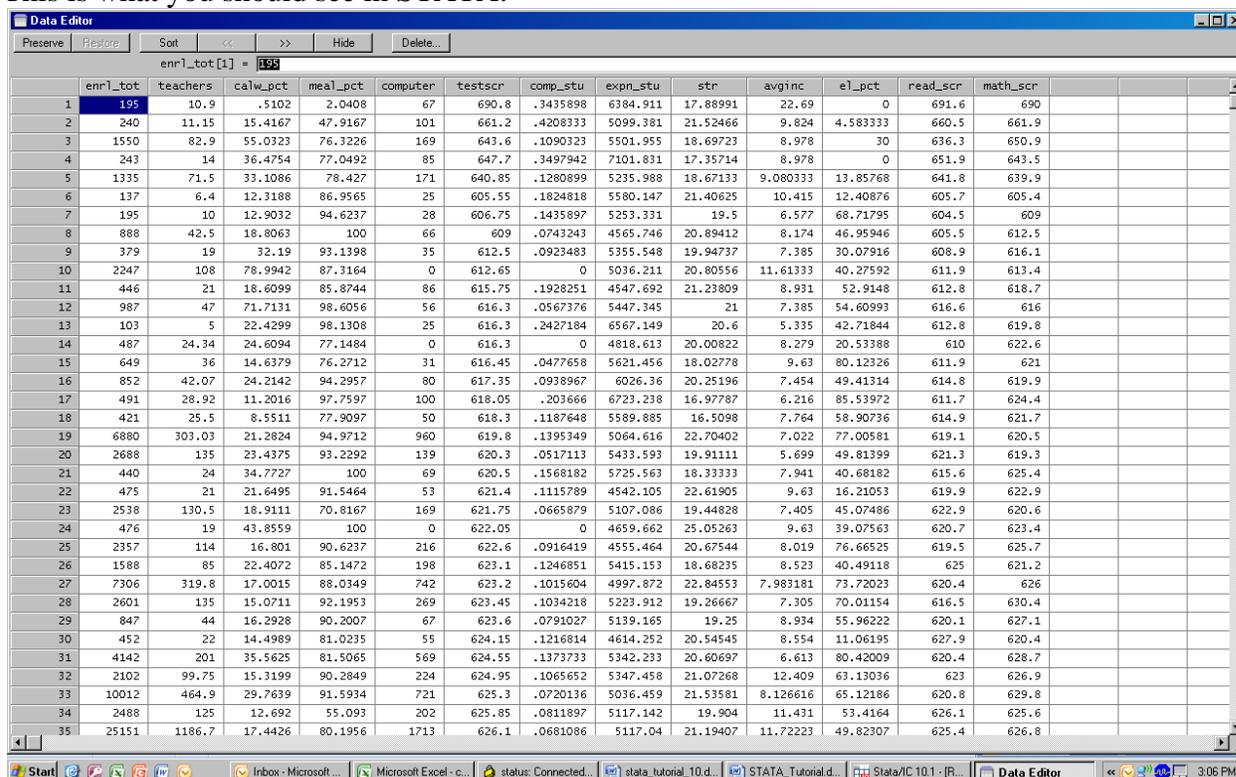
When you are done, push **Preserve**, and you are ready to save the file. Name it *caschool.dta.*

You can now reproduce Equation (4.7) from the textbook. Use the regression command you previously learned to generate the following output.

```
. reg testscr str, r

Linear regression                                Number of obs =      420
                                                 F(  1,   418) =    19.26
                                                 Prob > F      =   0.0000
                                                 R-squared     =   0.0512
                                                 Root MSE      =   18.581
```

testscr	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf. Interval]	
str	-2.279808	.5194892	-4.39	0.000	-3.300945	-1.258671
_cons	698.933	10.36436	67.44	0.000	678.5602	719.3057

(You can find the standard errors and the *t*-statistic on p. 129 of the Stock and Watson (2011) textbook. The regression R^2, sum of squared residuals (*SSR*), and standard error of the regression (*SER*) are presented in Section 4.3.)

f) Importing Data Files directly into STATA

Excel (Spreadsheet) Files

Even though the *cut and paste* method seemed straightforward enough, there is a second, more direct way to import data into STATA from Excel, which does not involve copying and pasting data points.

Start again with a new STATA file. In general, make sure your data is organized with the variable names in Row 1 of your spreadsheet with each column representing a different variable, and the observations in the rows beneath the variable names. Then, save your data set in Excel (or an alternative spreadsheet program) as a *.csv* file (this stands for comma separated values). Next, type the following command into the command window in STATA:

insheet using (filename)

where *(filename)* is the directory location of your file. (To find this, locate the file and right-click, selecting the **Properties** button. This should contain the location of the file to which you must add the filename; example C:\Econometrics\StockWatson\caschool.csv.) If your filename has any spaces or any symbol that appears on the number keys of the keyboard, then you should put quotation marks around your filename. STATA reads spaces as denoting separations between words, and therefore will only read the filename up until the first space or symbol, and then considers the rest to be a separate command.

NOTE: In order to *insheet* data, there must be no data already stored in memory. To get rid of any data that is already stored, type the command

clear

before "insheeting."

Once you have *insheeted* your data, you should see this reflected in your *Results box* and your variables should appear in your *Variables List box*. You can type *edit* to see your data in the data editor.

To save your data as a STATA file, click on **File** on the upper toolbar, then select **Save As**. When you save your file, make sure it is saved as a **.dta** file. This type of file can only be opened in STATA. Alternatively, you can type the command

save (filename)

where *(filename)* is the directory location and name of your file. If you have a previous version of this saved already, to overwrite the old version add *replace* after the save command. For example:

clear
save "C:\My Documents\test.dta", replace

If you wish to save a file that has been previously saved in the same directory location as the previous version, you may use the command *save, replace*.

Note: When you save a STATA dataset, you are really only saving the dataset as it exists at the time you chose to save. You are not retaining any of the analysis you may have conducted, although if you have changed the data since opening the file, these changes will be reflected.

As an exercise, copy the *caschool.xls* or *caschool.xlsx* data file from the Stock and Watson website and save the Excel file in some subdirectory on your computer as a .csv file. Then import the data set using the *insheet* command. Finally run the simple regression of *testscr* on *str* and check that your output contains 420 observations and corresponds to the STATA regression output in the previous section.

ASCII data

You can also import data from an ASCII file (text file). This assumes that you either saved data from a different source as an ASCII file or that you received data in ASCII file format. The file must be organized with one observation in each row, and the variables in the data set must be in separate columns.

Using the *infile* command, type the name of the variable that represents each column, followed by the file name. For example, consider an ASCII dataset that looks as follows:

```
10.75  12     6      1      0
16.50  16     3      0      0

…..

12.10  12     8      1      1
```

and which you want to import into STATA. Each row corresponds to observations on an entity (here an individual). The first columns above is the hourly wage, the second is years of education, the third is potential experience, the fourth is a binary variable which equals one if the individual belongs to a union and is zero otherwise, and the last column is another binary variable which takes on the value of one if the individual is married and is zero otherwise.

To import the data, you type the following command:

STATA dataset

To open a dataset that is already saved as a **.dta** file, you can either go to **File** and then **Open** to select your dataset, or you can type the command

use (filename)

This will open your dataset into STATA.

You can try doing this with the *caschool.dta* data set from the Stock and Watson website. Simply save that data set on your computer, then double click on it. This will open STATA with the data loaded already. Obviously this is the easiest method to import data into STATA.

g) Multiple Regression Model

Economic theory most often suggests that the behavior of a certain variable is influenced not only by a single variable, but by a multitude of factors. The demand for a product, e.g. LA Laker tickets, depends not only on the price of the product but also on the price of other goods, income, taste, etc. Similarly, the Phillips curve suggests that inflation depends not only on the unemployment rate, but also on inflationary expectation and possibly supply shocks, etc.

An extension of the simple regression model is the multiple regression model, which incorporates more than one regressor (see Equation (6.7) in the textbook on page 189).

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_k X_{ki} + u_i, \; i = 1, \ldots, n.$$

To estimate the coefficients of the multiple regression model, you proceed in a similar way as in the simple regression model. The difference is that you now need to list the additional explanatory variables. In general, the command is:

reg Y X1 X2 ... Xk, (options)

where *(options)* can be omitted (this is the default and gives you homoskedasticity-only standard errors) or can be replaced by various possible entries (e.g. "*r*" for heteroskedasticity robust standard errors).

See if you can reproduce the following regression output, which corresponds to Column 5 in Table 7.1 of the Stock and Watson (2011) textbook (page 238). The option used below is *(r)* to produce heteroskedasticity-robust standard error (STATA refers to these as "Robust Standard Errors").

```
. reg testscr str el_pct meal_pct calw_pct, r

Linear regression                                   Number of obs =      420
                                                    F(  4,   415) =   361.68
                                                    Prob > F      =   0.0000
                                                    R-squared     =   0.7749
                                                    Root MSE      =   9.0843

─────────────┬───────────────────────────────────────────────────────────────
             │              Robust
     testscr │      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
─────────────┼───────────────────────────────────────────────────────────────
         str │  -1.014353   .2688613    -3.77   0.000    -1.542853   -.4858534
      el_pct │  -.1298219   .0362579    -3.58   0.000     -.201094   -.0585498
    meal_pct │  -.5286191   .0381167   -13.87   0.000    -.6035449   -.4536932
    calw_pct │  -.0478537   .0586541    -0.82   0.415    -.1631498    .0674424
       _cons │   700.3918   5.537418   126.48   0.000      689.507    711.2767
─────────────┴───────────────────────────────────────────────────────────────
```

The interpretation of the coefficients is equivalent to that of a controlled science experiment: it indicates the effect of a unit change in the relevant variable on the dependent variable, *holding all other factors constant* ("ceteris paribus").

Section 7.2 of the Stock and Watson (2011) textbook discusses the *F*-statistic for testing restrictions involving multiple coefficients, the so called Wald test. To test whether all of the above coefficients are zero with the exception of the intercept, you can use the *test* command followed by each restriction that you want to test in parenthesis (STATA uses the name of the variable associated with the coefficient in combination with the restriction). Type

Test (str=0) (el_pct=0) (meal_pct=0) (calw_pct=0)

STATA will generate the following output:

```
. test (str=0) (el_pct=0) (meal_pct=0) (calw_pct=0)

 ( 1)   str = 0
 ( 2)   el_pct = 0
 ( 3)   meal_pct = 0
 ( 4)   calw_pct = 0

       F(  4,   415) =  361.68
            Prob > F =   0.0000
```

Note that the *F*-statistic is identical to the same statistic listed in the regression output.

See if you can generate the *F*-statistic of 5.43 following Equation (7.6) in the Stock and Watson (2011) text and listed at the bottom of page 223 (restrict the coefficients of *STR* and *Expn* to be zero).

h) Data Transformations

So far, we have only used data in regressions that already existed in some file that we either created or used. Almost always, you will be required to transform some of the raw data that you received before you run a regression. In STATA you transform variables by using the "*gen*" (as in generate) command. For example, Chapter 8 of the Stock/Watson textbook introduces the polynomial regression model, logarithms, and interactions between variables.

20

Let us reproduce Equations (8.2), (8.11), (8.18), and (8.37) here. The following commands generate the necessary variables[2]:

> *gen avginc2=avginc^2*
> *gen avginc3=avginc^3*
> *gen lavginc=log(avginc)*
> *gen ltestscr=log(testscr)*
> *gen strpctel=str*el_pct*

Note how the commands and generated variables are displayed in STATA, including those in red when you make a mistake in the command (e.g. genr instead of gen).
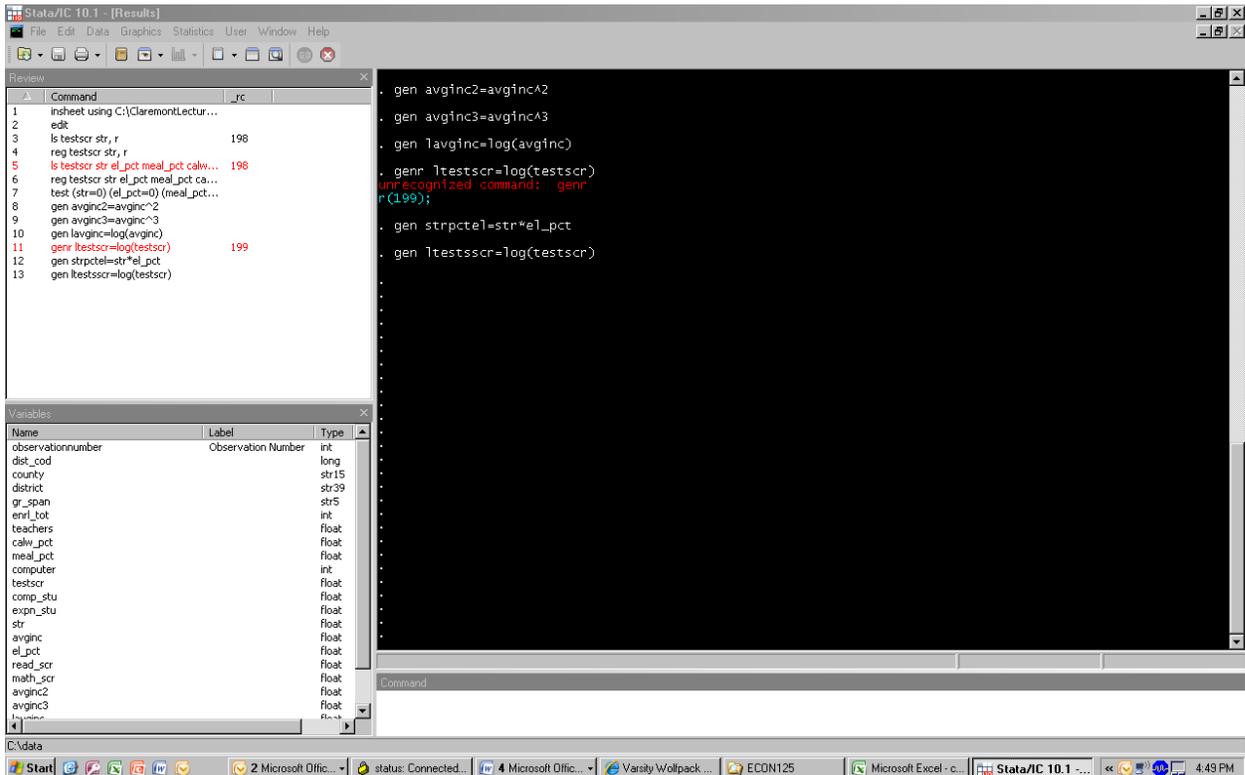
Next run the four regressions using the same technique as for multiple regression analysis. Finally save your workfile again and exit STATA.

Exercise

One of the problems with the type of tutorial you are working on is that you just follow

---

[2] For example, I have generated a variable called "avginc2", and assigned it to be the square of the previously defined variable "avginc". Note that I am generating variable names that are self-explanatory. They could have been called "variable1", "variable2", "variable3", etc. but it is a good idea to create variable names that you can remember.

instructions without internalizing them. A typical student will finish the tutorial with few problems but then little is retained. If I asked you to retrieve a data set and to run a few regressions, for example, would you be able to do that? Or would you say "how do I do this?"

Let's see how much you understood. Go to the Stock and Watson website for the 3rd edition (http://www.pearsonhighered.com/stock_watson). Go to *Student Resources* in the *Companion Web Site*, and download the CPS data set for Chapter 8 (*Data Sets for Replicating Empirical Results*: **CPS Data Used in Chapter 8**). Next open STATA (note that if you just double click on the cps_ch8.dta file, an error message will occur that tells you that insufficient memory was allocated). Before you open the *cps_ch8.dta* file, increase your memory (usually set at 1 MB by default). You can do this by typing in the command

set mem 10m

which increases the memory to 10 megabytes. In general, make sure to set the memory large enough to handle the data set, but small enough for your computer to handle the program (use *k* for kilobyte, *m* for megabyte, and *g* for gigabyte).

Then replicate the results for columns (1) from Table 8.1 on page 284 of the Stock and Watson (2011) textbook. Why do you think your results differ from those listed in the table? What if you found a way to restrict your sample to only include individuals who are at least 30 but not older than 64? To find a way to restrict your sample, look for **Help** and the *if* command. Then, restricting your sample to those individuals in that age group, replicate columns (1) to (3). For column (4), define *potential experience* as the Mincer experience variable (*age – Years of education – 6*).

Batch Files

So far, you have either clicked on buttons in STATA or used the "*Command Window*" to type executable statements (commands one by one, or line by line). But what if you wanted to keep a permanent record of all the transformations you made, regressions you tried, graphs you created, etc.? In that case, you would need to create a "program" that consists of a list of line commands similar to those that you used in the "*Command Window*" previously. After having created such a program, which is a "text" or "Ascii" file, you can then execute ("run") it and view the output afterwards (if you did not contain any errors). Batch files can also include loops and conditional branching (if you don't know what these are, not to worry). Batch files in STATA are called *Do-Files*.

Using STATA in batch mode has two important advantages over using STATA interactively:

- the *Do-File* provides an audit trail for your work. The file provides an exact record of each STATA command;
- 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 is 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.

Let's create such a program. Click on **New Do-File Editor** button. This opens the "*STATA Do-File Editor*" box. Type in, the following commands exactly as they appear below.

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

Here is the meaning of the seven lines of this program:

Line 1: This is an administrative command that tells STATA where to display the results of your analysis. STATA output files are called *log* files. The current line tells STATA to open a *log* file called *stata1.log* (you could have used any name, such as *love_metrix.log*, meaning, the word "stata1" is not required here). If there is already a file with the same name in the folder, STATA is instructed to replace it. Before you save the *Do-File*, replace the path in this line with the relevant path on the computer you are using.

Line 2: This line concerns the data set. As you learned earlier in the tutorial, datasets in STATA are called *dta* files. The dataset which you will use here is *caschool.dta*, which you downloaded earlier. The current line tells STATA the location and name of the dataset to be used for the analysis. Before you save the *Do-File*, replace the path in this line with the relevant path of the location where you saved *caschool.dta* to.

Line 3: This line also concerns the data set. It tells STATA to "describe" the dataset (a shorter version of the command is "*des*" instead of "*describe*"). This command produces a list of the variable names and any variable descriptions stored in the data set.

Line 4: This line tells STATA to create a new variable called *income* (a shorter version of the command is "*gen*" instead of "*generate*"). 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.

Line 5: This line tells STATA to compute some summary statistics (a shorter version of the command is "*sum*" instead of "*summarize*"). STATA will produce the mean, standard

deviation, etc.

Line 6: This line closes the file *stata1.log* which contains the output.

Line 7: This line tells STATA that the program has ended.

As long as you have replaced the path in line 1 and line 2 with the relevant paths from the computer you are working on, and if you downloaded/saved the California Test Score Data Set, then we are good to go. Save the *Do-File*, using the .*do* suffix. Next execute this *Do-File* by first opening STATA on your computer. Next, click on the **File** menu, then **Do…**, and then select the *stata1.do* file you just saved. This will "run" or "execute" the program.

You will be able to see the program being executed in the *Results Window*. Since the execution will not fit into one screen, you can scroll up and see everything that happened during the "run." Sometimes (although not here) you may 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.

To exit STATA, click on the usual exit button at the top right of STATA (alternatively click on **File** and then **Exit**.) STATA will ask you if you really want to exit, and you will respond **Yes**.

Your output has been saved in *stata1.log* and you can look at it by opening the file with any text editor (Notepad, for example) or in Word/WordPerfect. Here is what you should see:

```
-----------------------------------------------------------------------------------------------------------------------
 log:  your path here
 log type:  text
 opened on:   your date and time here

. use \your path here

. describe

Contains data from \your path here\caschool.dta
  obs:        420
  vars:        18                    15 Dec 2010 07:57
  size:      60,060 (94.3% of memory free)
-----------------------------------------------------------------------------------------------------------------------
         storage  display    value
variable name  type  format    label     variable label
-----------------------------------------------------------------------------------------------------------------------
observat       float  %9.0g
```

```
dist_cod      float  %9.0g
county        str18  %18s
district      str53  %53s
gr_span       str8   %8s
enrl_tot      float  %9.0g
teachers      float  %9.0g
calw_pct      float  %9.0g
meal_pct      float  %9.0g
computer      float  %9.0g
testscr       float  %9.0g
comp_stu      float  %9.0g
expn_stu      float  %9.0g
str           float  %9.0g
avginc        float  %9.0g
el_pct        float  %9.0g
read_scr      float  %9.0g
math_scr      float  %9.0g
```

Sorted by:

. generate income = avginc*1000

. summarize income

```
   Variable |    Obs      Mean   Std. Dev.    Min      Max
-------------+---------------------------------------------------------
     income |    420   15316.59   7225.89    5335    55328
```

. log close
   log: C:\your path here\stata1.log
 log type: text
 closed on: your date and time here

You now have an initial idea of how to work with *Do-Files* in STATA. The rest of this part of the tutorial will guide you through further commands and make the initial *Do-File* more complex. I suggest that you continue to work with the batch file you just created and then for you to add new lines to this program (if you use the *.pdf* version of this tutorial or have printed the tutorial using a color printer, then the new commands will appear in red).

```
#delimit ;
********************************************************;
*Administrative Commands;
********************************************************;
set more off;
clear;
log using \statafiles\stata1.log, replace;
********************************************************;
*Read in the Dataset;
********************************************************;
use \statafiles\caschool.dta;
des;
********************************************************;
*Transform Data and Create New Variables;
********************************************************;
***** Construct Average District Income in $s;
gen income = avginc*1000;
********************************************************;
*Carry Out Statistical Analysis;
********************************************************;
***** Summary Statistics for Income;
sum
        income;
********************************************************;
*End of Program;
********************************************************;
log close;
exit;
```

The new version of the *Do-File* carries out exactly the same calculations as before. However it uses four features of STATA for more complicated analysis. 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 the following line. This is useful because complicated commands in STATA are often too long to fit on a single line. (Make sure to place a ";" at the end of the seven old commands.) The above *Do-File* contains an example of a STATA command written on two lines: near the bottom of the file you see the command *sum 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 quite long, so it is good to get used to employing 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 (again, don't forget to add the seven ";" in the first version of the program).

The second new feature of the above *Do-File* is that many of the lines begin with an asterisk.

STATA ignores the text that comes after "*", so that these lines can be used for comments or to describe what the commands that follow are doing. Note that each of these lines ends with a semicolon. Without the semicolon, STATA would include the next line as part of the text description.

A final new feature in the program is the command

<p style="text-align:center">set more off</p>

This command eliminates the need to hit a key on your keyboard in the case when STATA fills the *Results Window* and stops displaying further results (the *-- more --* would appear); and there is the new command

<p style="text-align:center">clear</p>

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

Run the program and have a look at the new *log file*.

Next, change the previous version of the *Do-File* by adding commands until the new version looks as follows (again, new commands can be seen in red if your tutorial displays colors):

```
#delimit ;
*****************************************************;
*Administrative Commands;
*****************************************************;
set more off;
clear;
log using \statafiles\stata1.log, replace;
*****************************************************;
*Read in the Dataset;
*****************************************************;
use \statafiles\caschool.dta;
des;
*****************************************************;
*Transform Data and Create New Variables;
*****************************************************;
***** Construct Average District Income in $s;
gen income = avginc*1000;
***** Define variables for subset of data;
gen testscr_lo = testscr if (str<20);
gen testscr_hi = testscr if (str>=20);
*****************************************************;
*Carry Out Statistical Analysis;
*****************************************************;
***** Summary Statistics for Income;
```

```
sum
      income;
sum 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);
ttest testscr_lo=testscr_hi, unequal unpaired;
****************************************************;
*End of Program;
****************************************************;
log close;
exit;
```

There are three new features in this new version.

1)   New variables are created 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

$$gen\ testscr\_lo = testscr\ if\ (str<20)$$

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*).

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**

2)   The *ttest* command constructs tests and confidence intervals for the mean of a population or for the difference between two means (see Stock and Watson, 2011; 70-90). The command is used in two different ways in the program.

The first 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 really of interest, but the confidence interval for the mean is what we are looking for in this example.) The same command is then used for *testscr_lo* and *testscr_hi* (see section 3.2 and 3.3 in Stock and Watson (2011)).

The second form of the command is

ttest testscr_lo=testscr_hi, unequal unpaired

Executing this statement will test the hypothesis that *testscr_lo* and *testscr_hi* come from populations with the same mean. That is, the command computes the *t*-statistic for the null hypothesis that the (population) mean of test scores for districts with class sizes less than 20 students is the same as the mean of test scores for districts with class sizes greater than 20 students. The 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 (see section 3.4 in Stock and Watson (2011)).

3)   A third new feature in the *Do-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 define by the *gen* 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. In essence, the command instructs the program to overwrite the previously stored data.

You are now ready to execute ("run") the program as done before.

As before, change the previous version of the *Do-File* by adding commands until the new version looks as follows (again, new commands can be seen in red if your tutorial displays colors):

```
#delimit ;
********************************************************;
*Administrative Commands;
********************************************************;
set more off;
clear;
log using \statafiles\stata1.log, replace;
********************************************************;
*Read in the Dataset;
********************************************************;
use \statafiles\caschool.dta;
des;
********************************************************;
*Transform Data and Create New Variables;
********************************************************;
***** Construct Average District Income in $s;
gen income = avginc*1000;
***** Define variables for subset of data;
gen testscr_lo = testscr if (str<20);
gen testscr_hi = testscr if (str>=20);
********************************************************;
*Carry Out Statistical Analysis;
********************************************************;
***** Summary Statistics for Income;
sum
        income;
********************************************************;
***** Table 4.1 *****;
********************************************************;
sum str testscr, detail;
********************************************************;
***** Figure 4.2 *****;
********************************************************;
twoway scatter testscr str || lfit testscr str;
********************************************************;
***** Correlation *****;
********************************************************;
cor str testscr;
********************************************************;
***** Equation 4.11  and 5.8 *****;
********************************************************;
reg testscr str, robust;
********************************************************;
***** Equation 5.18 *****;
gen d = (str<20);
reg testscr d, r;
********************************************************;
sum 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);
ttest testscr_lo=testscr_hi, unequal unpaired;
********************************************************;
*End of Program;
********************************************************;
log close;
exit;
```

The new commands reproduce some of the empirical results shown in Chapters 4 and 5 of Stock and Watson (2011). There are several features of STATA included in the new commands which have not been used in the previous examples:

1) The *summarize* command ("*sum*") is now includes the option *detail*, which provides more detailed summary statistics. The command is written as

<div align="center">

sum str testscr, detail

</div>

This command tells STATA to compute summary statistics for the two variables *str* and *testscr*. The option *detail* produces detailed summary statistics that include, for example, the percentiles that are reported in Table 4.1 on p. 113 of Stock and Watson (2011).

2) The command

<div align="center">

twoway scatter testscr str || lfit testscr str

</div>

constructs a scatterplot of *testscr* versus *str* and includes the estimated regression line for the simple regression of the California Test Score Data Set, shown on p. 116 of Stock and Watson (2011).

3) The command

<div align="center">

cor str testscr

</div>

tells STATA to compute the correlation between the student teacher ratio and test scores.

4) Next you will reproduce equations (4.11) and (5.8) in Stock and Watson (2011) by using the *regress* (or short *reg*) command:

<div align="center">

reg testscr str, r

</div>

instructs STATA to run an OLS regression with *testscr* as the dependent variable and *str* as the regressor. The *robust* (short *r*) option tells STATA to calculate heteroskedasticity-

robust formulas for the standard errors of the regression coefficient estimators. Omitting this option results in the display of homoskedasticity-only standard errors.

5) The final innovation over the previous version of the *Do-File* is contained in the two commands following the line *Equation 5.18*. First a binary (sometimes referred to as dummy or indicator) variable "*d*" is created suing the STATA command

$$gen\ d = (str<20)$$

This variable is equal to 1 if the expression in parenthesis is true, that is, when the student teacher ratio is less than 20. Otherwise it is equal to 0, in other words, when the expression is false, or when the student teacher ratio $\geq$ 20. STATA allows you to use any of the relational operators defined above. The final *regression* command tells STATA to run a regression of test scores on the binary variable just created. The output reproduces equation (5.18) on p. 155 of Stock and Watson (2011).

Run the program now and look at the output in the *log*-file.

The upcoming *Do-File* will be the last program in this tutorial. Having understood all five should give you a solid grounding in STATA programming. As before, there are several commands added to the previous version of the *Do-File*. Add these commands to your older version until the new version looks as follows (new commands can be seen in red if your tutorial displays colors):

```
#delimit ;
****************************************************;
*Administrative Commands;
****************************************************;
set more off;
clear;
log using \statafiles\stata1.log, replace;
****************************************************;
*Read in the Dataset;
****************************************************;
use \statafiles\caschool.dta;
des;
****************************************************;
*Transform Data and Create New Variables;
****************************************************;
***** Construct Average District Income in $s;
gen income = avginc*1000;
***** Define variables for subset of data;
gen testscr_lo = testscr if (str<20);
gen testscr_hi = testscr if (str>=20);
****************************************************;
*Carry Out Statistical Analysis;
****************************************************;
***** Summary Statistics for Income;
```

```
sum
     income;
******************************************************.
***** Table 4.1 *****;
******************************************************,
sum str testscr, detail;
******************************************************,
***** Figure 4.2 *****;
******************************************************.
twoway scatter testscr str || lfit testscr str;
******************************************************,
***** Correlation *****;
******************************************************,
cor str testscr;
******************************************************,
***** Equation 4.11  and 5.8 *****;
******************************************************,
reg testscr str, r;
******************************************************.
***** Equation 5.18 *****;
gen d = (str<20);
reg testscr d, r;
******************************************************,
sum 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);
ttest testscr_lo=testscr_hi, unequal unpaired;
******************************************************,
***** Table 6.1 *****;
******************************************************,
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;
******************************************************,
***** Equation 7.5 *****;
******************************************************,
reg testscr str el_pct, r;
******************************************************,
```

```
***** Equation 7.6 *****;
*******************************************************;
replace expn_stu = expn_stu/2000;
reg testscr str expn_stu el_pct, r;
*******************************************************;
* Display Variance-Covariance Matrix;
*******************************************************;
vce;
*******************************************************;
***** F-test report in text;
*******************************************************;
test str expn_stu;
*******************************************************;
***** Correlations reported in text;
*******************************************************;
cor testscr str expn_stu el_pct meal_pct calw_pct;
*******************************************************;
*****Table 7.1 *****;
*******************************************************;
* Column (1);
reg testscr str, r;
display "adjusted Rsquared = " e(r2_a);
* Column (2);
reg testscr str el_pct, r;
display "adjusted Rsquared = " e(r2_a);
* Column (3);
reg testscr str el_pct meal_pct, r;
display "adjusted Rsquared = " e(r2_a);
* Column (4);
reg testscr str el_pct calw_pct, r;
display "adjusted Rsquared = " e(r2_a);
* Column (5);
reg testscr str el_pct meal_pct calw_pct, r;
display "adjusted Rsquared = " e(r2_a);
*******************************************************;
***** Appendix – rule of thumb F-Statistic;
*******************************************************;
reg testscr str expn el_pct;
test str expn;
reg testscr el_pct;
*******************************************************;
*End of Program;
*******************************************************;
log close;
exit;
```

The file produces several of the empirical results from Chapter 7 of Stock and Watson (2011). As before, some commands have been abbreviated when there is no possibility of confusion. The file uses abbreviations for STATA commands throughout (*generate* becomes *gen*, *regress* turns into *reg*, etc.).

In essence there are two new commands:

1) The first new command involves the testing of restrictions in equation 7.6 (page 218 of Stock and Watson (2011)). The command

reg testscr str expn_stu el_pct, r

instructs STATA to compute the regression. The command *vce* asks STATA to print out the estimated variances and covariances of the estimated regression coefficients. The command

test str expn_stu

gets STATA to carry out the joint test that the coefficients on *str* and *expn_stu* are both equal to zero.

2) The second new command is in the analysis of Table 7.1 on page 238 of Stock and Watson (2011). When STATA computes an OLS regression, it computes the adjusted R-squared ($\overline{R}^2$) as described in Section 6.4, page 194 of Stock and Watson (2011). However, STATA does not display all of the results it computes, including the adjusted R-squared. The command

display "Adjusted Rsquared = " e(r2_a)

instructs STATA to print out ("display") the adjust R-squared. Whatever appears between the two quotation marks (" ") will be displayed in your output (you did not have to display the words *Adjusted Rsquared* but could have chosen anything else, such as *My Measure of Fit*). However *e(r2_a)* tells STATA where to retrieve the stored result from and cannot be changed. The adjusted R-squared is not the only statistic that STATA stores and does not display. You can use the **Help** function or look in the *Reference* volume under *Saved Results* for the *reg* command to find other statistics.

Other Examples of *Do-Files*

You will find other examples of *Do-Files* on the accompanying Web site for the Stock and Watson (2011) econometrics textbook. You can download STATA *Do-Files* fro there to reproduce all of the analysis in Chapters 3-13. You will also find a STATA *Do-File* for the time series chapters 14-16 there. 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 14-16, as well as a tutorial for EViews.

3. 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 brackets [ ]. 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 sources listed at the beginning of this tutorial.

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 for your econometrics course. You should extend the list or create your own in addition to what is listed here.

Administrative Commands

 # delimit
> sets the character that marks the end of a command. For example, the command *# delimit ;* tells STATA that all commands will end with a semicolon. This command is used in *Do-Files*.

clear
> deletes/erases all variables from the current STATA session.

exit
> in a *Do-File*, the command tells STATA that the program has ended. If you type *exit* in the STATA *Command Window*, then STATA will close.

log
> controls STATA log files, which is where STATA writes output. There are two common uses of this command:
>
> - *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 bollom 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 #
> sets the maximum number of variables that can be used in a regression. The default maximum is 40. If you have a huge number of observations and want to run a regression with 45 variables, then you will need to use the command, where *#* is a number greater than 45.

set memory #m
>   is used in Windows and Unix versions of STATA to set the amount of memory used by the program. For details, see the discussion within the tutorial.

set more off
>   tells STATA not to pause and display the *–more—*message in the *Results Window*.

Data Management

describe
>   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
>   *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 deletes all of the other variables in the current STATA session.

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

save filename [, replace]
>   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*.

use filename
>   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 by using the command *gen ts_squared = testscr^2*.)

You can also use relational operators to construct binary variables. For example, in the forth batch file, the following command was included

$$gen\ d = (str<20);$$

This 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

$$gen\ ln\_ts = ln(testscr)$$

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

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

$$gen\ testscr\_lo = testscr\ if\ (str<20);$$

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

Statistical Operations

cor list of variables
> tells STATA to compute the correlation between each of the variables in *list of variables*

twoway scatter var1 var2 || lfit var1 var2
> produces a scatter plot of *var1* on the *Y*-axis and *var2* on the *X*-axis. If the || *lfit* part is included then the fitted OLS line is also displayed

predict newvarname [, residuals]
> when this command follows the *regress* command, the OLS predicted values or residuals are calculated and stored under the name *newvarname*. When the option *residuals* is used, the residuals are computed; otherwise the predicted values are computed and placed into *newvarname*.

Example:

<p align="center">reg testscr str expn_stu el_pct, r
predict tshat
predict uhat, residuals</p>

Here, *testscr* is regressed on *str*, *expn_stu*, *el_pct* (first command); the fitted values are saved and stored 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 the coefficient estimators. The option *noconstant* tells STATA not to include a constant (intercept) in the regression.

Examples:

<p align="center">reg testscr str, r
reg testscr str expn_stu el_pct, r</p>

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:

sum testscr str

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

sum testscr str, detail

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

test
> this command is used to test hypothese 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 list is to be carried out on the coefficients corresponding to the variables given in *list of variables*.

Example:

reg testscr str expn_stu el_pct, r
test str expn_stu

> Here *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 thest 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(#)]

> Here you test the null hypothesis that the population mena 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 computed a 90% confidence interval.

ttest testscr = 0 if (str<20);

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

ttest varname1 = varname 2 [if expression] [, level(#) unpaired unequal]

> tests 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 paried (they are not panel data), and the option *unequal* means that the population variables may not be the same. (Section 3.4 of Stock and Watson (2011) describes the equality of means tests under the *unpaired* and *unequal* assumptions.)

Examples:

ttest testscr_lo=testscr_hi, une unp;

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 of Stock and Watson, 2011).

ttest ts_lostr=ts_hisstr if elq1==1, unp une;

tests the null hypothesis that the population mean of *ts_lostr* 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 of Stock and Watson (2011).

4. FINAL NOTE

For a complete list of commands, consult the STATA *User's Guide* and *Base Reference Manuals* (3 volumes). In addition, there are more detailed manuals on *Graphics*, *Time Series*, *Data Management*, etc.; a total of 19 manuals which can be purchased for $495.00. Alternatively, you may want to use the "**Help**" command inside STATA. It will display details of STATA commands including options. Under the **Search…** tab, for example, you will find most of what you are looking for. As mentioned before, this tutorial is not intended to replace the *Reference* or *User's Guide*. The best way to learn how to use the program is to spend some time exploring and working with it.

STATA replication batch files for all the results in the Stock/Watson textbook are available from the Web site. You are invited to download these and study them.