

Stata 12: Getting Started Tutorial



Updated: September 2012

Table of Contents

Section 1: Introduction.....	3
1.1 About this Document	3
1.2 Documentation.....	3
1.3 Accessing Stata	3
1.4 Getting Help.....	4
Section 2: The Example Dataset	5
Section 3: An Overview of Stata 12	7
3.1 Starting and Navigating Stata 12	7
3.2 The Main Window	7
3.3 Variables List	9
3.4 Properties Window.....	9
3.5 Command Prompt	10
3.6 Review Window.....	10
3.7 The Do-File	11
3.8 Working Directory	13
Section 4: Importing and Exporting Data	14
4.1 Introduction.....	14
4.2 Importing Data	14
4.3 Creating a Dataset	16
4.4 Confirming, Browsing, and Editing Data	17
4.5 Compressing and Saving Data	20
4.6 Exporting Data	21
Section 5: Summarizing and Managing Variables.....	22
5.1 Introduction.....	22
5.2 Summarizing Data	22
5.3 Variables Manager	25
Section 6: Creating and Changing Variables	29
6.1 Introduction.....	29
6.2 Generating and Recoding Variables	29
6.3 Dropping Cases or Variables	31
6.4 Appending and Merging Data.....	32
Section 7: Reshaping Data	34
7.1 Introduction.....	34
7.2 Reshaping Data	34
Section 8: Help Menus and Log Files	36
8.1 Help Menus	36
8.2 Log Files	37
Section 9: Conclusion	39

Section 1: Introduction

1.1 About this Document

This document is an introduction to understanding and using Stata, a software package popular in the social sciences for manipulating and summarizing data and conducting statistical analyses. This is the first of two Stata tutorials, both of which are based on the 12th version of Stata, although most commands discussed can be used in earlier versions also.

In this “Getting Started” tutorial, we focus on familiarizing the reader with the Stata interface, navigating the different windows, importing and exporting data files, and summarizing, sorting, and managing variables. In the subsequent tutorial, “Data Analysis with Stata 12,” we dive into actual statistical tests and inferences. Readers with at least some basic statistical knowledge are best suited for these tutorials, although we do attempt to explain each process in as much detail as possible.

Similar to the SAS statistical software package, Stata can be intimidating to first-time users who are not familiar with the syntax language. However, Stata 12 has drop-down menu options for most analytic, graphical, and statistical commands (similar to, but not as extensive as, those found in SPSS). As tempting as the drop-down menus can be, we still recommend that you become familiar with the Stata syntax as it is more efficient and leads to fewer mistakes. However, we do present both options whenever possible.

1.2 Documentation

Among the many reasons why we prefer to use syntax over the drop-down menus is the extent of support material to turn to when you run into problems with your code. First and foremost, we recommend using the “help” feature within Stata itself (described in detail in Section 8 of the “Getting Started” tutorial). Additionally, you can use the following:

- 1) Stata manuals (some are available at the PCL for check-out)
- 2) Stata’s own website has a modest amount of FAQ’s in the support section:
<http://stata.com/support/faqs/>
- 3) The Department of Statistics and Data Sciences website to find more answers to FAQ’s: <https://stat.utexas.edu/software-faqs/stata>

1.3 Accessing Stata

If you are a faculty, student, or staff member at the University of Texas at Austin, you may access Stata 12 in the following ways:

- 1) License a copy from ITS Software Distribution Services (<http://www.utexas.edu/its/sds>).
- 2) Stata is also available at certain labs around campus, and your department may also provide it via a server or in one a lab room. Check with your advisor or chair on the availability of Stata in your department.

1.4 Getting Help

If you are a member of UT-Austin, you can schedule an appointment with a statistical consultant or send e-mail to stat.consulting@austin.utexas.edu . See stat.utexas.edu/consulting/ for more details about consulting services, as well as answers to frequently asked questions Stata and other topics.

Section 2: The Example Dataset

Throughout this document, we will be using a dataset called *cars_1993.xls*, which contains various characteristics, such as price and miles-per-gallon, of 92 cars from 1993. In order to follow along with the examples, please download this data by clicking [HERE](#).

Note that this is the same example dataset we use in the “SAS: Getting Started” tutorial, and the file is actually one of the example datasets from SAS, which provides information about the *cars_1993* file and is represented below:

Name: cars_1993

Reference: This represents a subset of the information reported in the 1993 Cars Annual Auto Issue published by Consumer Reports and from Pace New Car and Truck 1993 Buying Guide.

Description: A random sample of 92 1993 model cars is contained in this data set. The information for each car includes: manufacturer, model, type (small, compact, sporty, midsize, large, or van), price (in thousands of dollars), city mpg, highway mpg, engine size (liters), horsepower, fuel tank size (gallons), weight (pounds), and origin (US or non-US). The data are excellent for doing descriptive statistics by groups or an ANOVA or regression with price as the response variable. Note that violations of the assumptions are probably present and transformation of the response variable is most likely necessary.

Below is what the file should look like once you download and open it in Excel:

The screenshot shows a Microsoft Excel spreadsheet titled 'cars_1993.xls [Compatibility Mode] - Microsoft Excel'. The spreadsheet contains a dataset of 1993 cars. The columns are labeled as follows: A1: Manufacturer, B1: Model, C1: type, D1: Price, E1: CityMPG, F1: HighwayMPG, G1: EngineSize, H1: Horsepower, I1: FuelTank, J1: Passengers, K1: Weight, L1: Origin. The data is sorted by Manufacturer. The status bar at the bottom indicates 'Ready' and '100%' zoom.

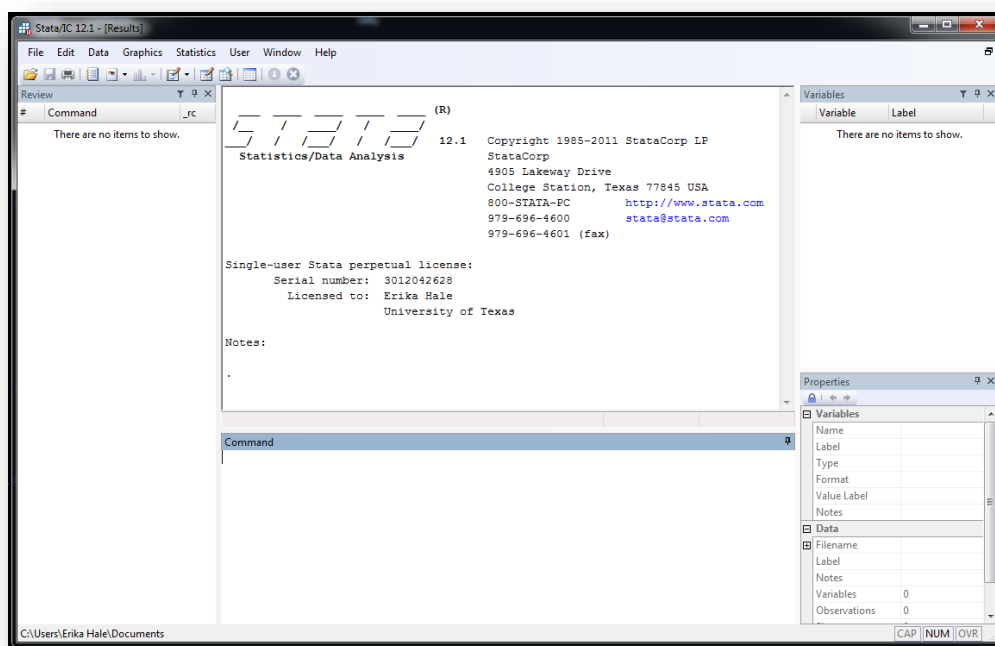
	Manufacturer	Model	type	Price	CityMPG	HighwayMPG	EngineSize	Horsepower	FuelTank	Passengers	Weight	Origin
1	Mazda	PX-7	3	32.5	17	25	1.3	255	20	2	2895	non-US
2	Chevrolet	Corvette	3	38	17	25	5.7	300	20	2	3380	US
3	Hyundai	Scoupe	3	10	26	34	1.5	92	11.9	4	2285	non-US
4	Honda	Prelude	3	19.8	24	31	2.3	160	15.9	4	2865	non-US
5	Honda	Accord	2	17.5	24	31	2.2	140	17	4	3040	non-US
6	Honda	Civic	1	12.1	42	46	1.5	102	11.9	4	2350	non-US
7	Geo	Storm	3	12.5	30	36	1.6	90	12.4	4	2475	non-US
8	Ford	Festiva	1	7.4	31	33	1.3	63	10	4	1845	US
9	Dodge	Stealth	3	25.8	18	24	3	300	19.8	4	3805	US
10	Ford	Mustang	3	15.9	22	29	2.3	105	15.4	4	2850	US
11	Geo	Metro	1	8.4	46	50	1	55	10.6	4	1695	non-US
12	Ford	Probe	3	14	24	30	2	115	15.5	4	2710	US
13	Suzuki	Swift	1	8.6	39	43	1.3	70	10.6	4	1965	non-US
14	Subaru	Justy	1	8.4	33	37	1.2	73	9.2	4	2045	non-US
15	Toyota	Celica	3	18.4	25	32	2.2	135	15.9	4	2950	non-US
16	Volkswagen	Corrado	3	23.3	18	25	2.8	178	18.5	4	2810	non-US
17	Volkswagen	Fox	1	9.1	25	33	1.8	81	12.4	4	2240	non-US
18	Pontiac	Firebird	3	17.7	19	28	3.4	160	15.5	4	3240	US
19	Mazda	323	1	8.3	29	37	1.6	82	13.2	4	2325	non-US
20	Lexus	SC300	4	35.2	18	23	3	225	20.6	4	3515	non-US
21	Mercury	Capri	3	14.1	23	26	1.6	100	11.1	4	2450	US
22	Pontiac	LeMans	1	9	31	41	1.6	74	13.2	4	2350	US

Section 3: An Overview of Stata 12

3.1 Starting and Navigating Stata 12

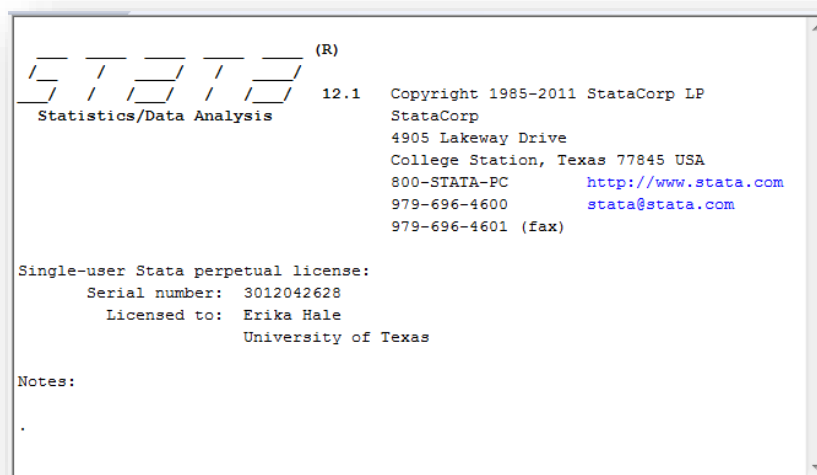
When you open Stata 12, you'll see the screen below. The main Stata interface is composed of five windows, listed clockwise from the center:

- 1) Main window
- 2) Variables list
- 3) Properties window
- 4) Command prompt
- 5) Review window (command history)

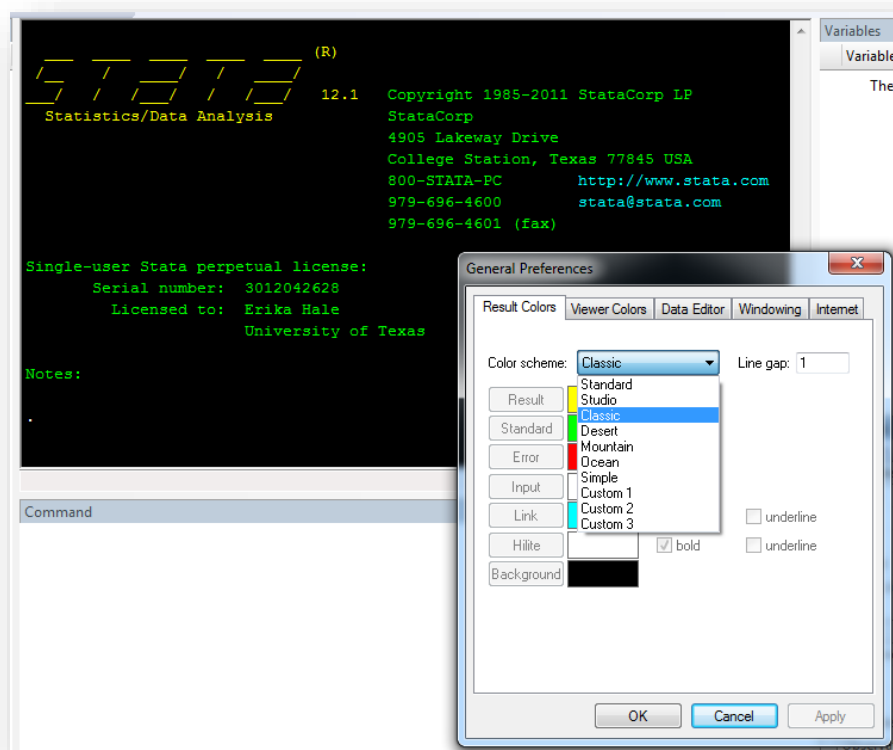


3.2 The Main Window

The main window is located in the center of the Stata window:

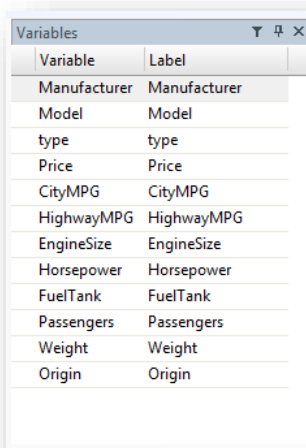


This window is where the output will appear when you run any commands. It also will write out each command that is submitted above any output. You can change the default look of the main window by going to **Edit → Preference → General Preferences** and change the color scheme:



3.3 Variables List

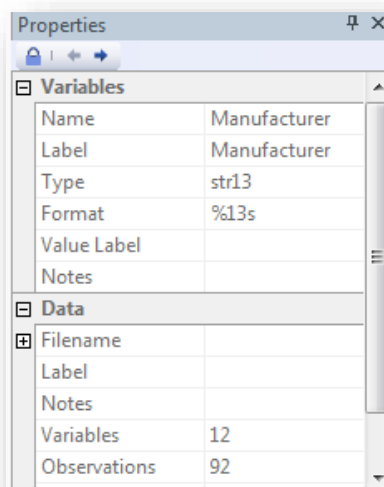
Directly to the right of the main window is a list of variables contained in the current dataset. When you open Stata, there will be no variables listed until you either call or create an active dataset. Below is what this list looks like for our sample *cars_1993* dataset, with each variable name and label listed (the default label is the variable name):



Variable	Label
Manufacturer	Manufacturer
Model	Model
type	type
Price	Price
CityMPG	CityMPG
HighwayMPG	HighwayMPG
EngineSize	EngineSize
Horsepower	Horsepower
FuelTank	FuelTank
Passengers	Passengers
Weight	Weight
Origin	Origin

3.4 Properties Window

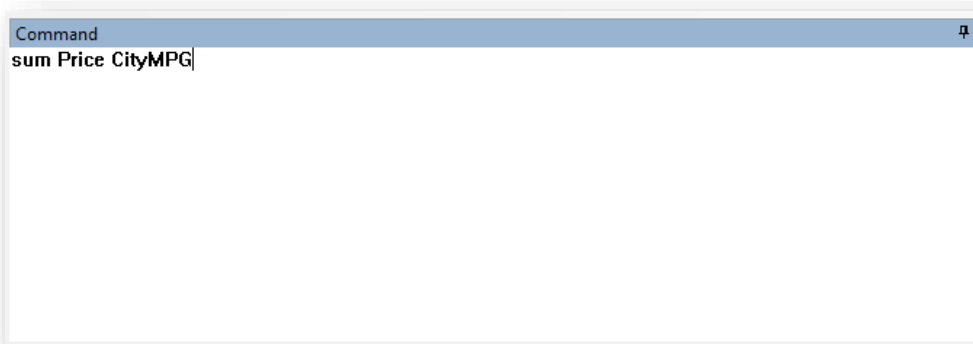
Beneath the variables list is the properties window, which lists variable and dataset details. For each variable in the active dataset, you can see the name, label, type, and any value labels you've assigned by clicking on the variable name in the Variables window (see Section 5.3). For the active dataset, you can also see the number of variables and observations, as well as the size and memory taken up by the file:



Properties	
Variables	
Name	Manufacturer
Label	Manufacturer
Type	str13
Format	%13s
Value Label	
Notes	
Data	
Filename	
Label	
Notes	
Variables	12
Observations	92

3.5 Command Prompt

To run a specific command, you can type it directly into the command prompt, located below the main window. In the screen shot below, I have asked Stata for a *summary* of the variables Price and CityMPG. By hitting **Enter**, the output for this command appears in the main window:



Main window shows:

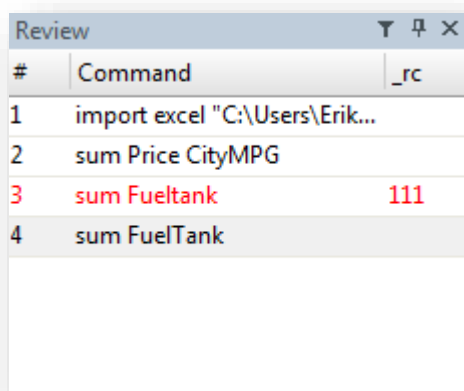
```
. sum Price CityMPG
```

Variable	Obs	Mean	Std. Dev.	Min	Max
Price	92	19.04891	8.623728	7.4	47.9
CityMPG	92	22.40217	5.63946	15	46

Although the command window is convenient for quick commands, we recommend running the majority of your code from a .do file (see Section 3.7).

3.6 Review Window

The review window serves as a running history of the commands that you have submitted to Stata since you have opened up the program. It is located along the left side of the Stata screen. Each time you run a command, either by typing it in the command prompt, running it from a .do file, or navigating through the drop-down menus, it appears in sequential order in the review window, like below:



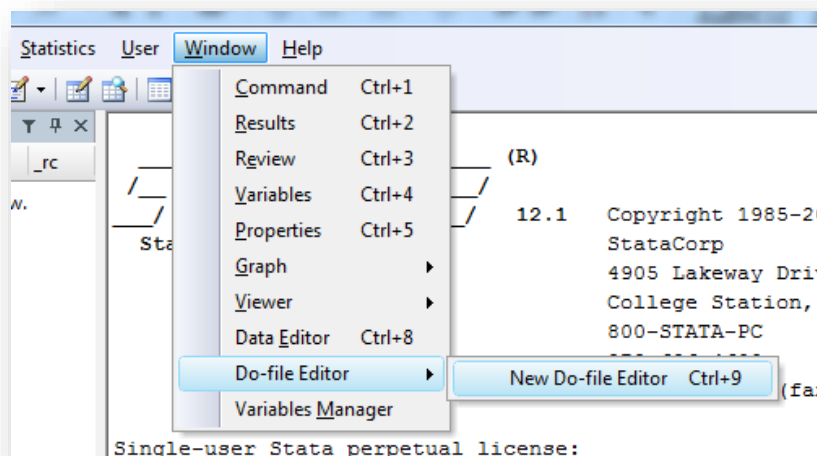
#	Command	_rc
1	import excel "C:\Users\Erik..."	
2	sum Price CityMPG	
3	sum Fueltank	111
4	sum FuelTank	

In the example above, I imported our example dataset (line #1) and then ran a command to summarize the Price and CityMPG variables (line#2). On the third line is a command that I attempted to run, but I incorrectly spelled the “FuelTank” as “Fueltank.” Any command that Stata can’t run shows up in the review window in red. This line also illustrates the fact that Stata is **case-sensitive**, and it therefore won’t recognize a variable name if the capitalization is incorrect. Immediately after the error, I corrected the **sum** command and Stata was then able to run a summary on FuelTank (line #4).

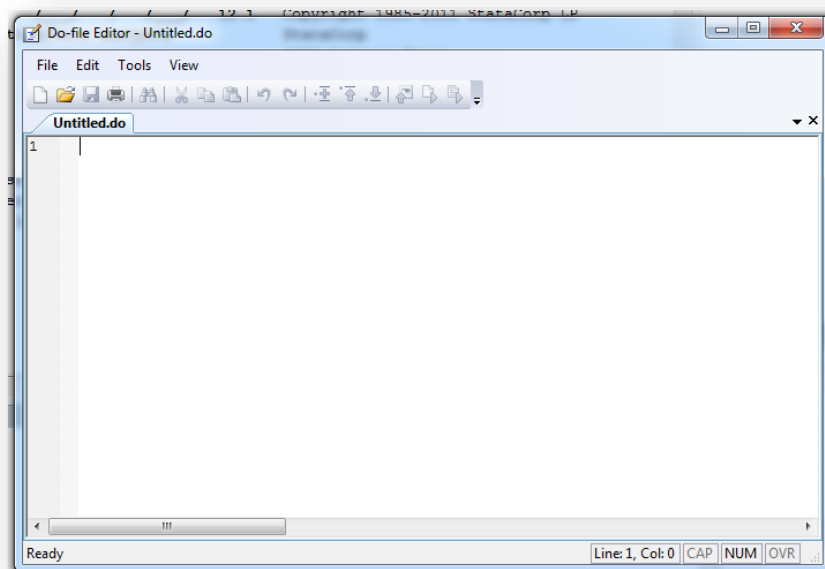
The review window is not only convenient for catching syntax errors, but also has the capability to re-run any command that it lists. By clicking on any line, the exact code that was used appears in the command prompt, and can from there be either edited or re-run by hitting **Enter**.

3.7 The Do-File

Knowing the uses of each of the previous five components is crucial to navigating and using Stata. However, there is another major component that is essential in efficiently running analyses in Stata called the “do file.” These files are essentially scripts which you can edit, save, and run within Stata, with the file extension *.do*. Because Stata does not provide a *.do* file at the start-up, you must create a new one (or open an existing one) by going to **Window → Do-File Editor → New Do-File Editor**. Or you can simply hit **Ctrl+9**:



Creating a new .do file will pop-up the following window:



This file is essentially a blank script, ready to be edited, saved, and run. We recommend **creating and saving a .do file** to house all of the commands you run on your dataset so that you can easily reference what you've done and re-run any steps as needed (also see Section 8 for information on log files). Any code you type in a .do file can be run in the main Stata window by highlighting it and hitting **Ctrl+D**.

Unlike the review window, which contains a history of commands used in the current Stata session, when you save a .do file, it is available for use during any future session. Another advantage is that you can run a chunk of code all at once by highlighting it in a

.do file and hitting **Ctrl+D** instead of entering it line-by-line in the command window. Finally, code contained in .do file is color-coded (similar to SAS), which can help you debug your syntax. As opposed to simply typing commands in the command prompt window, recognized commands in a .do file turn blue, string statements contained in quotation marks turn red, and comment lines turn green.

Just like when you write code in any programming language, it is extremely important to include comments within the file to help describe what each command is doing and to give a framework to the overall script. To specify a comment line, simply begin it with an `"*"`, which turns the entire line green and tells Stata to skip to the next line.

3.8 Working Directory

Each Stata session will have one working directory, which is a location on your computer (or any network drive you are connected to) that Stata will look in when you ask it to communicate with an external file. The commands that are associated with Stata's working directory are based on Linux commands.

To see what the current working directory is, type **pwd** in the command prompt (which stands for "path of the working directory") and the main window will display it:

```
. pwd
U:\
```

To change the working directory to some other path, type **cd** followed by the new directory:

```
cd "C:\Desktop\"
```

To get a list of the files contained in the working directory, type **dir** or **ls**. The list contains not only Stata files, but all files regardless of their format, and also contains the date and size of each file. Any folders within the working directory will have **<dir>** instead of the file size:

```
. ls
<dir>    9/04/12 10:08  .
<dir>    9/04/12 10:07  ..
<dir>    9/04/12 10:07  Subfolder1
25.0k    2/17/04 17:13  cars_1993.xls
665.4k   8/02/12 16:54  Stata_Tutorial.docx
```

Section 4: Importing and Exporting Data

4.1 Introduction

Now that you are familiar with the different components of Stata, this next section provides the code and drop-down menu options for importing and exporting datasets. Stata only recognizes one dataset at a time. You can start different sessions of Stata concurrently, and therefore have several datasets open at once. However, you won't be able to run commands on datasets that are active in different sessions of Stata. In order to do this, you must combine them into one dataset (see Section 6.4).

4.2 Importing Data

Before you import a dataset, whether you just opened a new Stata session or not, we recommend running the `clear` command, which clears out all active data from Stata's memory. You cannot bring in a dataset without clearing out the active dataset (you will get the following error):

```
no; data in memory would be lost
```

Stata's default file format for data files is *.dta*. For data that is already saved in *.dta* format, you can import it in direction with the `use` command. You must either type out the full path of the file, or change your working directory and then just `use` the file name:

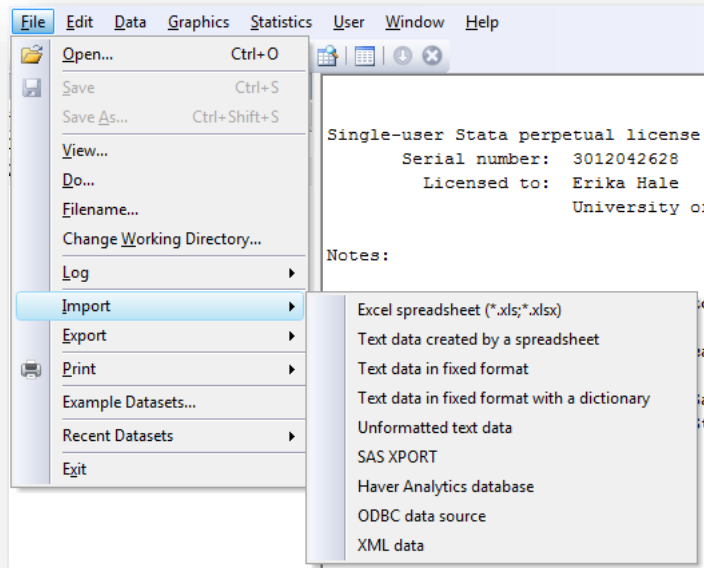
```
clear                                OR                                clear
use "C:\Desktop\statademo.dta"      cd "C:\Desktop\"
                                     use statademo
```

Stata 12 can read-in data that is in one of the following formats:

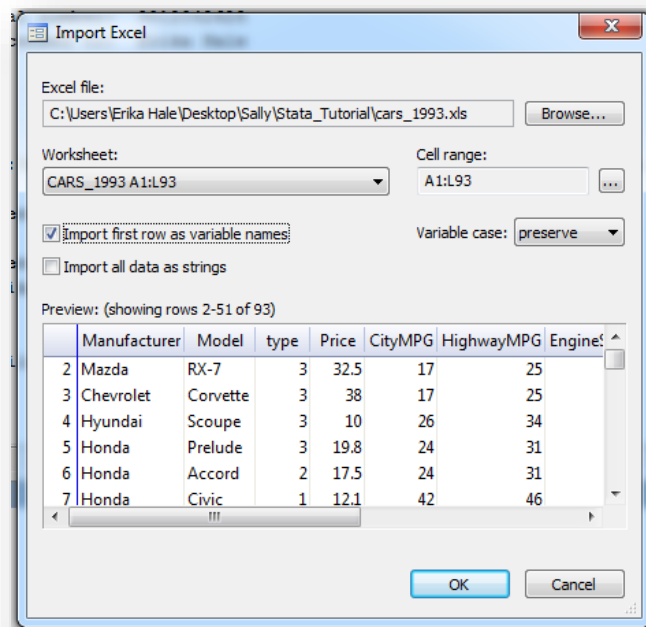
- 1) Excel spreadsheet (*.xls* or *.xlsx*)
- 2) Text data from a spreadsheet (*.csv*, etc.)
- 3) Text data in a fixed format (*ASCII*, etc.)
- 4) Text data in a fixed format with a dictionary
- 5) SAS XPORT transport file (*.xpt*)
- 6) Haver analytics file (*.dat*)
- 7) ODCB database file (*.dbf*, *.mdb*, etc.)
- 8) XLM data (*.xlm*)

There are two ways to import a dataset that is not in *.dta* format: run the command with syntax, either from a *.do* file or directly in the command prompt, or use a series of drop-down windows. Depending on the type of file you import, one way will be much easier than the other. Therefore, we will demonstrate both methods.

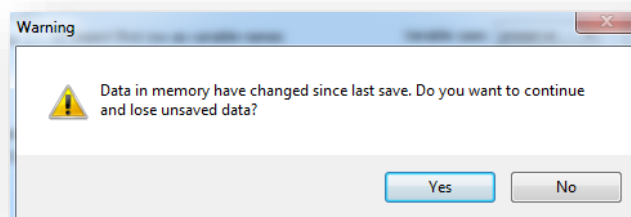
You will see these selections if you choose to import your data file with the drop-down menus. To do this, go to **File** → **Import** and choose the appropriate file type:



You will then be asked to select certain options depending on the type of file you want to import. For example, if your data is coming from an Excel file, *.xls* or *.xlsx*, you need to tell Stata which sheet you want to read and whether or not your variable names are contained in the first row:



If you already have a dataset in the current session, you will see the following message:



By hitting **Yes**, Stata will clear out the existing dataset and import the one you selected in its place. You can also do this by using the **clear** command prior to importing in a new dataset.

Using the drop-down menus for an Excel file is much easier than reading it in using syntax. However, if your file is saved as a *.csv*, we recommend using syntax. The code below is from a *.do* file, which you can tell because of the convenient color-coding. The first line is a comment, denoted by the “*” at the beginning and the green color. This means that Stata will not try to compile or execute this line. The second line tells Stata which folder on your computer to look in (which isn’t needed if the file is in the current working directory), while the third line contains the command for actually importing the *.csv* dataset, **insheet**, followed by the file name:

```
*Bring in spreadsheet
cd "C:\Desktop\Stata_Tutorial"
insheet using cars_1993.csv
```

Notice that the first words in the second and third lines, **cd** and **insheet**, both turn blue, telling you that they are commands which Stata recognizes. The directory specified in the second line is red, which Stata uses for anything in between quotation marks.

4.3 Creating a Dataset

Another way to read data into Stata is to input the data manually using the **input** command. Below is some example data that we read into the program:

```
input id gender str10 race
1 1 "Hispanic"
2 1 "White"
3 1 "White"
4 1 "White"
5 0 "White"
6 0 "Hispanic"
7 0 "Hispanic"
8 0 "White"
end
```


For variables with non-numeric values, you must specify the type and length of the variable before listing the variable name. As seen above, *race* is a string variable, so we included *str10* just before the variable name, indicating that it is a string of length 10. Because of the somewhat tedious syntax needed, we do not recommend entering in any sizable dataset with this method.

4.4 Confirming, Browsing, and Editing Data

Once you have either imported or created a dataset, you should notice several things. First, the variables in the dataset appear in the variables window. Also, just like with any other command, the command you used to get the data (**insheet**, **input**, etc.) is reported back to you in the main window and also shown in the review window.

There are now several things you can do the active dataset to confirm that it was imported correctly and has all the cases and variables you were expecting.

describe – outputs a summary of the active dataset, including the number of observations, number of variables and type of each, and the size of the overall dataset. All of this information is also available in the variables and properties windows:

```
. describe

Contains data
  obs:          92
  vars:          12
  size:         5,612
```

variable name	storage type	display format	value label	variable label
Manufacturer	str13	%13s		Manufacturer
Model	str10	%10s		Model
type	byte	%10.0g		type
Price	double	%10.0g		Price
CityMPG	byte	%10.0g		CityMPG
HighwayMPG	byte	%10.0g		HighwayMPG
EngineSize	double	%10.0g		EngineSize
Horsepower	int	%10.0g		Horsepower
FuelTank	double	%10.0g		FuelTank
Passengers	byte	%10.0g		Passengers
Weight	int	%10.0g		Weight
Origin	str6	%9s		Origin

```
Sorted by:
  Note:  dataset has changed since last saved
```

list varname – outputs data rows for variable or variables specified. For large datasets, you can ask Stata to only list rows 1 thru n with **in 1/n**.

```
. list Model type
```

	Model	type
1.	RX-7	3
2.	Corvette	3
3.	Scoupe	3
4.	Prelude	3
5.	Accord	2
6.	Civic	1
7.	Storm	3
8.	Festiva	1
9.	Stealth	3
10.	Mustang	3
11.	Metro	1
12.	Probe	3
13.	Swift	1
14.	Justy	1
15.	Celica	3
16.	Corrado	3

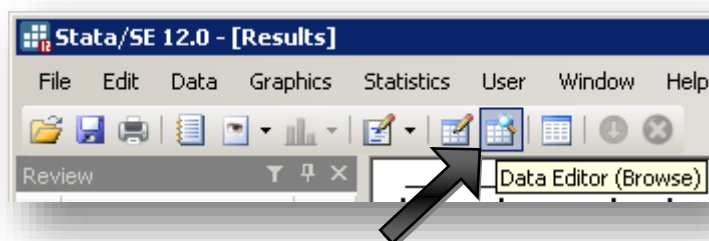
—more—

```
. list Model type in 1/5
```

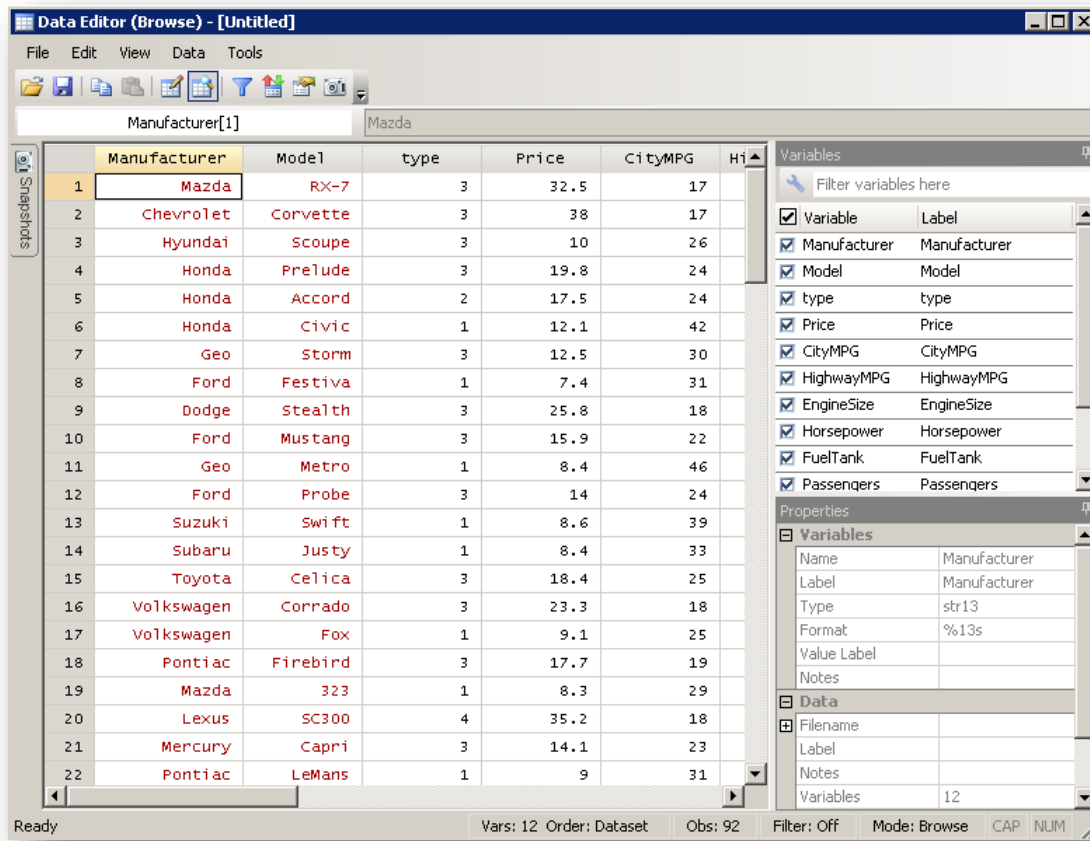
	Model	type
1.	RX-7	3
2.	Corvette	3
3.	Scoupe	3
4.	Prelude	3
5.	Accord	2

If you forget the **in 1/n** option, and Stata displays a large number of rows in the main window, you must scroll down the output by hitting **Spacebar** or some other key. To break the output and return control to the command window, hit **Q**.

You can also browse the active dataset by clicking the button in the toolbar shown below:

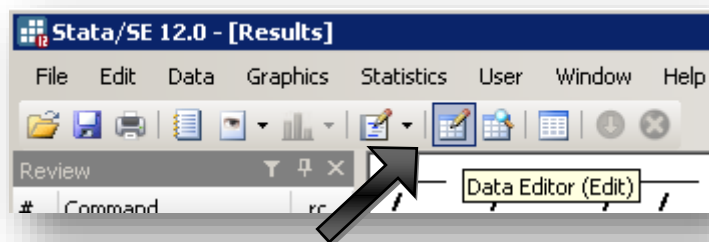


Clicking this button will open a new window that contains a spreadsheet-like display of the data with variables in the columns and records in the rows. The window also contains the variables and properties windows. Below is our *cars* dataset in the browse window:

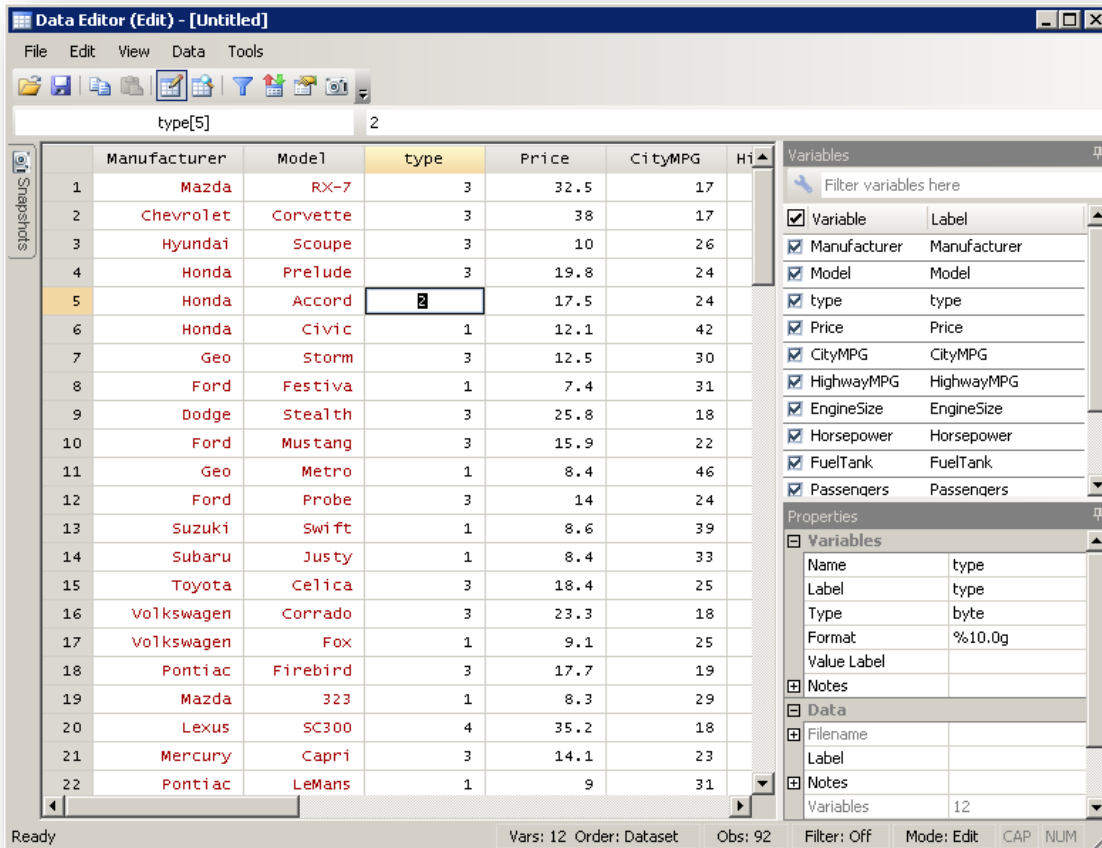


You'll notice that the title of the window is "Data Editor (Browse)." This means that you cannot change the content of the dataset, but only view it.

We will provide a variety of commands that transform and manipulate data in Section 5, but for now we will mention that there is a way to manually edit cases the active dataset. Right next to the "browse" button is another button that will bring up the dataset in a window that is updateable:



Hitting this will bring up a window that looks just like the browser but will allow you to change the values in any cell. In the screen below, I double-clicked on the "type" for the Honda Accord and then had the ability to update its value:



4.5 Compressing and Saving Data

To save the active dataset as a *.dta* file in the working directory, use the following command. If you want the file save in another directory, include the path in the *filename* below:

```
save filename
```

If you have already saved the dataset once before, or you would like to replace a file with the same name, add the **replace** option at the end. Otherwise, you will get an error that the file you are trying to save to already exists.

When working on a *.dta* dataset, any modifications you make will only take effect when you save the dataset. If you close the Stata session while unsaved changes have been made, it will ask you if you'd like to save it before closing.

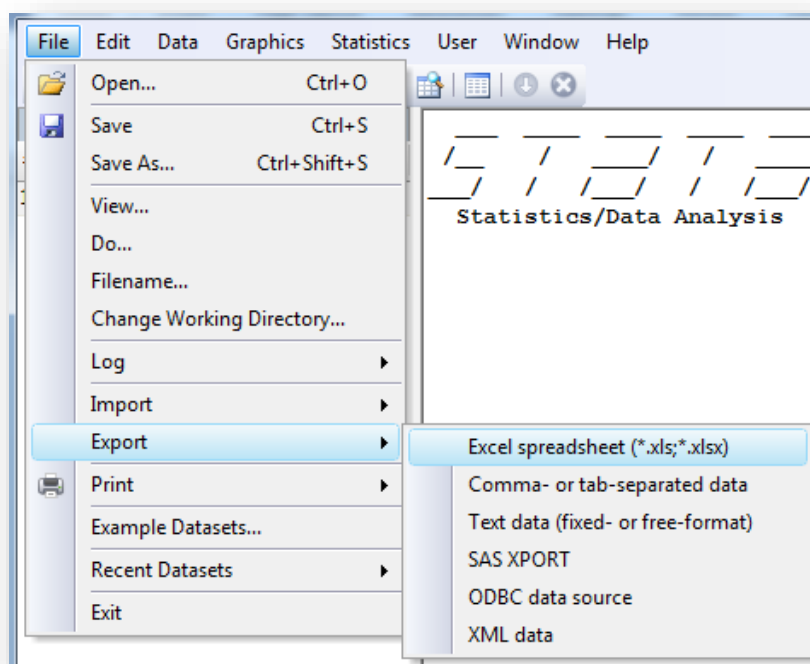
When you import a dataset with a different file extension (*.xls* or *.csv*), Stata does not modify the original file. So, for example, if you make some updates to an Excel spreadsheet within Stata and then use the save command, your original *.xls* file will stay

intact without any of the changes you made to the *.dta* file. See the following section for instructions on exporting the dataset in a different file format.

For larger datasets, you might want to reduce the memory they take up on your CPU by compressing the active dataset before you save it. To do this, just enter **compress** in the command window.

4.6 Exporting Data

Similar to importing files with extensions other than *.dta*, you can export datasets with the pull-down menus or by submitting a command with syntax. To use the menus, simply go to **File** → **Export** → ... and choose the format you wish to use:



To export a dataset in the command window or with a do-file, use the **outsheet** command followed by the variables you wish to include in the file. Then, type the location and name of the file preceded by **using**. The following example will export the variables *Model* and *type* into a file called “examplefile1.csv,” as a comma-separated file:

```
outsheet Model type using examplefile1.csv, comma
```

By default, the first line in the file will contain the variable names. To not include the variable names in your file, simply add the **nolabel** option at the end of the command.

Section 5: Summarizing and Managing Variables

5.1 Introduction

This section covers various ways of summarizing, labeling, and managing variables from your active dataset in preparation for any type of statistical test or analysis. For this section, we will again use the *cars* example data described in Section 2.

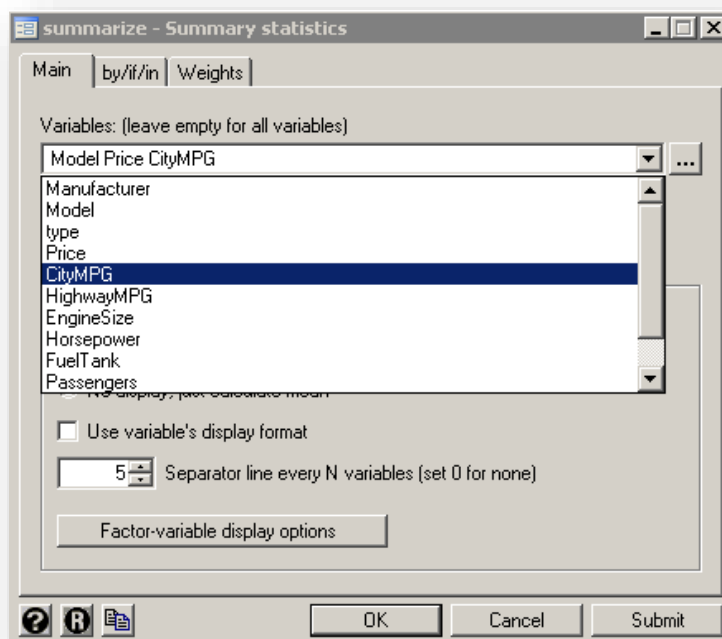
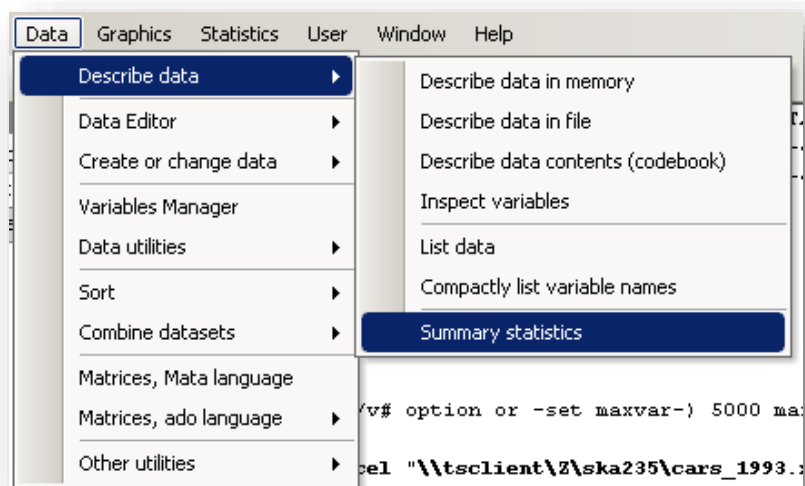
5.2 Summarizing Data

Before making any changes to your dataset, it might be helpful to get basic descriptive statistics of the variables. You can do this by using the **sum** command in the command prompt or by using the drop-down menus. Both methods are described below and give you an identical output. However, we again recommend using the command prompt or a .do file.

Below is the output for a summary of three variables from the *cars* dataset: *Model*, *Price*, and *CityMPG*. Note that *Model* is a string variable, and therefore no statistics are given. The **sum** command outputs the number of observations, mean, standard deviation, minimum, and maximum of numeric variables:

. sum Model Price CityMPG					
Variable	Obs	Mean	Std. Dev.	Min	Max
Model	0				
Price	92	19.04891	8.623728	7.4	47.9
CityMPG	92	22.40217	5.63946	15	46

And identical output is displayed by going to the **Data → Describe Data → Summary Statistics** menu. Once you select this, a window will pop up where you can choose the variables you want summarized from a dropdown menu. In this list, you can select more than one variable by simply clicking on any that you want, and they will be added to the command:



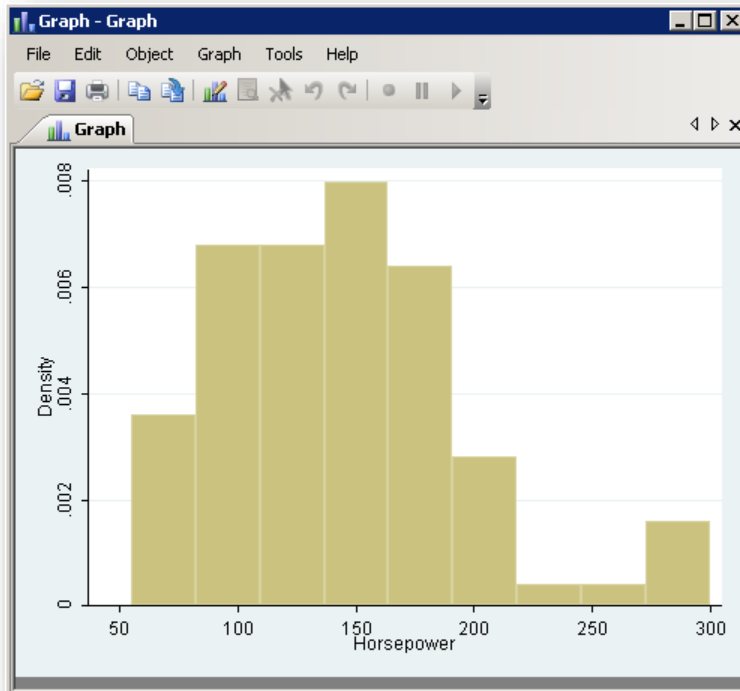
Another common way of summarizing a numeric variable is to create a histogram of the values. This can also be done with a simple command or by using the drop-down menus. If we want a histogram of the *Horsepower* of each of the cars in our dataset, we can type the following a .do file or directly in the command prompt:

```
hist Horsepower
```

When this command is run, Stata does two things. First, it outputs a line in the main window, giving the number of bins, starting value of the first bin, and width of each bin

which it applied to the graph by default. It also opens a new window with the histogram itself:

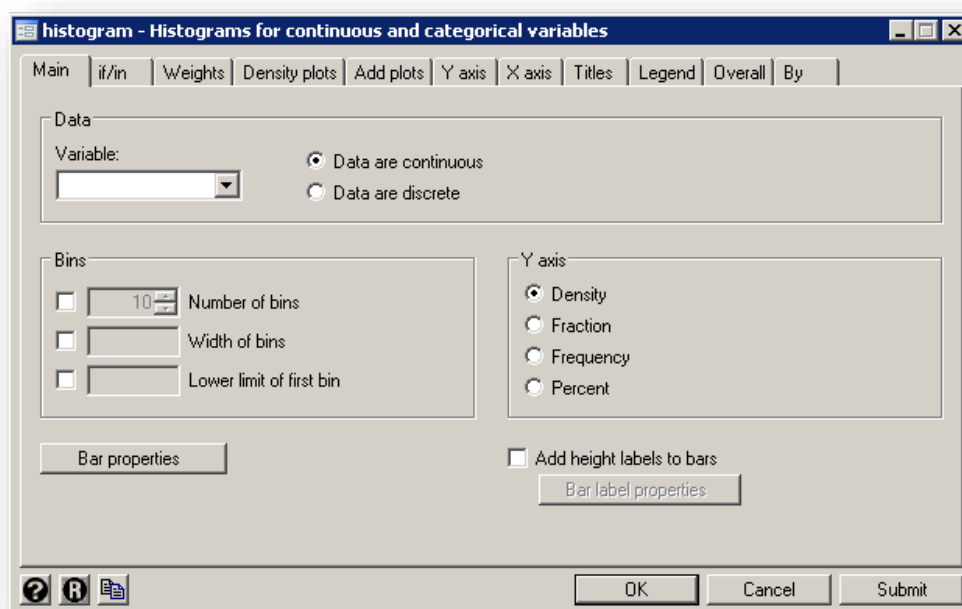
```
. hist Horsepower  
(bin=9, start=55, width=27.222222)
```



You can specify the number of bins that you want in the histogram by adding an option at the end of your command (see *help histogram* for other options with this command):

```
hist Horsepower, bin(12)
```

The same histogram will be produced by going to **Graphics** → **Histogram**:

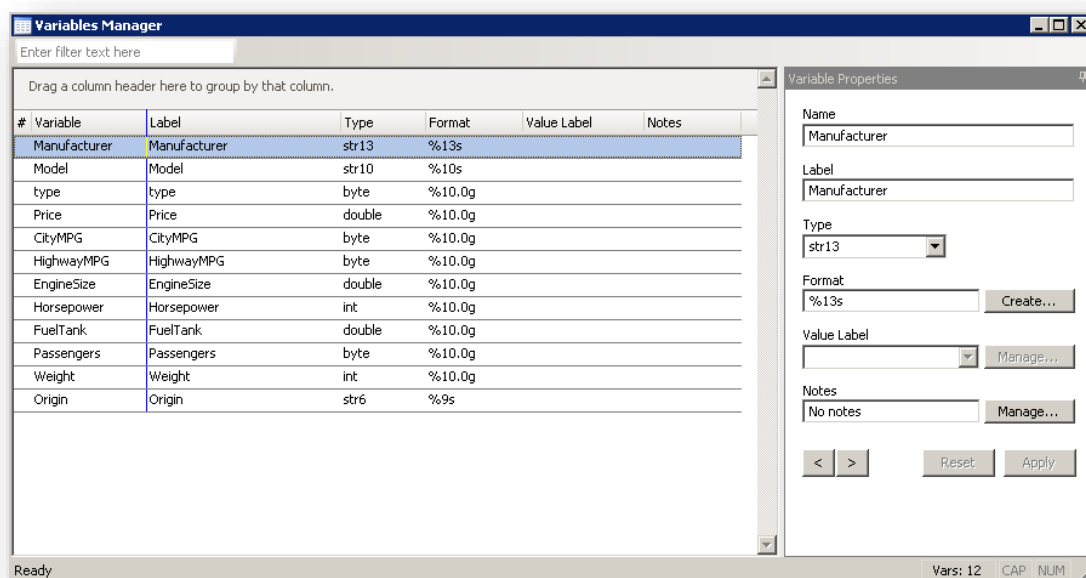


More descriptive statistics commands will be covered in the next Tutorial, “Data Analysis with Stata 12.”

5.3 Variables Manager

Stata has a toolbar button that will open the “Variables Manager,” which you can use to easily change the properties of the variables in your dataset. To open it, click the button show below:





Within the variables manager is the ability to edit each variable's name, label, type, format, and value labels. Click the specific variable you want to edit on the list, then make your changes in the "Variable Properties" window on the right and hit **Apply** for your changes to take effect.

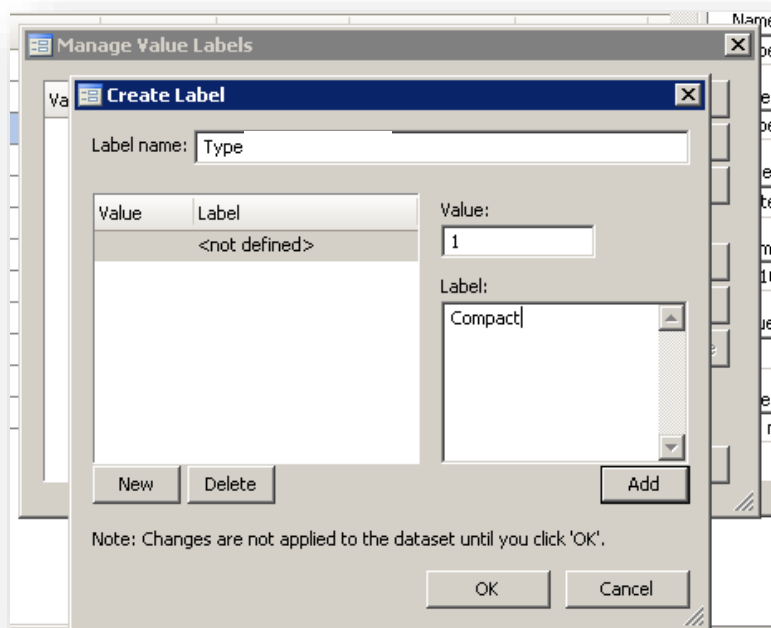
Let's say we wanted to add value labels to define each of the six "types" of vehicles:

1. Compact
2. Small
3. Sporty
4. Midsize
5. Large
6. Van

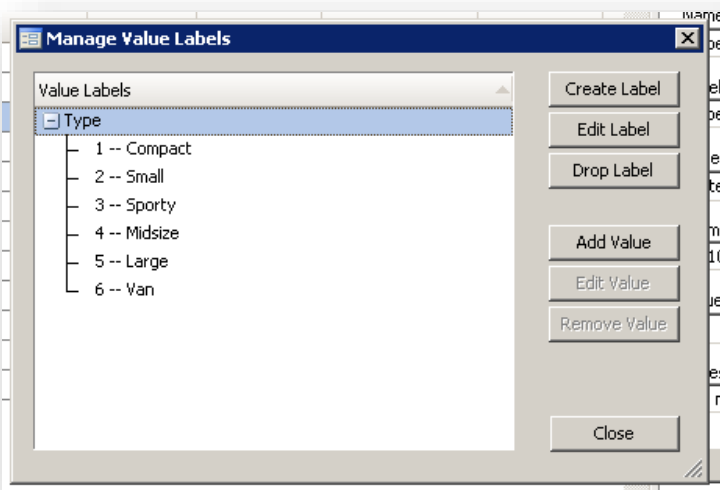
We can do this by selecting the *type* variable in the list and then hitting the Manage button next to the "Value Label" menu on the right side of the manager window. This will bring up the "Manage Value Labels" window. Notice that this button is inactive when you select a string variable (since value labels are associated with numeric variables only).

The "Manager Value Labels" window will show all *sets* of value labels, not the labels themselves. Each set of labels that you create will be available to assign to any variable in the dataset. So if, for example, you had a survey with choices "yes" and "no" for several different questions, you can define these labels as one set and then assign them to each variable that uses those labels.

To define a new set of labels, hit the **Create Label** button, which will open up another window. Here, enter the name or description of the *set* of labels, not the first label itself. For our example, let's just call it "Type," so that we know it relates to that variable. Then, enter the first value, "1" in the value box and type "Compact" in the label box:



Then hit **Add** to add it to the list. Repeat this procedure for each of the six labels, and you'll have the entire "Type" set of value labels. Then hit **OK**.



Finally, back in the main “Variables Manager” window, choose the “Type” set from the value label drop-down menu and hit **Apply**. You’ll now see “Type” in the “Value Label” column for that variable.

You can also assign value labels using the `label` command in the command prompt or a .do file. Stata actually gave us the syntax for doing this in the review window once we defined the labels with the “Variables Manager.” Remember that whether you enter a command directly in the prompt, from a .do file, or through the toolbar or drop-down menus, Stata records those commands in the Review window and also outputs them in the Main window:

```
. label define Type 1 "Compact" 2 "Small" 3 "Sporty" 4 "Midsize" 5 "Large" 6 "Van"

. label values type Type
```

Now, when you run a command on the “type” variable, the output will list the labels instead of “1, 2, 3,…” For example, we can see the type of the first 10 cars with the usual `list` command:

```
. list type in 1/10
```

	type
1.	Sporty
2.	Sporty
3.	Sporty
4.	Sporty
5.	Small
6.	Compact
7.	Sporty
8.	Compact
9.	Sporty
10.	Sporty

Section 6: Creating and Changing Variables

6.1 Introduction

Now that you have imported or created an active dataset, summarized any pertinent variables, and added appropriate labels, you might find it necessary to add additional variables, transform or edit existing ones, and make other modifications to your data. This section covers various ways of managing your dataset, which can be extremely important since you rarely end up with research data that contains every variable needed for your analysis.

6.2 Generating and Recoding Variables

Data analysts often need to create new variables based off of other data, including dummy variables, summary variables, or recodes (see next section). To do this in Stata, use the **generate** (or **gen**) command. For example, the following pair of commands will create a variable that is zero for each car and a variable that is the average mile-per-gallon consumption for each car, respectively:

```
gen dummy = 0
gen avgMPG = (CityMPG + HighwayMPG) / 2
```

Now let's say we want to update the *dummy* variable to be equal to one whenever a vehicle's horsepower is at least 200. We can do this with the **replace** command, as shown below:

```
replace dummy = 1 if Horsepower >= 200
```

An alternative way of creating a dummy variable is to generate a new variable followed by a logical statement. The following code will create the same variable as shown above, but only takes one line:

```
gen dummy = (Horsepower >= 200)
```

Just to visually confirm that our dummy variable is correct, we can list the *Horsepower* and *dummy* variables together for the first few cases:

```
. list Horsepower dummy in 1/10
```

	Horsepower	dummy
1.	255	1
2.	300	1
3.	92	0
4.	160	0
5.	140	0
6.	102	0
7.	90	0
8.	63	0
9.	300	1
10.	105	0

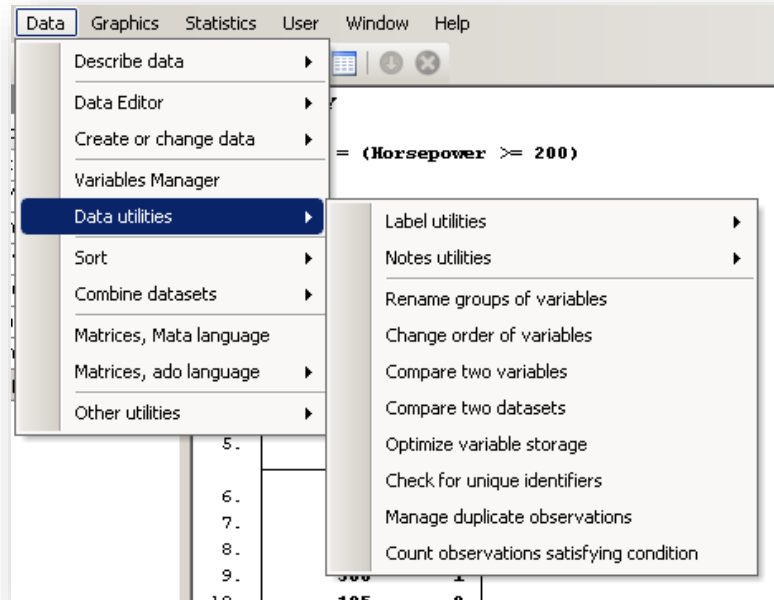
We can also recode all of the values within a variable with a single command without using **replace** repeatedly. For instance, use the **recode** command to update the *Weight* variable to be in thousands of pounds:

```
recode Weight (0/1000=1) (1001/2000=2) (2001/3000=3) ///
           (3001/4000=4) (4001/5000=5)
```

Combining the **recode** and **gen** commands, you can manipulate your data to suite many different needs. The code below will categorize the number of passengers for each car into “small,” “medium,” and “large” and dump these categories into a new variable called *Size*:

```
recode Passengers (0/3=1 Small) (4/6=2 Medium) ///
           (7/8=3 Large), gen(Size)
```

All of the commands used in this section, as well as many others, can also be run using the drop-down menus under **Data → Data Utilities...**



6.3 Dropping Cases or Variables

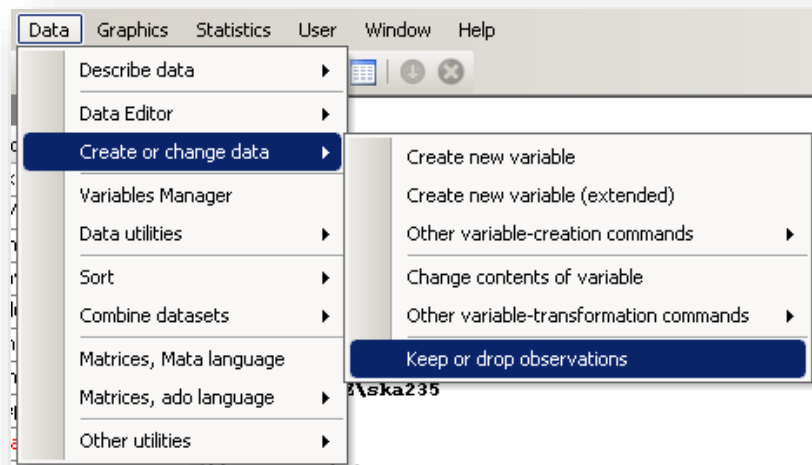
Stata's **drop** command can be used to either remove specific cases (rows) from your dataset, or delete entire variables. First, to remove entire variables, simply list those you wanted removed after **drop**. For example, let's delete the new variables we create in the previous sections:

```
drop avgMPG dummy Size
```

In order to drop selected cases from the dataset, follow the command with the criteria under which you want cases removed. If we wanted to remove all cars that were made outside of the US, we could type the following:

```
drop if Origin=="non-US"
```

You can do the equivalent with the **keep** command by entering the logical opposite of the criteria in the **drop** statement. These commands can also be run by going to **Data** → **Create or Change Data** → **Keep or Drop Observations...**



6.4 Appending and Merging Data

The **append** command will append cases from another file at the end of the active dataset. By default, the cases appended will be added to all common variables in the current dataset. However, you can keep other variables with the new cases with the **keep** option. To see other options for this command, type **help append** in the command prompt. Below is an example of appending cases from *dataset_B* to the active dataset:

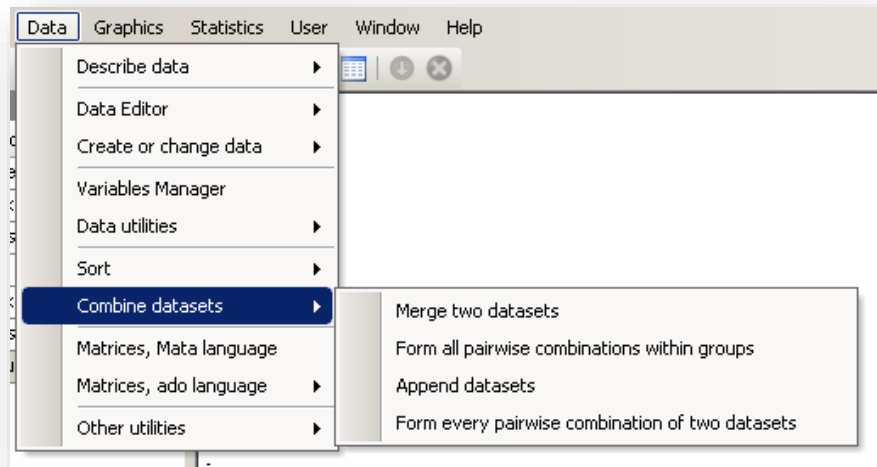
```
append using dataset_B
```

Merging two files with different variables for the same cases is slightly more complicated. To use the **merge** command, you must tell Stata which variable links the two files together (an ID variable) and also how you want the files merged (one-to-one, one-to-many, many-to-one, or many-to-many). Also, both files must be sorted by the linking variable, which you can do easily with **sort** followed by the variable name.

Below is an example where we merge two different datasets, each sorted by the variable *id* and each containing one row per *id* (one-to-one merge):

```
merge 1:1 id using dataset_B
```

You can use the drop-down menus to both append cases and merge variables:



Section 7: Reshaping Data

7.1 Introduction

For some types of analyses, like those run on longitudinal or repeated measures data, it is important to organize the data in a specific format. Stata refers to data as either “long” (where each row represents a single measurement), or “wide” (where each row represents an individual, with multiple variables for the different measurements). The following section explains how to transform your dataset from one format to another.

7.2 Reshaping Data

The command within Stata to transform your data from long to wide format, or vice versa, is quite simple. But understanding which one to use and when is a bit more complicated, and should be determined from which analysis you will be running.

To illustrate these commands, let us consider a repeated measures design on a class of students who took a pre-test (test #1), were subjected to some sort of treatment, and then given a post-test (test #2). The data could be represented in either of the ways shown below:

Long Format:

<u>ID</u>	<u>Test</u>	<u>Score</u>
1	1	82
1	2	87
2	1	90
2	2	91
3	1	78
3	2	87
...		

Wide Format:

<u>ID</u>	<u>Score1</u>	<u>Score2</u>
1	82	87
2	90	91
3	78	87
...		

In the long format, each student has two rows of data, corresponding to tests number 1 and 2. In the wide format, each row represents a single student, with two columns for the different test scores.

If our active dataset was in the long format pictured above, and we wanted to convert it into wide format, we would run the following command. This would be applicable if we wanted to run a repeated-measures ANOVA, which must be in wide format:

```
reshape wide Score, i(ID) j(Test)
```

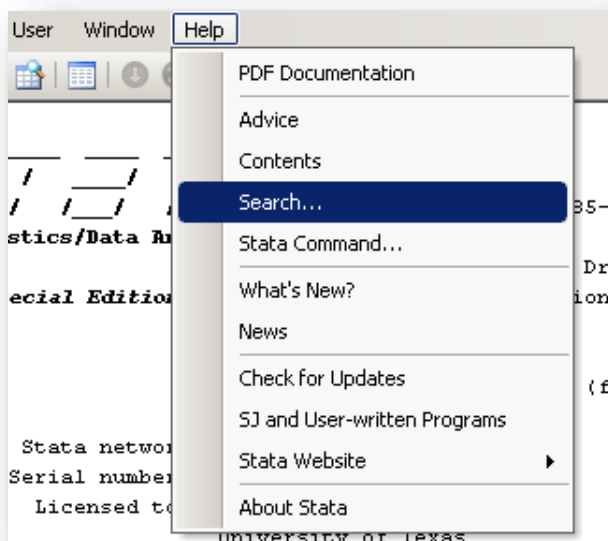
On the other hand, if our data were in wide format, and we wanted it to be converted to long format, we would run this command. This would be applicable if we wanted to run a linear mixed model, which needs the data to be in long format:

```
reshape long Score, i (ID) j (Test)
```

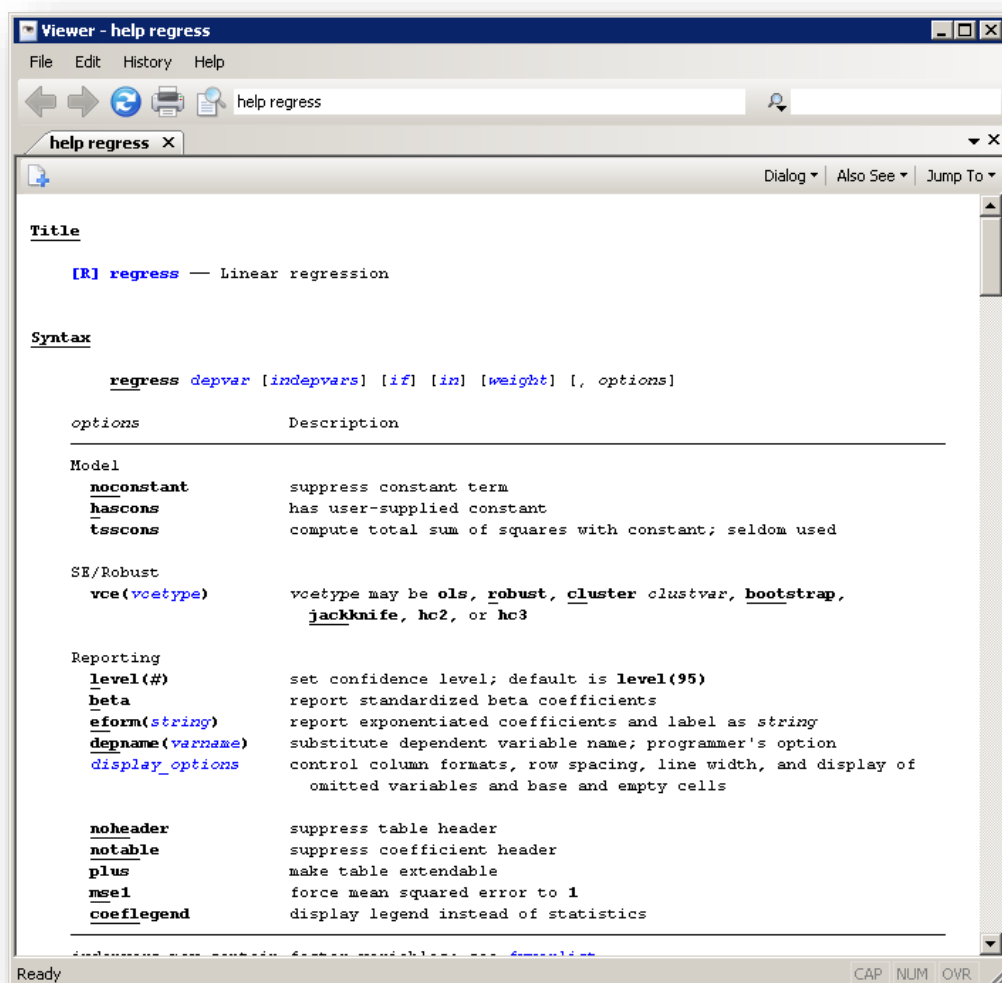
Section 8: Help Menus and Log Files

8.1 Help Menus

Stata 12 has thorough built-in help system that contains details about all functions and commands. It can be accessed from the “Help” menu seen below or by typing “help” followed by a command in the command prompt.



For example, if you wanted to know details about the linear regression function and a list of all the possible options for the command, you could simply type **help regress** in the command prompt and the following window will pop-up:



Contained in the help section for every Stata command is the specific syntax, equivalent drop-down menu selection (if applicable), description of what the command does, and details about options that the command recognizes. Most help pages also contain one or more examples of the function being used, which is often the most helpful thing to look at when having trouble understanding a command. We recommend getting in the habit of using the help menus whenever you are unsure about syntax, or wonder if Stata has the capability of running a certain type of analysis.

8.2 Log Files

In some situations, it might be beneficial to save exactly what commands you run, in the order you run them, along with all of the results. For example, if you were running a series of complicated analyses on pilot data that you anticipate needing to re-run later on the full dataset, knowing exactly what you ran would be helpful.

Stata has the capability of doing this with “log” files. A log file will record everything from the main window (including the actual code) from whenever you create it until you want it to stop. To make a new file, go to **File → Log → Begin...** Stata will then ask you to select a location and file name where it will save the log. Once you hit **OK**, the log starts running, and anything you do in the current session of Stata will be recorded.

You can pause the log by selecting **File → Log → Suspend**, and then resume by simply selecting **Resume**. The log will stop recording if you select **File → Log → Close** (or typing **log close** in the command prompt) or when you close out of the current Stata session.

Stata saves the log as an *.smcl* file, which can be viewed within Stata itself, or in Microsoft Word. However, if you open it in Word, all font styles and colors will be lost, which make it fairly difficult to distinguish between commands and output. Below is what a log will look like if you open in Stata by going to **File → Log → View...**

The screenshot shows the Stata Viewer window titled "Viewer - view \\austin.utexas.edu\disk\ska235\wtprofile\Desktop\Untitled.smcl". The log content is as follows:

```

name: <unnamed>
log: \\austin.utexas.edu\disk\ska235\wtprofile\Desktop\Untitled.smcl
log type: smcl
opened on: 31 Aug 2012, 16:30:38

. import excel "\\tsclient\Z\ska235\cars_1993.xls", sheet("CARS_1993") firstrow

. sum type Price CityMPG

```

Variable	Obs	Mean	Std. Dev.	Min	Max
type	92	3.130435	1.638932	1	6
Price	92	19.04891	8.623728	7.4	47.9
CityMPG	92	22.40217	5.63946	15	46

```

. list Horsepower in 1/10

```

	Horsepower
1.	255
2.	300
3.	92
4.	160
5.	140
6.	102
7.	90
8.	63
9.	300
10.	105

Section 9: Conclusion

Hopefully this tutorial has taught you the basics of navigating and understanding some of what Stata 12 is capable of and how it can be used to prepare datasets for analysis. Please see the next tutorial, “Data Analysis with Stata 12,” for detailed information on the types of statistical analyses that Stata can perform.

If you have any questions on the material presented here, or about other procedures in Stata that might be more appropriate for your data, please feel free to contact us at stat.consulting@austin.utexas.edu. If you have a question about Stata or other statistical software packages, feel free to set up an appointment with one of our consultants by visiting <http://stat.utexas.edu/consulting/free-consulting>.