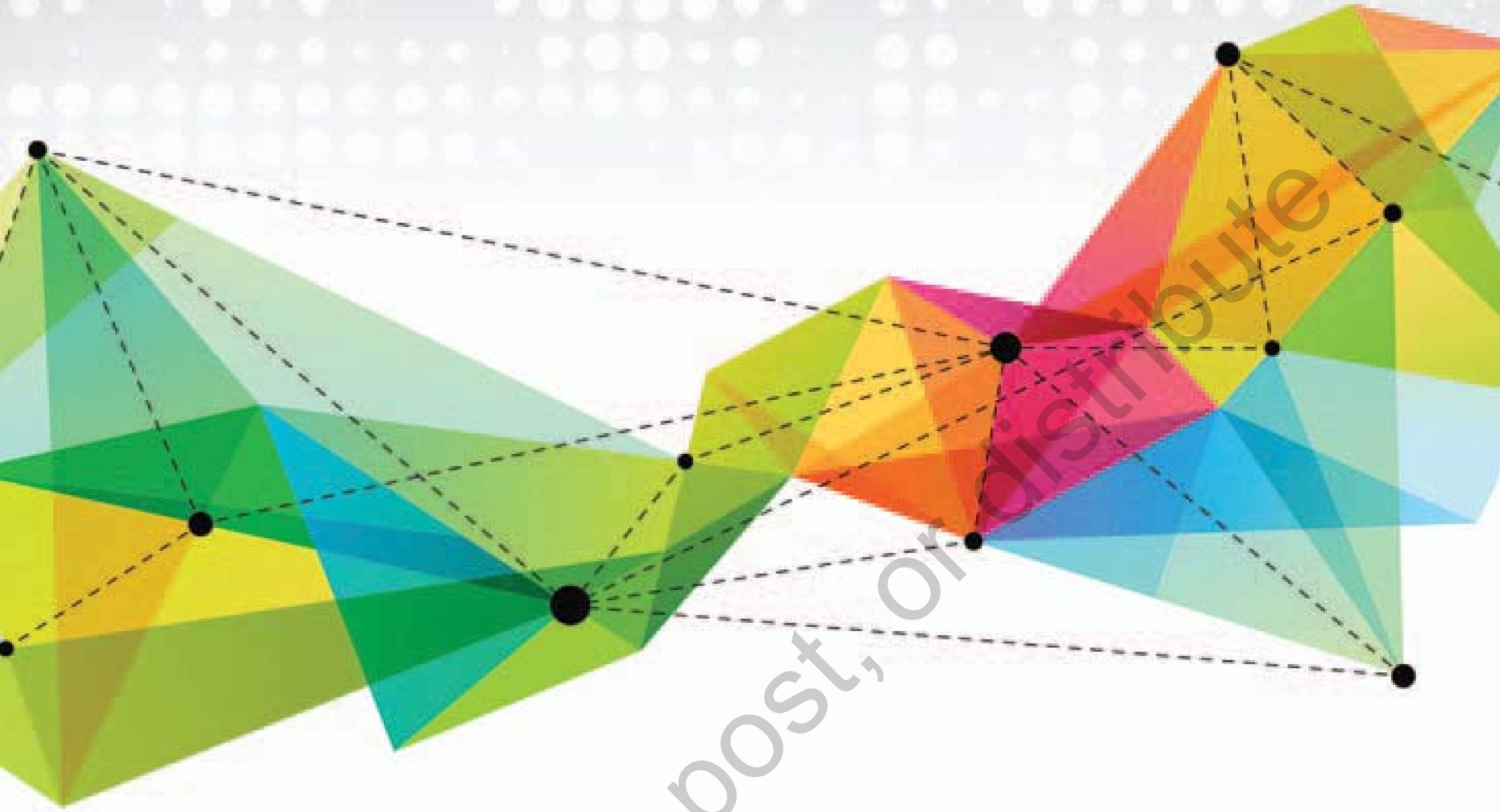


Applied **Ordinal Logistic Regression** Using Stata

Xing Liu

CHAPTER 1

Stata Basics

Objectives of This Chapter

This chapter begins with an introduction to the basics of Stata. The first section shows novice Stata users the interface, menus, and Stata toolbar. It also introduces the structure and rules of Stata commands, official and user-written commands, do-files, comments in do-files, and a do-file template. The second section shows readers various data management techniques, such as how to create a new variable, recode an existing variable, label a variable, and label values. The third section shows how to draw histograms, bar charts, box plots, and scatter plots, as well as how to use Graph Editor to edit graphs. After reading this chapter, you should be able to

- Start and exit Stata using different approaches.
- Enter data using Stata data-editor, and save a new data file.
- Open existing data files.
- Enter commands, create do-files, and save output.
- Create new variables, recode and label variables, and label values for categorical variables.
- Draw different types of graphs.

1.1 Introduction to Stata

Stata is a general-purpose statistical software package for data management, data analysis, and graphing. It can be run on almost all operating systems across Windows, Mac, Linux, and Unix. Stata is relatively small; yet it is powerful. It does not require much hard drive space. The space needed for version 8 or earlier is less than 30 MB. For versions 9 to 10, 100 MB of space should be good enough for the installation. With the inclusion of Stata manuals in the installation folder, versions 11 and 12 need about 250 MB and 500 MB of disk space, respectively; Stata 13 needs about 700 MB of hard drive space, and Stata 14 needs 900 MB. Excluding other files, the Stata executable and ado-files (i.e., the programs) in versions 13 and 14 only take about 150 MB.

Stata is fast. It starts fast! You will not need to sit there waiting. It also runs fast since it loads data directly into your memory when you work on your data and conduct analysis. Before Stata 12, if you had to work on a large dataset that exceeded the allowed amount of memory, you needed to set the memory size by using the `set memory` command. Starting with Stata 12, however, there are no longer memory constraints and you do not need to set it manually. It automatically adjusts your memory based on your needs via the new automatic memory manager.

Stata is a powerful tool for data management, graphics, and data analyses. As a general-purpose statistical software package, it is meant for conducting various statistical analyses. These analyses include, but are not limited to, the following techniques: basic statistical analyses, such as descriptive statistics, *t* test, analysis of variance (ANOVA), correlation, regression, and nonparametric tests; categorical data analyses for binary, ordinal, multinomial, and count outcomes; multivariate analyses, such as multivariate analysis of variance, cluster analysis, discriminant analysis, factor and principle component analysis, multidimensional scaling, correspondence analysis, and biplot; time series; survival analysis; propensity score analysis; structural equation modeling; longitudinal/panel data analysis; power analysis; survey data analysis; and multilevel modeling for continuous and categorical outcomes. The latest Stata 14 includes new statistical models for Bayesian analysis and item response theory (IRT).

Stata has nice programming capabilities. One advantage of Stata is that users can write programs with new features and functions that can be shared with other users in the Stata community. With the contributions of experts from various fields, new statistical techniques can be quickly implemented in Stata. For example, the user-written program `gllamm` for generalized linear latent and mixed modeling was developed by Rabe-Hesketh, Skrondal, and Pickles (2004) before the official release of the `xtmixed` program. We can search and install these user-written programs within Stata using the `search` or `findit` commands. After installation, these programs can be executed in the same way as the official Stata programs. So, when functions of interest are unavailable in the official programs, users can see whether they have been developed by other users.

Stata has complete documentation and is readily available to users. Starting with version 11, PDF manuals were installed within Stata so that users can now directly access them from the help menu. Stata 13 has 20 manuals with more than 10,000 pages, and the newest version, Stata 14, has 23 manuals with more than 12,000 pages. Stata also has a user-friendly help system. To get help for a particular command, you can either select it from the menu or type the `help`, `search` or `findit` commands on the command line. The help file you find is linked directly to its related entry in the PDF manuals.

Stata is mainly a command-driven statistical package. To execute a command, you need to enter it on the command line and press the **Enter** key. This task may seem daunting to a novice Stata user who may then turn around and look for a more user-friendly program. But wait a minute! Stata also has a graphic user interface (GUI), which was introduced in Stata 8. This point-and-click menu system helps new users to learn about the program. It makes it easier to familiarize yourself with the features of data management, graphics, and statistical tools. When you perform an analysis using the GUI, Stata displays the corresponding command in the **Preview** window and the output in the **Results** window. You can then save the command for future use.

Which flavor of Stata do you need? Stata offers several different flavors (or versions): Small Stata, Stata/IC (intercool), Stata/SE (special edition), and Stata/MP (multiprocessor). Among them, Stata/SE might be the most commonly used. All flavors have the same statistical functions but differ in how they handle the number of variables and observations, as well as the speed of computation. The Small Stata stands for "Stata for small computers" and can only handle up to 99 variables and 1,200 observations, so it is for smaller datasets. The other three flavors are the professional versions of Stata, and they almost do not have any limitation on observations as long as your computer memory allows for it. For example, Stata/IC and Stata/SE allow 2 billion observations and Stata MP allows up to 20 billion observations, but you need to have a computer with large memory. Stata/IC can handle a maximum of 2,047 variables, which is good enough for normal data analysis. Stata/SE is for a single-process computer, whereas Stata/MP is for a computer with a multicore processor. Both of them can handle 32,767 variables, and you rarely see a dataset with more variables than that.

1.1.1 Do You Still Need to Use Commands?

Since Stata has a well-built GUI system, why do we still need text commands in a non-DOS era? There is nothing wrong with using the GUI pull-down menus, and you should never feel inferior because you are using the functions via the GUI instead of via text commands. As long as you know how to use it, it will fulfill your needs. As you become an experienced Stata user, however, you will find it will be more efficient to type commands. There are three reasons for this efficiency. First, it saves you time. For example, to run a simple regression analysis with a dependent variable `y` and an independent variable `x`, you simply type:

```
regress y x
```

Second, you can save all your commands to a do-file so that you can replicate your analysis easily. In addition, you can edit your do-file for other analyses. For example, if we have three independent variables in a linear regression analysis, then we can modify the previous command as follows:

```
regress y x1 x2 x3
```

Third, you know what analysis you are doing by typing commands interactively. The GUI environment will allow you to point and click to any menus and produce results without knowing the underlying statistical techniques. Statisticians have always been concerned about abusing the use of statistical software when users have limited knowledge of the models. Stata helps alleviate this concern.

1.1.2 Stata at First Sight: Interface, Menus, and Toolbar

Stata can be started in three ways. First, you can run Stata by clicking the icon on the desktop. Second, you can start it by clicking the **Start** Menu on Windows, **All Programs**, and then **Stata**. Third, you can open it by double clicking a Stata data file.

To exit Stata, you can either type the command `exit, clear` and press **Enter** or use the pull-down menu. To use the menu, go to **File** and then click on **Exit**.

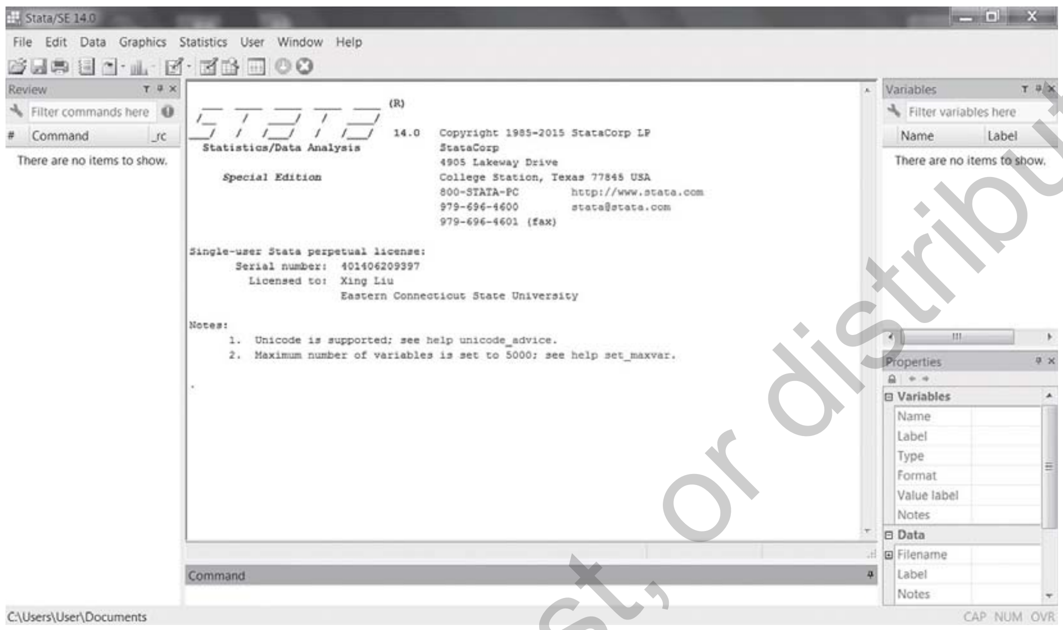
Stata Windows

Once you start Stata, you will see five windows (Figure 1.1): **Command**, **Results**, **Review**, **Variable**, and **Properties**.

Command window: In this window, you need to type a command and press the **Enter** key to execute it. You can copy a command from a Notepad or a Word document and paste it here. You can also edit a command before execution. You can only execute one command at a time in this window, whereas you can run a series of commands via the do-file. Normally you will also need to open a data file before running commands for data analysis.

Results window: This window displays the output after you execute a command. All the commands you entered in the Command window are echoed in the Results

Figure 1.1 Stata Windows



```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   14.0   Copyright 1985-2015 StataCorp LP
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
     Special Edition                  College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

Single-user Stata perpetual license:
       Serial number:  401406209397
         Licensed to:  Xing Liu
                       Eastern Connecticut State University

Notes:
      1.  Unicode is supported; see help unicode_advice.
      2.  Maximum number of variables is set to 5000; see help set_maxvar.

.
```

window, followed by the output. If you see `–more–` at the bottom of the Results window, it means that the screen is full and that you need to click on it so that you can see more results. To see all the output without showing this message, you can type the command `set more off`. The output can be copied and pasted into a text file or a Word document. It can also be saved into a log file, which will be explained in more detail next.

Review window: This window displays the commands that have been executed from the Command window. A nice feature of this window is that you can click on any command and bring it back to the Command window without retyping it. Then, you can modify it for the following analyses. You can also select any commands in the Review window and save them into a do-file.

Variable window: This window contains variable names in the dataset. Instead of typing variable names in a command, clicking any variable in this window will bring it to the Command window, which will be helpful to save time and reduce typos.

Properties window: This function was introduced in Stata 12. This window shows the properties of variables and the dataset. If you select a variable in the Variable window, the Properties window will display the variable name, label, type, format, value label, and notes. To modify these properties, you need to unlock the function first. To see the properties of the next variable, click on the arrow key at the upper right corner of the Properties window.

Stata Menus

Stata has eight pull-down main menus, including File, Edit, Data, Graphics, Statistics, User, Windows, and Help. These menus provide tools and features that can be used by pointing and clicking.

File menu: The purpose of this menu is to help you with files. Options in this menu help you open existing Stata datasets, graphs, and recent files; save data and graphs; open do-files; create and close log files; import and export data files; print results; open example datasets; and exit Stata.

Edit menu: The Edit menu helps you copy texts and tables from the output and paste them in another location so that you can save them.

Data menu: This menu provides functions for data management. By using this menu, you can describe data; edit data using the data editor; browse data using the data browser; create and change variables; sort variables; combine datasets; label datasets, variables, and values; add or delete notes to datasets; rename variables; add or drop variables or observations; identify duplicate observations; move the position of variables; create matrixes and work with them; and use other utilities.

Graphics menu: You can create and manage graphics by using various options on this menu.

Statistics menu: This menu provides the statistical analysis tools Stata offers. The list of Statistics menu options has grown as Stata has incorporated new statistical procedures into the package.

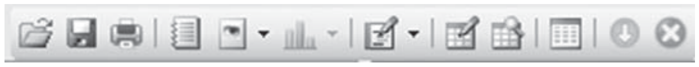
User menu: This menu is seldom used. Clicking on it brings you three empty submenus. It is useful only if you would like to add your own menu items for Data, Graphics, and Statistics menus.

Window menu: This menu contains a series of submenus, such as follows: Command, Results, Review, Variable, Properties, Graph, Viewer, Data Editor, Do-file Editor, and Variables Manager.

Help menu: This menu provides help and search facilities, such as having access to the PDF documents, getting advice, learning Stata via the contents, searching Stata commands, getting help with a command, finding resources for Stata learning, finding and installing user-written programs, and checking for updates.

Stata Toolbar

The toolbar, located below the main menus, comprises a set of icons. It helps you quickly access the most frequently used features. Familiarizing yourself with these icons will make the use of Stata more efficient.



These icons include Open, Save, Print Results, Log, Viewer, Graph, Do-File Editor, Data Editor (Edit), Data Editor (Browse), Variables Manager, Clear –More– Condition, and Break. Table 1.1 shows the icons of the toolbar, their titles, and their functions.

1.1.3 Creating a File and Entering Data

To create a new dataset, you can use the Stata Data Editor either by clicking on the **Data Editor (Edit)** icon on the Toolbar or by going to the **Data** Menu and selecting **Data Editor**. Figure 1.2 shows the window of Data Editor (Edit). Once the Data Editor window is opened, you can enter data just as you would in a spreadsheet. Stata automatically names your variables to `var1`, `var2`, `var3`, … from the left column to the right. Since the release of Stata 12, you can add/edit variable names, types, formats, variable labels, value labels, and notes in the Properties window at the bottom right corner. If you use an earlier version, you can change the variable name by double clicking on it. To save the data you have entered, click on the **Save** icon on the Toolbar or go to **File** and then **Save As**. You can also enter the command `save` on the command line with a file name, for example, `save data1`. The extension of the Stata dataset is .dta.

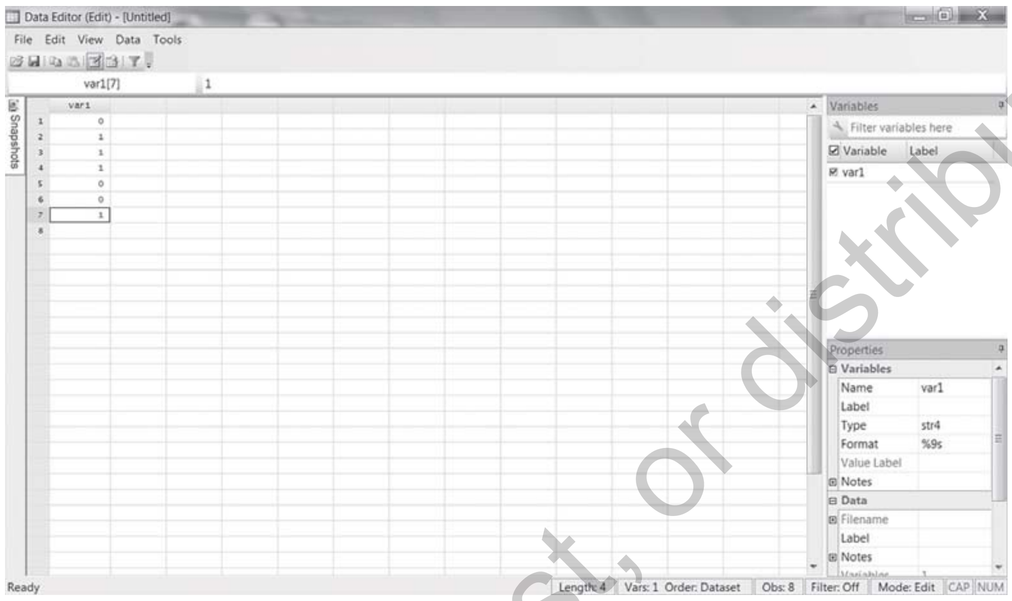
For Stata 13 users, although Stata 13 can read data saved by previous versions, older versions may not be able to read data created by newer versions. The command `saveold` can save a dataset in Stata 12 format. To create a dataset compatible with versions older than 12, use the user-written command `use13`.

For Stata 14 users, the improved `saveold` command can create a dataset readable by versions 11 to 13. Since Stata 14 now supports Unicode, whereas older versions do not, you may see that variable names in a dataset created by Stata 14 are not properly displayed in older versions. The command `use13` currently does not work in Stata 14. It is hoped that the issues of data transfer will be solved once `use14` is available. Be careful with data conversions. Another option is to use Stat/Transfer, a program for converting datasets among various formats. The latest Stat/Transfer 13 supports the data formats for all versions of Stata.

Table 1.1 Icons of the Toolbar, Their Titles, and Their Functions

Icon	**Icon Title**	**Functions**
📂	Open	Opens an existing Stata file (e.g., .dta, .gph, .do)
💾	Save	Saves the current dataset to your computer
🖨	Print	Prints your Stata output
📓	Log	Begins, closes, suspends, or resumes your log file
👁	Viewer	Opens the Viewer to get help
📊	Graph	Brings a graph window to the front
📝	Do-File Editor	Opens the do-file editor so that you can make or edit a do-file
✎	Data Editor (Edit)	Opens the data editor so that you can edit the current data file
🔍	Data Editor (Browse)	Opens the data editor so that you can browse rather than edit the current data file
▦	Variables Manager	Opens the variables manager so that you can change variable properties, such as the variable name, label, type, format, and value label
⬇	Clear –More– Condition	Tells Stata to show more output
⊗	Break	Stops executing the current program

Figure 1.2 Data Editor (Edit) Window



1.1.4 How to Open an Existing Data File

To open an existing Stata .dta file, you can use one of the following four ways:

- First, **double click** the Stata data file, which starts the Stata program, and Stata will read the data.
- Second, **enter** the command `use` followed by a file path on the command line. If a Stata data file is saved in the current working directory, you only need to type `use` followed by the file name. For example, **type** `use Chapter1.dta` and **press** the **Enter** key. If the file is saved elsewhere on the computer, you need to type the full path. The `use` command can also open a data file on the web followed by the web link. To open a dataset for Stata manuals, type the `webuse` command followed by the filename. In addition, the `sysuse` command can be used to open datasets stored in the system directories. For example, to use a system data file `auto.dta`, simply **type** `sysuse auto` in the Command window and press **Enter**.

- Third, click the **Open icon** on the Toolbar. Once you click on the icon, it will open the dialog. Select the dataset and then click on **Open**.
- Fourth, go to **File** on the main menu, and then click on **Open**. Select the data file and then click on **Open**.

Other Data Formats

Besides its own data format, Stata can read a variety of other data formats. For example, ASCII files, Microsoft Excel files (both .xls and .xlsx formats), and SAS transport format (SAS XPORT). Since the release of version 12, Stata can directly read Excel files using the `import excel` command or via the pull-down **File** menu.

1.1.5 The Structure of Stata Commands

Stata commands are concise, so do not be afraid of learning the command syntax. As a new user of Stata, you may ask, "Is it difficult to learn how to use Stata, particularly Stata commands?" The answer is "no." You just need to understand the structure of Stata syntax and get some practice on your computer. Now, let us look at several examples of Stata commands. Then, I will explain its generic structure.

The simplest structure of a command is the command name itself. For example, to get the descriptive statistics of all variables in a dataset, you just type the command `summarize.` If we want to see the descriptive statistics of a particular variable, type the variable name after the command `summarize`. For example, `summarize var1`. Entering more variable names after `summarize` will generate descriptive statistics for those variables.

To get a frequency of a categorical variable, type the command `tabulate` plus the variable name. For example, `tabulate var2`. If you get frequency tables for more than one variable, use the `tab1` command. For example, `tab1 gender class`.

Suppose you want to run descriptive statistical analysis on a subset of the data. To do this, you can use the `if` qualifier. For example, to see the descriptive statistics of students' reading scores in class 1 of two classes, enter the command `summarize reading if class == 1`.

If you want to see the descriptive statistics of a variable in the first 50 observations, you can use the `in` qualifier. For example, `summarize reading in 1/50`.

From the previous examples, we can induce a simple structure of Stata commands:

```
command [varlist] [if] [in]
```

The `command` stands for any Stata command. The `varlist` means variable list, which can be one variable or a list of variables. `If` stands for the "if qualifier," and `in` stands for the "in qualifier." In the command syntax, the command name is required and the elements in brackets are optional. When we enter a command, we should not enter any brackets, which are only used to explain the optional components in the command syntax here.

This simple structure of Stata commands can be expanded to a more generic one. We can add weight and command options to the syntax. The options are specified at the end and separated from the command by a comma. A prefix can also be added before the command. So the generic Stata command syntax you see in the manuals or in the command help dialog looks like this:

```
command [varlist] [if] [in] [weight] [, options]
```

A prefix can also be added before a Stata command. The `by` prefix is used to repeat a command for each category or group of the data. The `svy` prefix is used for the complex sampling survey data. The `xi` prefix creates dummy variables in an analysis.

Rules of Stata Commands

1. Stata commands should be error-free. In other words, you need to type them exactly.
2. Stata commands are case sensitive, so the lowercase and uppercase letters are different. Be careful when the variable names are mixed with lowercase and uppercase letters. When you name a variable, it is always a good idea to use lowercase.
3. Stata commands are typically in lowercase. It is rare to see a capital letter in Stata commands.

Official Commands and User-Written Commands

The official commands are those developed and released by Stata. They come directly with the Stata installation disk or files. Once Stata is installed, you can immediately execute these commands. In the Stata community, a wide range of Stata commands are developed by users. These commands do not come with your installation disk or files. They complement the official command by extending the capabilities of Stata. They are free and can easily be installed to your Stata. Most of the user-written commands were published in the *Stata Journal* or its predecessor, the *Stata Technical Bulletin*. Most of them are stored in the Statistical Software Component (SSC) Archive at Boston College (http://ideas.repec.org/s/boc/bocode.html) and can be accessed freely.

If you want to install a user-written program that you know the name of, just type the command `ssc install` followed by the program name. For example, to install a program named `ocratio`, which is used for continuation ratio models, type `ssc install ocratio`, and Stata will install it for you. You can also use the command `findit` to search a user-written program or to see whether the program has been installed. As a result, Stata will search official help files, FAQs, examples, and web resources, and it will provide a list of related links for this command. You can browse these links and decide which you want to install. Using the `findit` command gives you more control than directly installing it with the `ssc install` command.

If you want to check the version of a user-written program you have installed, type the `which` command followed by the program name. For example, if the `outreg2` program has been installed, type `which outreg2` and the version of the program will be displayed.

If you want to uninstall a user-written program previously installed from SSC, type `ssc uninstall` followed by the program name. If a program is not installed from SSC and you want to uninstall it, type the `ado, dir` command. After seeing the listed program in the Results window, type `ado uninstall` followed by the program name.

Different from the official Stata programs, the user-written programs cannot be updated automatically. If you want to update a user-written program to the latest version, type the `adoupdate` command. Stata will check the status of all the installed user-written programs and list the numbered programs that need to be updated. By typing the `adoupdate, update` command, all recommended programs will be updated. If you only want to update one of the listed programs, specify the program name after the `adoupdate` command.

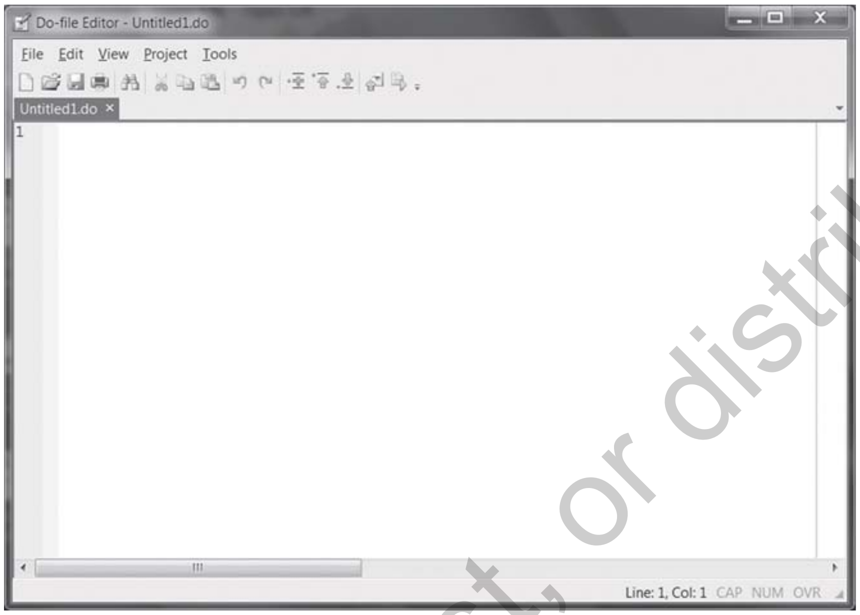
1.1.6 Do-Files

If you use the Command window to enter commands, you can enter one piece at a time. Do-files can help you put a list of commands in one file and execute them together as a batch. Using do-files helps you to organize commands, keep a record, and understand what you have done when you need them in the future. It is also helpful when you collaborate with other researchers on a research project. Your colleagues can simply replicate the analyses using the do-files you provide, and they can modify them for new analyses. It will save you time when you need to replicate your statistical analyses. If you are an instructor, these do-files are also helpful when preparing your instruction and for grading students' assignments.

To create a do-file, you can use the Do-file Editor by clicking the **Do-file Editor** icon on the toolbar, or you can just type the command `doedit`. This will open up the Do-file Editor. Figure 1.3a shows the window of the Do-file Editor. After entering a list of commands, you can save it from the **Review** window as a do-file with a .do extension, for example, chapter1.do. To execute this do-file, type `do chapter1.do`.

You should save the do-file to the current working directory. Otherwise, you will need to type the whole path with the file name when you execute it. To check your current working directory, type the command `pwd` or `cd`.

Figure 1.3a Window of the Do-file Editor


Do-file Editor - Untitled1.do
File Edit View Project Tools
Untitled1.do ×
1
Line: 1, Col: 1 CAP NUM OVR

A good habit when you create a do-file is to have comments in the file. The comments may include time, project name, who wrote it, and for what purposes. The clearer your comments are, the better your documentation will be. The comments in the do-files will not be executed but will be displayed in your output. There are several ways to write comments. A comment would start with a * after a double slash, //, or after a triple slash, ///. For a long comment, you can use a combination of /* and */. Stata recognizes it as a comment if you follow these four formats. Please note that the triple slash (///) is often used as a line-join indicator for long command lines in do-files. It tells Stata that the following command line is a continuation of the current line and both should be executed together. Users may feel confused with the two functions of the triple slash. In addition, a comment with //, ///, and /* */ can only be used in do-files or ado-files, and it should not be entered in the Command window.

Long (2009) provided detailed suggestions on how to write do-files. He suggested that the do-files be "robust and legible" (p. 50). They are robust when researchers can produce the same results when rerunning the same command or conduct the same analysis using a different computer. They also need to be legible so that researchers can understand what analyses can be done using these do-files. Specific suggestions for making robust do-files by Long (2009) mainly include making the do-files independent of each other, using version control so that the do-files work with previous versions, and excluding directory locations so that the do-files can be run in the current working directory. Another suggestion is to set the seeds when you generate random numbers so that the same results can be produced.

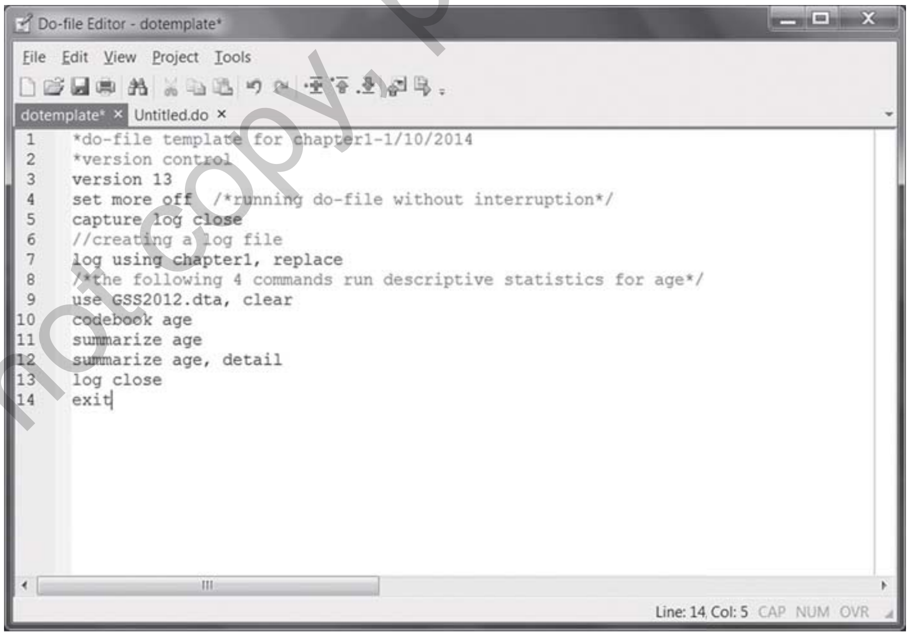
To make the do-files legible, Long (2009) also suggested using various comments, employing alignment and indentation, setting line size, limiting the use of abbreviations, and being consistent in the way of creating do-files.

How to Create a Do-File Template

A do-file template is a template you can use to create do-files for any statistical analysis. It includes some common commands that ease your work. Figure 1.3b is an example of a do-file template with comments.

- Line 1: A comment with the content information and the date.
- Line 2: A comment for the version control.
- Line 3: The command for version control. This command is useful when you need to run commands in older versions. The command `version 13` tells us that the do-file will work in this version.

Figure 1.3b Example of a Do-File Template


Do-file Editor - dotemplate*
File Edit View Project Tools
dotemplate*
Untitled.do
Line: 14, Col: 5 CAP NUM OVR

```
1   *do-file template for chapter1-1/10/2014
2   *version control
3   version 13
4   set more off  /*running do-file without interruption*/
5   capture log close
6   //creating a log file
7   log using chapter1, replace
8   /*the following 4 commands run descriptive statistics for age*/
9   use GSS2012.dta, clear
10  codebook age
11  summarize age
12  summarize age, detail
13  log close
14  exit
```

- Line 4: The `set more off` command is used so that your do-file can be run without interruption. A comment follows the command.
- Line 5: If you would like to create log files so that the output will be automatically saved, use the `capture log close` command before creating the log files. This command tells Stata that no other log files are open or in use.
- Line 6: A comment for creating a log file using a command below.
- Line 7: The command `log using chapter1` creates a log file to save the output.
- Line 8: A comment for the following four commands.
- Lines 9–12: The core commands for descriptive statistics.
- Line 13: The `log close` command closes the log file after the analysis is completed.
- Line 14: The `exit` command closes Stata. This command is optional in the template. It can be omitted if you would like to continue your data analysis.

1.1.7 How to Save Stata Results

When you execute a command from the Command window or from a pull-down menu, Stata will display the output in the Results window. There are two ways to save the output. First, a simple way is to copy and paste it. You can copy the selected output as text and paste it into a Word document or a text file. If you paste it into a Word document, then you need to specify the font as `Courier New` and set the font size to 9 or smaller. Otherwise, it will not be shown properly since the table will be misaligned. You can also copy and paste it as a table or as a picture. One disadvantage of this method is that you need to do it manually. It will be tedious and time-consuming to use the copy-and-paste method if your output is long or if you save different results to separate folders.

The second and most common way to save results is to create a log file, which is more efficient than the copy-and-paste method. By using a log file, you can save all of your results at the beginning of your Stata session. The log file can be created in either of the following two formats. One is a plain text format with the extension .log, and the other is a Stata Markup and Control Language format (.smcl), which is the default log format. The plain text format log is preferred since it can be easily opened and edited by other programs, whereas the log in .smcl extension can only be opened by Stata itself. To create or start a log file, type:

```
log using filename, text
```

The “`filename`” is the log file you name. The “`text`” means that the format is plain text. If you want to replace a previous log with the same name, you can add “`replace`” to the command. So the modified command is:

```
log using filename, replace text
```

If you want to add the current log file to an existing log file, you can add the append option as follows:

```
log using filename, append text
```

To suspend your logging, type:

```
log off
```

You can resume your logging by typing:

```
log on
```

To close your log file after you complete an analysis, simply type `log close`.

If you use the point-and-click menu to create a log, go to **File**, select **Log**, and click on **Begin**. Provide a name for your new log file, select either the .txt or .smcl format type, and then click **Save**.

A log file will be saved in the current working directory. If you are uncertain of your working directory, type `pwd` in the **Command** window. You can use the command `cd` to change the working directory.

1.1.8 What If I Have a Question? How Do I Get Help?

We have many questions when we use Stata. Thankfully, Stata has a complete, convenient help system, which provides rich resources for users in different ways.

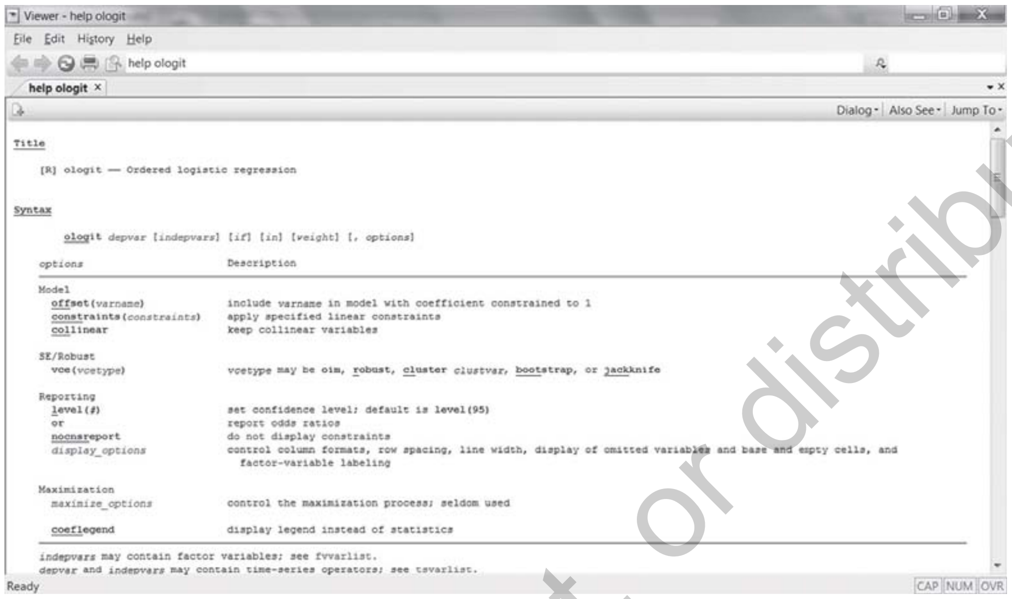
First, Stata itself provides help. Stata has online help and search facilities, which provide a wide range of help. If you have a question for a particular command, you can either click on **Help** from the menu or type `help`, `search` or `findit` commands with the name of the command on the command line. For example, if you are looking for the syntax of the `ologit` command for the proportional odds model for ordinal outcomes, you could type `help ologit`. The screenshot shown in Figure 1.4 shows the syntax for `ologit` with descriptions. Examples are also provided in the help file, which is truncated in the screenshot.

The same screenshot can be reached if you use the pull-down menu. Go to **Help** on the menu, click on **Stata Command**, and then enter `ologit` in the command box. You can see the **help ologit** after clicking on **OK**.

The help command only focuses on a particular command. The other three options for broader help are the `search` command, the `hsearch` command, and the `help contents` command.

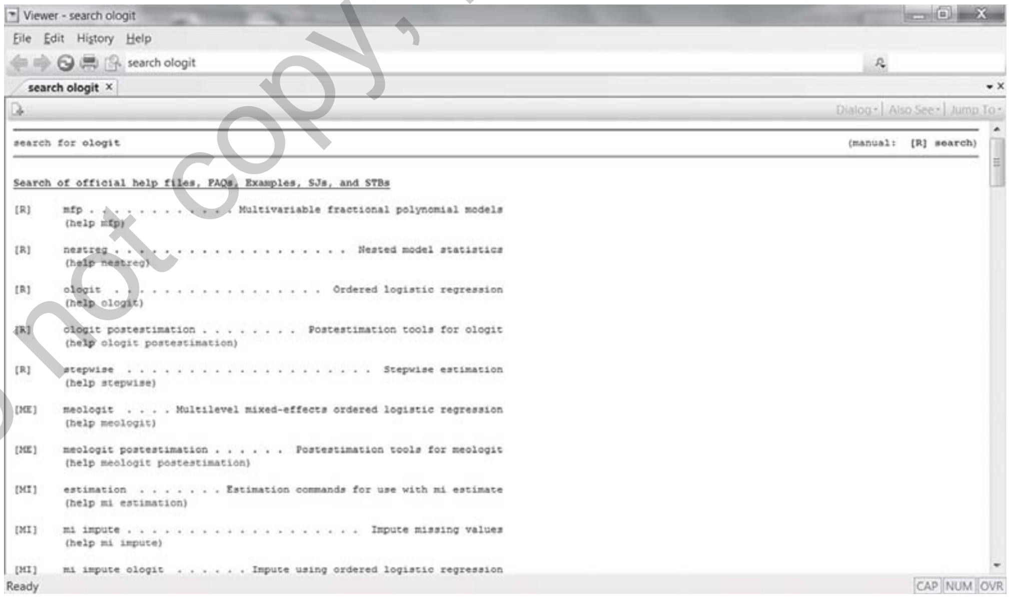
The `search` command provides a keyword search. For example, `search ologit` provides a keyword search for the `ologit` command, which is displayed in Figure 1.5.

Figure 1.4 Screenshot of the `help ologit` Command



```
Title

    [R] ologit — Ordered logistic regression


Syntax

        ologit depvar [indepvars] [if] [in] [weight] [, options]

    options                       Description
    ------------------------------------------------------------------------------------------------
    Model
      offset(varname)             include varname in model with coefficient constrained to 1
      constraints(constraints)    apply specified linear constraints
      collinear                   keep collinear variables

    SE/Robust
      vce(vcetype)                vcetype may be oim, robust, cluster clustvar, bootstrap, or jackknife

    Reporting
      level(#)                    set confidence level; default is level(95)
      or                          report odds ratios
      nocnsreport                 do not display constraints
      display_options             control column formats, row spacing, line width, display of omitted variables and base and empty cells, and
                                    factor-variable labeling

    Maximization
      maximize_options            control the maximization process; seldom used

      coeflegend                  display legend instead of statistics
    ------------------------------------------------------------------------------------------------
    indepvars may contain factor variables; see fvvarlist.
    depvar and indepvars may contain time-series operators; see tsvarlist.
```

Figure 1.5 Screenshot of the `search ologit` Command



```
search for ologit                                                           (manual:  [R] search)

Search of official help files, FAQs, Examples, SJs, and STBs

[R]     mfp . . . . . . . . . . . . . Multivariable fractional polynomial models
        (help mfp)

[R]     nestreg . . . . . . . . . . . . . . . . . . .  Nested model statistics
        (help nestreg)

[R]     ologit  . . . . . . . . . . . . . . . . .  Ordered logistic regression
        (help ologit)

[R]     ologit postestimation . . . . . . . .  Postestimation tools for ologit
        (help ologit postestimation)

[R]     stepwise  . . . . . . . . . . . . . . . . . . . .  Stepwise estimation
        (help stepwise)

[ME]    meologit  . . . . Multilevel mixed-effects ordered logistic regression
        (help meologit)

[ME]    meologit postestimation . . . . . .  Postestimation tools for meologit
        (help meologit postestimation)

[MI]    estimation  . . . . . . . Estimation commands for use with mi estimate
        (help mi estimation)

[MI]    mi impute . . . . . . . . . . . . . . . . . . .  Impute missing values
        (help mi impute)

[MI]    mi impute ologit  . . . . . . Impute using ordered logistic regression
```

The `hsearch` command searches the contents of a command on the Internet. For example, typing the `hsearch ologit` command brings you to the screenshot shown in Figure 1.6.

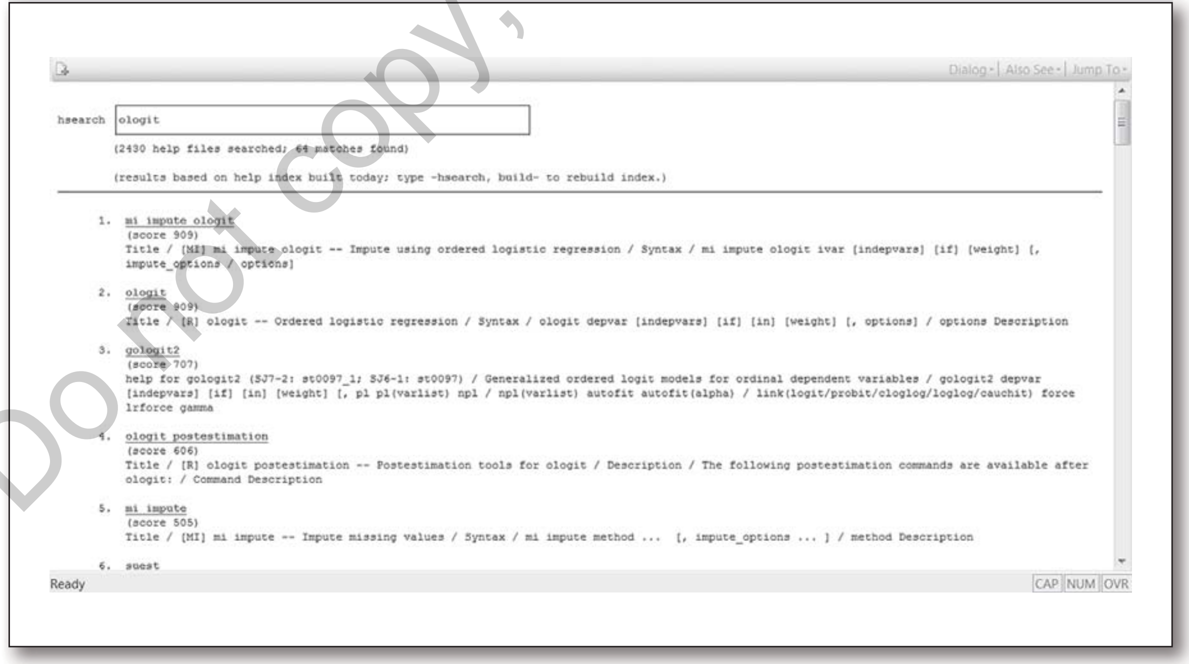
The `help contents` command helps users learn about Stata in a general way. It is not for a particular command, so it is not followed by any Stata command names. If you type `help contents`, you will get the screenshot displayed in Figure 1.7. Topics are organized into the following categories: getting started, data manipulation and management, utilities, graphics, statistics, matrix commands, programming, and interface features. Clicking on any category brings you to the PDF documentation, help, and search files related to that topic.

Second, Stata has provided complete PDF documentation since version 11. Users have access to these manuals directly from the menu or the installation folder within Stata. Since the help file is linked directly to its related entry in the PDF manuals, users can focus on a specific command in the manuals via the help file.

Third, you can watch Stata tutorials on YouTube. Stata has an official YouTube channel for users to learn Stata under the username "StataCorp". Stata periodically uploads short videos regarding various topics that show users how to use Stata. They are free to watch at any time.

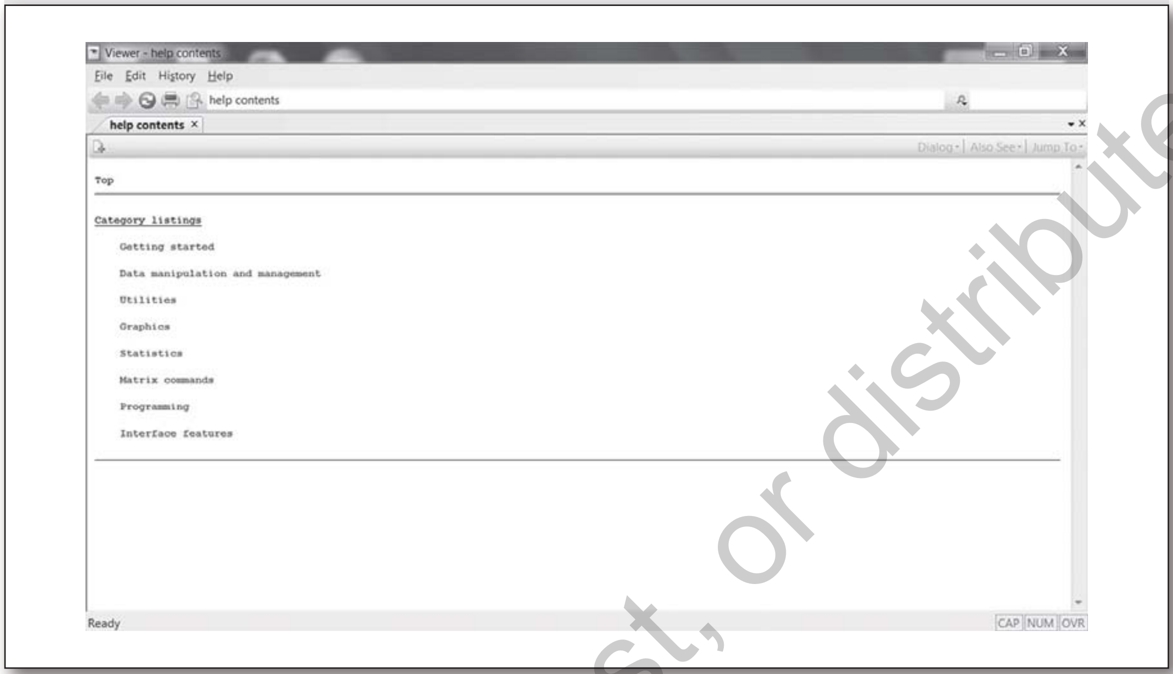
Fourth, you can join the Stata listserver. The free listserver is organized for Stata users. After signing up, you can post your questions and get answers from the Stata user community. Even if you are not a member of the listserver, you can still benefit

Figure 1.6 Screenshot of the `hsearch ologit` Command



```
hsearch  ologit
         (2430 help files searched; 64 matches found)

         (results based on help index built today; type -hsearch, build- to rebuild index.)

    1.  mi impute ologit
        (score 909)
        Title / [MI] mi impute ologit -- Impute using ordered logistic regression / Syntax / mi impute ologit ivar [indepvars] [if] [weight] [,
        impute_options / options]

    2.  ologit
        (score 909)
        Title / [R] ologit -- Ordered logistic regression / Syntax / ologit depvar [indepvars] [if] [in] [weight] [, options] / options Description

    3.  gologit2
        (score 707)
        help for gologit2 (SJ7-2: st0097_1; SJ6-1: st0097) / Generalized ordered logit models for ordinal dependent variables / gologit2 depvar
        [indepvars] [if] [in] [weight] [, pl pl(varlist) npl / npl(varlist) autofit autofit(alpha) / link(logit/probit/cloglog/loglog/cauchit) force
        lrforce gamma

    4.  ologit postestimation
        (score 606)
        Title / [R] ologit postestimation -- Postestimation tools for ologit / Description / The following postestimation commands are available after
        ologit: / Command Description

    5.  mi impute
        (score 505)
        Title / [MI] mi impute -- Impute missing values / Syntax / mi impute method ...  [, impute_options ... ] / method Description

    6.  suest
```

Figure 1.7 Screenshot of the `help contents` Command



```
Top

Category listings

    Getting started

    Data manipulation and management

    Utilities

    Graphics

    Statistics

    Matrix commands

    Programming

    Interface features
```

from it. After searching your question online, you may find that it has been answered by other users via the listserver.

Fifth, you can get help from the Stata website (www.stata.com). Stata provides FAQs, technical support, onsite training, third-party workshops or short courses, and NetCourses. Stata also publishes books and journals (the *Stata Journal* and its predecessor, the *Stata Technical Bulletin*).

1.2 Data Management

1.2.1 Creating a New Variable

When you work on a research project, you often need to create new variables, recode an existing variable to a new variable, or combine several variables into one variable. There are also situations when you need to label a variable or label values for a categorical variable. In this section, we will briefly introduce commands to fulfill these tasks.

The `generate` command is used to create a new variable.

```
generate var1 = 1
```

This command creates a variable `var1` with a value of 1 for all observations.

```
generate var2 = var1
```

This command creates a variable `var2`, which is the same as the original variable `var1`.

```
generate var3 = var1+var2
```

This command creates a variable `var3`, which is the sum of the two variables `var1` and `var2`.

Now, let's see an example using the General Social Survey 2012 (GSS 2012) dataset. I would like to create a new variable `help`, which is the sum of two variables `helppoor` and `helpsick`.

```
. generate help = helppoor + helpsick
(707 missing values generated)
```

The `replace` command is used to change the values or contents of an existing variable. To protect your data, you should avoid making changes to existing variables. Any time you change or modify a variable, it is always a good habit to create a new variable that is the same as the original one and then make changes to the new one. Therefore, the two commands `generate` and `replace` are often used together. For example, suppose you have a variable `sex` with a value of 1 for males and a value of 2 for females. If you want to recode the value of males to be 1 and females to be 0, type:

```
generate male = 1 if sex ==1
replace male = 0 if sex ==2
```

The following output is displayed.

```
. generate male = 1 if sex ==1
(1088 missing values generated)

. replace male = 0 if sex ==2
(1088 real changes made)
```

The double equal sign "==" is different from the single equal sign "=". The former sign is one of the logical operations, and it works with the `if` qualifier, whereas the single equal sign does not work with the `if` qualifier. Instead, it is used when we generate or recode a variable or replace the value of a variable.

You can also create a dummy or binary variable with a combination of the `generate` and `replace` commands. For example, let us create a variable `education` with a value of 1 for respondents' highest year of school completed greater than 13.53 years, and a value of 0 otherwise. You type:

```
generate education = 0
replace education = 1 if educ > 13.53
```

```
. generate education = 0

. replace education =1 if educ > 13.53
(953 real changes made)
```

Instead of using two commands, a shortcut for doing this is to type:

```
generate education = educ > 13.53
```

Let us see a more complex example. If we would like to create a new categorical variable `SES`, according to the family income `realinc`, then we enter the following five commands:

```
generate SES = realinc
replace SES = 1 if realinc >= 0 & realinc <= 10412
replace SES = 2 if realinc >= 10412.5 & realinc <= 22050
replace SES = 3 if realinc >= 22051 & realinc <= 40425
replace SES = 4 if realinc >= 40426
```

The first command creates a new `SES`, which is the same as the variable `realinc`. The second command creates the level 1 for `SES` if the value of `realinc` is equal to or greater than 0 and less than 10,412. The third command creates the level 2 for `SES` if the value of `realinc` is equal to or greater than 10,412.5 and less than 22,050. The fourth command creates the level 3 for `SES` if the value of `realinc` is equal to or greater than 22,051 and less than 40,425. The fifth command creates the level 4 for `SES` if the value of `realinc` is equal to or greater than 40,426.

The following is the Stata output. As we can see, there is no error message so all the commands are executed successfully.

```
. generate SES = realinc
(216 missing values generated)
```

```
. replace SES = 1 if realinc >= 0 & realinc <= 10412
(415 real changes made)

. replace SES = 2 if realinc >= 10412.5 & realinc <= 22050
(560 real changes made)

. replace SES = 3 if realinc >= 22051 & realinc <= 40425
(422 real changes made)

. replace SES = 4 if realinc >= 40426
(577 real changes made)
```

1.2.2 Recoding a Variable

The recode command is used to recode values of an existing variable. For example:

```
recode educ (min/13.53 = 0) (13.54/max = 1)
```

This command tells Stata to recode the values of the variable `educ` from the minimum to 13.53 to 0, and from 13.54 to the maximum to 1.

Since you do not want to overwrite the original variable, generate a new variable when recoding its values:

```
recode educ (min/13.53 = 0) (13.54/max = 1),
generate(education)
```

This command tells Stata to generate a new variable `education` by recoding the values of the variable `educ` from the minimum to 13.53 to 0 and from 13.54 to the maximum to 1. The following is the Stata output.

```
. recode educ (min/13.53 = 0) (13.54/max = 1), generate (education)
(1969 differences between educ and education)
```

Let us see another example. We would like to recode the continuous variable `income` into a new categorical variable `inclevel`, which has three categories. The values between the minimum and 9 of the variable `income` will be coded to 1 for the new variable, and the values from 10 to 11 and from 12 to the maximum will be coded to 2 and 3, respectively. We enter the command:

```
recode income(min/9 =1)(10/11 = 2)(12/max =3),
generate(inclevel)
```

The following is the Stata output.

```
. recode income (min/9 = 1) (10/11 = 2) (12/max = 3), generate (inclevel)
(1733 differences between income and inclevel)
```

You can use the `recode` command to reverse a variable. For example, a survey item uses a Likert scale of 1 to 5, with 1 = strongly agree and 5 = strongly disagree. You want to reverse the order and define strongly disagree to be 1 and strongly agree to be 5. The following is the syntax:

```
recode goodlife (1=5)(2=4)(3=3)(4=2)(5=1), generate
(gliferev)
```

The command tells Stata to reverse the values of the variable `goodlife` and create a new variable `gliferev`.

The following is the Stata output without any error message.

```
. recode goodlife (1=5) (2=4) (3=3) (4=2) (5=1), generate (gliferev)
(1108 differences between goodlife and gliferev)
```

1.2.3 Labeling a Variable

Labeling shows the meaning of a variable. To label a variable, use the `label variable` command as follows:

```
label variable varname "label text"
```

For example, if you want to label a variable `efficacy` with the text `mathematics self-efficacy`, enter:

```
label variable efficacy "mathematics self-efficacy"
```

Another example:

```
label variable watchtv "number of hours in watching TV"
```

Now, using the GSS 2012 dataset, let us label the new variable `gliferev` with the text `standard life of you will improve (recoded)`. You would enter:

```
label variable gliferev "standard life of you will improve
(recoded)"
```

1.2.4 Labeling Values

To analyze a categorical variable, first you need to code the text data into numeric values. Next, you need to make sense of these values by labeling them. The purpose of labeling values is to define the numeric values of a categorical variable. This will make your analysis easier and interpretation clearer. For example, say a variable `gender` is coded as 1 and 2 in your data. Without proper labeling of these two values, people will feel confused when reading your data and output. Once you label values of a categorical variable, the label will appear in your output when you conduct an analysis.

In Stata, labeling values involves two steps:

- First, you use the `label define` command to define a label, which connects numeric values to meaningful descriptions of categories.
- Second, you use the `label values` command to assign the label to that variable. For example, suppose we want to label values of a variable `gender` with 1 for female and 0 for male. The following are the commands:

  ```
  label define gendlabel 1 “female” 0 “male”
  label values gender gendlabel
  ```

The first command tells Stata to define a label named `gendlabel` with 1 for female and 0 for male. The second command assigns the label `gendlabel` to the variable `gender`.

Let us take a look at an example. We would like to add value labels to the newly created variable `gliferev`. The scale is from 1 to 5, with 1 = strongly disagree, 2 = disagree, 3 = neither, 4 = agree, and 5 = strongly agree.

First, we use the command `label define` to define a label named `glifelabel`. The following is the command:

```
label define glifelabel 1 “strongly disagree” 2 “disagree”
3 “neither” 4 “agree” 5 “strongly agree”
```

Then, we attach the value label `glifelabel` to the variable `gliferev`. The command is as follows:

```
label values gliferev glifelabel
```

The following is the Stata output. Using the command `codebook gliferev`, we can see the labels are correctly listed.

```
. label define glifelabel 1 "strongly disagree" 2 "disagree" 3 "neither" 4 "agr
> ee" 5 "strongly agree"

. label values gliferev glifelabel

. codebook gliferev

-------------------------------------------------------------------------------
gliferev                            standard life of you will improve (recoded)
-------------------------------------------------------------------------------

                  type:  numeric (byte)
                 label:  glifelabel

                 range:  [1,5]                        units:  1
         unique values:  5                        missing .:  0/1974
       unique mv codes:  2                       missing .*:  642/1974

            tabulation:  Freq.   Numeric  Label
                            71         1  strongly disagree
                           306         2  disagree
                           224         3  neither
                           549         4  agree
                           182         5  strongly agree
                             6        .c
                           636        .i
```

1.2.5 The **egen** Command

The `egen` command is an extension of the generate command. It has functions that are unavailable in the `generate` command or more efficient in tasks done by the `generate` command. For example, you can use this command to create a mean, standard deviation, minimum, maximum, median, or mode of a variable. You can also create a new variable, which is a summation of several items in a survey. For example, let us enter the following command:

```
egen var4 = rowtotal(var1 var2 var3)
```

It creates a variable with a total row score of `var1`, `var2`, and `var3`.

Another example is to create a variable with a row mean score of a set of variables:

```
egen mean = rowmean(var1 var2 var3)
```

If you use the generate command, you need to type and enter an equation like this:

```
generate mean = (var1 + var2 + var3)/3
```

The `egen` command computes more accurate results of row means than the `generate` command when there are missing values among the original variable. For example, if some observations are missing in one of the above three variables, the `egen` command computes the mean based on the values of two existing variables. However, the `generate` command creates a missing value if any of the three variables are missing.

1.2.6 How to Deal With Missing Values When Recoding Variables

Most missing values are coded as a period (.) in a dataset. In Stata, all missing values take on the largest possible values, which are greater than any nonmissing values. In other words, nonmissing values are less than any missing values. So an expression `var1 < .` means nonmissing values in `var1` since nonmissing values in `var1` are less than the system missing values.

For example, using the ELS:2002 data, we want to generate a binary variable `proficiency` and code it as 1 when students' math scores (`bytxmirr`) are larger than 38.1, and 0 otherwise.

```
. generate proficiency = 0

. replace proficiency = 1 if bytxmirr > = 38.1
(8280 real changes made)
```

Using this command, all the missing values (if there are any) will be coded to be 1 since they are larger than any nonmissing values. To exclude missing values, we need to add another "if" condition, `if bytxmirr < .`. The following is the Stata command.

```
. drop proficiency

. generate proficiency = 0

. replace proficiency = 1 if bytxmirr > = 38.1 & bytxmirr <.
(8004 real changes made)
```

To rerun the commands, we need to drop the originally created variable `proficiency`. With the new command, the missing values have been coded correctly.

In addition to being coded as `.`, different missing codes are allowed in Stata. For example, `.a`, `.b`, `.c`, ... `.z`. If these missing codes appear in your data, check the meaning of them. Identify reasons for these missing codes, and decide whether you want to code the other types of missing codes to the default missing (`.`) for your analysis.

If there are different types of missing codes in your data, and you want to exclude missing values in some of the variables, use the following expression:

```
!missing(var)
```

1.2.7 Other Useful Data Management Commands

The following commands will be briefly introduced, but the examples using real data will be omitted here due to space limitations:

1. Combining data
   - The `append` command can be used to add cases to the existing variables. When we have two datasets containing the same variables with the same variable types, this command can be applied to combine different cases into one dataset.
   - The `merge` command can be used to add variables from two datasets that have a common unique identification variable. Please note that the command ID variable should be sorted before using the `merge` command.

2. Reshaping data

The `reshape` command is useful when we reorganize data into different forms. The `reshape long` command reorganizes the dataset in the long form, whereas the `reshape wide` command transforms the dataset in the wide form. For example, in longitudinal data analysis, a person-level dataset is in the wide form, a multivariate layout with one record per individual; on the other hand, a person-period dataset is in the long form with multiple records for each individual, representing each time-point for data collection.

3. Converting variable types

A variable can be coded in either a numeric or a string format. A numeric variable deals with numbers, whereas a string variable contains text data. The `encode` command can be used to convert a string variable into a numeric variable. For example, if a string variable `group` is coded as "small," "medium," and "large," we can convert it to a numeric variable `size` with the following command: `encode group, generate(size)`.

4. Using the `display` command as a calculator

The `display` command can be used a calculator. After typing the command and the math expression, you can see the calculated results in the output.

```
. display -2*[-734.755 -(-701.147)]
67.216
```

1.3 Graphs

One advantage of Stata is that it can draw various publication-quality graphs. In this section, some basic functions in Stata Graphics will be introduced. I will focus on histograms, bar charts, box plots, and scatter plots. For each type of graph, I will start from the Stata command to the pull-down menu. Commands for Stata graphics can be either simple or complex. They are simple because the basic types of graphs can be easily created using the `graph` command, which is a combination of graphs and the type of graph you would like to draw. For example, the `graph bar` is the command for bar graphs, `graph pie` is for pie charts, and `graph twoway scatter` is for scatter plots. The one-word `histogram` command, as its name suggests, handles histograms.

For each type of graph, Stata offers rich options, which may seem complicated. You may make graphs either by entering the command or via the pull-down menus of the GUI. The GUI is helpful when you draw a complex graph that normally needs a long syntax. When you use the GUI to draw graphs, it automatically generates Stata commands that are shown in the output in the **Results** window. You may save the commands into a do-file so that the graphs can be easily reproduced or edited. When you make any changes or modifications to the existing graphs, Stata displays corresponding commands.

In many situations, you will need to modify your graphs. Although you can do it by entering commands or using the menus, thanks to the **Graph Editor**, you can directly edit your graphs based on your needs. The **Graph Editor** is a function that was added to Stata in version 10. Instead of typing long, complete commands on the command line, you may create a simple graph first, and then explore various options you need by using the **Graph Editor**.

In the following sections, I will show you how to create basic graphs, such as histograms, bar charts, box plots, and scatter plots. At the end, I will introduce the Graph Editor for when you need to modify graphs.

1.3.1 Histograms

The histogram is one of the most frequently used graphs, and it is used to present in a visual manner a frequency distribution of data. In a histogram, scores appear on a horizontal scale and frequency counts are displayed on a vertical scale. Histograms are normally used for continuous variables. They can also be used for ordinal variables if their underlying traits are continuous.

The Stata command for histograms is `histogram`. For example, to see the distribution of the variable `price` using the auto.dta data file, simply enter the `histogram price` command. The histogram is shown in Figure 1.8.

```
. histogram price
(bin=8, start=3291, width=1576.875)
```

In the histogram in Figure 1.8, the distribution is shown as density, which is the default. If you would like it to be shown as frequencies, fractions, or percentages, you would add the options of frequency, fraction, and percent. For another example, if you wanted to draw a histogram of students' math scores (Figure 1.9) using the ELS:2002 data, you would enter the following command with the option `frequency`:

```
. histogram BYTXMIRR_nomiss, frequency
(bin=42, start=12.523, width=1.3618096)
```

1.3.2 Bar Charts

There are two different types of bar charts in Stata. The first type is similar to histograms since bars in both types of graphs display frequency counts. However, in a

Figure 1.8 Histogram of Price

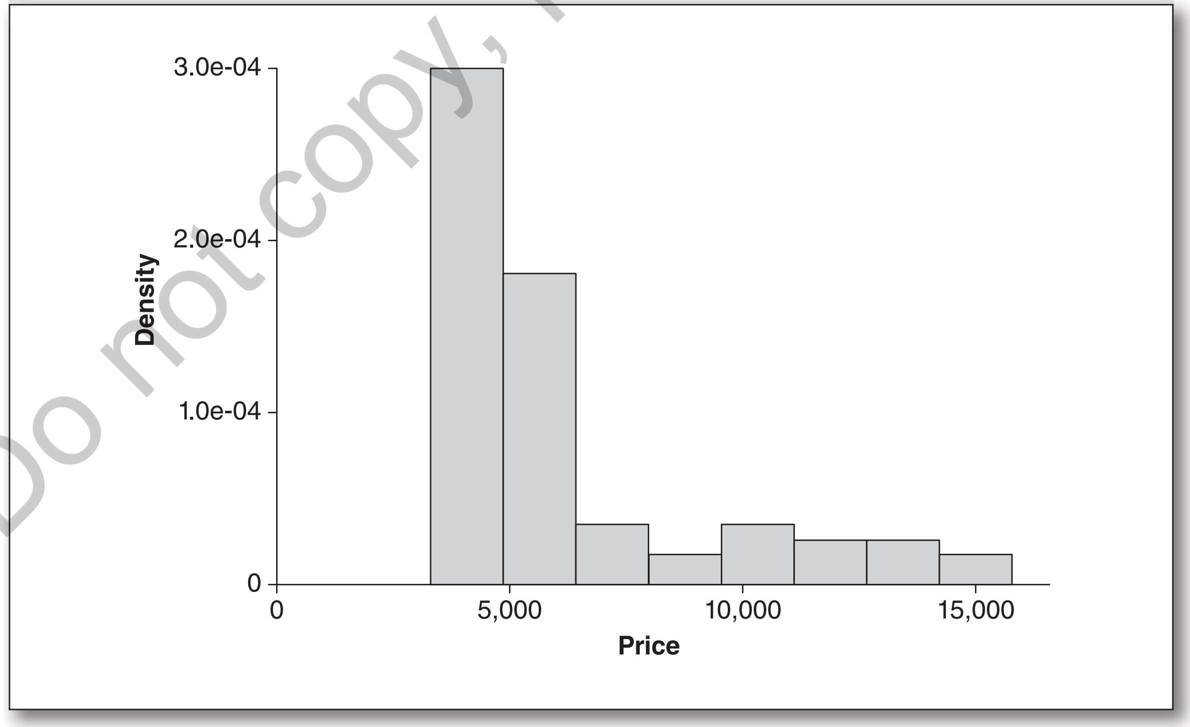
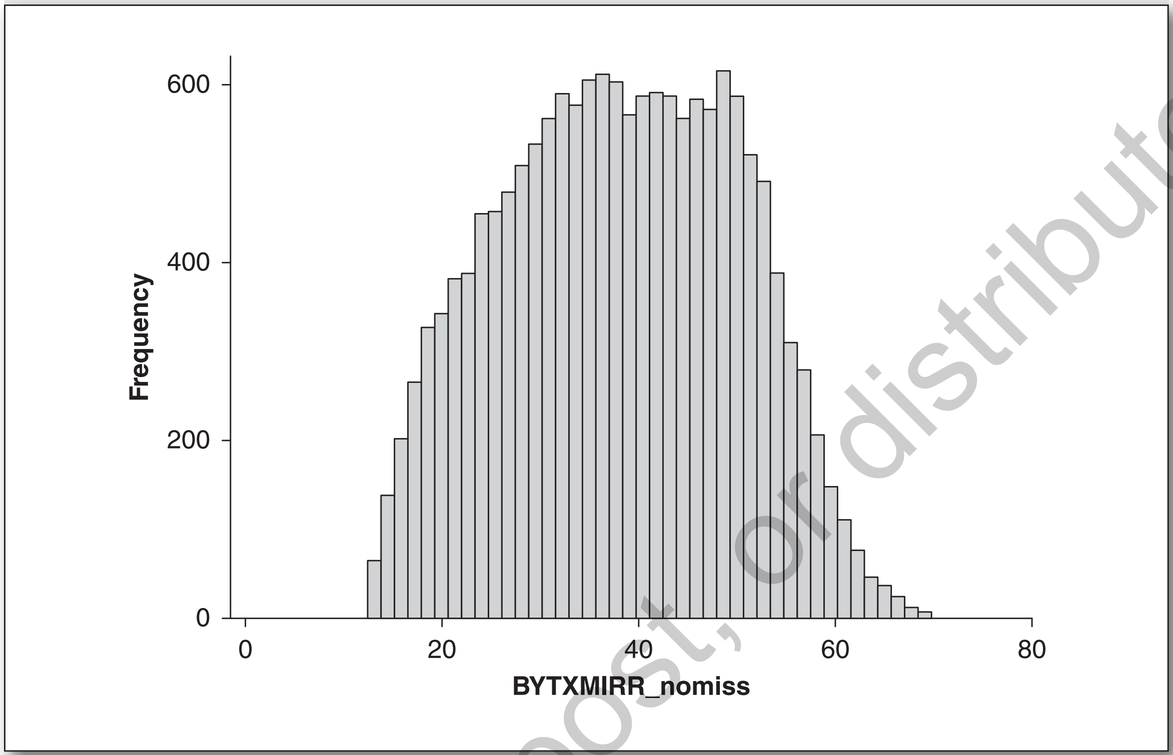


Figure 1.9 Histogram of Math Achievement



histogram, each bar is separated, whereas there is no space between bars in a bar chart. Bar charts are normally used to display frequencies for categorical variables. The command for this type of bar chart is the same as that for histograms. For example, to draw a bar chart for a categorical variable BYINCOME, type the following command:

```
histogram BYINCOME, frequency
```

You will then see the following output and the graph shown in Figure 1.10:

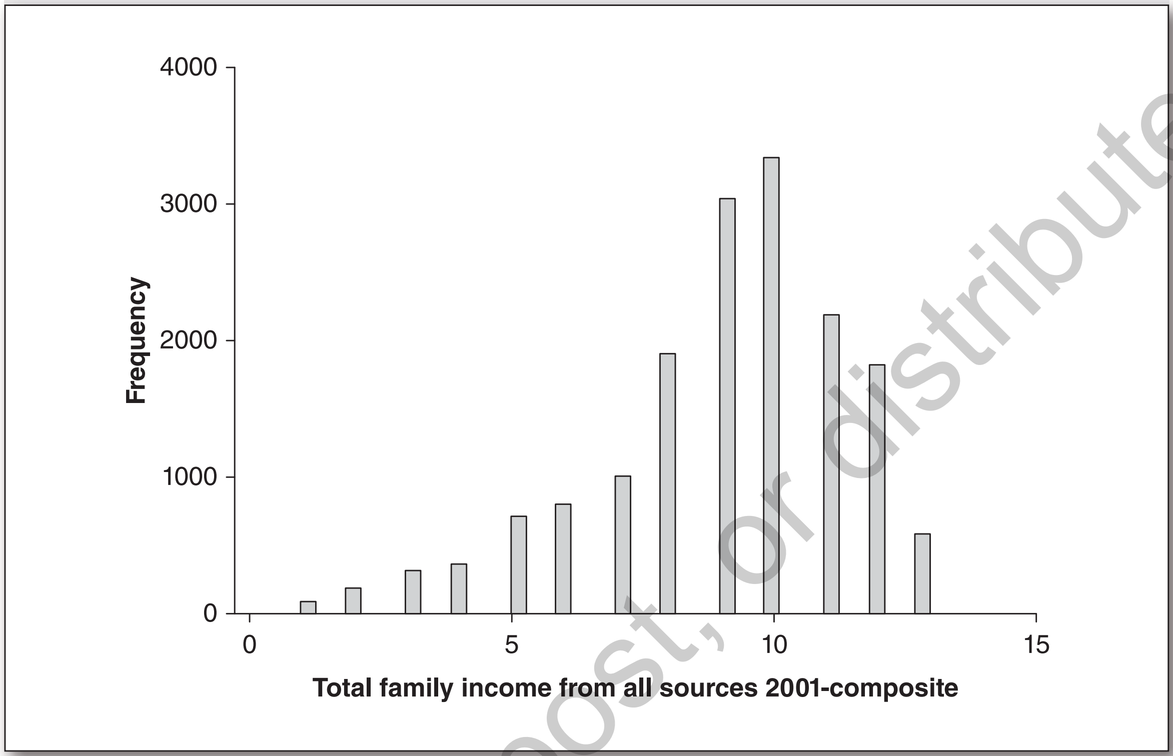
```
. histogram BYINCOME, frequency
(bin=42, start=1, width=.28571429)
```

If the variable is not coded as categorical in the dataset, you need to add the option discrete to draw a bar chart. The same bar graph can be drawn if you type the following command:

```
histogram BYINCOME, frequency discrete
```

The second type of bar chart is for a continuous variable across categories. The graph bar command does not produce the first type of bar charts we introduced

Figure 1.10 Bar Chart of Family Income



earlier. Instead, it shows a two-way vertical bar chart, with the y axis for the continuous variable and the x axis for the categorical variable. Statistics of the continuous variable, such as mean, median, sum, count, minimum, maximum, or percentiles from 1 to 99 can be shown on the y axis, where mean is the default. A basic bar chart can be obtained if you enter the following command:

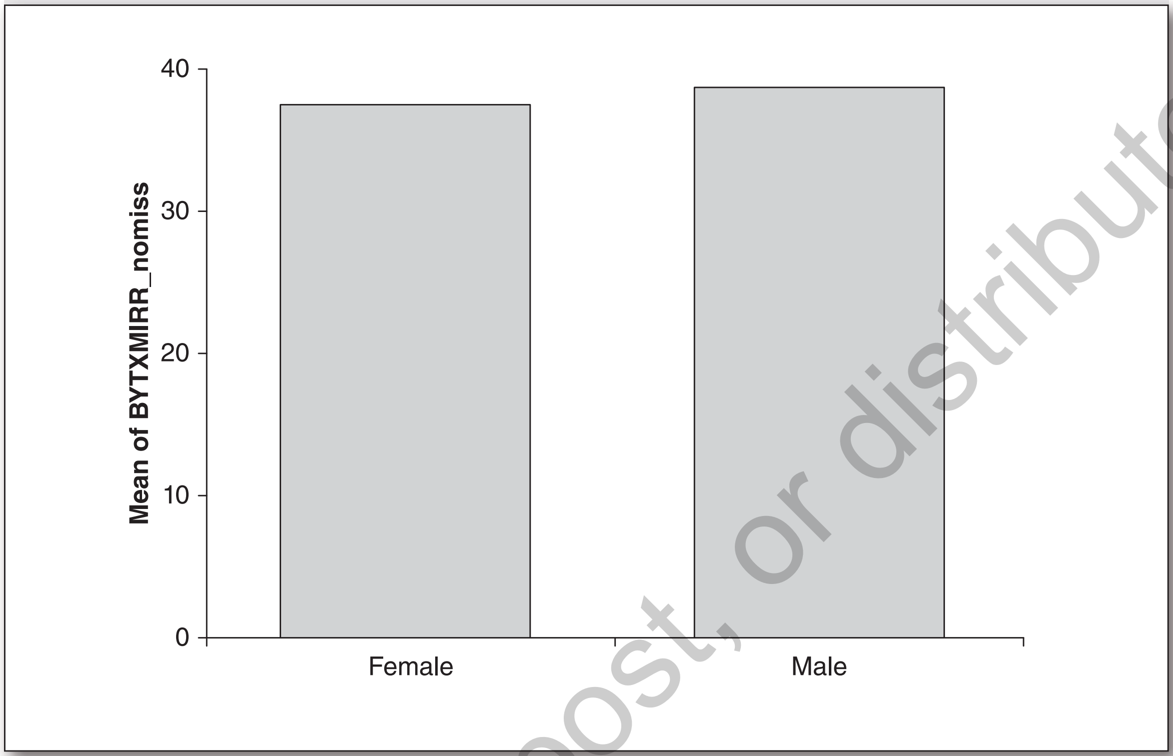
```
graph bar varname
```

This command only shows one bar with the mean of the variable on the y axis. This might not be the graph you want. If you would like to draw bar charts for a continuous variable across a categorical variable, you need to use the right command with the `over` option. For example, the command `graph bar yvar, over(xvar)` tells Stata to draw a bar chart for a continuous variable `yvar` over the categorical variable `xvar`.

The following `graph bar` command draws bar charts (Figure 1.11) for the math achievement score (`BYTXMIRR_nomiss`) over the categorical variable `BYGENDER`:

```
. graph bar BYTXMIRR_nomiss, over(BYGENDER)
```

Figure 1.11 Bar Chart of Math Achievement Over Gender



1.3.3 Box Plots

Box plots are useful for displaying a distribution and identifying outliers for a continuous variable. They display the 25th and 75th percentile, median, whiskers, and outliers. In a box plot, the lower and upper ends of the box indicate the 25th and 75th percentile, respectively. The width of the box indicates the interquartile range. The vertical line in the box is the median, which is the 50th percentile. The vertical lines below and above the box are whiskers, which indicate the spread of your data. Observations beyond the whiskers are shown as dots, which are outliers.

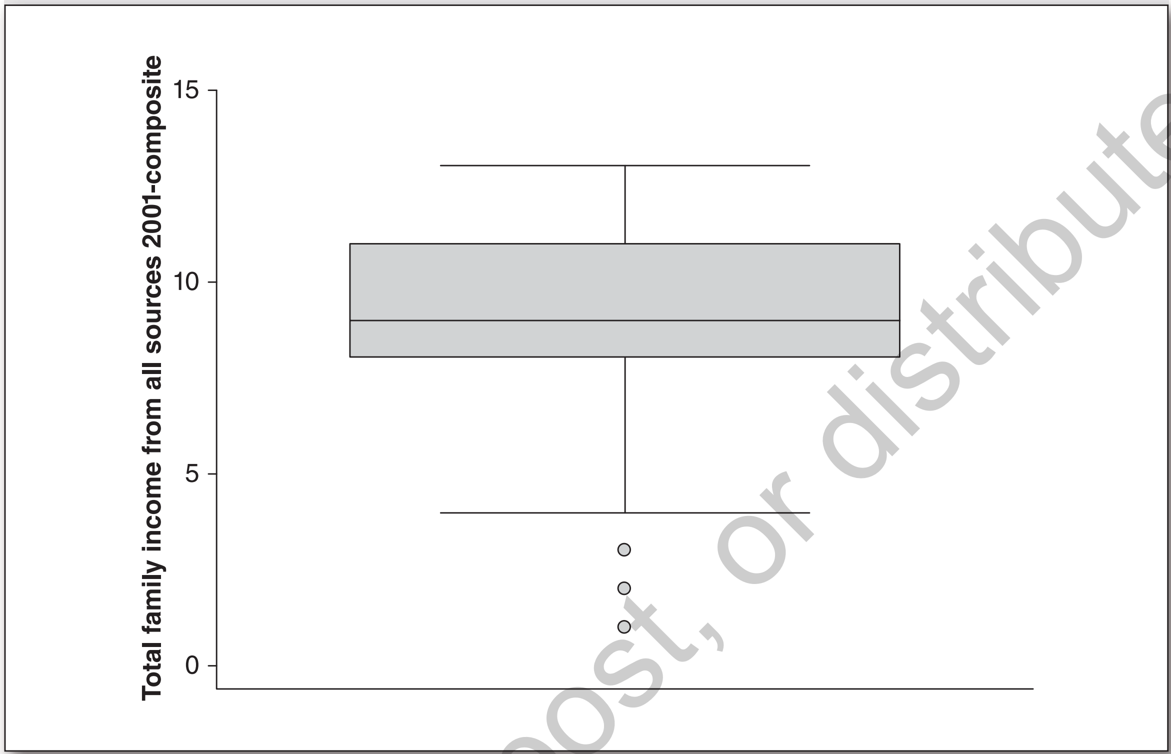
To get a basic box plot, enter the following command:

```
graph box var1
```

This command tells Stata to draw a box plot for the variable `var1`. Let us draw a box plot for the variable `BYINCOME` using the ELS:2002 data (Figure 1.12):

```
. graph box BYINCOME
```

Figure 1.12 Box Plot of Family Income



To display several box plots in one graph, enter the command with several variables:

```
graph box var1 var2
```

For example, the following command draws box plots for the two variables BYINCOME and BYSES2 in one graph (Figure 1.13):

```
. graph box BYINCOME BYSES2
```

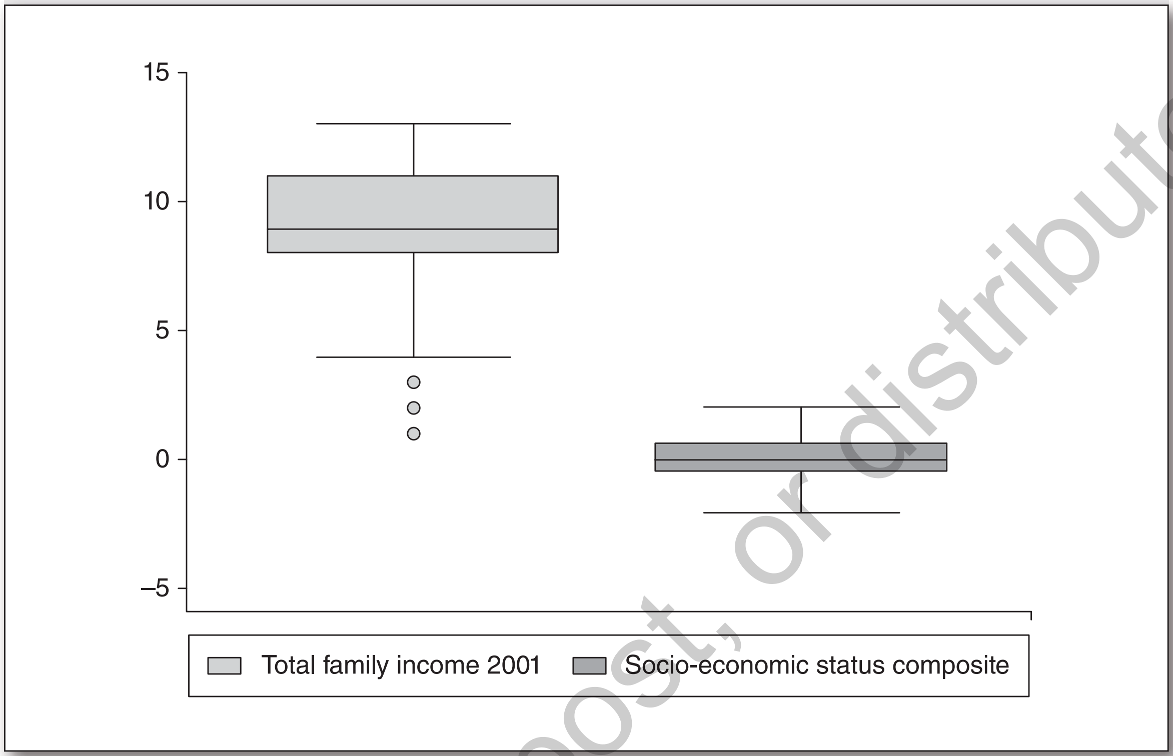
1.3.4 Scatter Plots

Scatter plots are used to show a relationship between two variables. It is a two-dimensional graph, with the x axis displaying values of one variable and the y axis displaying values for the other variable (Figure 1.14).

To see a scatter plot of two variables `var1` and `var2`, type the following command:

```
twoway scatter var2 var1
```

Figure 1.13 Box Plots of Family Income and Socioeconomic Status



This command tells Stata to draw a two-way scatter plot for `var1` and `var2`. For example:

```
. twoway scatter BYSES2 BYTXMIRR_nomiss
```

For illustration purposes, we may focus on a subsample of 200 cases so that we can get a clearer picture. You still use the same `twoway scatter` command with the `in` qualifier:

```
twoway scatter BYSES2 BYTXMIRR_nomiss in 1/200
```

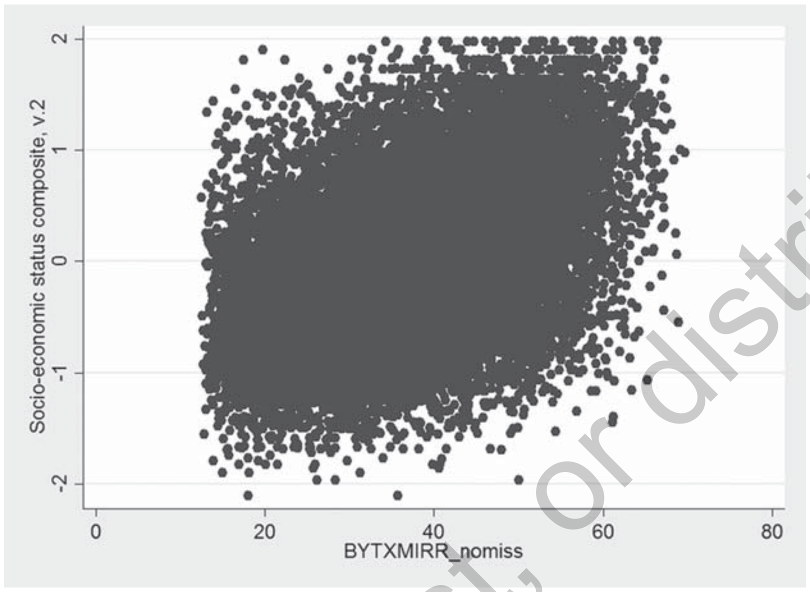
Figure 1.15 shows the scatterplot of math scores and SES for the subsample.

1.3.5 How to Save Graphs

Stata does not automatically save the graphs you create. The graphs are independent from the log output files; therefore, they are not saved into log files. To save your graph, enter the `graph save` command. For example:

```
graph save graphname
```

Figure 1.14 Scatterplot of Math Achievement and Socioeconomic Status



The graph will be saved in the .gph format, which can only be opened by Stata. Using the `graph export` command, Stata can also save graphs into different formats that can be easily opened by other programs. These formats include Encapsulated PostScript (.eps), PDF (.pdf), Portable Network Graphics (.png), PostScript (.ps), and Windows Metafile (.wmf). For example, to save the graph in the .wmf format, enter the following command:

```
graph export graphname.wmf
```

The created graph can be copied and pasted into a Word document, but this method would be tedious if you needed to save a lot of graphs all at one time.

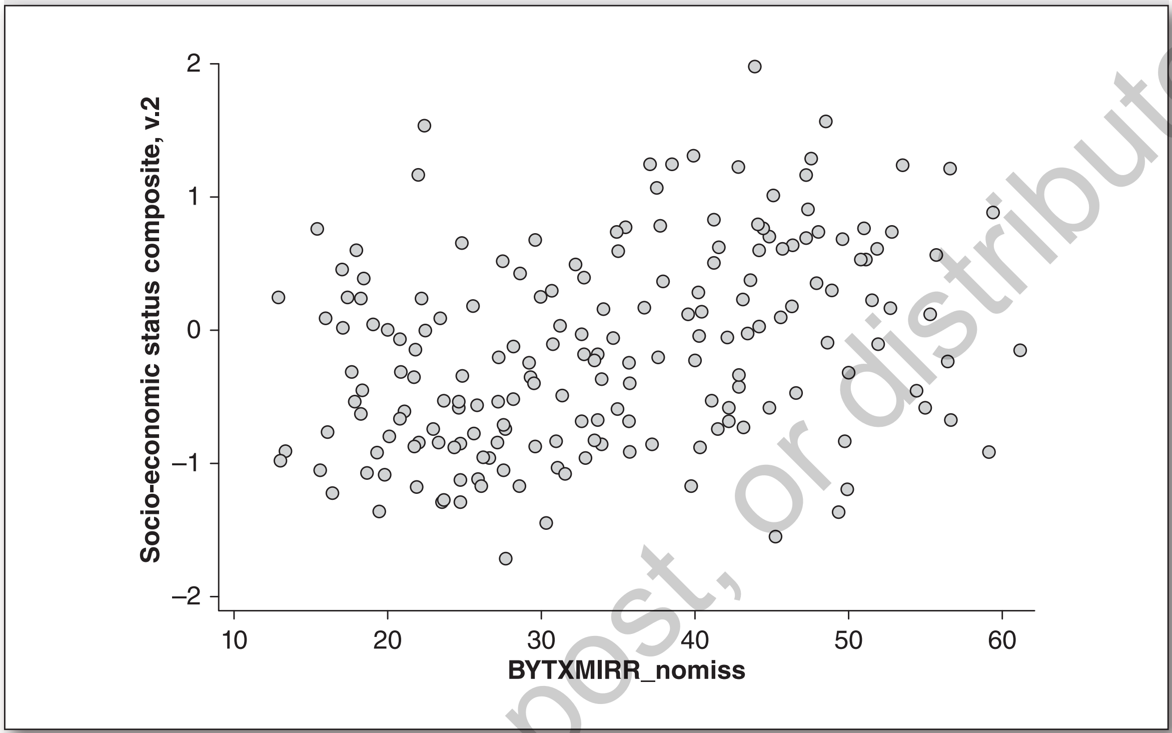
To open an existing graph for editing, enter the `graph use` command:

```
graph use filename
```

1.3.6 Stata Graph Editor

The Stata Graph Editor is a good helper if you need to make edits to your graph. Please keep in mind that it is an editor, so it cannot create a new graph. You first need to have a graph at hand to make edits.

Figure 1.15 Scatterplot of Math Achievement and Socioeconomic Status: Subsample of 200

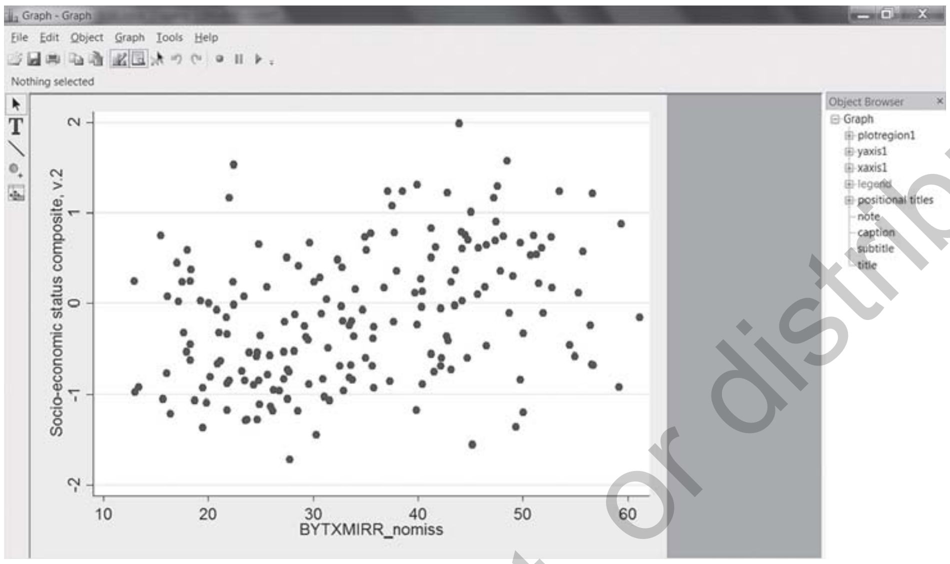


What can the Graph Editor do? It is powerful and convenient. By using the Editor, you can select and make edits on objects, as well as add texts for title, subtitle, caption, note, x axis, y axis, and the legend. You can also add lines and arrows, add markers, and edit the graph grid.

There are three ways to start the Graph Editor. First, after the file is open, go to **File**. Then click **Start Graph Editor**. Another option is to click on the Graph Editor icon on the toolbar. The third option is to right click on the graph, and then select **Start Graph Editor**. When it starts, it displays as shown in Figure 1.16.

You can see five tool buttons on the left of the graph and the Object Browser on the right. The five tools in sequence are a pointer tool for selecting objects; three tools for adding texts, lines, and markers; and a tool for editing the grid. The Object Browser shows you all the objects or contents of a graph, which include the plot region, y axis, x axis, legend, and positional titles. If you want to make edits on any one of the objects, you just need to click on it, and it will be selected. We will see that it is highlighted in the Object Browser and in the graph if it exists. It also opens the Contextual Toolbar, which shows the properties of the object.

Figure 1.16 Graph Editor



In Figure 1.17, I will show you how to change the title of the x axis.

First, let us click on the title under the object `xaxis1`. Once it is selected, you will see the title of the x axis, `BYTXMIRR_nomiss`, is also highlighted in Figure 1.17.

Next, double click or right click on the title. You will see a dialogue box for the textbox properties displayed in Figure 1.18. In the text box, type the new title for the x axis: “Math Scores of High School Students”. Click **Apply**, and you will see the new title appear in the graph.

As a second example, I will add a title for the graph. At the bottom of the **Object Browser**, next to **subtitle**, select **title**. Since there is no title in the graph, double click **title**. It will open up a dialog box for **textbox properties** (Figure 1.19). **Enter** the title “Relationship between Math Achievement and SES” in the text box, and then click on **Apply**.

You can see that the new title appears in Figure 1.20. In addition to entering the title name in the textbox properties, a shortcut is to enter it in the Contextual Toolbar, which is located just above the graph.

For more information about data management and graphics, see Acock (2014), Baum (2006), Cameron and Trivedi (2010), Hamilton (2012), Juul (2014), Kohler and Kreuter (2012), Longest (2015), and Mitchell (2010, 2012a). Refer to Baum (2009) and Long (2009) for advanced topics, such as programming in data management.

Figure 1.17 Editing the X-Axis Title With the Graph Editor

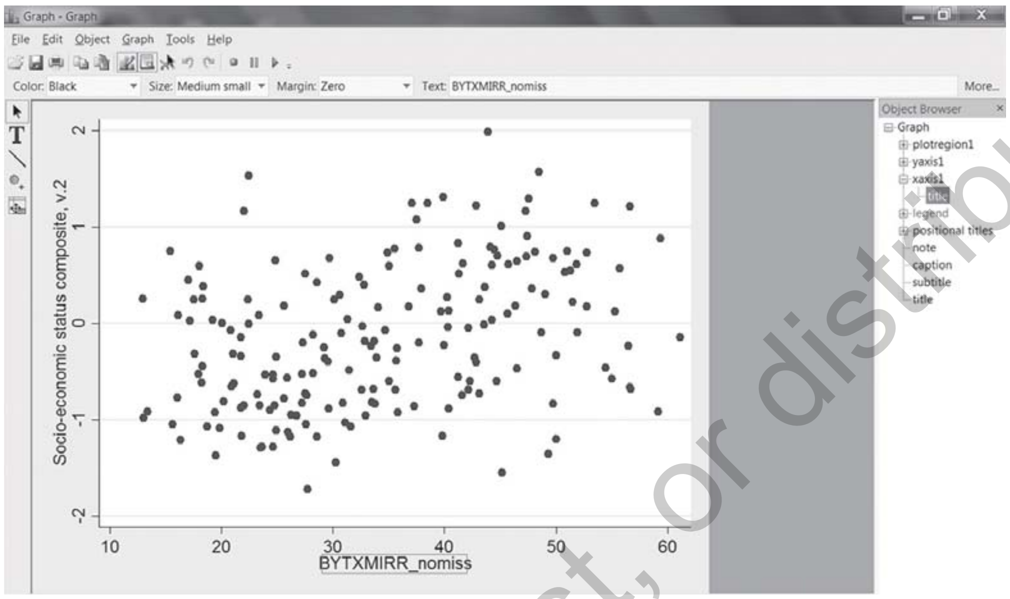
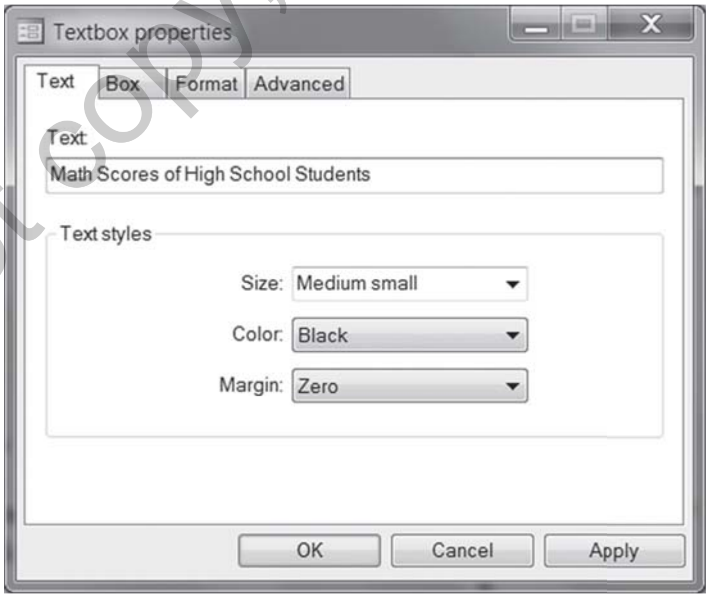


Figure 1.18 Dialogue Box for the Textbox Properties



Text: Math Scores of High School Students

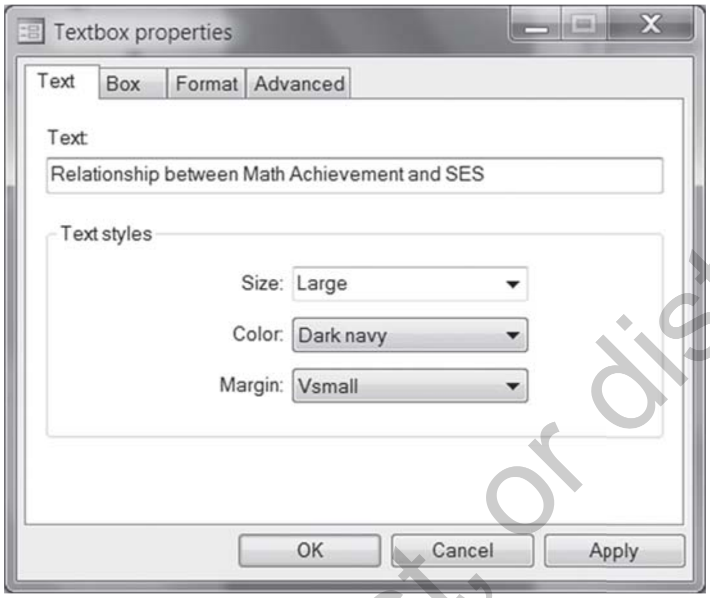
Text styles

Size: Medium small

Color: Black

Margin: Zero

Figure 1.19 Adding a Graph Title Using the Textbox Properties



Text: Relationship between Math Achievement and SES

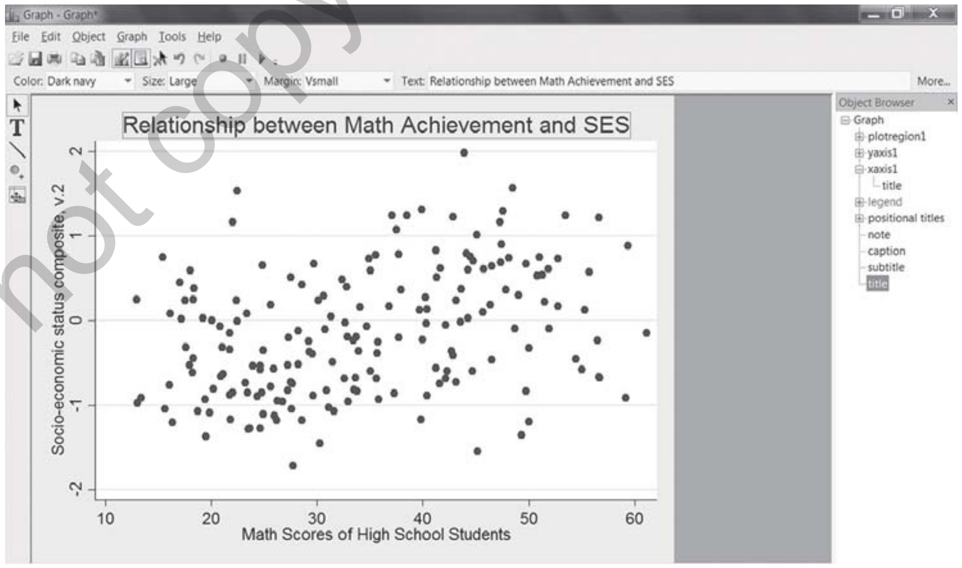
Text styles

Size: Large

Color: Dark navy

Margin: Vsmall

Figure 1.20 A New Title in the Graph


Graph - Graph*
File Edit Object Graph Tools Help
Color: Dark navy
Size: Large
Margin: Vsmall
Text: Relationship between Math Achievement and SES
More...
Relationship between Math Achievement and SES
Socio-economic status composite, v.2
Math Scores of High School Students
Object Browser
Graph
plotregion1
yaxis1
xaxis1
title
legend
positional titles
note
caption
subtitle
title

1.4 Summary of Stata Commands in This Chapter

```
*1.2 Data Management

*1.2.1 Creating a new variable
use chap1-gss2012, clear
generate help = helppoor+helpsick

*Recoding the values of an existing variable
generate male = 1 if sex ==1
replace male = 0 if sex ==2

*Creating a dummy variable
generate education = 0
replace education =1 if educ > 13.53

*Another way to create a dummy variable

*drop education first since it has already existed
drop education
generate education = educ > 13.53

*A more complex example of creating a new categorical variable
generate SES = realinc
replace SES = 1 if realinc >= 0 & realinc <= 10412
replace SES = 2 if realinc >= 10412.5 & realinc <= 22050
replace SES = 3 if realinc >= 22051 & realinc <= 40425
replace SES = 4 if realinc >= 40426

*1.2.2 Recoding a variable

*Recoding values of an existing variable: Overwriting the original variable
recode educ (min/13.53 = 0) (13.54/max = 1)

*Recoding values of an existing variable: Generating a new variable
drop education
recode educ (min/13.53 = 0) (13.54/max = 1), generate(education)

*Another example
recode income(min/9 = 1)(10/11 = 2)(12/max = 3), generate(inclevel)

*Reversing the order of a variable
recode goodlife (1=5)(2=4)(3=3)(4=2)(5=1), generate(gliferev)

*1.2.3 Labeling a variable

*label variable varname "label text"
*label variable efficacy "mathematics self-efficacy"
*label variable watchtv "number of hours in watching TV"
*label variable gliferev "standard life of you will improve (recoded)"

*1.2.4 Labeling values

*Example 1

*label define gendlabel 1 "female" 0 "male"
*label values gender gendlabel
```

```
*label values gender gendlabel

*Example 2
label define glifelabel 1 "strongly disagree" 2 "disagree" ///
3 "neither" 4 "agree" 5 "strongly agree"
label values gliferev glifelabel
codebook gliferev

*1.2.5 egen

*egen var4 = rowtotal(var1 var2 var3)
*egen mean = rowmean(var1 var2 var3)
*generate mean = (var1 + var2 + var3)/3

*1.2.6 Dealing with missing values
use chap1-gss2012, clear

*Example 1
generate proficiency = 0
replace proficiency = 1 if bytxmirr > 38.1

*Example 2
drop proficiency
generate proficiency = 0
replace proficiency = 1 if bytxmirr > = 38.1 & bytxmirr <.

*1.2.7 Other useful commands: display
display -2*[-734.755 -(-701.147)]
```

***1.3 Graphs**

```
*1.3.1 Histograms
sysuse auto, clear
histogram price
use chap1-els2002, clear
histogram BYTXMIRR_nomiss, frequency
histogram BYINCOME, frequency
histogram BYINCOME, frequency discrete

*1.3.2 Bar charts
graph bar BYTXMIRR_nomiss, over(BYGENDER)

*1.3.3 Box plots
graph box BYINCOME
graph box BYINCOME BYSES2

*1.3.4 Scatter plots
twoway scatter BYSES2 BYTXMIRR_nomiss
twoway scatter BYSES2 BYTXMIRR_nomiss in 1/200

*1.3.5 How to save graphs

*graph save graphname
*graph export graphname.wmf
*graph use filename
```

1.5 EXERCISES

Use the GSS 2012 data for the following problems.

1. Find the variable `happy`, and recode it to a new variable `happyrev`. Recode the values of 1, 2, and 3 in `happy` into the values of 3, 2, and 1, respectively, for the new variable.
2. Label the new variable `happyrev`, `happiness`, and then label its values 1 "not too happy", 2 "pretty happy", and 3 "very happy".
3. Produce a histogram for `educ`.
4. Draw a scatter plot to explore the relationship between `satfam7` and `satfin`.
5. Save the graph with a name.