A Gentle Introduction to Stata

5th Edition

ALAN C. ACOCK
Oregon State University

A Stata Press Publication
StataCorp LP
College Station, Texas

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845
Typeset in $\LaTeX\,2_\varepsilon$
Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Print ISBN-10: 1-59718-185-4
Print ISBN-13: 978-1-59718-185-3

Library of Congress Control Number: 2016935690

Contents

4 Working with commands, do-files, and results

4.1 Introduction

Throughout this book, I am illustrating how to use the menus and dialog boxes, but underneath the menus and dialogs is a set of commands. Learning to work with the commands lets you get the most out of Stata, and this is true of any statistical software. Even the official Stata documentation is organized by command name, as illustrated by the *Stata Base Reference Manual*, which has more than 2,800 pages explaining Stata commands.

With the logical organization of the menu system, you may wonder why you need to even think about the underlying commands. There are several reasons: Entering commands can be quicker than going through the menus. More importantly, you can put the commands into files that are called *do-files*, allowing for easy repetition of a series of commands. These do-files allow you to replicate your work, something you should always ensure you can do. When you collaborate with co-workers, they can use your do-file as a way to follow exactly what you did. It is hard enough to remember all the commands you create in a session, and if there is a delay between work sessions, it is impossible to remember all those commands. Even when you are using the menu system, it is useful to save the commands generated from the menus into a do-file, and Stata has a way to facilitate saving these commands; we will soon learn how to do this.

What is a command? What is a do-file?

A command instructs Stata to do something, such as construct a graph, a frequency tabulation, or a table of correlations. A do-file is a collection of commands. "Do-file" is a good name because it is so descriptive: the series of commands in the file tells Stata what to "do". A simple do-file might open a dataset, summarize the variables, create a codebook, and then do a frequency tabulation of the categorical variables. A do-file can include all the commands you use to label variables and values; it can recode variables, average variables, and define how you treat missing values. Such a do-file might be only a few lines long, but complicated do-files can be thousands of lines long.

4.2 How Stata commands are constructed

Stata has many commands. Here are some of the commands covered in this book:

`list`	List values of variables
`summarize`	Summary statistics
`describe`	Describe data in memory or in file
`codebook`	Describe data contents
`tabulate`	Tables of frequencies
`generate`	Create or change contents of variable
`egen`	Extensions to generate
`correlate`	Correlations (covariances) of variables or coefficients
`ttest`	Mean-comparison tests
`anova`	Analysis of variance and covariance
`regress`	Linear regression
`logit`	Logistic regression, reporting coefficients
`factor`	Factor analysis
`alpha`	Compute interitem correlations (covariances) and Cronbach's alpha
`graph`	The graph command

Stata has a remarkably simple command structure. Stata commands are all lowercase. Virtually all Stata commands take the following form: *command varlist* `if/in`, *options*. The *command* is the name of the command, such as `summarize`, `generate`, or `tabulate`. The *varlist* is the list of variables used in the command. For many commands, listing no variables means that the command will be run on all variables. If we said `summarize`, Stata would summarize all the variables in the dataset. If we said `summarize age education`, Stata would summarize just the `age` and `education` variables. The variable list could include one variable or many variables. After the variable list come the `if` and `in` qualifiers regarding what will be included in the particular analysis. Suppose that we have a variable called `male`. A code of `1` means that the

participant is a male, and a code of `0` means that the participant is female. We want to restrict the analysis to males. To restrict the analysis, we would say `if male == 1`. Here we use two equal signs, which is the Stata equivalent to the verb "is". So the command means "if male is coded with a value of 1". Why the two equal signs? The statement `male` = 1 literally means that the variable called `male` is a constant value of 1, but males are coded as `1` and females are coded as `0` on this variable. Sometimes we want to run a command on a subset of observations, and so we use the `in` qualifier. For example, we might have the command `summarize age education in 1/200`, which would summarize `age` and `education` in the first 200 observations.

Each command has a set of *options* that control what is done and how the results are presented. The options vary from command to command. One option for the `summarize` command is to obtain detailed results, summarizing the variables in more ways. If we wanted to do a detailed summary of scores on age and years of education for adult males, the command would be

```
. summarize age education if male == 1 & age > 17, detail
```

The command structure is fairly simple, which is helpful for us because it is absolutely rigid. This example used the ampersand (`&`), not the word "and". If we had entered the word "and", we would have received an error message. Here are more examples with `if` statements:

```
. summarize age education if sex == 0
. summarize age education if sex == 1 & age > 64
. summarize age sex if sex == 0 & age > 64 & education == 12
```

When you have missing values stored as `.` or `.a`, `.b`, etc., you need to be careful about using the `if` qualifier. Stata stores missing values internally as huge numbers that are bigger than any value in your dataset. If you had missing data coded as `.` or `.a` and entered the command `summarize age if age > 64`, you would include people who had missing values. The correct format would be

```
. summarize age if age > 64 & age < .
```

The `< .` qualifier at the end of the command is strange to read (less than dot) but necessary. Table 4.1 shows the relational operators available in Stata.

Table 4.1. Relational operators used by Stata

Symbol	Meaning
`==`	is or is equal to
`!=` or `~=`	is not or is not equal to
`>`	is greater than
`>=`	is greater than or equal to
`<`	is less than
`<=`	is less than or equal to

The `in` qualifier specifies that you will perform the analysis on a subset of cases based on their order in the dataset. If we had 10,000 participants in a national survey and we wanted to list the values in the dataset for **`age`**, **`education`**, and **`sex`**, this list would go on for screen after screen after screen, which would be a waste of time. We might want to list the data on **`age`**, **`education`**, and **`sex`** for just the first 20 observations by using **`in 1/20`**. The `1` is where Stata should start (called the first case), the "/" is read as "to", and the `20` is the last case listed. Thus **`in 1/20`** tells Stata to do the command for the cases numbered from 1 to 20, or for the first 20 cases. The full command is

```
. list age education sex in 1/20
```

Listing just a few cases is usually all you need to check for logical errors. Most Stata dialog boxes include an if/in tab for restricting data.

If your dataset contains few variables, it may be easier to just leave the Data Editor (Browse) open with the data while you are typing your commands instead of running a listing. Any changes made by your commands will appear immediately in the Data Editor (Browse). You can open the Data Editor (Browse) by typing the **`browse`** command or the **`browse, nolabel`** command in the Command window or by clicking on the toolbar icon that looks like a spreadsheet with a magnifying glass (see figure 2.3).

The final feature in a Stata command is a list of options. You must enter a comma before you enter the options. As you learn more about Stata, the options become increasingly important. If you do not list any options, Stata gives you what it considers to be basic results. Often these basic results are all you will want. The options let you ask for special results or formatting. For example, in a graph, you might want to add a title. In frequency tabulation, you might want to include cases that have missing values. One of the best reasons for using dialog boxes is that you can discover options that can help you tailor your results to your personal taste. Dialog boxes either include an Options tab or have the options as boxes that you can check on the Main tab. The most common mistake a beginner makes when typing commands directly in the Command window is leaving the comma out before specifying the options.

Here are a few Stata commands and the results they produce. You can enter these commands in the Command window to follow along.

```
. use http://www.stata-press.com/data/agis5/firstsurvey_chapter4
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          id |        20        10.5     5.91608          1         20
      gender |        20         1.5    .5129892          1          2
   education |        20       14.45    2.946452          8         20
      sch_st |        18    3.444444    1.149026          2          5
     sch_com |        20         3.5    1.395481          1          5
-------------+--------------------------------------------------------
      prison |        17    3.176471    1.550617          1          5
     conserv |        19    2.947368    1.544657          1          5
```

This **`summarize`** command does not include a variable list, so Stata will summarize all variables in the dataset. It has no **`if/in`** restrictions and no options, so Stata summarizes all the variables, giving us the number of observations with no missing values, the mean, the standard deviation, and the minimum and maximum values. The statistics for the **`id`** variable are not useful, but it is easier to get these results for all variables than it is to list all the variables in a variable list, dropping only **`id`**.

We can add the **`detail`** option to our command to give more detailed information. Do this for just one variable.

```
. summarize education, detail

                     Years of education
-------------------------------------------------------------
      Percentiles      Smallest
 1%            8              8
 5%          9.5             11
10%         11.5             12       Obs                  20
25%           12             12       Sum of Wgt.          20

50%         14.5                      Mean              14.45
                        Largest       Std. Dev.      2.946452
75%         16.5             17
90%           18             18       Variance       8.681579
95%           19             18       Skewness      -.1636124
99%           20             20       Kurtosis       2.522208
```

As expected, this method gives us more information. The 50% value is the median, which is 14.5. We also get the values corresponding to other percentiles, the variance, a measure of skewness, and a measure of kurtosis (we will discuss skewness and kurtosis later).

Next we will use the **`list`** command. Here are four commands you can enter, one at a time, to get three different listings:

```
. list gender education prison in 1/5

     +----------------------------------------+
     | gender   educat~n               prison |
     |----------------------------------------|
  1. |  woman         15             too long |
  2. |    man         12    much too lenient  |
  3. |    man         16          about right |
  4. |    man          8                    . |
  5. |  woman         12          about right |
     +----------------------------------------+

. list gender education prison in 1/5, nolabel

     +-----------------------------+
     | gender   educat~n   prison  |
     |-----------------------------|
  1. |      2         15        4  |
  2. |      1         12        1  |
  3. |      1         16        3  |
  4. |      1          8        .  |
  5. |      2         12        3  |
     +-----------------------------+

. numlabel _all, add
. list gender education prison in 1/5

     +-------------------------------------------+
     |   gender   educat~n                prison |
     |-------------------------------------------|
  1. | 2. woman         15           4. too long |
  2. |   1. man         12   1. much too lenient |
  3. |   1. man         16        3. about right |
  4. |   1. man          8                     . |
  5. | 2. woman         12        3. about right |
     +-------------------------------------------+
```

The first command shows the first five cases. Notice that the `education` variable appears at the top of its column as `educat~n`. We can use names with more than eight characters, but some Stata results will show only eight characters. For names with more than eight characters, Stata will keep the first six characters and the last character and insert the tilde (~) between them. Because we assigned value labels to `gender` and `prison`, the value labels are printed in the list. However, notice that the numerical values are omitted.

The second command adds the `nolabel` option, which gives us a listing with the numerical values we used for coding but not the labels. The next command, `numlabel _all, add`, adds a numeric value to each label for all variables (the `_all` tells Stata to apply this command to all variables). If we wanted to remove the values later, we would enter `numlabel _all, remove`. Finally, the last listing gives us both the values and the labels for each variable.

4.3 Creating a do-file

You may be asking yourself how you will ever learn to use all the options and qualifiers. One way is to read the Stata documentation, but it is often easier to use the menus and

record the command in a do-file. Stata has a simple text editor called the Do-file Editor in which you can enter a series of commands. You can run all the commands in this file or just some of them. You can then edit, save, and open the commands in a do-file at a later date. Saving these do-files means not only that you can replicate what you did and make any needed adjustments but also that you will develop templates you can draw on when you want to do a similar analysis. To open the Do-file Editor window, use **Window ▷ Do-file Editor ▷ New Do-file Editor**. You can also open the Do-file Editor by clicking on the toolbar icon that looks like a notepad; see figure 4.1 for the icon in Stata for Windows.

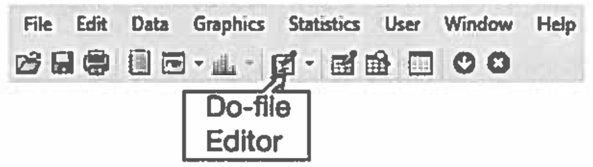


Figure 4.1. The Do-file Editor icon on the Stata menu

Stata allows you to have multiple do-files open at once, but we will use only one do-file in this book. When you gain confidence using Stata, the ability to have several do-files open simultaneously is a useful feature. You might use one do-file primarily for data-management purposes; you might use another for general analysis of the data; you might use a third for analyzing a subset of the data. You might even work on a couple of different projects in one session. Using multiple do-files in a session is sort of like juggling balls: some of us are better at it than others. For now, one Do-file Editor window is all we need. When several do-files are open, clicking on the arrow to the right of the icon shows the name of each open file. A blank do-file appears in figure 4.2.

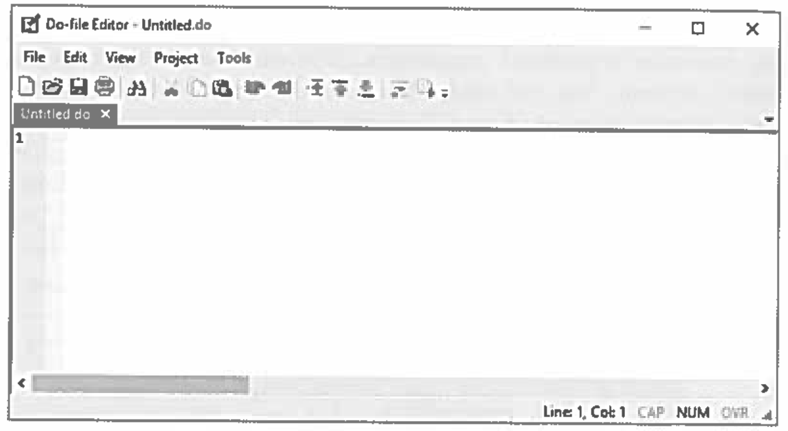

Do-file Editor - Untitled.do
File Edit View Project Tools
Untitled.do
Line: 1, Col: 1 CAP NUM OVR

Figure 4.2. The Do-file Editor of Stata for Windows

When you click on another window, the Do-file Editor can be hidden by it. You can bring the Do-file Editor to the front again by clicking on it in the system toolbar or by using the Alt+Tab key combination to move through the open windows until the Do-file Editor is highlighted. You can avoid having the window disappear by arranging your desktop so that other windows do not overlap with the Do-file Editor window. If you have two monitors, it is convenient to have the main Stata window on one monitor and the Do-file Editor on the other monitor.

The Do-file Editor lacks special features, such as underlining or the special fonts you would get with a word processor such as MS Word. Word processors add a lot of hidden features that would be confusing to Stata. The Do-file Editor is a plain-text editor. It does, however, have some nice features for writing a do-file. As you can see in figure 4.3, the toolbar has many features that you would expect in an editor.

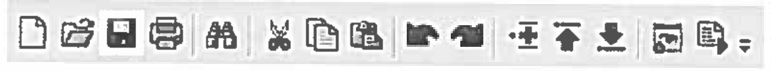
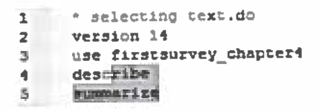


Figure 4.3. The Do-file Editor toolbar of Stata for Windows

The first 10 icons on the left are fairly standard. The first four icons let you open a new do-file, open an existing do-file, save your file, and print the file. The fifth icon allows you to perform a search, although most people prefer to use Ctrl+f to perform a search. The next three icons will cut, copy, and paste, although most people prefer to use Ctrl+x, Ctrl+c, and Ctrl+v (Cmnd+x, Cmnd+c, and Cmnd+v for Macs), respectively. The curved arrows provide undo and redo functions, like a word processor. The next three icons have to do with bookmarks, marking lines in a file you are editing so you can easily jump back to them later. The second from last icon will show content of the file in a Viewer window. Finally, the last icon runs either your entire do-file or the part of your do-file that you have selected. This is the key icon for running your do-file.

You select lines of code in the same way you would in a word processor, by highlighting them. If you want to select a section of several lines, you do not need to highlight all the lines completely; you can highlight some of each line. Here is an example where we want to run two commands, **describe** and **summarize**. In figure 4.4, we have selected only part of the **describe** command but all the **summarize** command.



```
* selecting text.do
version 14
use firstsurvey_chapter4
describe
summarize
```

Figure 4.4. Highlighting in the Do-file Editor

The Do-file Editor automatically numbers your lines. Only the number 1 for the first line appeared in figure 4.2 because the rest of the file is blank. In long do-files, the line numbers help you find your way around. In the lower right, the Do-file Editor also gives the exact location of the cursor, such as "Line: 1, Col: 0" (again, see figure 4.2).

I recommend that you include the name of the file as a comment in the first line of the do-file. Placing an asterisk (*) at the beginning of the line marks the text as a comment that Stata prints but does not interpret. I include this line because my do-file will have the name of the file that created the output so that I know where to find the file if I need to change something later on. Let's type `* my_first.do` at the top of our blank do-file.

Another way of adding comments, especially long comments, to a file is by typing `/*` just before the comment and `*/` right after the comment. Anything between the `/*` and `*/` will be treated as a comment. Figure 4.5 shows what we will use in our example do-file. This type of long comment is helpful for organizing a long do-file into sections with a new comment explaining the purpose of each new section.

We will now add our first command, **version 14**. This command can be important because it will execute the commands as they were written in Stata 14. Stata is continually being improved, and updates may include changes to the commands that you are using. By including the version number you are using when you create the do-file, you ensure that everything will work as intended even when you are using future versions of Stata. This is called version control. It is how Stata maintains compatibility when new versions add new capabilities.

Notice that the comments appear in the Do-file Editor in a green font. This makes them easy to find in a long do-file. Stata commands appear in a bold blue font.

Next we need to open the dataset. In our do-file, we type

```
. use firstsurvey_chapter4.dta, clear
```

Stata automatically looks in our working directory for this file. If we already have a dataset open, the **clear** option will clear those data from Stata's memory. If we had a dataset open with changes made to the data and we did not include the **clear** option with our **use** command, Stata would issue a warning because it does not want us to accidentally clear an active dataset before we save our changes.

You do not have to store your data and do-files in the working directory. You can create a separate directory for each project, or you can use an external flash drive. To open a dataset that is not saved in the working directory, you must specify the full path in your command. Say that our dataset, **firstsurvey_chapter4.dta**, was saved to an external flash drive in a folder called **first project**. We would type

```
. use "E:\first project\firstsurvey_chapter4.dta", clear
```

The quotation marks are required if your folder name or filename contains embedded spaces like the space in **first project** above.

The folder path will vary for Mac and Unix. For example on a Mac if your file is in the **Documents** folder, you would type

```
. use "Users/username/Documents/firstsurvey_chapter4.dta", clear
```

After using Stata for some time, it is usually best to organize your folders in such a way that each project has a separate folder creating a good work flow. Scott Long has written a wonderful book called *The Workflow of Data Analysis Using Stata* (2009), which is available from Stata Press at http://www.stata-press.com/books/wdaus.html. You should read this book if you plan to be involved in a series of major projects using Stata.

The next two commands we need to type are **describe** and **summarize**, which will describe the dataset and then perform a summary of the variables, giving us the number of observations, mean, standard deviation, minimum value, and maximum value. Unless there are too many variables in your dataset, these are good commands to include at the beginning of your do-file.

If you are wondering how Stata knows when it is done with one command and ready to start another, the answer lies with the Enter key. Every time you press the Enter key, Stata assumes you are either done or starting a new command. Having a hard return between them makes **describe** and **summarize** two separate operations. All programs have some way to designate the end of a command. SAS, for example, ends each procedure with a semicolon. The semicolon in SAS is the same as the Enter key in Stata.

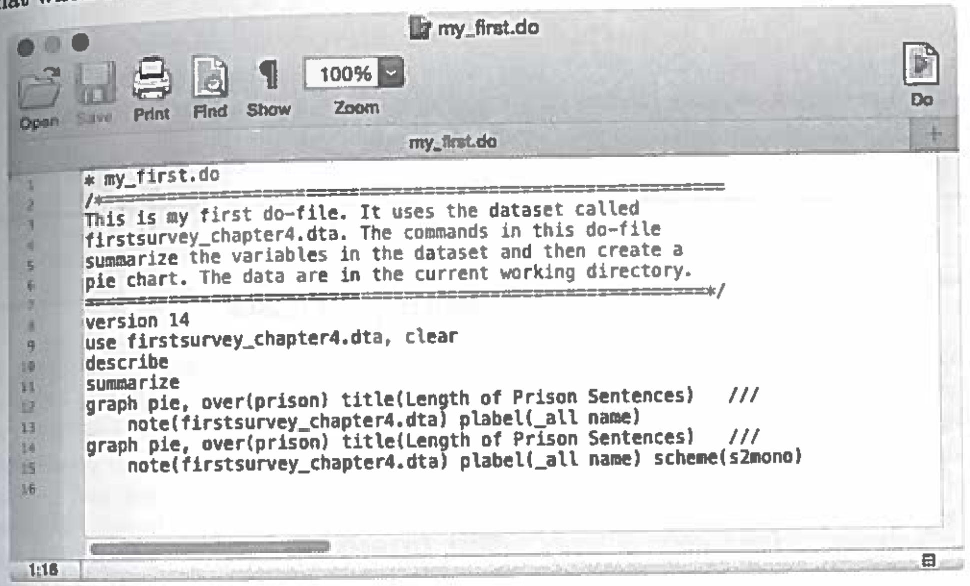
In general, one line equals one command in a do-file. This is a great way to distinguish one command from another as long as each command is short enough to fit on one line. One line equals one command. What happens when you have a long command that extends for more than one line? Stata needs a way to know that when you press the Enter key, you are not really done with the command. The solution is to put /// at the end of a line. This tells Stata that the next line is a continuation of the previous

one. We illustrate the use of /// in the **graph pie** commands shown in figure 4.5. (We will cover these commands in the next chapter. For now, just enter them into the Do-file Editor.)

If you are using dialog boxes to create your commands, you can click on the Copy button in the bottom left corner to copy the command and then press Ctrl+v in the Do-file Editor to paste the command into your do-file.

You must remember to save your do-file. You do this much like you would save a file in any other program; that is, click on File ▷ Save as.... Then you can type the name of the file; in our example, we are using **my_first.do**. You can also browse to find the project folder where you want to save the file, such as **E:\first project**. Until you have more experience, it is probably best to store your do-files and your data in the same folder.

Figure 4.5 shows our do-file, **my_first.do**, as it appears in the Do-file Editor. Notice that when we saved the do-file, the filename now appears at the top of the Editor.



```
* my_first.do
/*=============================================================
This is my first do-file. It uses the dataset called
firstsurvey_chapter4.dta. The commands in this do-file
summarize the variables in the dataset and then create a
pie chart. The data are in the current working directory.
=============================================================*/
version 14
use firstsurvey_chapter4.dta, clear
describe
summarize
graph pie, over(prison) title(Length of Prison Sentences)   ///
    note(firstsurvey_chapter4.dta) plabel(_all name)
graph pie, over(prison) title(Length of Prison Sentences)   ///
    note(firstsurvey_chapter4.dta) plabel(_all name) scheme(s2mono)
```

Figure 4.5. Commands in the Do-file Editor window of Stata for Mac

Are you convinced that you should keep a do-file? I hope so. Imagine that one month from now, you decide to create a pie chart for another variable, **conserv**, from **firstsurvey_chapter4.dta**. You can open the **my_first.do** file and replace **over(prison)** with **over(conserv)**. Click on the icon to run the do-file, and you are done.

As things get more complex, having a set of do-files that perform certain tasks becomes extremely valuable. One thing you might do is create an MS Excel spreadsheet to keep a record of the do-files you create. You might have three columns: the name and path of the do-file, the dataset it uses, and a brief statement of the purpose of each do-file. Eventually, you will have 30 or more do-files, and you can scan this listing to find one that is close to what you plan to do next to serve as a template. If you start doing several major projects, you probably should have a spreadsheet like this for each of the projects.

Stata do-files for this book

The webpage for this book, http://www.stata-press.com/data/agis5/, has do-files for each chapter along with the datasets used. You can copy the do-files and datasets to your computer and reproduce the results in this book. These do-files may also be useful as templates when you do your own work because they can be modified as you need. You will have to change the paths in these files to the paths where you have stored your data unless you have stored the data in your current working directory, `C:\Users\`*username*`\Documents`. You could also change the current working directory. If you stored the data files in `C:\Users\`*username*`\Documents\agis5`, you could enter the command `cd "C:\Users\`*username*`\Documents\agis5"` to make that the current working directory. Then to open a file for Stata to use, you would simply type `use` *filename*, where *filename* is the name of the data file you want to use.

4.4 Copying your results to a word processor

Many people start using Stata to do their homework for a class in statistical methodology. In those cases, the datasets are often fully prepared, and you will not need to keep a record of how the data were created, how you labeled variables, how you recoded some items and dealt with missing values, or how you produced your results. In fact, many of the analyses will have short results. Here it may be simplest and best to save your results by highlighting the text that you want to save in the Results window, right-clicking on the highlighted text, and then selecting Copy from the menu (or using Ctrl+c after you have highlighted the text to copy). You can then paste the text into your favorite word processor. It is a good idea to include the commands that are in the Results window, because these give you a record of what you did. Except when there is no data manipulation, commands like these are no substitute for a do-file that includes everything you did in preparing the data.

When you copy results from Stata's Results window to your word processor, the format may look like a hopeless mess because Stata output is formatted using a fixed-width font; when you paste your results, things will likely not line up properly. The simplest solution to this alignment problem is to change the font and probably the font size, depending on your margins. I usually use the Courier or Courier New font at 9 point. Sometimes the lines may still wraparound, and you may need to widen the

margins. Most Stata results will fit nicely if you have 1-inch margins and a 9- or 10-point Courier font. With most word processors, you change the font of the lines that contain the results by highlighting the lines you want to change and then selecting the Courier font and the 9-point font size.

Saving tabular output

If you are using the Windows version of Stata and have tabular output (say, the results of a `summarize` command), you may want to select just that portion of the text that appears as a table, right-click on it, and select Copy Table, Copy Table as HTML, or Copy as Picture from the menu. You can then paste this text into your word processor. The Copy Table as HTML option pastes it as a table like one you would see on a webpage. However, if you are working with a specific style format, such as the APA requirements for tables, you will need to recenter the table in your word processor to meet those requirements. The Copy as Picture option works nicely as long as you do not need to format the output to a particular style. The Copy as Picture option takes a picture of the output you highlighted. If you paste this picture into a word processor, it will look just like it did in the Results window. You will then be able to resize it, but you will not be able to edit it.

If you are using the Mac version of Stata and select Edit ▷ Copy as picture, Stata will copy the table as both a TIFF image and as a PDF image to the Clipboard. When you paste the file into another application, that application will automatically determine which version of the file to take from the Clipboard.

As you progress in your class or as you start to use Stata in your own research, you will find that copying and pasting is really not up to the task of creating a record of your work. For more complex work, you will want to use log files, which we will now discuss.

4.5 Logging your command file

Stata can write a copy of everything that is sent to the Results window into a file. The file is called a *log file*. When you start logging, you can create a new file, or you can add on to the end of an existing log. You can temporarily suspend logging at any time and then restart it again. If you do not have a running Stata session, start one now, and let's take a look at output logs.

We can open a log by selecting File ▷ Log ▷ Begin..., which will bring up the file selector window. Navigate to the directory in which we want to keep our log, and then enter a name (or select one from the list). By default, Stata will save this log using a markup language called Stata Markup and Control Language (SMCL) that only Stata can read in a Viewer or Results window. The log will have a *filename*`.smcl` name, where

`.smcl` is the extension. It looks nice in Stata but in another application, although it is a text file, the SMCL tags will be displayed because the word processor or other text editors will not understand them.

The other file format is called "log", which is a simple text file that your word processor can read. We need to pick the option of having a log file rather than a SMCL file from the dialog box. At the bottom of the *Begin logging Stata output* dialog box, we can specify *Save as type*. From the pulldown menu, select *Log (*.log)* for the type. Like the Results window, the `.log` format uses a fixed-width format, and if you insert this file into a word processor, you need to make sure that the font is fixed width (for example, Courier) and the font size is small enough (for example, 9 point). Make sure to select a location where you can find this log file. We can insert this log into an open file in our word processor, or we can open the log file itself in our word processor. Because the extension will be `.log`, when we open it into a word processor, we need to make sure that the word processor is not just looking for certain other extensions. For example, in MS Word, you would need to browse for the log file where the type is `*.*` rather than `*.docx`.

If you are using MS Word as your word processor, you have two options when working with a `.log` file. You can open it as a new document in Word as long as you remember to change the file type from `*.docx` to `*.*` or `show all`. If you do this, the Word document will have a fixed font. You may need to change the font size, say, to 9 point, or make the margins wider. If you have an existing Word document, you need to use the **Insert** option rather than the **Open** option, and insert the text where you want it.

Open a new log file called **`results.log`**. Make sure to specify that it is a log file rather than a SMCL file. Then run a **`summarize`** command on the file. Now open the log by selecting **File ▷ Log ▷ View...**, which will show us the basic information about our file, where it is stored, and the date and time it was created, and will give the results of our summary in a Viewer. When we go back to our Command window, the Viewer will seem to disappear, but it will be on the taskbar at the bottom of our screen. We can click on it to open it again.

Run a few more commands, such as a tabulation and a graph. Now select the Viewer from the Windows toolbar, and the Viewer returns, but it goes only as far as the original **`summarize`** command. However, at the top of the Viewer window is a **Refresh** button. Clicking on this button will update the Viewer to include the **`tabulate`** and the **`graph`** commands. What happened to the graph? Unfortunately, the log does not include the graphic output in the log file.

Experienced users use log files a lot. For beginners, log files may not be necessary. Also, if you make a lot of mistakes and need to run each command several times before you get it just right, the log will have all the bad output that you do not want, along with the good output that you do want. You might want to pause the log file while you try out a command or do-file. Then when you have the command the way you want it, you can restart the log to minimize the bad output. To do this, select **File ▷ Log ▷ Suspend** or **Resume**, respectively.

A strength of using log files is that they provide a record of both your commands and your results. The commands precede each result, so you can immediately see what you did. The advantage of the do-file from the Do-file Editor is that the commands are all together in a compact form, so it is easier to review the programming you did. A major limitation of the log file is that it does not save graphic output, such as the pie chart we did in this chapter. Graphs need to be saved by right-clicking within the graph and then choosing an option, whether you want to save it (select **Save as...**) or copy it (select **Copy**) so you can paste it into your word processor.

4.6 Summary

Let's review what you have learned in this chapter:

- How to open a Do-file Editor and copy commands from the Results window
- How to use the dialog boxes in Stata to generate Stata commands and then copy these to the Do-file Editor
- How to string a series of commands together in the Do-file Editor and add comments to this file
- How to run an individual command and groups of commands from the Do-file Editor
- How to save your do-file and retrieve it for later analysis
- How to cut and paste between Stata and a word processor, like Word
- How to create and view a log file

This is a lot of new knowledge for you to absorb. Never feel bad if you need to review this material because you forgot a step along the way. Even as you gain experience with Stata, this chapter will be a handy resource.

This chapter focused on the mechanics of using the Do-file Editor, writing a simple do-file, and saving your results. As you go through the following chapters, you will learn how to write more complex do-files and do more complex data management.

The rest of the book will focus on performing graphic and statistical analysis, building on what we have done so far. Most people learning a statistics program want to learn how to do analyses rather than what we have done so far. Still, what we have done so far sets the groundwork for doing the analyses. Chapter 5 will go over graphic presentations and descriptive statistics.

4.7 Exercises

1. Open `firstsurvey_chapter4.dta` by selecting File ▷ Open.... Open a Do-file Editor, and copy into the Editor the command that opened the dataset. Open the dialog box to summarize the dataset, run the **`summarize`** command for all the variables, and copy this command from the Results window into the Do-file Editor. Save this do-file as `4-1.do` in a place where you can find it.

2. Open the do-file you created in the first exercise, and add appropriate comments. Save the new do-file under the name `4-2.do`.

3. Open the file `4-2.do`. Put your cursor in the Do-file Editor just below the command that opened the dataset and above the command that summarized the variables (you will need to insert a new line to do this). Type the **`describe`** command in the Do-file Editor. Add a command at the bottom of the file that gives you the median score on education. Save the new do-file under the name `4-3.do`.

4. Open `4-3.do`, run the **`describe`** command and the command that gave you the median score on education, highlight the results, paste the results into your word processor, and change the font so that it looks nice.

5. Open `4-3.do`. Open a log file with the log file type. Call the file `4results.log`. Run the entire `4-3.do` file, and exit Stata. Open a new session in your word processor, and open your log file into this session. Format it appropriately.