THE ESSENTIALS

Now that you are familiar with the basic components of Stata and data files, it is time to begin performing statistical analyses. The thought of conducting statistical operations on top of learning a new computer program can be a doubly daunting task—often bringing with it a considerable amount of anxiety. This chapter is explicitly designed to help alleviate this common and natural emotional reaction to learning Stata to conduct statistical operations. This chapter has three primary goals. First, it presents a conceptual approach to learning Stata commands that has been shown not only to help learn the necessary operations but also to assuage the fears of "memorizing" seemingly endless commands. Second, the basic structure or format of Stata commands is covered. Regardless of whether the actual operation that a command performs is straightforward or complex, all Stata commands follow a very similar structure. Knowing this underlying format will help you process each newly presented operation more easily. Finally, this chapter discusses the 5 essential commands of Stata. These 5 commands form the foundation of statistical and data management operations for the vast majority of research projects. Therefore, once you have completed this chapter, you will have mastered a significant portion of using Stata to accomplish your research. Doing so will hopefully minimize anxiety and increase confidence when approaching the more nuanced topics covered in the subsequent chapters.

INTUITION AND STATA COMMANDS

Perhaps one of the more intimidating aspects of Stata is that it operates primarily, and most effectively, using a syntax, command-driven interface. As most readers have become accustomed to a Windows, point-and-click interface, this more "DOSesque" system may be unfamiliar and unusual. Furthermore, many users may be disheartened by the thought of trying to memorize numerous, odd-sounding commands.

These very valid concerns are why this book uses a new approach for teaching Stata commands. This method is founded on the idea that instead of viewing Stata as some black box that only spits out the right results when told exactly what to do, it is more beneficial to see Stata as an extremely smart colleague whom you are asking to produce some calculations very quickly. The latter perspective will help you remember that although Stata is a statistical computer program, it is designed by people. When these people thought about what to call particular commands, they did their best to give them names that made sense.

Taking the latter approach helps facilitate a more intuitive approach to Stata. Rather than considering the numerous command names that need to be "memorized," it is more effective to think "what would I call a command that would tell a computer to do a cross-tabulation?" Or alternatively, you can think "if my colleague and I had been working together for a long time, how might I tell him or her that I needed a cross-tabulation in a shorthand way." Generally, thinking in this manner leads you to the correct answer, `-tabulation-` or `-tab-` for short. This intuitive approach should help you learn and retain Stata commands more easily and effectively. It should also help minimize worries about the prospect of using Stata.

There are times when this type of thinking may not lead to exactly the right command. For example, if you thought "what would I tell my colleague if I wanted him or her to erase a variable from the data set," you may think "erase" or "delete." The actual command for this operation is `-drop-`. But approaching the new command in this way should lead you more quickly to the correct command and will help make the actual command make more sense and be more easily remembered.

Thus, as we embark on learning all the wonderful things Stata can do, keep this intuitive approach in mind. Remember you are simply working with a really smart colleague. At times communication may become strained, but with a bit of dialogue and understanding, you will be able to conduct very effective analyses.

A CLOSER LOOK: COMMANDS VERSUS POINT-AND-CLICK

Often, new Stata users are apprehensive about using Stata because of its command-driven interface, rather than a Windows, point-and-click-based system. Sometimes, this concern may tempt users to disregard learning Stata commands and instead rely solely on its Menus and point-and-click operation. Although this path may seem appealing, there are several reasons to fight the urge.

First, using the point-and-click method is not any easier, in terms of the amount of information you need to know. That is, even when using a Windows-based program, you still need to learn which menus to open, which button to use for a particular operation, and the correct options to choose. This method may seem easier than learning the commands, but it is not due to a difference in quantity of information to be attained. The distinction in the two methods rests mainly in the familiarity with using menus and Windows to perform operations. But at one time, this method was probably intimidating as well. Just as many people have come to feel very comfortable using Windows-based computer programs, with a little practice, the Stata syntax, command-based interface will seem just as straightforward.

Second, and perhaps even more important, there are real advantages to knowing the command-based aspect of Stata. For the majority of operations, the command-based interface is much quicker than the menus. What can take several point-and-clicks to process through the necessary layers of options can usually be typed in a few short words. Furthermore, although similar operations can be performed using either method, the command-based format makes it much easier to save and replicate your data manipulation and analyses. Often, you need to make adjustments to previously conducted procedures and run them again. As will be shown in the What Is a Do File? section of Chapter 3, using the commands along with "do files" makes this process much more straightforward. Additionally, if you continue to use Stata, you will find that many of the more advanced capabilities of the program rely on the command format.

THE STRUCTURE OF STATA COMMANDS

This section provides an overview of the components of Stata commands. Much more detail and specific examples are covered throughout the chapter, which will help clarify each aspect. Every command that is performed in

Stata has the same basic structure, which can be written in generic terms as follows:[1]

```
command varname(s) [if varname==value] [, options]
```

Command

Any statistical or data operation you want to perform in Stata has a name. For example, if you would like to delete an entire variable from the data, the command would be `-drop-`. These commands are generally the first item that is typed in the Command window (or "do file," covered in Chapter 3).

Most commands have two forms: (1) a full command name and (2) an abbreviated command name. The abbreviated command name contains the minimum number of characters required to uniquely specify that command. If a command has an abbreviation, you can type as many of the characters as you would like, as long as it contains the minimum abbreviation. For example, the full command to perform a linear regression is `-regress-`, but the abbreviated command name is `-reg-`. Therefore, you could type `-regress-`, `-reg-`, `-regr-`, `-regre-`, or `-regres-`, and the same operation would be performed. This book always introduces a command using the full command name, but often, an abbreviated command name is used for the sake of simplicity after this first use.

Variables

After the command, you must specify the variable or variables on which you want to perform that operation. For example, if you wanted to delete a variable named `gender`, you would type `drop gender` into the Command window. Particular commands, as will be discussed in more detail, either accommodate multiple variables or even require multiple variables to be specified. If, for instance, you wanted to create a cross-tabulation, you would specify two variables after the appropriate command.

[1] One aspect that is not included in this very basic presentation of the command structure is the implementation of weights. Advanced users interested in including weights, and other variants of this basic syntax, should see a full discussion of this topic in the Stata Help Files: Structure and Language section of Chapter 8.

if Statements

There may be times when you want to perform an operation only on certain types of cases. As an example, you may want to produce a cross-tabulation table that includes only the males in your data set. To do so, you would type an `if` statement after you have entered the command and variables. Generally, these `if` statements take the form of a particular variable or variables equaling some value.

The `if` statements are completely optional, meaning you do not have to enter them when performing a command, which is why they are shown in brackets above. You need to type an `if` statement only when you wish to perform the operation on a selected set of cases in the data.

Options

Most Stata commands include options that can be invoked with them. As the name suggests, option statements are optional. Options perform some extension or modification of the basic command, such as requesting additional statistical measures or a different formatting of the output. When a Stata command does not produce exactly what you would like by default, you can often obtain what you are looking for through the use of options. When each command is covered throughout the book, the most helpful options will be detailed as well. The Stata Help Files section of Chapter 8 explains how to learn all the possible options for each command.

Executing a Command Using the Command Window

Once you have determined which command you need to use, which variables you want to perform it on, and whether you would like to use an `if` statement or options, you are ready to execute the command.

First, be sure that you have the Command window selected by clicking the mouse when the cursor is anywhere in the Command window. Next, you will type the command, variable name(s), and any desired `if` statements or options. Instead of actually typing a variable name, you can click on the little arrow that appears when your cursor is over a particular variable in the Variables window or double-click on the variable in the Variables window to have the variable name appear in the Command window. Once you have finished entering all the information, you will press **Enter**.

Pressing **Enter** tells Stata to perform the operation and causes output to be displayed in the Results window. Note that some commands may wrap onto more than one line in the Command window. This scenario is completely acceptable. Stata treats everything typed before you press **Enter** as a single command. Thus, for each command you wish to perform, you need to type all the required information, and press **Enter** (i.e., you cannot type several commands in succession in the Command window). A method for performing multiple commands at once is covered in the next chapter.

THE 5 ESSENTIAL COMMANDS

The following section provides a closer look at the foundation commands of Stata. These 5 commands accomplish a significant portion of the analyses and data management that is needed for many research projects. This section should be seen, however, as an introduction to these commands. It explains the basics of each command, which for many users may be all that is needed. More of the specifics and nuances for each command are covered in the chapter devoted to that particular statistical operation. Therefore, the goal of this section is threefold. First, it provides essential commands that perform some of the most frequently used operations. Second, it gives you a framework on which to place all of the more advanced topics to come in later chapters. Third, it should give you confidence to tackle those more advanced topics. When you grasp these core concepts, you are in a great position to become an effective Stata user.

All the examples that follow use the `Chapter 2 Data.dta` file, available at study.sagepub.com/longest3e. This data set contains 7 variables for 25 cases from the National Study of Youth and Religion (NSYR) data (see the Preface for more information on how these data were collected). This subsample of the full data was selected so that it would be possible for you to double-check the following analyses by producing them by hand if it is helpful. As mentioned in Chapter 1, it is a good idea to save a copy of the data file you are working with so that you always have a backup of the original data.

tabulate

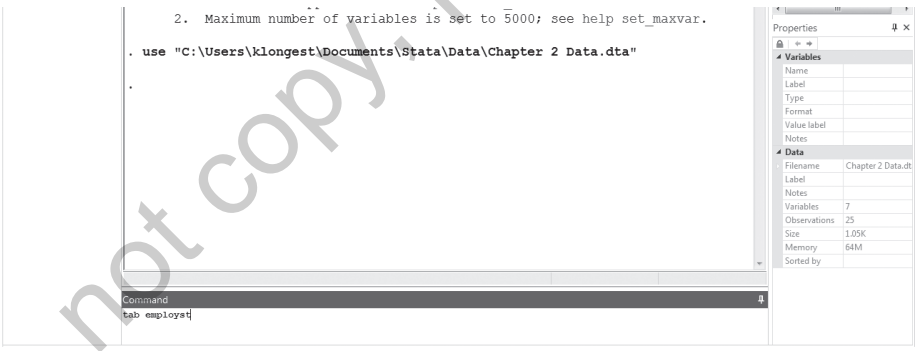
The first two essential commands, `-tabulate-` and `-summary-`, both produce basic descriptive information about variables, which is why they generally

are the first analytic operation performed for the vast majority of research studies. Again, this section will provide more of an overview on how to use these commands, whereas Chapter 4 will present much greater detail on the specifics of using these commands, as well as more detail on extensions to each.

One of the first analytic processes taught in statistics courses is how to construct a frequency distribution table. Notice, if you were asking a really smart colleague to produce a distribution that tabulates the values of each of the cases, you might tell him or her to "tabulate" the data. The abbreviated command name for `-tabulate-` is technically `-ta-`, but it will probably be easier to remember `-tab-` as a shortened version.

To see what the `-tab-` command does, let's look at the `employst` variable, which comes from a question asking about the respondents' current employment status. To examine this variable, select the Command window (i.e., click the mouse while the cursor is over the Command window), and type `tab employst` (or alternatively, type `tab` and place your cursor over `employst` in the Variables window, and double-click it or click the small arrow that appears). When you have typed the command, the screen should look similar to the screen shot presented in Figure 2.1.

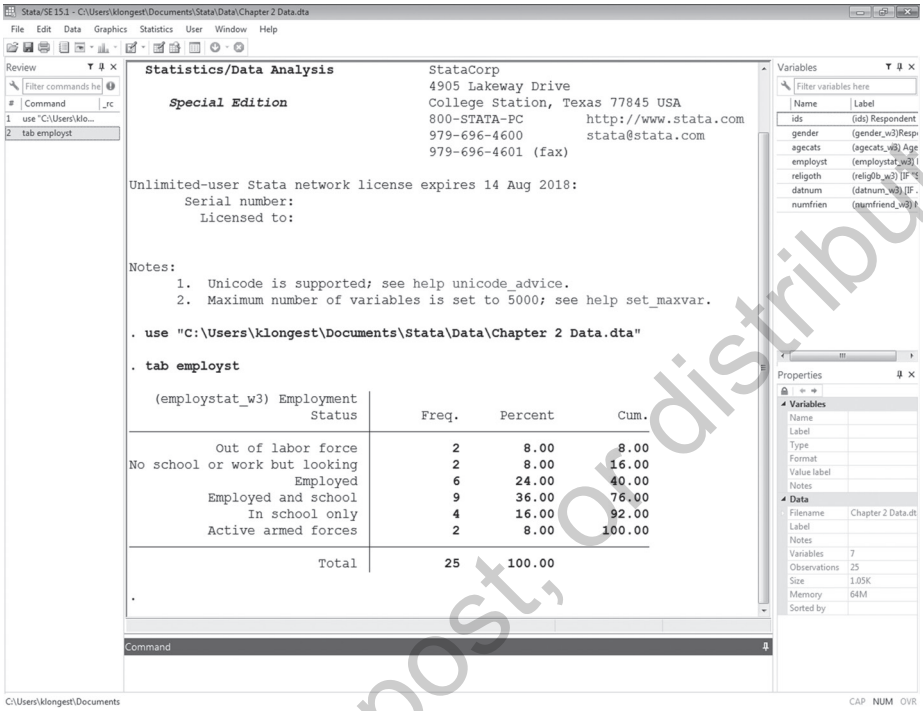
FIGURE 2.1 ⬢ COMMAND TYPED IN COMMAND WINDOW



Now simply press **Enter**, and your screen should look like Figure 2.2.

Before addressing the unique aspects of the information provided by the `-tab-` command, examine some of the general output that is produced by all Stata commands. First, you see what the command "did" displayed in the Results window. In this case, this is a distribution table showing the frequency, percentage, and cumulative percentage for each category of the variable `employst`.

FIGURE 2.2 ⬢ RESULTS OF –TAB– COMMAND



Next, just above this output, Stata presents the exact command that was executed to produce the output. In this case that is `tab employst`. This same information is also stored in the Review window. These three components are produced for every command you enter in the Command window.

The output of the –`tab`– command produces a table showing the frequency, percentage, and cumulative distribution of the given variable. For the `employst` variable, six cases are `Employed`, which is 24% of the sample. Also notice that it displays the total number of cases that fall into at least one of these categories of the variable. In this case, there are 25 cases that were coded into one of the presented categories.[2] The top left corner of the

[2] For these introductory examples, none of the variables have any missing data, meaning all the cases have valid answers for all the variables. Clearly, this situation may not always be the case with real data. Handling such missing data will be covered in more detail in the later chapters that more thoroughly discuss using the commands to complete statistical analyses. Specifically, see the Data Management: Missing Data section of Chapter 3 and the Advanced Convenience Commands: mark and markout section of Chapter 8.

table lists what is called the "variable label." These labels usually provide a brief description of the variable, such as "Employment Status." Working with these labels is covered in more detail in Chapter 3.

As with all commands, the `-tab-` command contains several options that can be invoked to perform different or additional operations beyond this default procedure. One of the more useful options for the `-tab-` command is `-sort-`. The `-sort-` option tells Stata, as it sounds, to rearrange (i.e., sort) the table so that it lists the categories in descending order of frequency. Remember, options are always typed after entering a comma, meaning you would type `tab employst, sort` in the Command window, and press **Enter**. When you do this, the following output is presented:

```
tab employst, sort
```

(employstat _ w3) Employment Status	Freq.	Percent	Cum.
Employed and school	9	36.00	36.00
Employed	6	24.00	60.00
In school only	4	16.00	76.00
Out of labor force	2	8.00	84.00
No school or work but looking	2	8.00	92.00
Active armed forces	2	8.00	100.00
Total	25	100.00	

As you can see, the same basic information (frequency, percentage, and cumulative percentage) is displayed, but now the categories are ordered so that you can easily see which one contains the most respondents and which contains the least. For employment status, the most common category is "Employed and school," whereas being "Out of labor force," "No school or work but looking" (for a job), and "Active armed forces" are all tied for the least common responses.

There are several other options you can use with `-tab-`, and most of them are covered in the Frequency Distributions section of Chapter 4. But for now, it is only necessary to understand the basic form of how options are invoked, as it is similar for all other commands. Also, it should be noted that as with commands, options have full names and abbreviated names. The full name will always be presented when it is first introduced, with the most common abbreviation being used in all the following instances.

In addition to producing a distribution of one variable, the -tab- command can generate a cross-tabulation between two variables. For instance, you may be interested to know if there is a difference in the respondents' employment status by gender. To make this comparison, you would want to see the distribution of employment status for males and the distribution of employment status for females. One method for displaying this information is to invoke the -tab- command and list both variables instead of just one. Type `tab employst gender` in the Command window, and press **Enter**. Doing so produces the following results:

```
tab employst gender
```

(employstat _ w3) Employment Status	(gender _ w3)Respondent gender Male	Female	Total
Out of labor force	0	2	2
No school or work but looking	0	2	2
Employed	2	4	6
Employed and school	3	6	9
In school only	3	1	4
Active armed forces	1	1	2
Total	9	16	25

From this table, you can see that there are more females (16) than males (9) in this sample. Additionally, females are more prevalent in all the categories, except in the "In school only" and "Active armed forces" categories. But these differences in frequencies could stem from the greater number of females overall. Therefore, to know if there is a true relationship, you would want to know the percentage of females in each category compared with the percentage of males in each category.

To produce the necessary figures, you can again think intuitively. You need to ask Stata to produce a set of percentages based either on the rows or on the columns. Because you believe that gender is the causal variable (i.e., the independent variable), you would want the percentages in the columns. That is, you want to be able to compare the proportion of *all females* who are employed with the proportion of *all males* who are employed. Therefore, the percentages need to be calculated within the columns. To have your smart colleague make this calculation, you might consider telling him or her, for shorthand, "columns." Following this logic, the option to present

these percentages is `-column-`. (If you wanted the percentages in the rows, as you might have guessed, the option would be `-row-`.) Type `tab employst gender, col` in the Command window, and press **Enter**.

```
tab employst gender, col
```

```
+--------------------------+
| Key                      |
|--------------------------|
|      frequency           |
|  column percentage       |
+--------------------------+
```

(employstat _ w3) Employment Status	(gender _ w3) Respondent gender Male	Female	Total
Out of labor force	0 0.00	2 12.50	2 8.00
No school or work but looking	0 0.00	2 12.50	2 8.00
Employed	2 22.22	4 25.00	6 24.00
Employed and school	3 33.33	6 37.50	9 36.00
In school only	3 33.33	1 6.25	4 16.00
Active armed forces	1 11.11	1 6.25	2 8.00
Total	9 100.00	16 100.00	25 100.00

A key is displayed at the top of the table to indicate what each of the numbers in each cell represents, in this case the frequency and the column percentage. Using the percentages, you can more accurately compare the relationship of gender on employment status. When examining only the frequencies, it appeared that females were more likely to be employed because 4 females reported being employed compared with only 2 males. But using the percentages, it now appears as if there is no difference by gender in terms of being employed because 25% of females are employed and just more than 22% of males are employed, a difference of less than 3%.

A CLOSER LOOK: COMMAND SHORTCUTS

You might have noticed that the three previous commands were very similar, with the latter two simply adding a new option. You may wonder if there is a shortcut to reproduce these commands and add the new option without having to type out the entire command again. Fortunately, there are actually two methods to save you from this repetitive typing. To practice both methods, imagine that you wanted to see the percentages based on the rows in addition to the columns.

The first method uses the Review window. To start, find the command in the Review window that is closest to what you would now like to perform. In this case, you would select the command that first produced the column percentages (i.e., `tab employst gender, col`). Move your cursor onto this command, and click on it. The command appears in the Command window. Now you can select the Command window, type `-row-` at the end of the command, so that it now would read `tab employst gender, col row`, and press **Enter**.

For the second method, be sure that you have the Command window selected (i.e., your cursor should be flashing in the Command window). Then press the **Page Up** key. The command you just ran is now displayed in the Command window. If you continue pressing the **Page Up** key, you will cycle through all the commands you have invoked during the current session. You can find the one you are looking for, type `-row-` at the end, and press **Enter**.

While both of these methods are efficient shortcuts for rerunning commands during one session of Stata, the What Is a Do File? section of Chapter 3 explains an even more effective method for running similar commands over multiple sessions of Stata.

summary

Often you need to analyze variables that contain numerous categories, such as income or number of people in a county (i.e., interval-ratio variables). For such variables, it would be possible to produce a distribution table, but it might not be very useful because each category would most likely contain only a few cases. In the current data set, the variable `-datnum-` comes from a question on the NSYR that asked how many romantic relationships the respondent has been involved in during his or her lifetime. The question used an open-ended response format, so that respondents could report any number between 0 and 100. With only 25 respondents in the current subsample, it is likely that each case has its own unique value, making a distribution table not very helpful.

In such a case, it would be more helpful to see figures that describe the basic pattern of values without actually seeing each case's specific response. Thinking intuitively, you would be asking your smart colleague to "summarize" the variable with a few numbers. These summary statistics are often referred to as measures of central tendency and variability. The command to produce these statistics is -summary-, or -sum-. To produce these figures, type `sum datnum` in the Command window, and press **Enter**. Doing so presents the following results:

```
sum datnum
```

Variable	Obs	Mean	Std.Dev	Min	Max
datnum	25	7.56	6.083311	1	20

The default -sum- command (i.e., with no options included) produces several descriptive statistics. Moving from left to right: the number of observations with data on the given variable (`Obs`), its arithmetic mean (`Mean`), the standard deviation (`Std. Dev.`), and the smallest (minimum or `Min`) and the largest (maximum or `Max`) value any case reports for that variable are presented. The figures are all based on observations with information on that variable in the current data set. For example, respondents in the NSYR could have possibly reported having dated up to 100 people, but no one in the current subsample claimed to have dated more than 20 people.

One helpful feature of the -sum- command is that it allows for more than one variable to be entered in a single command line. For example, try entering `sum datnum agecats` in the Command window, and press **Enter**. The following output will be displayed:

```
sum datnum agecats
```

Variable	Obs	Mean	Std. Dev	Min	Max
datnum	25	7.56	6.083311	1	20
agecats	25	20.04	1.619671	18	23

The same information is displayed, but now the descriptive statistics for both number of people dated and age are shown. The -sum- command accepts as many variables as you would like to enter.

There may be times when you would like to know additional descriptive statistics besides the ones displayed by the -sum- default, such as the

percentile values or the kurtosis. To have Stata produce these statistics, you can invoke the -detail- option with the -sum- command. In the Command window, type `sum datnum, detail`, and press **Enter**. Now the results appear as shown below:

```
sum datnum, detail
```

```
        (datnum _ w3) [IF HAS BEEN IN A ROMANTIC
      RELATIONSHIP OR HAS BEEN MARRIED J:4. How
-------------------------------------------------------------------------------
    Percentiles                 Smallest
     1%         1      1
     5%         2      2
    10%         2      2  Obs           25
    25%         3      2  Sum of Wgt.   25
    50%         5         Mean           7.56
             Largest      Std. Dev.      6.083311
    75%        10     15
    90%        20     20  Variance      37.00667
    95%        20     20  Skewness       .9859665
    99%        20     20  Kurtosis       2.722225
```

Now, in addition to the observations, mean, minimum, and maximum, the -sum- command has produced multiple percentile values, the variance, skewness, and kurtosis score. Many users may be interested in the median value of particular variables. Although Stata does not indicate a value as the "median," the figure listed as the 50% value is equivalent to the median value. Just as with the default -sum- command, you can enter multiple variables in one command line, and a detailed descriptive statistic table for each one will be displayed.

generate

Frequently, when you are working with data, you need to generate new variables that do not initially exist in the data set. For example, you might realize that instead of using respondents' actual age, you need a measure that indicates how many years older than 18 they are, essentially treating age 18 as the 0 or starting age. Unfortunately, there is no variable in the current data set that holds the exact information. Therefore, you need to create a new variable in order to achieve your goal. Specifically, you have to generate a variable that represents the number of years over 18 a person is.

Intuitively, if you were going to tell your smart colleague to make a new variable, what single-word command would you use? Perhaps you might

consider "create," which makes sense, but another similar option would be "generate." If you initially thought "create," remember that although your intuition may not have led you to the exact right command name, thinking in this way should now help you remember the correct one: `-generate-`. As opposed to the previous commands you have learned, `-tab-` and `-sum-`, the `-generate-` command (abbreviated `-gen-`) is not a data analysis command. That is, it does not produce results of statistics or figures. Rather, `-generate-` is a data management command that is used to create new variables that you can then analyze. But remember, all Stata commands follow a similar structure.

So the first word you will type in the Command window is the command name `-gen-`. Next, you need to type what the newly generated variable name will be. Although you can name this new variable just about anything you would like, there are a few general tips when deciding on a new variable name. As mentioned in the Preface, Stata is case-sensitive. This feature is why it is an effective strategy to use variable names that contain only lowercase letters. It is possible to name a variable `gender`, `Gender`, or even `GENDER`. But there are times at which you will need to type this variable name. It is generally quicker, usually easier, and less likely to produce mistakes if you use only lowercase letters in variable names. Next, try to be as succinct as possible. You might have to type this variable name several times, so the shorter the better. But you must balance this succinctness with a concern for clarity. It might be tempting to call your new variable something like `newvar1`. Although you might remember what that variable represents right now, after creating several new variables it might get confusing. This scenario is why it is best to give your new variable a name that will help you remember exactly what it represents.

The variable that you need to create in the given example is going to indicate the number of years past 18 the respondent is. Therefore, you might call this new variable `agep18`. This variable name tells you that the variable is the person's age "post" 18. Again, you should name the variable whatever will help you remember what it contains using the fewest characters possible.

Once you have decided on the new variable's name, type it into the Command window. As of now, the command should read `gen agep18`. Next, you have to tell Stata (or your smart colleague) what exactly should be in this new variable. Or, in other words, you need to indicate what this new variable will *equal*. Thus, after the new variable's name, you need to

type an equal sign, making the command `gen agep18=`. Do not worry about whether there are spaces before or after the equal sign; Stata does not care about how many or few spaces there are in a command line. From here, you can probably determine that the remainder of the command line should indicate exactly what the new variable will contain. This portion of the command is similar to a formula or equation and can contain just about any operation you can think of.

Most often, what follows the equal sign in a `-generate-` command is one or some combination of three elements: numbers, mathematical functions, and existing variables. It is important to remember that whatever you type after the equal sign is applied to every case in the data set (unless you invoke an `-if-` statement, which is described next). That is, in a way, Stata goes through each case, one by one, and executes the formula you type after the equal sign. For example, try typing `gen examp=200` in the Command window, and press **Enter.** Notice that in the Variables window there is a new variable listed with the name examp. Now produce a distribution of this new variable (i.e., type `tab examp` in the Command window, and press **Enter**). The results should be as follows:

```
gen examp=200

tab examp

    Examp |      Freq.     Percent        Cum.
----------+-----------------------------------
      200 |         25      100.00      100.00
----------+-----------------------------------
    Total |         25      100.00
```

This table shows that all 25 cases have been assigned a value of 200 for this new variable.

For the purposes of the substantive example, you need a variable that represents the number of years after 18 a respondent is. To create such a variable, you would need to subtract 18 from the person's current age. Using numbers alone, you might consider typing `gen examp2=21-18` to generate a variable that will represent the number of years past 18 for all 21-year-olds. If you execute this command and again produce a distribution of it (i.e., type `tab examp2` in the Command window, and press **Enter**), the results should be displayed as follows:

```
gen examp2=21-18

tab examp2
```

```
     examp2 |      Freq.     Percent        Cum.
------------+-----------------------------------
          3 |         25      100.00      100.00
------------+-----------------------------------
      Total |         25      100.00
```

This new variable contains the correct information for what was typed. A person who is 21 is 3 years past 18 years old. But you can see that it has assigned 3 to every case. If you look back at the summary table you produced earlier for the `agecats` variable (to do so, simply use the scroll bar on the right side of the Results window or the **Page Up** key after selecting the Results window), you can see that not everyone in the sample is 21 years old. At least one of the respondents is 18 years and another is 23 years, meaning that these cases would be 0 and 5 years past 18, respectively.

To produce a variable that contains the correct information for each case, you need to tell Stata to generate a new variable that is the case's actual age minus 18. Instead of putting "21," as you did for the last example, you need Stata to use each case's actual age and then subtract 18. Which variable contains this information that you already have in Stata? The variable `agecats`. Thus, the correct command to produce a variable that represents each person's number of years past 16 would be `gen agep16=agecats-18`. Notice, you simply replace "21" with the variable name that contains each case's actual age. Stata now knows to go through each case, one by one, and take that case's value in the `agecats` variable, subtract 18, and then enter this into the new variable. Type this command into the Command window, and press **Enter**. Then, display a distribution table of this new variable (i.e., `tab agep18`). The table should look like the one shown below:

```
gen agep18=agecats-18

tab agep18
```

```
    agep18 |      Freq.     Percent        Cum.
-----------+-----------------------------------
         0 |          7       28.00       28.00
         1 |          2        8.00       36.00
         2 |          6       24.00       60.00
         3 |          4       16.00       76.00
         4 |          5       20.00       96.00
         6 |          1        4.00      100.00
-----------+-----------------------------------
     Total |         25      100.00
```

Now you can see that the cases have been assigned different values based on what their actual age is. If you want to double-check to make sure that this new variable contains the correct information, you could create a cross-tabulation of the original age variable by this new variable. Type `tab agecats agep18` in the Command window, and press **Enter**. The following results table should be displayed:

```
tab agecats agep18
```

(agecats _ w3) Age variable collapsed into one year categories	agep18						Total
	0	1	2	3	4	5	
18	7	0	0	0	0	0	7
19	0	2	0	0	0	0	2
20	0	0	6	0	0	0	6
21	0	0	0	4	0	0	4
22	0	0	0	0	5	0	5
23	0	0	0	0	0	1	1
Total	7	2	6	4	5	1	25

The table shows that you have created the new variable correctly. All 7 cases who are 18 years old are coded as being 0 years past 18 in the `agep18` variable. The 2 cases who are 19 are coded as 1 years past 18, and so on.

It is possible to use multiple variables to generate a new measure. For example, perhaps you wanted a single measure to indicate a respondent's overall sociability. In the current data, there are two measures that could serve as indicators of that concept, the number of people dated (`datnum`) and the number of close friends reports (`numfrien`). For a single measure, you might consider adding the number of friends to the number of people dated. To create this variable, follow the same steps as above. First, enter the command `-gen-`. Next, enter the new variable name you would like to use: `gen totsocial` (total sociability). Then type an equal sign and put in the formula needed to correctly generate the new variable.

This latter portion is the slightly tricky part. But think of what information you need and where it is contained. You need the number of people each person reports to have dated (i.e., the `datnum` variable)

and the number of close friends (i.e., `numfrien` variable). Then, what mathematical operation do you need to perform on these pieces of information? Add the two together. Therefore, the full command that you need to enter into the Command window should read as `gen totsocial=datnum+numfrien`.

Notice that the "+" is used to tell Stata to add the value of each case on `datnum` to that case's value on `numfrien`. See the "A Closer Look: Mathematical Operators and Their Symbols" box for a list of the most commonly used mathematical operators and their Stata symbols.

Just as before, it is generally a good idea to double-check that the new variable contains the information that you intended. Here, because the new variable was the result of an operation performed on two other variables, simply displaying a cross-tabulation will not provide all the information. Although there are several options to conduct this evaluation, perhaps the easiest method with the tools you have already learned is to open the Data Browser and check the values of a few cases on `datnum`, `numfrien`, and `totsocial` to make sure that the calculation resulted in the correct information being coded in the new variable. Now that you know how to use the Command window, you can actually open the Data Browser by typing `browse` in the Command window and pressing **Enter**. Or you can type `browse datnum numfrien totsocial`, and only the variables listed will be displayed in the Data Browser window. Once you have opened the Data Browser, you should see that the new variable `totsocial` does indeed represent the number of people each case has dated plus the number of friends they report.

replace (if)

Sometimes, instead of creating a new variable, you might need to alter the values of an existing variable. There are several reasons why you would want to perform such an operation. You might realize that you need to make a replacement due to a mistake in the data entry process. For example, perhaps all the 18-year-olds are actually 19 years old. More frequently, you might want to combine categories for substantive reasons. For example, you might decide that you do not want to make the distinction between being out of the labor force and being out of work but looking for a job. Therefore, you would want to replace the values for the out of work but looking for a

A CLOSER LOOK: MATHEMATICAL OPERATORS AND THEIR SYMBOLS

When you create new variables in Stata, you often need to use a mathematical operation or function. Each of these operations has a particular symbol that Stata recognizes. Several are probably obvious, such as "+" for addition, but others might not be so clear. The following is a list of the most commonly used operations/functions and their symbols.

Addition	+
Subtraction	–
Multiplication	*
Division	/
Power	^
Greater Than, Less Than	>, <
Greater Than or Equal To, Less Than or Equal To	>=, <=
Absolute Value	abs(number/variable name)
Natural Log	ln(number/variable name)
Square Root	sqrt(number/variable name)

Also note that Stata follows the traditional order of operations. For example, if for some reason you wanted the variable `totsocial` to contain the natural logarithm of total friends and dating partners, you could type the following command:

```
gen lntotsocial=ln(datnum+numfrien)
```

Stata will conduct what is inside the parentheses first (i.e., the addition) and then take the natural logarithm of the result.

job cases to be equal to the value of those who are coded as unemployed so that these two categories would have the same value. In each of these scenarios, you would be asking your smart colleague to take the current values of that variable and replace them with a new value. In one word, you would ask your colleague to `-replace-` the old values with the new values.

In many respects, the -replace- command works very similarly to the -gen- command. Instead of creating a new variable, the -replace- command simply changes the values of a current variable. For the sake of practice, assume that the problem with age noted above had been discovered (i.e., all 18-year-olds were actually 19). Before starting the replacement process, it is helpful to produce a distribution of the existing variable to compare the new version of the variable after you replace the necessary values. Typing `tab agecats` in the Command window and pressing **Enter** produces the following results:

```
tab agecats
```

(agecats _ w3) Age variable collapsed into one year categories	Freq.	Percent	Cum.
18	7	28.00	28.00
19	2	8.00	36.00
20	6	24.00	60.00
21	4	16.00	76.00
22	5	20.00	96.00
23	1	4.00	100.00
Total	25	100.00	

You can see that there are 7 cases that need to have their value of 18 replaced to equal 19. Therefore, once you have made the replacement, the new version of `agecats` should have 9 cases that equal 19.

To fix this problem, you need to change all the cases that are currently coded as 18 to equal 19 on the `agecats` variable. As with previous commands, begin by typing the command into the Command window: `replace`. Next, you need to enter the name of the existing variable whose values you would like to replace: `replace agecats`. To keep this order in mind, it can be helpful to think about how you would ask a colleague to perform this operation (i.e., "please, replace (the values of the) agecats (variable)"). Now, just as with the -gen- command, you have to tell Stata what the new values should equal. In this case, you are asking Stata to set the new values to equal 19, so the command at this point should read `replace agecats=19`.

Again, just as with -gen-, what comes after the equal sign are usually numbers, mathematical operators, and variable names.

But you are not done at this point. The -replace- command, just like -gen-, performs whatever you enter after the equal sign in a case-by-case fashion on all cases, unless otherwise told not to. Therefore, if you pressed **Enter** with only the current command line typed, Stata would go through each case and, because it has no further information, replace each respondent's value of agecats to equal 19. This change is not what you are trying to accomplish. Instead, you need to tell Stata that it should only change the value of a case *if* that case currently equals 18 on the agecats variable.

To perform this operation, you need to invoke an -if- statement. Whenever you use an -if- statement, you are telling Stata to perform the command that precedes the -if- only on cases for which the expression that follows the -if- is true. Most often, what will follow the -if- statement in a command line is a variable name with an equal to, greater than, or less than condition. In the current example, you want Stata to replace the value on agecats to equal 19 if the case's value on agecats equals 18. This command may be starting to sound a bit complicated, but remember to think about how you would ask someone to complete this operation (i.e., "please, replace (the values of the) agecats (variable) (to) equal 19 if (the value of the) agecats (variable) equals 18"). Following this form leads to the correct structure of the command: `replace agecats=19 if agecats==18`.

But wait. You probably noticed that the equal sign after the -if- in the command line includes *two* equal signs. This is not a typo. When you want to use an equality expression in an -if- statement, you must type two equal signs. Again, this process may sound confusing, but there is a pretty straightforward rule of thumb to help keep this difference straight. Whenever you are telling Stata to make something equal to something else (i.e., to change a value so that it becomes equal to something else), then you only use one equal sign. In the first part of this command, you are telling Stata to make the values on the variable agecats to become equal to 19, meaning you only need to type one equal sign. Whenever you are telling Stata to evaluate or assess whether something is equal to something else, then you need two equal signs. In the second part of the above command, you are asking Stata to perform the replacement only after checking to see if that case's value on agecats is equal to 18, meaning you need to type two equal signs.

Type the -replace- command from above (`replace agecats=19 if agecats==18`) into the Command window, and press **Enter**. Then produce a distribution table of the `agecats` variable (i.e., `tab agecats`). When you do, you will see the following table:

```
replace agecats=19 if agecats==18

tab agecats
```

(agecats _ w3) Age variable collapsed into one year categories	Freq.	Percent	Cum.
19	9	36.00	36.00
20	6	24.00	60.00
21	4	16.00	76.00
22	5	20.00	96.00
23	1	4.00	100.00
Total	25	100.00	

The table shows that the replacement has been made correctly. All 7 cases that previously had been coded as 18 have now been set to equal 19. None of the values of the other cases have been altered, meaning that the -if- statement operated successfully.

You may be realizing about now that there is a danger in using the -replace- command. Once you have performed a replacement, there is no reverting back. Now that you have changed the 7 cases that were previously recorded as 18 to equal 19, there is no way to separate them from the 2 cases that were originally coded as 19. So if you discovered that you had mistakenly thought those cases need their ages changed, it would be impossible to change them back. This situation is precisely why it is strongly recommended that you always work with a duplicate copy of the original data set and keep an original copy of the data file as a backup. Doing so will always provide you with a method for recovering the original version of any variable you may change. This scenario also illustrates the utility of "do files," explained in the What Is a Do File? section of Chapter 3, as a method for saving all your commands, in case you need to replicate a portion of your analyses after making such a mistake.

A CLOSER LOOK: THE "DREADED" ERROR MESSAGE

So far, you have done everything correctly. Unfortunately, even the most experienced Stata users make mistakes. When you enter a command incorrectly, Stata displays an error message telling you that something went wrong. This error message is accompanied by a clickable link that displays more information about the particular error. Most frequently, error messages are the result of a typographical error (e.g., typing the name of a variable incorrectly). Sometimes, however, error messages result from not entering the command or an option appropriately. One of the first such instances you will probably encounter is the use of the double equal sign after an –if– statement. For example, type the command you just conducted into the Command window, but delete one of the equal signs after the –if– portion of the command. The (incorrect) command should read `replace agecats=19 if agecats=18`. When you press **Enter**, you will see the following results:

```
replace agecats=19 if agecats=18

invalid syntax
r(198);
```

The error message is "invalid syntax," and "r(198)" is the clickable portion that tells you exactly why you received the error message. If you click on that portion of the message, Stata gives more information about the error. Note that at the end of this description, it states "Errors in specifying expressions often result in this message." Expressions include –if– statements, meaning Stata is telling you to double-check to make sure you have used the correct operator, which in this case you have not because you only entered one equal sign. You will also note that while the incorrect command is still recorded in the Review window it is in red font, indicating that it resulted in an error.

Additionally, there is a safer way to use the –replace– command to prevent this scenario from occurring in the first place. For this example, assume that you have now learned that the one case that has been recorded as being 23 years old should have actually been coded as 22 years old. But you are concerned that at some point you may want to be able to identify this miscoded case (perhaps to analyze why the case may have been incorrectly recorded). What you are looking to do then is create a second version of the `agecats` variable that you could use to replace the value of the 23-year-old case with 22. Before delving into how to create this copy, can you think of a way, using the commands you have already learned, to do so?

To create this copy, you could use the -gen- command. You want to tell Stata to generate a new variable that is equal to the current agecats variable. Remember the structure of the -gen- command: command new-variable-name = new-value. Therefore, type gen agecatsrp= agecats in the Command window, and press **Enter**. Now, create a cross-tabulation of the original agecats variable and your newly created copy (tab agecats agecatsrp). Doing so produces the following results:

```
gen agecatsrp=agecats

tab agecats agecatsrp
```

(agecats _ w3) Age variable collapsed into one year categories	agecatsrp					Total
	19	20	21	22	23	
19	9	0	0	0	0	9
20	0	6	0	0	0	6
21	0	0	4	0	0	4
22	0	0	0	5	0	5
23	0	0	0	0	1	1
Total	9	6	4	5	1	25

The two variables are identical, which is what you were looking to create. Now you can use the -replace- command just as you did before to change the value for the case coded as 23 to 22 on your newly created copy of the agecats variable. Remember, you are only asking Stata to replace a case to equal 22 if that case currently equals 23 on the agecats variable, meaning you need an -if- statement in your -replace- command. So you would type replace agecatsrp=22 if agecats==23 in the Command window, and press **Enter.** Next produce a distribution table of the copied agecatsrp variable, now with the replaced value, and the original agecats variable (tab agecats agecatsrp):

```
replace agecatsrp=22 if agecats==23

tab agecats agecatsrp
```

(agecats _ w3) Age variable collapsed into one year categories	agecatsrp 19	20	21	22	Total
19	9	0	0	0	9
20	0	6	0	0	6
21	0	0	4	0	4
22	0	0	0	5	5
23	0	0	0	1	1
Total	9	6	4	6	25

The values from the original `agecats` variable are listed in the rows, and the new version's values are in the columns. The table shows that the two versions are similar, except that the one case that is coded as 23 in the original version has now been replaced to equal 22 in the new version. Notice that using this process allows you to be able to identify the case(s) that was replaced through a comparison of the old version of the variable with the copied and replaced version of the variable. This ability may be needed at some point in your analyses, and if nothing else, you now will always have the original version of the variable in case you should need it.

Although this particular example may seem a bit superfluous, there are more pertinent situations when you will need to replace the values of an existing variable. Even more frequently, you will find this method of creating a new variable (either as the copy of an existing variable or based on a particular calculation) followed by single or even multiple replacements to be quite useful.

One typical example of a more complicated replacement is the creation of a dichotomous or nominal variable that is based on multiple conditions. Consider that you want to create a dichotomous variable (sometimes referred to as a dummy variable) that serves as an indicator of respondents being "isolated." For present purposes, define "isolated" as having dated fewer than 2 people and having 2 or fewer friends. Based on this definition, you are going to count anyone who has dated 2 or fewer people *and* has 2 or fewer friends as isolated. As you can see, the conditions for being included are somewhat complicated. Fortunately, the combination of `-gen-` and `-replace-` help create this new variable.

To start, you need to create a new variable that can serve as the indicator of being isolated. To do so, you need to use the `-gen-` command, but remember, you need to tell Stata what the new variable should equal. In the end, you want a dichotomous variable, meaning it should have two possible values. Typically, such variables are coded with one category equal to 0 and the other equal to 1. In the given example, the variable would be coded 0 if the case is not isolated and 1 if the case satisfies the requirements of being isolated.

A useful practice when creating these types of variables is to begin by coding every case on the new value to equal 0. Doing so essentially starts with the premise that no one is included in the indicator category. Then you can tell Stata which cases need to be replaced, to equal 1, if they satisfy the determined criteria to be included in the indicator. Thus to start, type `gen isol=0` into the Command window, and press **Enter**.

Now you have the new variable in which every case has been set to equal 0. You need to replace the cases that fit the requirements for being isolated to equal 1. Notice, however, that based on the criteria there are two conditions that must be met, meaning you need to use a slightly more specific `-if-` statement. The `-replace-` command starts very similarly to what you did above: `replace isol=1 if`. From here, you need to think about what you need to tell Stata so that it correctly replaces the cases to equal 1 if they meet all the requirements for being counted as isolated.

The first component is that the case must have only dated 2 people or fewer. Therefore, the first clause in the `-if-` statement should clarify this condition: `replace isol=1 if datnum<=2`. Notice, instead of using a double equal sign as before, here you can use the less than or equal to symbol because you are telling Stata to include all cases that are equal to or less than 2 on `datnum`.

But you are not quite done. If you were to hit **Enter** at this point, all cases that have dated 2 or fewer people would be replaced to equal 1. But you want cases to equal 1 (i.e., to be in the isolate group) if they have dated 2 or fewer people *and* have 2 or fewer friends. Therefore, you need to add another condition to the `-if-` statement. Again, try to keep in mind what you would tell your smart colleague to do if you wanted him or her to actually move a group of people who should be counted as isolated. You might tell that person to "move everyone who has dated 2 or fewer people *and* who has 2 or fewer friends." If you replace each "has" in that statement with a less than or equal to sign and the conjunctions with the appropriate symbols, you will have the correct Stata command: `replace isol=1 if`

`datnum<=2 & numfrien<=2`. Now you can press **Enter**. When you do so, you will see the following results:

```
gen isol=0

replace isol=1 if datnum<=2 & numfrien<=2

(2 real changes made)
```

These results tell you that 2 cases had their values changed on the variable `isol`. As before, it is good to double-check that the correct cases were replaced. You can use the Data Browser window to do so. Or you can produce a cross-tabulation of `datnum` and `numfrien` to ensure that the correct cases were altered.

A CLOSER LOOK: LOGICAL OPERATORS AND THEIR SYMBOLS

When you use compound `-if-` statements, you need to use logical operators. Each of these operations has a particular symbol that Stata recognizes. The following is a table of the most commonly used operators and their symbols.

And	&
Or	|
Not Equal To	!=
Is Equal To	==

Also note that as with mathematical operators, it is helpful to use parentheses to ensure that the intended order of operators is followed. For example, if you wanted the cases to be coded as isolated if they are younger than 20 years old and have dated 2 or fewer people or have 2 friends or fewer, you might type

```
replace isol=1 if agecats<20 & datnum<=2 | numfrien<=2
```

Given this exact command, however, Stata would code people who are younger than 20 years and have dated 2 or fewer people as equal to 1. It would also code people who have 2 or fewer friends as equal to 1. Note that this replacement is not what you intended because Stata grouped the first two clauses rather than the second two. To fix this scenario, use parentheses to explicitly denote the correct grouping:

(*Continued*)

(*Continued*)

```
replace isol=1 if agecats<20 & (datnum<=2 | numfrien<=2)
```

Now Stata will know that for a case to equal 1, it has to first be less than 20 years old and then the case either needs to have dated 2 or fewer people or have 2 or fewer friends.

To produce such a table, you want to create a standard cross-tabulation as you did above. Except now you only want to display the cases that were replaced to equal 1 on the new variable so that you can check to make sure those cases have the appropriate values on `datnum` and `numfrien`. In other words, you only want the –`tab`– command to apply to cases that are now equal to 1 on the `isol` variable. To limit the table to only these cases, you should invoke another –`if`– statement. Type `tab datnum numfrien if isol==1` in the Command window, and press **Enter**. Remember, because you are asking Stata to assess whether a case is coded as 1 on the `isol` variable (i.e., comparing each case's value to 1), you need the double equal sign.

```
tab datnum numfrien if isol==1
```

(datnum _ w3) [IF HAS BEEN IN A ROMANTIC RELATIONSHIP OR HAS BEEN MARRIED J:4. How	(numfriend _ w3) N:1. Now for the next set of questions, I'll be asking some things 2	Total
1	1	1
2	1	1
Total	2	2

The table only shows the values on `datnum` and `numfrien` for the two cases that were changed to be equal to 1 on the `isol` variable. And as you can see, they both satisfy the requirements of having dated 2 or fewer people and having 2 or fewer friends. Using this type of –`if`– statement with other

commands is covered in later chapters that examine specific analyses. For now, you should be able to use it to assess whether these types of replacements were completed successfully.

recode

One of the limitations of the `-replace-` command is that it can only replace one value at a time. That is, in the `agecats` example above, you needed to type two commands to change the values of 18 to 19 and 23 to 22. You cannot type `-replace-`, in one command line, to make `agecats` equal to both 19 and 23 based on different conditions. In some respects, this limitation could be seen as a strength of the `-replace-` command because it makes you be very careful and explicit when you make changes to the data. But it can also become quite cumbersome when you need to alter several values of an existing variable. Fortunately, Stata has a command, `-recode-`, that allows multiple replacements using only one command line.

Intuitively, you may not arrive at the word "recode" for a command that asks a colleague to change the values of a variable. But the command name provides a good deal of intuitive information that should help you remember it. The primary distinction between `-replace-` and `-recode-` is that with `-replace-` you are literally overwriting the value of a variable (or a case) with another value. Although you used an `-if-` statement to isolate this overwriting to particular cases, the default operation of `-replace-` is to change the values of every case in a variable. `-recode-`, on the other hand, is asking Stata to take a current value and change it (i.e., recode it) into another value. This different operation means you have to tell Stata the old value you would like to change and what new value you would like it to be recoded to.

For this example, consider that you have decided to create an ordinal variable out of the currently interval-ratio variable `datnum`. You want to distinguish minimal daters, moderate daters, and frequent daters. Therefore, you need to recode the current 13 categories into 3. To make the three distinctions, it is best to begin by looking at a distribution of the variable. Type `tab datnum` into the Command window, and press **Enter**.

```
tab datnum
```

tab datnum (datnum _ w3) [IF HAS BEEN IN A ROMANTIC RELATIONS HIP OR HAS BEEN MARRIED J:4 How

	Freq.	Percent	Cum.
1	1	4.00	4.00
2	4	16.00	20.00
3	4	16.00	36.00
4	1	4.00	40.00
5	3	12.00	52.00
6	2	8.00	60.00
7	1	4.00	64.00
9	1	4.00	68.00
10	2	8.00	76.00
11	1	4.00	80.00
15	2	8.00	88.00
20	3	12.00	100.00
Total	25	100.00	

Based on these results, you could decide to classify anyone who has dated 3 or fewer people as minimal daters, 4 to 7 as moderate daters, and anything more than 7 as frequent daters. You could use the steps shown above to create a new variable and then replace its values using appropriate `if` statements. Although this method would produce the desired result, it would require four separate commands (one to create the new variable, and one for each of the three categories).

To simplify this process, you can use the `-recode-` command. The structure of the `-recode-` command is similar to all the ones you have been working with in this chapter. Start by typing the command name and the original variable whose values you are changing: `recode datnum`. Now you need to tell Stata which values to recode and what new values they should take. You could choose any three numbers to represent the three new categories, but it is generally easiest to use 1, 2, and 3 to code these types of ordinal variables. You will be asking Stata to recode any cases that are currently coded as having dated 1 through 3 people to now equal 1 (minimal daters), currently coded as having dated 4 through 7 people to now equal 2 (moderate daters), and anyone who has dated 7 through 20 people should be changed to equal 3 (frequent daters). The `-recode-` statement, therefore, requires that three separate conditions be specified.

The command you need to type in the Command window to accomplish this alteration is `recode datnum (1/3=1) (4/7=2) (8/20=3)`. Before you press **Enter**, take a moment to examine the command. The first new symbol you probably notice is the slash "/." This is the operator for "through," which tells Stata to change all values from and including the number listed before the "/" through and including the value listed after the "/." It might be confusing that the "/" operator is used both to symbolize the mathematical function of division and this logical function of "through." Fortunately, the `-recode-` command is virtually the only situation in which "/" is used to signify "through." In basically every other possible scenario, "/" is a symbol for division.

Next, you see that you only need to use a single equal sign. Here you are telling Stata to set the values of the cases on a particular variable to be equal to something else, which means you only need the one equal sign. Finally, if you look at the distribution of the `datnum` variable shown above, you might notice that there are no people who have dated 8 people. The final condition in the `-recode-` command tells Stata to recode all values from and including 8 through and including 20 to equal 3. It does not matter that none of the cases in the current data are actually coded as 8; there are also no cases coded as 13 or 18. Stata is simply looking at the range, and any case that has a value that falls in that range will be recoded.

Now you may be ready to press **Enter**, but pause for just another moment. Think about what you are asking Stata to do. Stata will be changing all the values of the `datnum` variable to be equal to 1, 2, or 3. There might be a situation later in your analyses where you want to know the exact number of people each case has dated. If you continue with the `-recode-` command as written, you would have no way to return to the original values (short of reopening the original data file).

You already know one way to protect against this danger. Just as with the `-replace-` command, you could create a new variable that is the exact copy of the `datnum` variable and then recode this duplicated version. This method would definitely work, but it would require an additional step. Fortunately, the `-recode-` command has a useful option to protect against this danger that does not require an additional command.

You can invoke the `-generate(newvar)-` option to automatically create a new variable that contains the recoded version of the original variable. This option is slightly different from the ones you have worked with before

in that it requires you to type not only the option but also something else in the parentheses. The "newvar" portion of the option denotes that you need to type a name for the new variable that will be created and hold the recoded values.

Putting everything together, keep what you have already typed and add the generate option into the Command window, `recode datnum (1/3=1) (4/7=2) (8/20=3), gen(datlevs)`, and press **Enter**. Now you have told Stata the values on `datnum` that need to be recoded, the values they should take to create a new variable to hold these recoded values, and the name of that new variable (`datlevs`). As always, it is useful to check to make sure the command recoded everything in the way you intended. Type `tab datnum datlevs` in the Command window, and press **Enter**. The following cross-tabulation is presented:

```
recode datnum (1/3=1) (4/7=2) (8/20=3), gen(datlevs)

tab datnum datlevs
```

(datnum _ w3) [IF HAS BEEN IN A ROMANTIC RELATIONSHIP OR HAS BEEN MARRIED J:4.	RECODE of datnum ((datnum _ w3) [IF HAS BEEN IN A ROMANTIC RELATIONSHIP OR HAS BEEN MARRIED]			
How	1	2	3	Total
1	1	0	0	1
2	4	0	0	4
3	4	0	0	4
4	0	1	0	1
5	0	3	0	3
6	0	2	0	2
7	0	1	0	1
9	0	0	1	1
10	0	0	2	2
11	0	0	1	1
15	0	0	2	2
20	0	0	3	3
Total	9	7	9	25

You can see from this table that the cases from the original `datnum` variable were recoded into the correct values on the new `datlevs` values. For example, the 1 case who reported having dated 1 person, the 4 who had dated 2 people, and the 4 who had dated 3 people are now all equal to 1 (minimal daters) on the new `datlevs` variable. Additionally, Stata automatically has added some

information to the datlevs variable, shown at the top of the table, to help you remember that it is the recoded version of the datnum variable.

As you work with quantitative data, whether it is a secondary data set or one that you have collected yourself, one of the most frequent alterations you will need and/or want to make is this type of recategorization. Surveys often ask for more precise responses than you may need. Therefore, being able to collapse multiple categories into broader groupings is one of the most vital skills in an analyst's repertoire. The -recode- command is the most effective command for making this type of change.

A CLOSER LOOK: MULTIPLE COMMANDS TO THE SAME ENDS

You may have noticed that the -replace- and -recode- commands seem similar and that you could use each to achieve the same goal. You will notice several other instances throughout this book in which multiple pathways, using various commands, lead to the same outcome. This equifinality may at first seem confusing and may even lead to some frustration by making it seem as if you have to remember when to use which command.

But viewed from another perspective, this feature is one of Stata's greatest strengths—its flexibility and adaptability. You do not need to worry about which combination of commands is the "right" way, as long as the combination you use produces the desired end. Some users may prefer, for a variety of reasons, to use a combination of -gen- and -replace- rather than -recode- with the -gen(newvar)- option, whereas others may even prefer to use a combination of -gen- and then -recode-. All three patterns would create a similar variable, so you could use whichever command combination seems most straightforward, comfortable, or even easiest to remember. In these types of situations, there is no one correct way to accomplish the task. Stata allows users to construct the pattern that works for them. So do not try to remember each and every possible method; rather, practice the one in which you have the most confidence. As long as the outcome is what you are looking for, then that is the "right" way.

NONESSENTIAL, EVERYDAY COMMANDS

The 5 commands that have been covered to this point are the most essential commands both in their utility and in their applicability to learning the foundation of Stata. There are, however, a handful of other commands that are frequently used, although perhaps not explicitly essential to the completion of a research project. These commands do not "fit" neatly into

any one substantive section, which is partly why they have been included in this somewhat miscellaneous subsection. Most of the commands are fairly straightforward and therefore are described relatively briefly. Still the details provided explain how and when to use them effectively.

rename

When you use secondary data sets, the variable names may not always be extremely informative. There are data sets that even name variables based on the survey question (e.g., q112), making it difficult to remember what information is stored in the variable.

Chapter 1 explained two ways to alter the name of variables using the point-and-click interface. The command to do so is similarly straightforward, making all three efficient alternatives. The command name is `-rename-` followed by the existing variable name and the new name you want it to be called.

For example, in the current data, it may be helpful to have the `gender` variable named `female` so that it is clear that the variable is an indicator of being female (i.e., female is coded as 1).

Type `rename gender female` into the Command window, and press **Enter**. The variable `female`, in place of `gender`, now appears in the Variables window.

drop/keep (if)

Although it is generally not necessary, there are times when you might want to eliminate a variable or particular cases from a data set. The command to accomplish both tasks is `-drop-`, but the structure to accomplish either task is slightly different.

To drop a variable or variables from the data set, simply enter the command followed by the names of the variables to be deleted. In the current data set, the `examp` and `examp2` variables were generated only as illustrations. Because you do not need them for your analyses, you could choose to eliminate them from the data set.

Type `drop examp examp2` in the Command window, and press **Enter**. The two variables are now eliminated from the data file and no longer appear in the Variables window.

When you need to eliminate particular cases, think about how you might ask your smart colleague to do so. For instance, you might realize that the one 23-year-old case was included in the data set by mistake and should be completely eliminated. You might ask your smart colleague to "drop cases if they are 23 years old." This verbal command is similar to the Stata command to drop cases. You must invoke an `-if-` statement immediately after the `-drop-` command.

Type `drop if agecats==23` in the Command window, and press Enter. Remember that the double equal sign is needed in the `-if-` clause because you are asking Stata to assess whether a statement is true. If you now produce a distribution table of the `agecats` variable by typing `tab agecats` into the Command window and pressing **Enter**, the table displays as follows:

```
drop if agecats==23

tab agecats
```

(agecats _ w3) Age variable collapsed into one year categories	Freq.	Percent	Cum.
19	9	37.50	37.50
20	6	25.00	62.50
21	4	16.67	79.17
22	5	20.83	100.00
Total	24	100.00	

The variable now only ranges from 19 to 22, and the total number of cases is 24 instead of 25.

The `-keep-` command is implemented exactly like the `-drop-` command but produces the opposite result of *retaining* only the cases or variables specified to keep in the command line. For example, you might decide that you only want to keep respondents who are younger than 21 years of age. Now instead of specifying whom to eliminate, using the `-if-` statement, you need to tell Stata whom to keep. Because you only want to keep cases who are younger than 21 years of age, you would type `keep if agecats<21` into the Command window, and press **Enter**. After doing so, typing `tab`

`agecats` into the Command window and pressing **Enter** displays the following table:

```
keep if agecats<21

tab agecats
```

(agecats_w3) Age variable collapsed into one year categories	Freq.	Percent	Cum.
19	9	60.00	60.00
20	6	40.00	100.00
Total	15	100.00	

The variable now only ranges from 19 to 20, and the total number of cases is 15 instead of 24.

One note of caution to keep in mind when deciding whether to drop (or keep) selected cases from a data set: It may seem appropriate to completely eliminate cases that do not pertain to particular analyses. For instance, you may only be interested in the number of friends among young adult females. If so, you might consider dropping all the males from the sample. There is no technical problem with doing so, but it is not the most effective approach. You could of course keep a backup data file that contains the full sample. But all the data manipulations you make on your restricted data would then need to be redone if your research interests change and you decide to include males in the analyses. It is generally a better strategy to use an –if– statement with your desired analyses. In the above example, if you wanted the cross-tabulation shown in this chapter to only include people who were younger than 23 years of age, you could type `tab employst gender if agecats<23`. This latter command would accomplish the same ends as if you dropped the 23-year-old respondent and then produced the cross-tabulation, but it would not permanently alter the data set.

describe

The –describe– command (shortened –desc–) can be used to present information about the entire data set or about particular variables. It can be

entered completely by itself, which will display detailed information about the data file and each variable included in the data set. The inclusion of the Properties window (starting with Stata 12) makes this a less needed command, but for users of Stata 11 or older, it is still very helpful.

Type `desc` into the Command window, and press **Enter** to see this full set of results. Invoking the `-short-` option provides only the information about the data. Type `desc, short` into the Command window, and press **Enter**. The following, more concise results are displayed:

```
desc, short

 Contains data from C:\Documents and Settings\
 klongest\My Documents\Stata\Data\Chapter 2\Chapter 2
 Data.dta
 obs:             15
 vars:            12
 size:            915
 Sorted by:
 Note: dataset has changed since last saved
```

The results indicate how many cases (`obs`) are in the date file, how many variables (`vars`) are included in the data, and the size of the data file.

If you want information about only particular variables, you can enter their names after the `-desc-` command. Type `desc female datlevs` into the Command window, and press **Enter**. The information shown below is presented:

```
desc female datlevs

Variable  Name   type    Format  label  variable label
------------------------------------------------------------------------------
female    Byte   %12.0f  Gender  (gender _ w3)Respondent
                                 gender
datlevs   Int    %9.0g   RECODE of datnum ((datnum _
                         w3) [IF HAS BEEN IN A
                         ROMANTIC RELATIONSHIP OR HAS
                         BEEN MARRIED]
```

Several aspects of the variables are shown, including the type and format (both discussed in Chapter 1). Additionally, this display lists the value label

(label) attached to the variable and its variable label. Both of these aspects of variables are discussed in the Data Management: Working With Labels section of Chapter 3. But for now it is useful to know that the -desc- is a helpful command to obtain a quick summary of all this information.

display

One interesting side utility of the Command window is that it can serve as a calculator. When you execute the -display- (shortened -disp- or even -di-) command followed by a mathematical formula, Stata will display the answer in the Results window.

For instance, if you wanted to double-check the percentage of males who are employed, displayed in the employst by gender cross-tabulation, you can type di (2/9)*100 in the Command window, and press **Enter**. The correct result, 22.22, appears in the Results window.

The -di- command follows a standard mathematical order of operations and uses the same symbols for operators discussed in the "A Closer Look: Mathematical Operators and Their Symbols" box. Whenever you enter numbers that are four digits or more, the comma(s) should not be typed. If you wanted to know the square root of 50,345, you would type di sqrt(50345) into the Command window, *not* (50,345).

Although it may not be the most vital command, the -di- command is nonetheless extremely useful and just one more positive feature of Stata.

set more off

When you run a command that produces lengthy results (e.g., a frequency distribution of a variable with numerous categories), the Results window is forced to scroll. Stata, however, by default prevents the results from scrolling past one page view. When this situation arises, Stata pauses the display of the results and displays the word more at the bottom of the screen. Essentially, Stata is asking you whether you would like to see "more" of the results. To see the rest of the results, you can either click on the displayed more with your mouse, or you can press any button on your keyboard. At first, this default operation may seem a bit frustrating, but Stata is actually saving you the work of scrolling back up to see the first set of results by giving you a chance to review them before continuing.

If you prefer to have Stata simply show all the results without pausing, you can enter the command -set more off- in the Command window, and press **Enter**. Furthermore, if you would like to change the default setting so that Stata never pauses while displaying the results, you can invoke the -permanently- option along with the -set more off- command. If you would ever like to revert to the original default and have Stata pause when displaying lengthy results, you can invoke the -set more on- command.

Summary of Commands Used in This Chapter

5 Essential Commands

tab

```
tab employst
tab employst, sort
tab employst gender
```

sum

```
sum datnum
sum datnum agecats
sum datnum, detail
```

generate

```
gen examp=200
tab examp
gen examp2=21-18
tab examp2
gen agep18=agecats-18
tab agep18
tab agecats agep18
gen totsocial=datnum+
  numfrien
browse datnum numfrien
  totsocial
```

replace if

```
tab agecats
replace agecats=19 if
  agecats==18
tab agecats
gen agecatsrp=agecats
tab agecats agecatsrp
replace agecatsrp=22 if
  agecats==23
tab agecats agecatsrp
gen isol=0
replace isol=1 if datnum<=2
  & numfrien<=2
tab datnum numfrien if
  isol==1
```

recode

```
tab datnum
recode datnum (1/3=1)(4/7=2)
  (8/20=3), gen(datlevs)
```

Nonessential, Everyday Commands

rename

```
rename gender female
```

drop

```
drop examp examp2
drop if agecats==23
tab agecats
keep if agecats<21
tab agecats
```

desc

```
desc
desc, short
desc female datlevs
```

display

```
di (2/9)*100
di sqrt(50345)
```

set more off

```
set more off
set more off, permanently
set more on
```

A Closer Look: The Dreaded Error Message

```
replace agecats=19 if
  agecats=18
```

Exercises

Use the original `Chapter 2 Data.dta` for the following problems.

1. Produce a distribution table for the `numfrien` variable.
2. Produce a distribution table for the `numfrien` variable that is sorted by frequency.
3. Create a cross-tabulation of `numfrien` and `agecats`.
4. Reproduce the cross-tabulation of `numfrien` and `agecats` so that the percentages of each age category are displayed.
5. Generate a new variable that is the the number of people dated (`datnum`) per year since the respondent turned 16 (i.e., it is the division of total dating partners by years past age 16).
6. Generate a new variable that replaces all respondents who report dating (`datnum`) 20 people to be equal to 15.
7. Generate a new variable that is an indicator of people who are 20 years old (`agecats`) and have dated (`datnum`) between 5 and 10 people.
8. Recode, and create a new variable containing the recoded values, the `agecats` variable into a two-category variable representing the respondents being younger than 21 years old versus being 21 years of age or older.
9. Rename the `numfrien` variable to be called `frndnum`.
10. Using the Command window, calculate the product of 976 and 543.