Using SPSS: A Little Syntax Guide

Andrew F. Hayes, Ph.D.
www.afhayes.com

There are many general-use statistics programs on the market today, including SPSS (also known as "IBM Statistics"), SAS, STATA, Minitab, JMP, and others. There are also some good ones that are freely available. An example is R, which is very powerful but can be quite daunting to the newcomer. Each program has its strengths and weaknesses relative to others, in terms of capabilities, flexibility, ease of use, widespread adoption, and price.

But SPSS remains one of the dominant programs in the social sciences and business fields, probably for historical reasons in part (many educators learned it and so continue to teach it to the next generation), but also because it has a very friendly "point-and-click" user interface that makes it simple to use. But underneath this user interface is a very powerful and versatile programming language that many don't know about. This programming language built into SPSS allows the user to speak to SPSS in "syntax" or "code" rather than a series of points and clicks with the graphical user interface (GUI). Like learning any language for the first time, learning to talk to SPSS with syntax takes some patience and practice, and so many who do know about it opt not to use it because doing so isn't as easy, at least not at first, as just pointing-and-clicking one's way through analytical life.

This document was written to start you on what will be a life-long journey as you build SPSS syntax programming skills over time. This *Little Syntax Guide* was *not* written with the intention of teaching you much about data analysis, although you may learn something about that along the way. It is assumed that you are using this because you want to learn about writing and using SPSS syntax, not about data analysis. Perhaps you are doing so in the context of a course you are taking with a textbook that offers formal instruction on the theory and practice of data analysis. If your only goal reading this is to learn about how to analyze data, you should consult a different book than this one, listen carefully to your professor, attend class regularly, and so forth. This book won't suit your needs. And it should now go without saying that by reading this, you won't learn how to conduct various analyses through a series of click-here-drag-there instructions like you will find in some books written about SPSS. If you want to learn about how to use SPSS with the GUI, stop reading now and look elsewhere for advice. With the exception of a section on data entry, this book does not provide such instruction.

This *Little Syntax Guide* also was not written as a substitute for the more comprehensive *Command Syntax Reference* available through the SPSS help menu. The reference manual contains more information that you would ever need to know about SPSS syntax, the capabilities of SPSS, and how to get it to generate output for various procedures. But at over 2000 pages, the *Command Syntax Reference* can be a tad overwhelming. In this *Little Syntax Guide,* I provide only some basic guidance on SPSS syntax for different data management and analysis problems that are commonly confronted by students taking their first statistics course or two. Use it as a springboard for diving deeper into learning about writing SPSS syntax through other sources.

Why Learn Syntax?

Once you start writing SPSS syntax, its advantages and benefits will become apparent, and when you become comfortable doing it, you'll probably never go back to exclusively point-and-click data analysis. But I have found it necessary with many skeptical students and others new to data analysis to make the argument why it is necessary to inflict such pain—and painful it is at first for some—when pointing and clicking seems so much easier. I have also been asked by colleagues just what I tell my own students, as they often find themselves training those who arrive at their labs eager to get involved but knowing little about how to speak to a piece of data analysis software without the aid of a heavily-exercised index finger and mouse. So before beginning, I lay out eleven reasons why learning syntax is a good thing to do, worth the effort, and even necessary for a credible, replicable science.

First, syntax functions as a natural documentation for your thinking as you work with a data analysis problem. There is no better way of remembering the details of an analysis you conducted than looking at the code you wrote that conducted the analysis in the first place. It is not uncommon for considerable time to pass between when an analysis is conducted, written up for publication or some other form of reporting, and reviews from the peer review process come back or new data become available. Remembering accurately how you constructed a variable, which cases you deleted from an analysis and why, or the exact variables perhaps of many in the data that you used as predictors and outcome, or independent and dependent, can be difficult given such time lags and nearly impossible if your only reminder is your imperfect memory of what you clicked here and dragged there and in what sequence.

Second, with a program containing the syntax for an analysis, it is much easier to replicate an analysis you did earlier when new data are added, or to tweak the analysis if you want to modify it in some fashion. For instance, you might want to repeat an analysis with a new dependent variable or a new covariate. To do so, you simply make the modification and rerun the program you wrote rather than redoing all the pointing-and-clicking in the original sequence (assuming you can even remember what you did earlier and in what sequence).

Third, use of syntax makes it easier for you (or someone you consult) to troubleshoot when something goes wrong or an analysis generates errors. A complete analysis often involves substantial data modification or transformation, construction of new variables, selective processing of certain cases, and various other things each of which can produce errors when the data are ultimately subjected to analysis. Without some kind of record of the many steps involved—a record provided by your syntax—it can be difficult to figure out what step in the sequence is producing the problem. When you have the syntax used throughout the procedure, it is much easier for you to debug the flaws in the logic of your reasoning or thinking that is producing the error. And without your syntax, it will be next to impossible for anyone you consult for assistance to help you figure out the source of your problem.

Fourth, learning about and writing syntax facilitates collaboration with others. Data analysis is sometimes a group activity, and large scale research projects require communicating with many different people over time. It is much easier to share and pass around a program containing the code for an analysis that a collaborator can edit and run than a list of point-and-click instructions that is difficult to describe and follow. People you work with, especially someone who controls the money in a research project, likely will expect you to be able to communicate with him or her about what you are doing by providing the code you have written for various analyses you have conducted related to the project. And there is a good chance your collaborators will already know something about writing SPSS code and will expect you to have this skill. So at a minimum, if you don't know how to do it, you will find yourself a little embarrassed. At worst, lack of this knowledge will render you unqualified as a collaborator.

Fifth, some features of SPSS can only be accessed through command syntax. Although the point-and-click menu-driven system is convenient, not everything that SPSS can do has a corresponding menu or dialog box. As one example, consider the menu for constructing a matrix of correlations between a set of variables. If you were to choose 10 variables using the point-and-click menu, the only output option is the production of a 10 X 10 matrix of correlation coefficients. This matrix will contain 100 correlations, with the correlation between variable *i* and *j* provided twice, once above the diagonal of the matrix and once below, as well as 10 correlations between a variable and itself. This is a lot of redundancy in a large matrix that may not even fit on your screen or a piece of paper when printed. Furthermore, you may not even care about most of these correlations. Suppose you only care about the correlation between eight of these variables with the other two, and thus only 16 correlations are of interest to you. As will be seen later, using syntax, you can ask SPSS to generate only the 8 X 2 matrix of correlations you care about. This is not possible using the point-and-click menu system. There are many other examples of options and analyses available in SPSS that can be accessed only through command syntax.

Sixth, learning how to communicate with your statistics program using syntax helps you develop a skill that is generalizable to other statistical packages and programming languages. When I recruit a student to my lab, one of the first questions I ask is whether he or she knows any programming languages. It doesn't matter that much to me what languages the person knows. What is important is whether the person has some experience communicating with a computer in this fashion. If so, I know I will be able to teach this person the things he or she needs to know to work in my lab. Once you have acquired a programming skill and a programming mindset, it is a lot easier to pick up new languages, even those that are quite different than the ones you already know.

Seventh, as you pursue additional coursework in statistics, you will find that most instructors and the books they use will assume that you already have some background in writing syntax in one or more statistical programming languages. So learning how to write code early in your education will help you to advance further and at a more rapid pace than if you wait until you get into those more advanced classes, where you will be behind others in the class who have already acquired this skill.

Eighth, sharing code is increasingly becoming an expectation when publishing. The open science movement and concerns about the "replication crisis" facing some areas has resulted in a push for greater transparency in the way science is conducted. Increasingly, journals and funding agencies are requiring investigators to make their data available as well as the code used to manage the data and conduct various analyses reported in white papers and published articles. It is likely that in time, you will have to provide syntax to editors, public archives, and funding agencies as a condition of the receipt of money and publishing your findings.

Ninth, one day you may end up teaching a data analysis course using SPSS. It is far easier to teach the use of SPSS by providing syntax that accomplishes a particular analysis than it is to do so using set of point-here, click-there, drag-this set of instructions in visual or text form. And whereas SPSS syntax tends to be pretty stable over time from version to version (a new version of SPSS seems to come out about once per year), the appearance of menus and where things are located is less stable. You will find yourself having to update teaching materials far more often when you provide screen shots of menus and dialog boxes than you will if you provide syntax. And a few lines of syntax take much less space on a piece of paper, on a PowerPoint slide, or a book than a bunch of pictures of menus, dialog boxes, sub-dialog boxes, and so forth. You will be able to convey more information in less space when your teaching materials are based on SPSS syntax. This is one of the reasons (of several) I focus on SPSS syntax in two of my books on data analysis (*Regression Analysis and Linear Models*; and *Introduction to Mediation, Moderation, and Conditional Process Analysis;* both available through www.guilford.com).

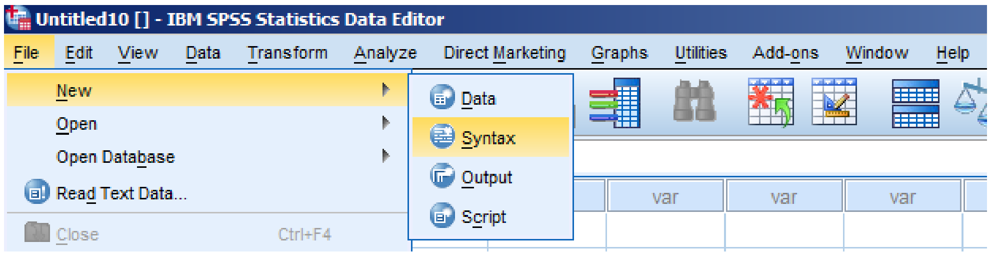
Tenth, knowing SPSS syntax is a marketable skill. Almost any undergraduate or graduate student who has taken a data analysis course can report experience using a programs such as SPSS. Such skills listed on a resume do little to distinguish applicants for a job from each other and do not offer much competitive advantage when attempting to secure employment. Far fewer people right out of college or graduate school can truthfully claim (in writing and during interviews) that they can start the job ready to tackle some of the more complicated data management and programming tasks that characterize the needs of commercial industries and government by writing code. Pointing-and-clicking generally does not get you very far when dealing the kinds of data and research problems that characterize the nonacademic world. Though many companies and government offices leave their more complex programming tasks to dedicated computer programmers, being able to work closely with such programmers and communicating with them in code is a job skill that will distinguish you from other applicants. The chances are good that your employer will not rely on a GUI-based statistics language for many of its statistical and data management tasks, so having experience writing in code in any statistical package makes it easier to make the leap to the world of statistical and database programming outside of the academic world using languages that *require* code for operation.

Eleventh, last but not least, and as crazy as it sounds, there are people who believe that your software preference and how you interact with your preferred program is diagnostic of your intelligence and sophistication. SPSS users are perceived by some as soft scientists who don't really know what they are doing and are somehow lesser scientists as a result of their preference. These people believe *real* scientists write code. Why some feel a need to berate SPSS users because of their tendency to use (or rely on) the point-and-click interface that has made it so popular is beyond me. All software packages have their strengths and weaknesses, and people make their choice of preferred software program for a variety of reasons. But I can't deny that you will encounter people like this from time to time. If you come into a lab or research group, begin a collaboration with someone, meet with a data consultant, or otherwise approach someone in the scientific community with a question related to analysis and don't have code to show them or the skill developed to work with software in this way, don't be surprised if you are met with a raised eyebrow or a condescending look down the nose once you identify yourself as an SPSS user. You can fight the stereotype by being a counterexample and by educating such people about the power found under the hood of the point-and-click interface.

Data Entry

Before you can do any kind of data manipulation or analysis using SPSS, you have to have some data and get it into SPSS in the form of a "data file." Data can come in many formats, but data files generally don't create themselves. Fortunately, there are many public archives that store data in SPSS format, such as the Pew Center for the People and the Press, which has an excellent repository of data files from their regular polls of the public (http://people-press.org/category/datasets/). However, someone created those files, and if you are the investigator conducting a study, most likely the person that ends up creating your files will be *you*. Here I describe the basics of data entry using SPSS command syntax. But for completeness, I also talk about using the point-and-click GUI (Graphical User Interface) for data entry. Following this section, rarely do I make reference to speaking to SPSS using the user interface.

The first thing you must do before writing a program, whether to enter data or conduct an analysis, is to tell SPSS you want to create a new syntax file. Select "File"→ "New"→"Syntax" from within SPSS, as below



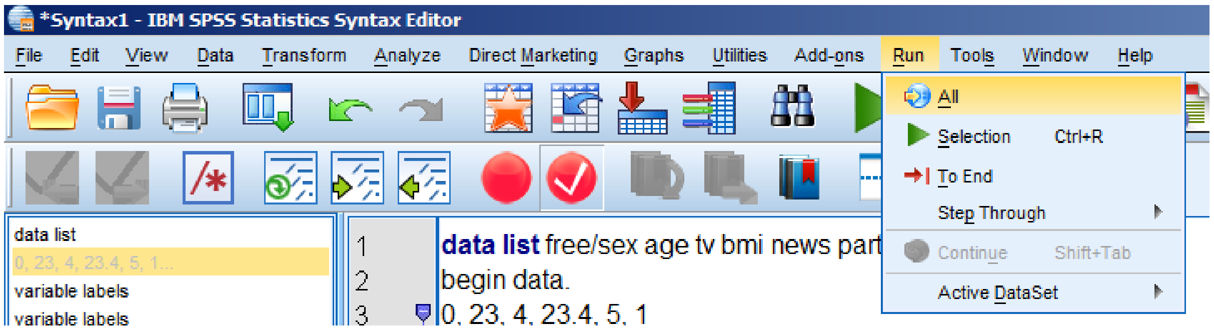
A blank window will pop open that looks like an empty word processing document. In this window you will type a program to tell SPSS about your data and what you want to do with it.

The program on the next page reads a small data set into SPSS formatted in FREE field format with 6 variables and 8 records or “cases” (rows), saves it to a file named “example.sav” in an existing folder called “data” on the “c:” storage device, opens the file, assigns labels to variables, assigns values to codes of categorical variables, and defines missing values. The save command is written using Windows conventions for referring to file locations and storage devices in a program, and the .sav extension on the file name is used by the computer as well as SPSS to identify the file as an SPSS-formatted data file. Users of MacOS would use a Mac convention for referring to file locations, as discussed later.

```
data list free/sex age tv bmi news partyid.
 begin data.
 0, 23, 4, 23.4, 5, 1
 0, 18, 3, 27.6, 3, 1
 1, -99, 1, 32.4, 0, 3
 0, 20, 4, 31.4, 2, 2
 1, 27, 4, 23.4, 5, 1
 1, 20, 3, 27.6, 3, 3
 0, 43, 1, 32.4, -99, 1
 1, 34, 2, 31.4, 2, 1
 end data.
 save outfile = 'c:\data\example.sav'.
 get file = 'c:\data\example.sav'.
 variable labels sex 'SEX: Sex of participant'.
 variable labels age 'AGE: Age of participant'.
 variable labels tv  'TV: Hours of TV in a typical day'.
 variable labels bmi 'BMI: Body Mass Index'.
 variable labels news 'NEWS: News use hours per day'.
 variable labels partyid 'PARTYID: Political party self-identification'.
 value labels sex 0 'female' 1 'male'.
 value labels partyid 1 'Democrat' 2 'Republican' 3 'Neither'.
 missing values all (-99).
```

SPSS doesn’t care about type case. Whether you list commands in UPPERCASE, lowercase, or a MixTuRe doesn’t matter. However, all commands must end with a period. One of your biggest frustrations when learning SPSS syntax will be the result of forgetting to terminate a command with a period.

To execute the program, choose “Run”→”All” with the syntax window open:

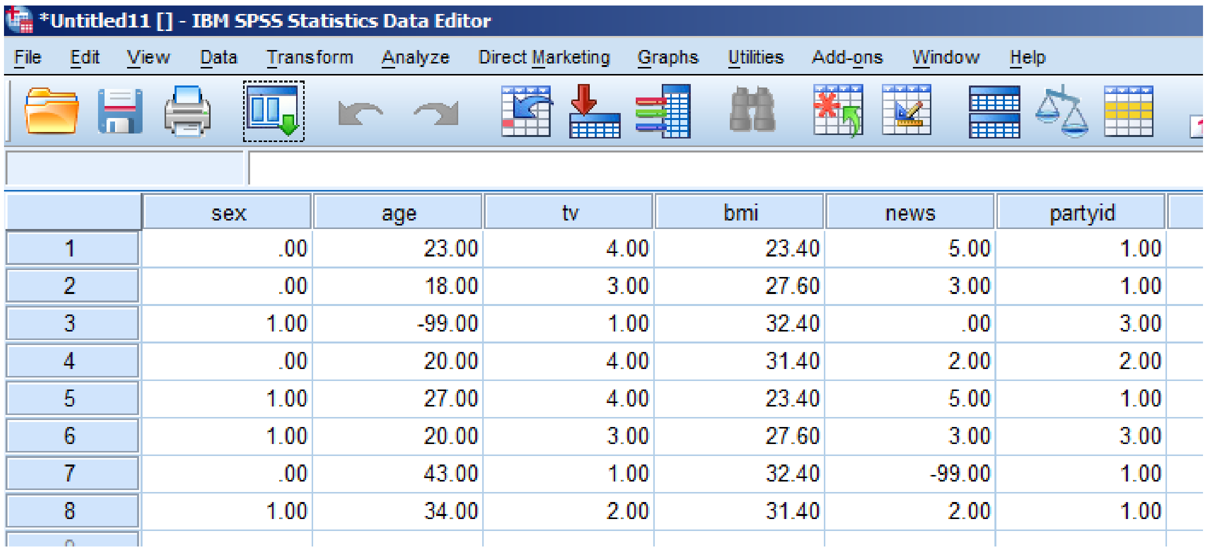

*Syntax1 - IBM SPSS Statistics Syntax Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Run Tools Window Help
All
Selection Ctrl+R
To End
Step Through
Continue Shift+Tab
Active DataSet

data list
0, 23, 4, 23.4, 5, 1...
variable labels
variable labels

```
data list free/sex age tv bmi news part
begin data.
0, 23, 4, 23.4, 5, 1
```

Alternatively, you could simply select all the text in the syntax window by clicking and dragging the mouse across it, and then clicking the "play" button on the icons bar or choose "Run"→"Selection".

If you have successfully executed the commands above, you should be able to see the data in SPSS in a data window. When you open SPSS, the data window is blank. But now that you have created a data file, you should see something that looks like this:


*Untitled11 [] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help

	sex	age	tv	bmi	news	partyid
1	.00	23.00	4.00	23.40	5.00	1.00
2	.00	18.00	3.00	27.60	3.00	1.00
3	1.00	-99.00	1.00	32.40	.00	3.00
4	.00	20.00	4.00	31.40	2.00	2.00
5	1.00	27.00	4.00	23.40	5.00	1.00
6	1.00	20.00	3.00	27.60	3.00	3.00
7	.00	43.00	1.00	32.40	-99.00	1.00
8	1.00	34.00	2.00	31.40	2.00	1.00

Now a bit about these commands.

DATA LIST

The list of variable names in the DATA LIST command tell SPSS that these will be the names of the variables. As there are six names listed, SPSS will know there are six variables in your data set. A variable name is best kept to 8 characters at most, although SPSS will allow you to have longer variable names if you want. But they generally must contain only letters or numbers, although there are a few exceptions. For example, if you feel you want to put a space in a variable name, use the "_" character. You can also use @. Either of these is acceptable to SPSS. Although you can put numbers in a variable name, not as the first character. You can't use some characters such as %, #, & in a variable name. Best to stick to letters and numbers and the occasional underscore ("_") when possible.

SPSS will read through the data between BEGIN DATA and END DATA like you read a book, from top left to bottom right, row by row. When it starts at the first data point in row 1, it considers this a new "case" (e.g., a participant in a study, a respondent to a questionnaire, and so forth). It assigns data points to cases and variables as it reads left to right. So the first value gets assigned to case #1, variable "sex," the second value gets assigned to case #1, variable "age," the third value gets assigned to case #1, variable **"**tv," and so forth. When it runs out of variable names as it is reading, it assumes that the next data point gets assigned to the next case, and it starts assigning to variables at the beginning of the variable list. SPSS continues reading until it runs out of data. Note that if SPSS runs out of data before it reaches the end of the variable list in a cycle, SPSS will give you an error. If this occurs, you have a data entry problem. For example, you may have forgotten to enter a value for one or more variables for one or more cases in your data set.

The data in the gray box above are in "comma delimited" format, meaning that each data point is separated by a comma. A similar data set in "space delimited" format would look like

```
0 23 4 23.4 5 1
0 18 3 27.6 3 1
1 -99 1 32.4 0 3
0 20 4 31.4 2 2
1 27 4 23.4 5 1
1 20 3 27.6 3 3
0 43 1 32.4 -99 1
1 34 2 31.4 2 1
```

SPSS would not care which you used. It would read data in either format happily and you don't have to tell it what you are doing. In fact, you can even mix formats if you want, as in

```
0 23, 4 23.4 5 1
0 18 3 27.6 3 1
1 -99 1, 32.4, 0 3
0 20 4 31.4 2, 2
1, 27 4 23.4, 5 1
1 20 3, 27.6 3 3
0 43 1, 32.4 -99 1
1 34 2 31.4, 2, 1
```

SPSS doesn't care. It essentially treats commas and spaces the same when it reads the data (except when used to "skip" when reading a missing value, as below). In fact, SPSS doesn't even care if you make the data look pretty, in rough row and column format. You could also just list a big string of numbers, as in

```
0, 23, 4, 23.4, 5, 1, 0, 18, 3, 27.6, 3, 1, 1, -99, 1, 32.4, 0, 3, 0, 20, 4,
31.4, 2, 2, 1, 27, 4, 23.4, 5, 1, 1, 20, 3, 27.6, 3, 3, 0, 43, 1, 32.4, -99,
1, 1, 34, 2, 31.4, 2, 1
```

Obviously, this is a bit harder for people to read, but SPSS can read it just fine. With six variables in the DATA LIST command, it will assign the first six values to case #1, the next six to case #2, and so forth, in the order the variables are listed in the DATA LIST command.

You may find when attempting to read data that you get some kind of error or warning. Don't ignore errors SPSS gives you when reading data. If you get an error message, most likely SPSS hasn't read the data correctly, and anything you do with it could, as a result, be incorrect.

MISSING VALUES

In this data file, I have used -99 to represent a "missing value," meaning that for that case, data on that variable were not available. When using a numerical value to represent a missing data point, **it is important to tell SPSS so by using the MISSING VALUES command**. The MISSING VALUES command above tells SPSS that -99 is the missing value code for ALL variables in the data file. SPSS will not see -99 as a number, but rather as an absence of data. So the -99 won't be used in any statistical computations when it occurs in a variable. You could also use a period "." to denote a missing value, as in

```
0 23 4 23.4 5 1
0 18 3 27.6 3 1
1 . 1 32.4 0 3
0 20 4 31.4 2 2
1 27 4 23.4 5 1
1 20 3 27.6 3 3
0 43 1 32.4 . 1
1 34 2 31.4 2 1
```

Or, using comma delimited entry, a second comma to "skip" that data point, as in

```
0, 23, 4, 23.4, 5, 1
0, 18, 3, 27.6, 3, 1
1,, 1, 32.4, 0, 3
0, 20, 4, 31.4, 2, 2
1, 27, 4, 23.4, 5, 1
1, 20, 3, 27.6, 3, 3
0, 43, 1, 32.4,, 1
1, 34, 2, 31.4, 2, 1
```

There is no need to tell SPSS that the resulting blank cells in the data file represent missing data. Thus, the use of "." or a comma-skip is sometimes a safer way to denote missing data, because there is no way this could be mistaken for real data, as opposed to -99, which might be seen as real data in an analysis if you forget to tell SPSS (or other people you give the data to) that -99 isn't real data. If you wanted assign -99 as the missing value code for only some of the variables, list those variables by name in the MISSING VALUES command, as in

```
missing values age news (-99).
```

You could even have different missing values for different variables. For example, if you had used -98 for age but -99 for news as missing values, try

```
missing values age (-98) news (-99).
```

VARIABLE LABELS

The VARIABLE LABELS command is fairly self-explanatory. This command is used to assign a descriptive label to your variable. You will refer to variables in your SPSS programs by their names, not their labels. The labels appear in the output when you do something with a variable, and also in a part of the data file which you can see by selecting "Variable View" once an SPSS data file has been constructed. These labels provide you or others who look at your data information about just what the variable names refer to in terms of the things you are measuring. I have adopted the habit of listing the name of the variable in the data file in the label itself (as in this example), so that I won't have to remember the variable name in the file if I am programming while looking at some SPSS output.

VALUE LABELS

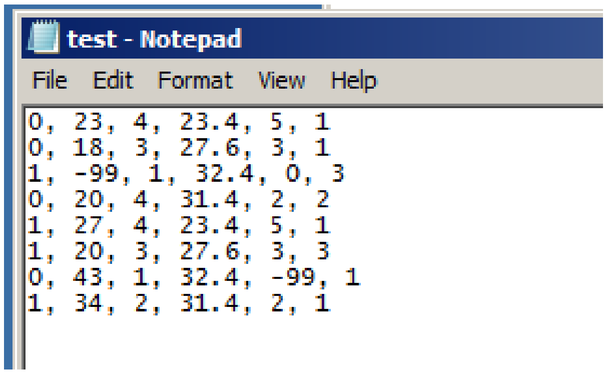
Although SPSS can read "string" data, meaning data that include letters rather than just numbers, it is best to avoid the use of string data when possible because many statistical procedures you are likely to use don't allow string-formatted variables. Rather, better to use numerical codes and then assign a label to let you know what those codes mean. For example, rather than using M for males and F for females, I have used the numerical codes 0 and 1 in the SPSS program above, and then assigned a value label for my own information and others who might see the data. This is done with the VALUE LABELS command. For instance,

```
value labels sex 0 'female' 1 'male'.
```

tells SPSS that for variable "sex," a 0 is a male and 1 is a female. SPSS doesn't care what these labels are. They are more for you so that you will remember what the codes mean. These labels will appear in output when appropriate.

Reading Data from a Storage Device

In this example above, the data were a part of the SPSS program, sandwiched between the BEGIN DATA and END DATA sections of the program. Sometimes you may find you want to read data that is stored in a file rather than embedded in the syntax program itself. This is easy enough to do. Suppose, for example, that the data we read above instead resided as a file named test.txt on the G: storage medium you are using in a folder named DATA. That file might look like this:



```
0, 23, 4, 23.4, 5, 1
0, 18, 3, 27.6, 3, 1
1, -99, 1, 32.4, 0, 3
0, 20, 4, 31.4, 2, 2
1, 27, 4, 23.4, 5, 1
1, 20, 3, 27.6, 3, 3
0, 43, 1, 32.4, -99, 1
1, 34, 2, 31.4, 2, 1
```

This file could be read into SPSS using the DATA LIST command, as such

```
data list file = 'g:\data\test.txt' free/sex age tv bmi news partyid.
```

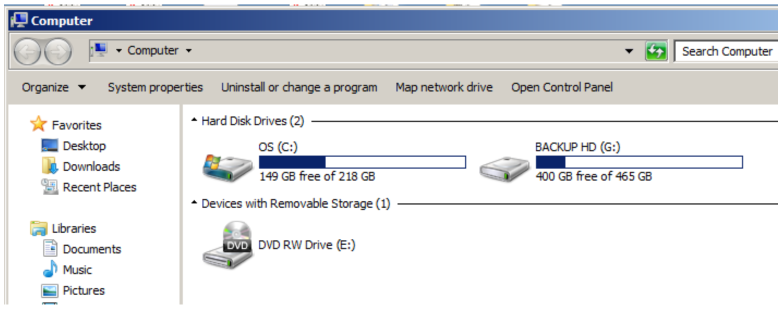
Variable and value labels, missing values, and so forth could then be assigned as described above. There would be no need for BEGIN DATA and END DATA commands because you aren't provided the data in the program itself. MacOS users would use Mac conventions for referring to file locations, such as `'MyHD\data\test.txt'` for a file on the "MyHD" drive in a folder named "data".

Saving SPSS Format Data to a Storage Device

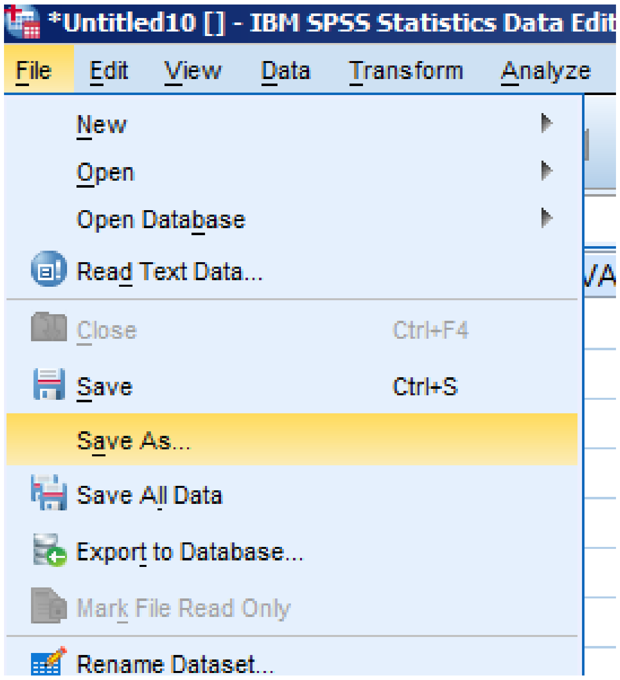
Once you have data entered, you will probably want to save it for safe keeping, use later, and so forth. To save your data in the format of an SPSS data file to an external storage device such as a USB drive you have two options. The first option is to write the command to tell SPSS to save the data file, and where to save it to. For instance, by executing the command

```
save outfile = 'g:\example.sav'.
```

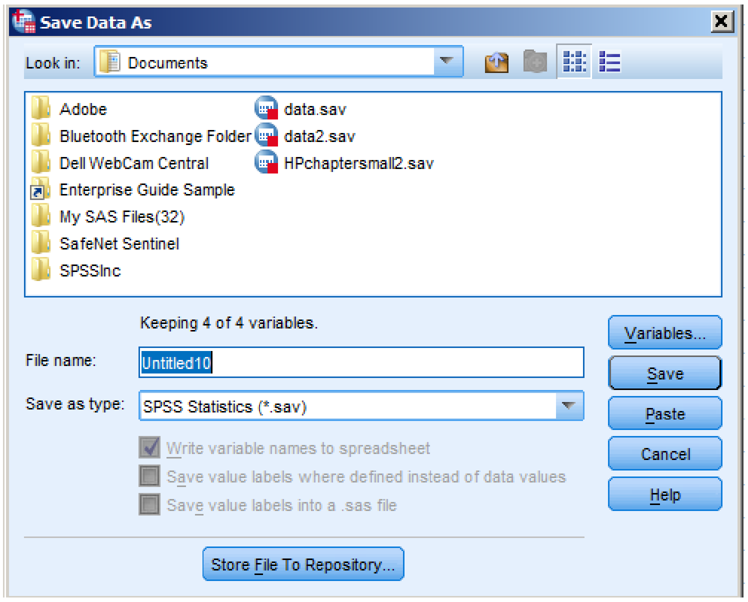
SPSS will save the data file to a storage device labeled "g:" on your machine. As can be seen below, "g:" on my computer is the backup hard disk:



Mac users should use MacOS conventions for specifying the file path, such `'MyHD\data\example.sav'` Alternatively, once the data file has been created, you could save the data using the SPSS GUI. To do this, click on the data spreadsheet so that it is the one you see on your monitor, and then select "File"→ "Save As..."



If this is the first time you are saving this file after executing SPSS, you can also select the "Save" option. If the file doesn't currently have a name or you chose "Save as..." you will see a window pop up, as below.


Save Data As
Look in: Documents
Adobe
Bluetooth Exchange Folder
Dell WebCam Central
Enterprise Guide Sample
My SAS Files(32)
SafeNet Sentinel
SPSSInc
data.sav
data2.sav
HPchaptersmall2.sav
Keeping 4 of 4 variables.
File name: Untitled10
Save as type: SPSS Statistics (*.sav)
Write variable names to spreadsheet
Save value labels where defined instead of data values
Save value labels into a .sas file
Variables...
Save
Paste
Cancel
Help
Store File To Repository...

Navigate using standard Windows or MacOS procedures to find the storage device and location you'd like to save the file. Type the file name in the "File name:" box and click the "OK" button. When completed, the window will disappear and you should see the data window again.

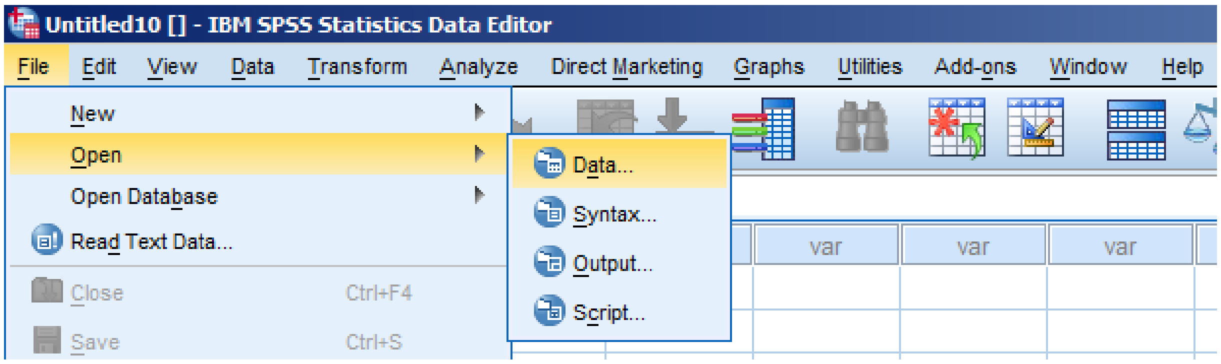
Note that SPSS data files will be created with (and should have in order to load them) a ".sav" extension. An extension is a convention computer operating systems use to assign icons to files and to tells the computer which programs should open which types of files. If you don't put the extension on the file name, SPSS will do it for you automatically, depending on the type of file you are saving. Of course, if you choose not to save the data file, no big deal really. You can just recreate it by reexecuting the program that constructed it in the first place, assuming you have saved the syntax file (see below).

Loading an SPSS Formatted Data File

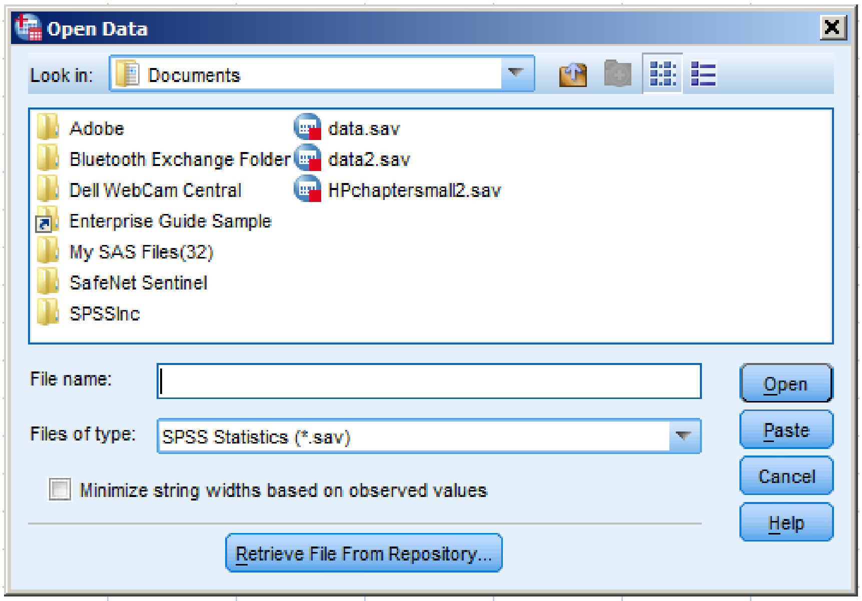
If you already have data in SPSS format saved to a USB drive or on the hard disk of your machine, you will want to load it into SPSS in order to do any work with the file. You can do this using syntax, or using the GUI. For example, executing the command

```
get file = 'g:\example.sav'.
```

will open an SPSS data file called "example.sav" stored on the G: device. This example is based on Windows conventions for file paths. Mac users should use MacOS conventions such as `'MyHD\data\example.sav'`. Using the GUI instead, select "Data…" which is an option under "File"→"Open", as below.



When you select "Data..." the window below should pop open. Navigate to the location where file resides and then click on it in the window and then click "Open". Once the file is opened, you should be taken back to the SPSS data spreadsheet window.



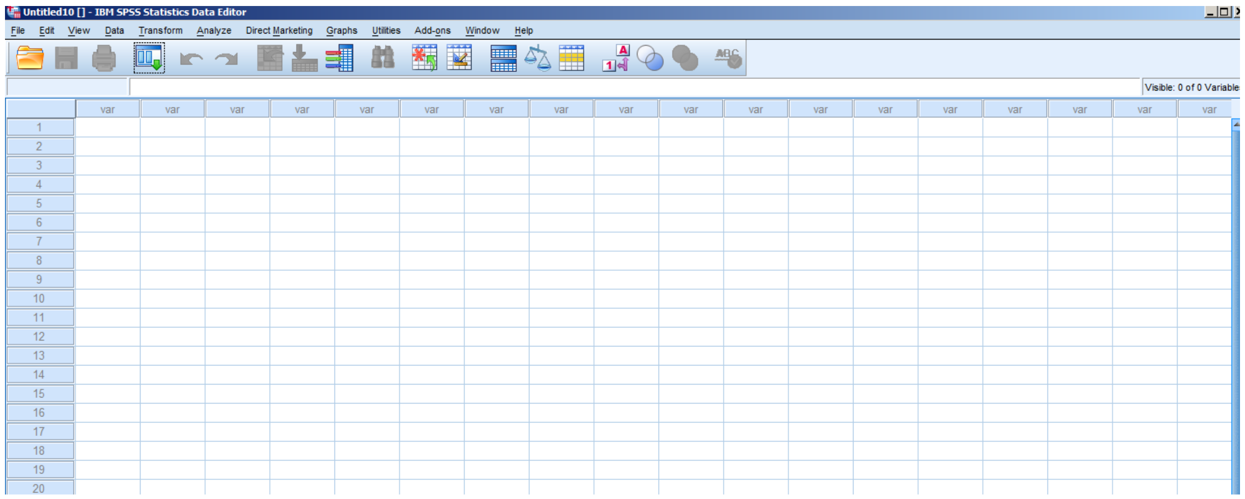
Saving and Loading an SPSS Syntax File

One thing that you can't do using SPSS syntax is save a syntax file. This must be done using the SPSS GUI. Choose "File"→"Save" or "Save As.." when the syntax window is open to save a syntax program you have written. Note that SPSS syntax files will be saved with a ".sps" extension. Or if you want to open an existing syntax file, chose "File"→ "Open"→ "Syntax".

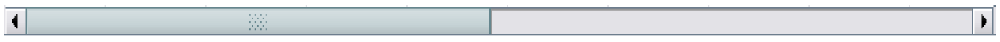
Entering Data in SPSS using the Menus

Some people find it easier to enter data for the first time using the menus. Once you learn SPSS syntax, you probably won't care which approach you use. In this section, I temporarily suspend lessons about the use of syntax by focusing on data entry using the data spreadsheet and pointing-and-clicking.

If you have not already done so, execute SPSS. Once running, you will see a window that is essentially a spreadsheet where you enter data into SPSS. Whenever you execute SPSS, the new data window will open and be ready for input. The data window looks like this:

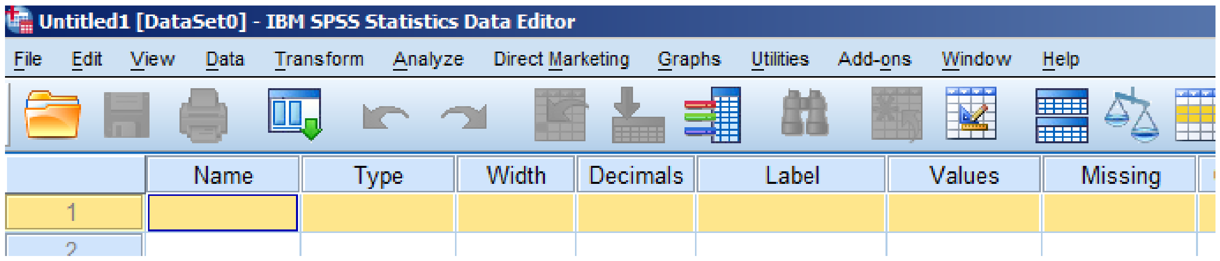

Untitled10 [] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help
Visible: 0 of 0 Variable
var

A data set can be quite large, but the data window can only be so large. Thus, the window will only show you part of the whole data set if you are working with a large file. You can move around inside larger data files using the arrow keys on the keyboard, or by “scrolling” the window around the data file using the arrow keys on the window itself. Just click the arrow to move the window in that particular direction:

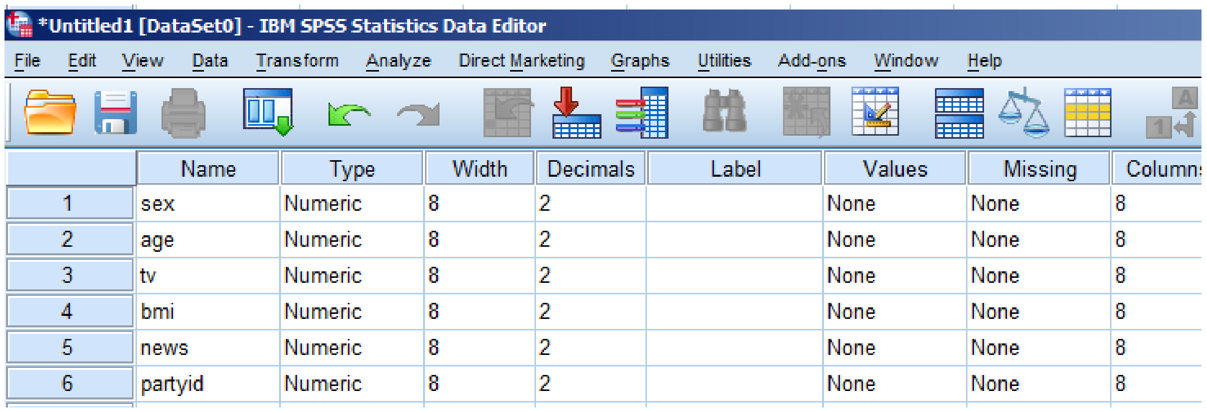
click here  or here.

You will also find a “scroll bar” for the vertical direction.

To enter data, you first must define the variable(s). To do so, move the mouse up to the first column labeled “var” and double click on the mouse. The following window will appear:

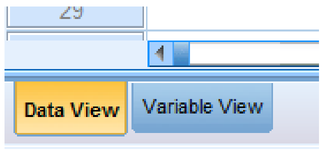

Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help
Name
Type
Width
Decimals
Label
Values
Missing

Alternatively, you can click on the "Variable View" tab near the bottom of the screen. In the first row, under the "Name" column type in the name for the variable you want to create. Once you type in a variable name, other columns will come to life. Continue entering your variable names for the data file until you are done, as such:

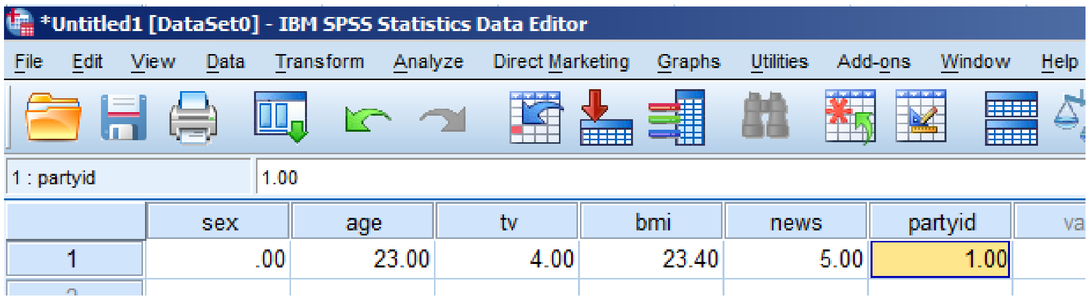

*Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help

	Name	Type	Width	Decimals	Label	Values	Missing	Column
1	sex	Numeric	8	2		None	None	8
2	age	Numeric	8	2		None	None	8
3	tv	Numeric	8	2		None	None	8
4	bmi	Numeric	8	2		None	None	8
5	news	Numeric	8	2		None	None	8
6	partyid	Numeric	8	2		None	None	8

Now, click the "Data View" tab near the bottom of the screen:



This will take you back to the data spreadsheet. Notice now that the columns are named as you named them earlier. To enter your first row of data, move the mouse to row #1 under the first variable column and click. Then, type in the data for the first "case" (the first row), as below, hitting the right arrow key to move the next column after you have entered a data point. Proceed entering the data until you are finished. Once you get to the end of the first row, you should have something that looks like this:


*Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help

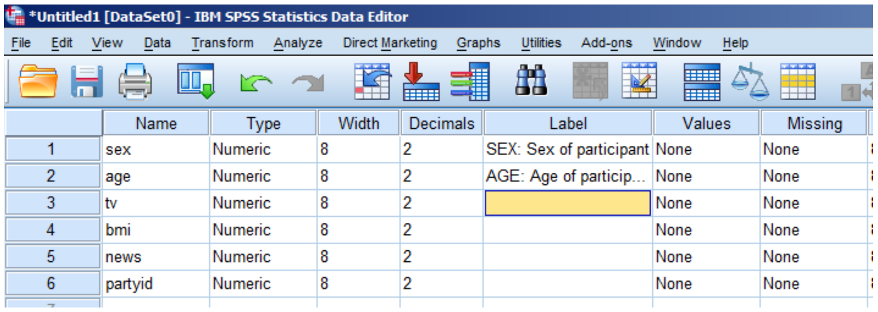
1 : partyid 1.00

	sex	age	tv	bmi	news	partyid
1	.00	23.00	4.00	23.40	5.00	1.00

Keep entering the data, row by row, until you are done. Hit the return key or use the down and left arrows to move the cursor to the beginning of the next row. Feel free to experiment. In time, you will master it. When you are done entering data, it is a good idea to save it to disk or it will be erased when you exit SPSS.

Variable Labels

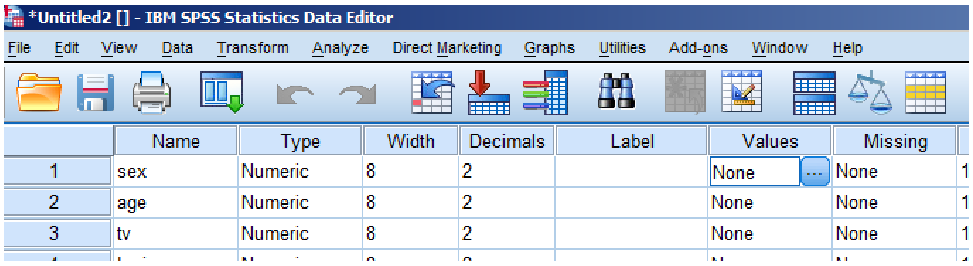
Let's now assign some descriptive labels to the variables. In the command syntax, we used the VARIABLE LABELS command. Using the GUI, this is done by going into "Variable View" by clicking the tab at the bottom of the data window, selecting a column under "Label" and entering the label you want for the variable in that row. Continue entering variable names until you are done.


*Untitled1 [DataSet0] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help

	Name	Type	Width	Decimals	Label	Values	Missing
1	sex	Numeric	8	2	SEX: Sex of participant	None	None
2	age	Numeric	8	2	AGE: Age of particip...	None	None
3	tv	Numeric	8	2		None	None
4	bmi	Numeric	8	2		None	None
5	news	Numeric	8	2		None	None
6	partyid	Numeric	8	2		None	None

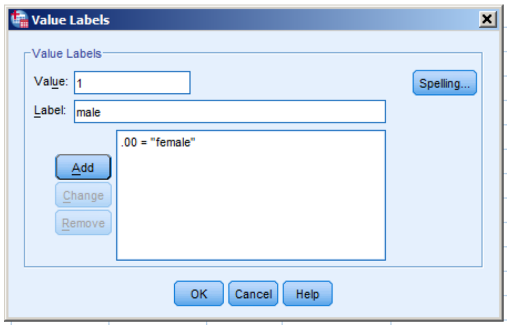
Value Labels

Descriptive labels are assigned to numerical codes for each variable in the "Variable View" section of the data. Observe under "Values" (see below) that there are no labels attached to any data values for any variables.


*Untitled2 [] - IBM SPSS Statistics Data Editor
File Edit View Data Transform Analyze Direct Marketing Graphs Utilities Add-ons Window Help

	Name	Type	Width	Decimals	Label	Values	Missing
1	sex	Numeric	8	2		None	None
2	age	Numeric	8	2		None	None
3	tv	Numeric	8	2		None	None

To add labels, select the cell under "Values" in the row for the variable you want to add value labels to. For instance, to add the labels "Male" and "Female" to the 0 and 1 codes for sex, click the "Values" cell in the "Sex" row and click the "…" button (see above). A "Value Labels" window will pop open, as below. Select the value of the variable you want to assign a code to in the "Value" box, and then provide a label in the "Label" box. Then click "Add". Continue until you have assigned all labels for the values you want to label, and click "OK". Repeat for any variables to which you want to assign value labels.



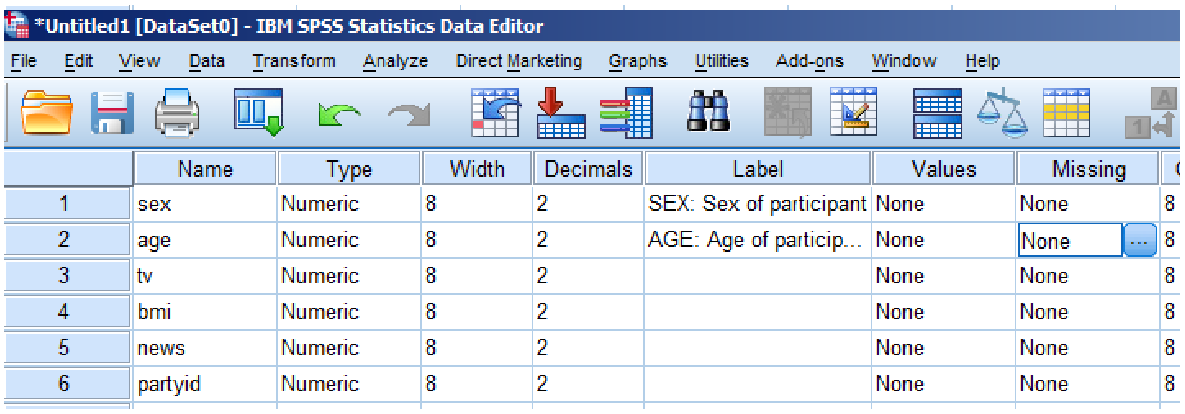
Value: 1

Label: male

.00 = "female"

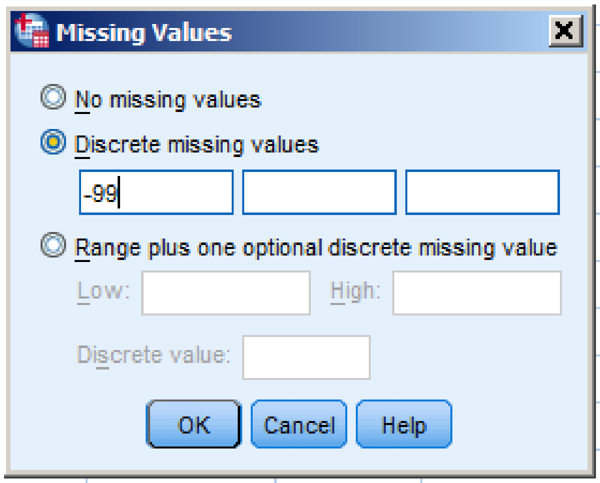
Missing Values

In the data spreadsheet, missing values can be represented either as blank cells (i.e., cells with a "."), or with a numerical code that you then label as a missing to tell SPSS not to treat those data as real whenever data analyses are conducted. Notice in the Variable View window under "Missing" that currently there are no missing values assigned.



	Name	Type	Width	Decimals	Label	Values	Missing	
1	sex	Numeric	8	2	SEX: Sex of participant	None	None	8
2	age	Numeric	8	2	AGE: Age of particip...	None	None	8
3	tv	Numeric	8	2		None	None	8
4	bmi	Numeric	8	2		None	None	8
5	news	Numeric	8	2		None	None	8
6	partyid	Numeric	8	2		None	None	8

Let's assign -99 as missing for the variable named age. First, select the "None" cell under "Missing" in the "Age" row, and click the "…" button. This will open a "Missing Values" box. Select "Discrete missing values" and enter -99, as below. Then click "OK".



No missing values
[✓ Discrete missing values]
-99
Range plus one optional discrete missing value
Low:
High:
Discrete value:

Observe in the variable view window that -99 is now assigned as a missing value code for age.

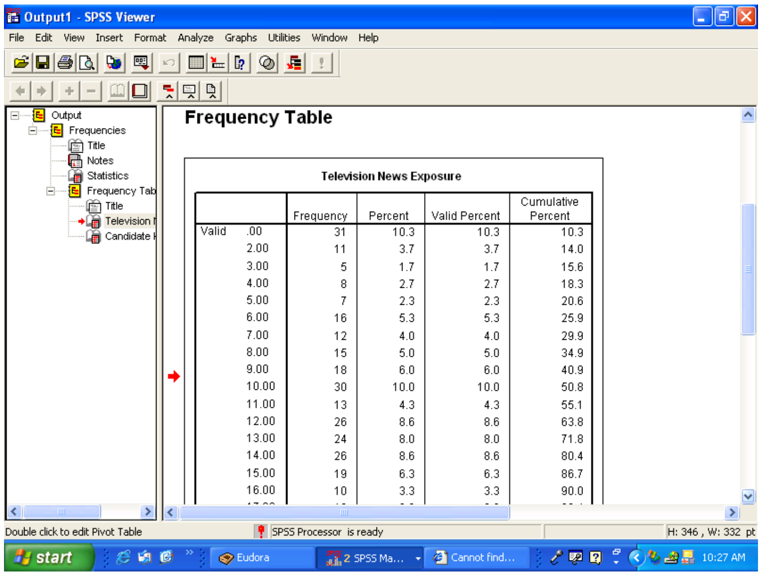
Cutting, Pasting, and Printing SPSS Output

There will be times when you want to move SPSS output to some other word processing program, such as Microsoft Word, or print part of the SPSS output you have generated.

Cutting and Pasting

Different programs will treat the output differently, and even different versions of the same program may require a slightly modified procedure. If what I describe here does not work, play around a bit until you get what you want.

Once you have generated some output, you will find in the SPSS output window a number of "objects." Some of the objects are titles for the output, and others will be graphs, tables, etc. First, select the graph, table, or whatever, that you want to move to another program. You should see a thin box appear around that object one you have selected it, as below. So long as that box remains around the object, you know that it is selected. When you have selected what you want, move the mouse to the top of the SPSS menu bar and click "Edit," and select "Copy". You can also choose "Copy Special" if the option is available to you. The Copy Special option allows you to choose the format that the resulting output will be in when you attempt to paste it. Or, rather than selecting "Edit" at the top of the SPSS menu bar, try right clicking your mouse. This may open up an option allowing you to copy the object without having to use the Edit menu.

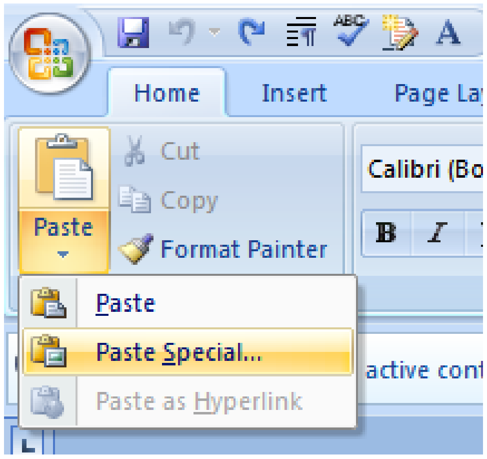


Frequency Table

Television News Exposure

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	.00	31	10.3	10.3	10.3
	2.00	11	3.7	3.7	14.0
	3.00	5	1.7	1.7	15.6
	4.00	8	2.7	2.7	18.3
	5.00	7	2.3	2.3	20.6
	6.00	16	5.3	5.3	25.9
	7.00	12	4.0	4.0	29.9
	8.00	15	5.0	5.0	34.9
	9.00	18	6.0	6.0	40.9
	10.00	30	10.0	10.0	50.8
	11.00	13	4.3	4.3	55.1
	12.00	26	8.6	8.6	63.8
	13.00	24	8.0	8.0	71.8
	14.00	26	8.6	8.6	80.4
	15.00	19	6.3	6.3	86.7
	16.00	10	3.3	3.3	90.0

Once your selection is copied, you then need to "paste" it to the other program. Different programs will have different ways of doing this, but most likely you will find a "Paste" button or option somewhere on one of the menus in the program you are using. For example, in Microsoft Word, there is a "Paste" button in the top left corner, as below.



Clicking Paste will put a copy of the image you selected in SPSS into the other program. Your program may ask you to choose the format of the object, such as JPEG file, text, and so forth. I recommend not pasting as text unless the object you selected is just text. By choosing an image format (such as JPG, PNG) you can then rescale the image in your program easily, making it larger or smaller. If you have a "Paste Special" option, I recommend choosing this rather than the simple "Paste" option. With this option, your program is almost certainly going to ask you what format you want to use.

Printing

If you want to print output from the SPSS window, you have two options. The first option is to print everything in the output window, and the second is to print only objects in the output window that you have selected. To select output, follow the procedure described in the section above on cutting and pasting. Once you are ready to print, choose "Print" under the "File" menu at the top of SPSS. This will open up a dialog box asking you whether you want to print the entire output window or just the section of output you have selected. Choose all visible output or selected output and then click "OK".

Creating or Transforming Variables

You will often find yourself wanting to construct a new variable in your data that is based on the values of other variables in data. Or you may want to modify an existing variable. Transformations and variable construction can be done with the COMPUTE command. This command operates on data in the columns of a data file to produce a new column (or modifies an existing column) containing the results of the operation when applied to each row in the data. The syntax is very simple:

```
compute variable = argument.
```

where `variable` is any existing or new variable name, and `argument` is any operation that will affect the resulting values of variable.

A few things to keep in mind about the COMPUTE command:

- If the variable following `compute` does not exist, SPSS will create it.
- If the variable following `compute` exists already, SPSS will modify it based on the argument.
- Any variable you list *after* the equals sign in a `compute` line must exist. If it does not exist, an error will result. SPSS can't do operations involving a variable that does not exist.
- If case is missing on a variable in `argument`, then typically the variable being constructed is set to a missing value. There are a few exceptions to this.

The second point above is especially important. If you use a variable name immediately following `compute` that is already being used, SPSS will change the values of the variable once the command is executed. One the command is run, you may not be to undo the modification.

Here are some examples of the COMPUTE command:

`compute c = 1.`	Sets variable c to 1 for all cases in the data.
`compute c = a + b.`	Creates or modifies a variable c that is the sum of values of two existing variables a and b. If a case is missing on either a or b, c will be set to missing.
`compute c = (a+b)/2`	Creates or modifies a variable c that is set to the arithmetic mean of the values in variables a and b. If a case is missing on a or b, c will be set to a missing value.
`compute d = mean(a,b,c)`	Creates or modifies a variable d that set to the mean of the values in existing variables a, b, and c. If a case is missing on a, b, or c, the mean will be based on only those values that are not missing.
`compute d = mean.2(a,b,c).`	Creates or modifies a variable d that is the mean of values in variables a, b, and c but only for those cases that have data for at least two of these variables. If at least two values are not available, d is set to missing.
`compute c = 4*ln(b).`	Creates or modifies a variable c that is four times the natural log of value in variable b.
`compute c = b**2.`	Creates or modifies a variable c that is the square of the

	values in b. This could also be written `compute = b*b.`
`compute c = sqrt(abs(d)).`	Creates of modifies a variable c that is the square root of the absolute value of variable d.
`compute c = (b < 2).`	Creates or modifies a variable c that will be equal to 1 if the value in existing variable b is less than 2. Otherwise, c will be set to zero. (In most languages, “true” is represented 1 and “false” is represented zero. So this code is defining c by whether the argument in parentheses is true or false).
`compute c = rv.normal(0,1).`	Creates or modifies a variable c that is a randomly selected standard normal deviate (i.e., a value from a normal distribution with mean of 0 and standard deviation of 1).

See the *Command Syntax Reference* for additional examples.

Transformations resulting from the COMPUTE command will not be applied to the data until a data analysis or file operation command is given or an EXECUTE command is run. A message on the bottom of the screen “Transformations pending…” will alert you to the fact that your COMPUTE command has not yet been completed. To complete the operation, type `execute` in the syntax and then execute it.

Here are some mathematical operators that you will find useful:

`+`	addition
`-`	subtraction
`/`	division
`*`	multiplication
`<`	less than
`>`	greater than
`<=`	less than or equal to
`>=`	greater than or equal to
`<>`	not equal to
`abs`	absolute value
`sqrt`	square root
`ln`	natural logarithm
`**2`	square (i.e., raise to the power of 2). Use “3” in order to cube, and so forth.
`exp`	exponentiation (i.e., raise *e* to the power of)

SPSS respects order of operations in mathematics. So multiplication and division is done before addition and subtraction, for example. But operations in parentheses are done before operations not in parentheses. So the command

```
compute a = b+2/c.
```

divides 2 by the value held in variable c, and then adds the value in variable b to the result. But the command

```
compute a = (b+2)/c.
```

first adds 2 to the value in variable b, and then divides this sum by the value in c. These are very different. Suppose for a particular case, b = 4 and c = 8. The first example above yields a = 4.25 whereas the second example yields a = 0.75.

It is good idea to label variables as soon as you have created them. Alternatively, have a section of your program that produces labels for variables that you have constructed after data were entered. For example, if you had a variable named BMI and you wanted to create a new variable identifying whether a person was considered “overweight” by criteria used by the Centers for Disease Control (defined as BMI of at least 30), you might do something like this:

```
compute overw = (bmi >= 30).
variable labels overw 'Overweight by CDC criteria'.
value labels overw 0 'not overweight' 1 'overweight'.
```

Assigning variable labels and value labels immediately after you construct a variable is a good way of documenting what you are doing, in addition to adding labels to output that will be more meaningful to you when looking at output than just an abstract variable name with values of 0 and 1.

Standardizing a Variable

There are occasions where you might want to analyze a variable after the measurements have first been standardized. Standardizing a variable involves converting each measurement in the distribution to deviations from the mean in standard deviation units. That is, if x_i is a case *i*’s measurement on variable X, then case *i*’s standardized value Z_i on *X* is

$$Z_i = \frac{x_i - mean(X)}{SD_X}$$

where *mean*(*X*) is the mean of *X* and SD_X is the standard deviation of *X*. The resulting distribution of values of *Z* will have a mean of 0 and a standard deviation of 1. You could standardize a variable using the COMPUTE command, but only this would require first calculating the mean and standard deviation of the variable. For instance, if variable “hello” has mean 2 and standard deviation 3, you could create a new version of hello that is standardized by executing

```
compute zhello = (hello-2)/3.
```

The values in the resulting column labelled "zhello" will be the corresponding measurements for hello but rescaled so that the mean of zhello is 0 and the standard deviation is one. Each value in zhello is how many standard deviations above (if zhello is positive) or below (if zhello is negative) the mean the corresponding case's measurement on hello resides in the distribution.

An alternative is to use the DESCRIPTIVES command, introduced later, but tacking on the SAVE option, as in

```
descriptives variables = hello/save.
```

This command will generate a new variable in the data set called "zhello" that contains the standardized values of hello. Using this procedure, you don't have to know the mean and standard deviation of the variable, and the computations will be done to a much higher degree of precision.

Conditioning Computations Using IF

Operations defined by the COMPUTE command are applied indiscriminately to every row in the data file. Sometimes you want an operation to be carried out only if a certain condition is met. For instance, you might want the operation to be carried out only for males, or only for people younger than 20. This is typically done with the IF command. The IF command works much like the COMPUTE command in terms of the structure of the argument, but you don't need to write COMPUTE. Instead, you say IF. The syntax structure is

```
if (argument1) variable = argument2.
```

Where `argument1` is any argument that, if true, results in the execution of the rest of the command, `variable` is any existing or new variable, and `argument2` is any operation that will affect the values that result and stored in `variable.` Here are some examples of the IF command:

`if (b = 1) c = 2.`	Creates or modifies a variable c that is set equal to 2 for cases with a value of b equal to 1.
`if ((a+b)<=3) c = 0.`	Creates or modifies a variable c that is set to 0 if the sum of the values in existing variables a and b is less than or equal to 3.
`if ((a>b) or (c>b)) k=d.`	Creates or modifies a variable k that is set to the value in variable d if either the value in variable a is greater than the value in variable b or the value in variable c is greater than the value in variable b.
`if (a=1 and b=1) c=ln(d).`	Creates or modifies a variable c that is set to the natural log of d if the values of a and b are both equal to 1.

For additional examples, see the *Command Syntax Reference*.

The syntax structure of the IF command following `argument1` is largely identical to `compute`. So any of the mathematical operations available to you in `compute` can also be used following `argument1`. In order for SPSS to determine if `argument1` is true (in which case the rest of the operation proceeds), any variable listed in `argument1` must already exist in the data, otherwise an error will result. And as with COMPUTE, any variables in `argument2` must also exist, or an error will result.

The IF command produces a result only if the condition specified in `argument1` is true. If the argument is false for a particular case in the data, then any variable created by the IF command is set to missing for that case. But if the variable already exists and the argument is false for a particular case, the value currently held in variable will remain for that case. Its value is untouched.

The COMPUTE and IF commands are often used in conjunction. For example, suppose you have a variable called group which has the values 1 and 2. The command set

```
compute d1 =-1.
if (group=2) d1=1.
```

will first create a new variable `d1` set equal to -1 for all cases, but then change it to 1 for cases with value 2 on group. The resulting variable d1 is set to 1 for cases with `group` = 2 and it is set to -1 for cases with `group` = 1. Note that two IFs would accomplish the same thing:

```
if (group=1) d1=-1.
if (group=2) d1=1.
```

There is a subtle difference here, however. What if a case was missing on `group`? The first set of commands would set `d1` to 1 for cases missing on `group`, whereas the second set of commands would set cases missing on group to missing on `d1`.

You may find yourself sometimes wanting to convert missing data into something else. For instance, suppose you wanted to create a new variable called `miss` set to 1 if a case is missing on a variable named `hello`, and 0 for all cases not missing on `hello`. The syntax would be

```
compute miss=0.
if missing(hello) miss = 1.
```

Like the COMPUTE command, operations conducted with IF will not be applied to the data until a data analysis or file operation command is given or an EXECUTE command is run. A message on the bottom of the screen “Transformations pending” will alert you to the fact that your IF command has not yet been completed. To complete the operation, type `execute` in the syntax and then execute it.

An Illustration

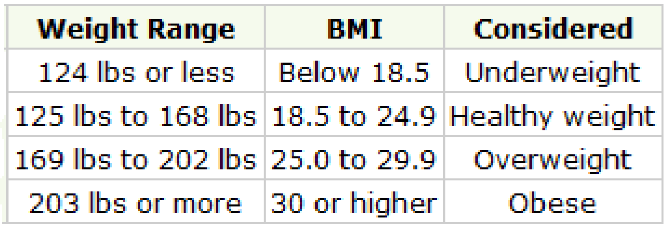
Here is a practical illustration of the use of the IF command. The program below constructs a data set containing the body mass index of 20 women and 20 men (a person’s body mass index, or BMI, is the percentage of the person’s body weight that is fat). The data are entered such that the first 20 are women, and the second 20 are men.

```
data list free/bmi.
begin data.
19.28 22.20 20.89 22.86 22.37 24.52 17.30 22.50 25.73 19.49
20.03 20.69 23.81 23.52 22.12 21.95 23.61 21.69 19.11 19.88
21.05 26.33 24.98 21.00 23.64 28.47 21.21 30.28 20.49 27.89
24.29 23.38 24.29 18.57 23.69 23.65 31.43 28.47 28.24 24.98
end data.
variable labels bmi 'Body mass index'.
```

Knowing that the first 20 rows of the data are women and the next 20 are men, a new variable can be created coding sex (0 = female, 1= male) using the row number in the data file. The code below does this. The $casenum variable you can’t see in the data, but SPSS understands it---it is an internal variable. $casenum holds the row number in which a case resides in the data file.

```
if ($casenum < 21) sex = 0.
if ($casenum > 20) sex = 1.
variable labels sex 'Sex of the person'.
value labels sex 0 'female' 1 'male'.
```

The Centers for Disease Control (CDC) operationalizes various categories of weight using the BMI. According to the CDC



Weight Range	BMI	Considered
124 lbs or less	Below 18.5	Underweight
125 lbs to 168 lbs	18.5 to 24.9	Healthy weight
169 lbs to 202 lbs	25.0 to 29.9	Overweight
203 lbs or more	30 or higher	Obese

Source: http://www.cdc.gov/obesity/defining.html

Using the CDC criteria, we can create a new variable that codes whether a person is underweight, of healthy weight, overweight, or obese. The code that does so is

```
if (bmi < 18.5) cdc = 1.
if (bmi >= 18.5 and bmi <= 25) cdc = 2.
if (bmi >= 25 and bmi < 30) cdc = 3.
if (bmi >= 30) cdc = 4.
variable labels cdc 'CDC: CDC weight status category'.
value labels cdc 1 'underweight' 2 'healthy weight' 3 'overweight'
  4 'obese'.
execute.
```

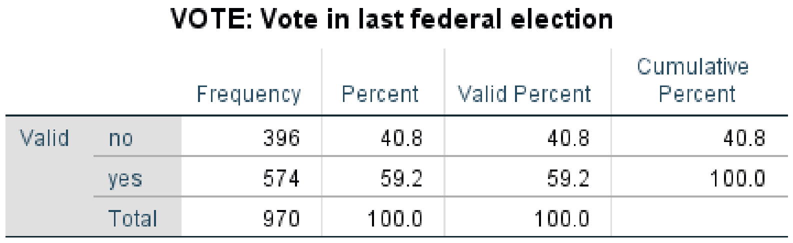
To see the results of this code, run it all at once and take a look at the data file SPSS produced.

Univariate Data Description

SPSS offers many options for generating statistics used for describing a variable in a data set, such as how frequently certain measurements occur, or how to characterizing the “center” or “spread” of a distribution of measurements. A frequency table, for instance, can be produced in SPSS using the FREQUENCIES command. It has a variety of additional options for generating other statistics that will be of interest as well as some graphical depictions of data. For example

```
frequencies variables = vote.
```

will produce a table containing how frequently different values of “vote” appear in the data, as in the output below:



```
                VOTE: Vote in last federal election

                                                        Cumulative
                   Frequency   Percent   Valid Percent    Percent
Valid   no               396      40.8            40.8       40.8
        yes              574      59.2            59.2      100.0
        Total            970     100.0           100.0
```

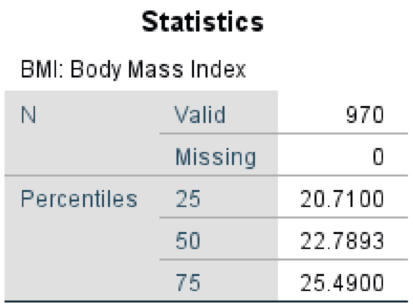
In the data, vote is coded 0 = no, 1 = yes. Because those values are assigned to the data as value labels, we see “no” and “yes” in the table rather than 0 and 1. You can add as many variables to the list you want, and SPSS will generate a frequency table for each of the variables in the list. For example:

```
frequencies variables = vote news tv.
```

The FREQUENCIES command has many options to generate other statistics. For example, suppose you want to know the 25th, 50th, and 75th percentiles of the distribution of BMI, but you don’t want to see the frequency table itself. Try

```
frequencies variables = bmi/percentiles = 25 50 75/format notable.
```

The resulting output is

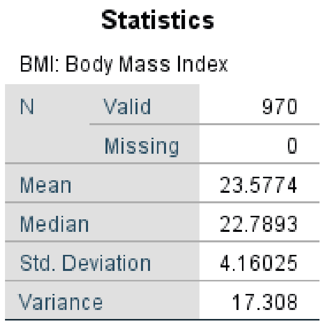


```
            Statistics
BMI: Body Mass Index
N             Valid         970
              Missing         0
Percentiles   25        20.7100
              50        22.7893
              75        25.4900
```

The “format notable” option suppresses the frequency table (try leaving this off and see what happens). Or you might want to see the mean, median, standard deviation, and variance of the distribution of BMI in the data:

```
frequencies variables = bmi/statistics mean median stddev variance
   /format notable.
```

which yields the output below.

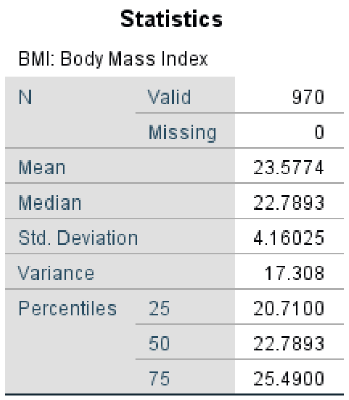


```
            Statistics
BMI: Body Mass Index
N               Valid        970
                Missing        0
Mean                     23.5774
Median                   22.7893
Std. Deviation           4.16025
Variance                  17.308
```

You could combine these if you wanted to, as such:

```
frequencies variables = bmi/statistics mean median stddev variance
   /percentiles = 25 50 75/format notable.
```

which generates

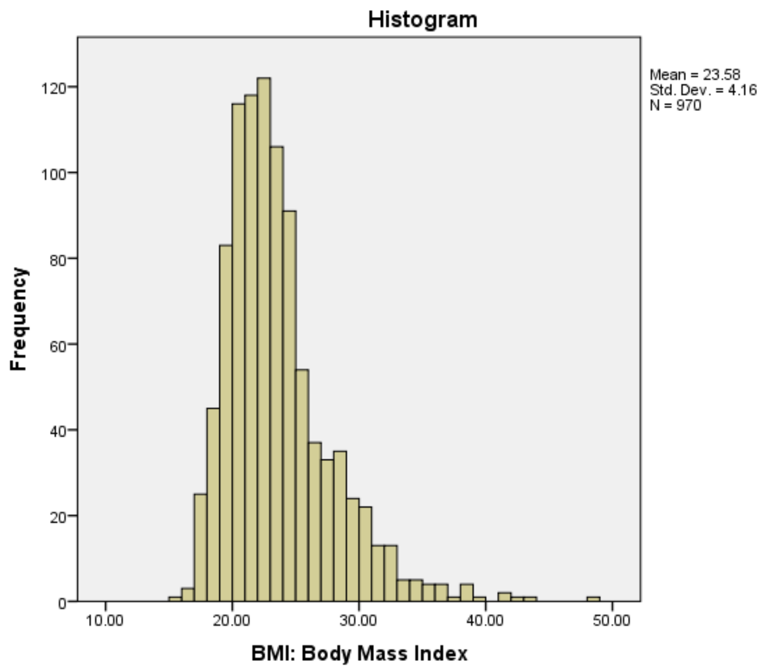


```
            Statistics
BMI: Body Mass Index
N               Valid        970
                Missing        0
Mean                     23.5774
Median                   22.7893
Std. Deviation           4.16025
Variance                  17.308
Percentiles     25       20.7100
                50       22.7893
                75       25.4900
```

The FREQUENCIES command also has an option for producing a histogram. Simply request it by adding the option to the command:

```
frequencies variables = bmi/histogram/format notable.
```

If you double-click on the histogram that will put it into edit mode. Many things are editable in SPSS figures. Play around and you will find ways of making this much nice looking that what SPSS produces by default.



But you don't have to use the FREQUENCIES command to produce a histogram. Try this instead:

```
graph/histogram=bmi.
```

SPSS also has a number of other commands that producing basic descriptive statistics. For example, the DESCRIPTIVES command produces a table containing the mean, standard deviation, minimum, and maximum value observed in the data. You can put more than one variable in the variable list to produce a compact table describing many variables simultaneously. For example, try

```
descriptives variables = bmi news tv gpa.
```

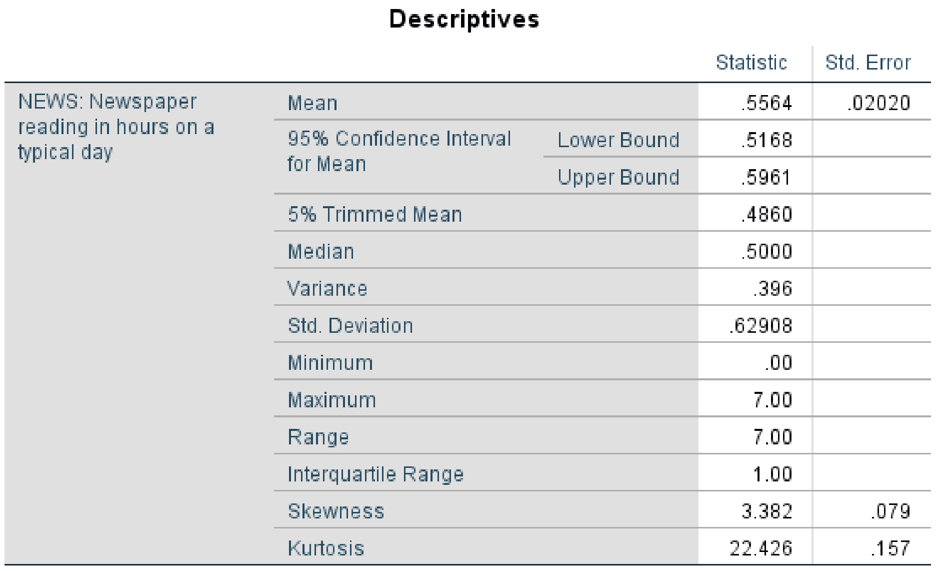
which produces as output

Descriptive Statistics

	N	Minimum	Maximum	Mean	Std. Deviation
BMI: Body Mass Index	970	15.14	48.61	23.5774	4.16025
NEWS: Newspaper reading in hours on a typical day	970	.00	7.00	.5564	.62908
TV: TV hours on an average day	970	.00	20.00	2.3469	1.48140
GPA: Grade point average	970	2.00	4.00	3.0817	.38677
Valid N (listwise)	970				

An alternative is the EXAMINE command. It produces many other statistics by default relative to what DESCRIPTIVES does. For instance:

```
examine variables = news/plot=none.
```



Descriptives

			Statistic	Std. Error
NEWS: Newspaper reading in hours on a typical day	Mean		.5564	.02020
	95% Confidence Interval for Mean	Lower Bound	.5168	
		Upper Bound	.5961	
	5% Trimmed Mean		.4860	
	Median		.5000	
	Variance		.396	
	Std. Deviation		.62908	
	Minimum		.00	
	Maximum		7.00	
	Range		7.00	
	Interquartile Range		1.00	
	Skewness		3.382	.079
	Kurtosis		22.426	.157

Try leaving off "plot = none" and see what happens.

Dealing with Grouped Data

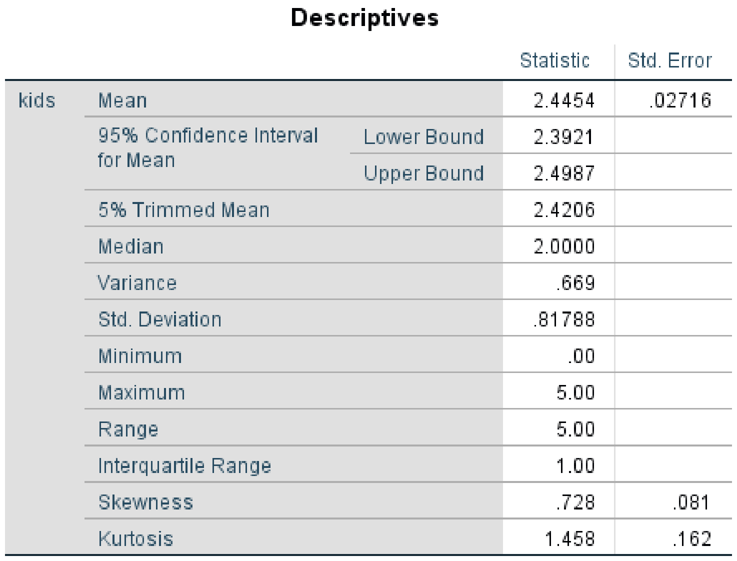
Sometimes data may take the form of a frequency distribution which counts the number of observations with a particular value on a particular variable. For instance, consider the table below, which tabulates the number of people out of 907 who gave a particular response to the question "what is the ideal number of children in a family?"

Response	Frequency
0	10
1	30
2	524
3	252
4	71
5	20

It is possible to calculate various statistics on data in this form without having to go to all the hassle of typing in all 907 responses. To calculate the mean and standard deviation of the ideal number of children, or various other statistics of interest, use the frequency of responses to weight those responses using the WEIGHT command in SPSS. For example, the program below enters the data in table form and then calculates various statistics using the EXAMINE command after telling SPSS to weight each response by the number of people who gave that response.

```
data list free/kids freq.
begin data.
0 10
1 30
2 524
3 252
4 71
5 20
end data.
weight by freq.
examine variables = kids.
weight off.
```

The resulting output is



Descriptives

			Statistic	Std. Error
kids	Mean		2.4454	.02716
	95% Confidence Interval for Mean	Lower Bound	2.3921	
		Upper Bound	2.4987	
	5% Trimmed Mean		2.4206	
	Median		2.0000	
	Variance		.669	
	Std. Deviation		.81788	
	Minimum		.00	
	Maximum		5.00	
	Range		5.00	
	Interquartile Range		1.00	
	Skewness		.728	.081
	Kurtosis		1.458	.162

Observe that even though the data file SPSS generates contains only 6 rows, SPSS is saying there are 907 cases in the data, which is the total number of responses to the question in the table. The mean response is 2.45 and the standard deviation is 0.82, based on the 907 responses.

Describing Differences Between Groups

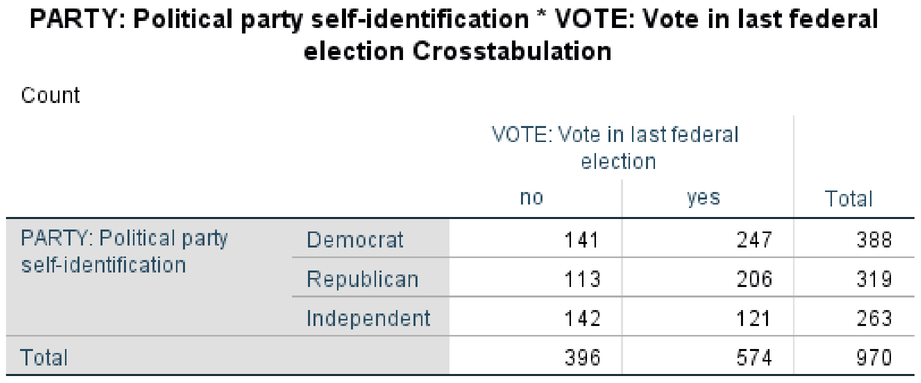
Research is usually about some kind of comparison, often between two or more groups on some variable of interest. These groups may be naturally occurring (such as men and women) or constructed artificially, as when conducting experiments that involve random assignment to conditions. SPSS has a variety of commands that can be brought into service for describing data in a format that allows to do quick comparisons between those groups.

Crosstabulation

For example, we might want to know whether Democrats or Republicans were more likely to vote relative to Independents. The CROSSTABS command tabulates the frequencies of one variable at values of another variable. For example, the command below produces a crosstabulation of political party self-identification with whether or not the person voted in the last election.

```
crosstabs tables = party by vote.
```

The output shows that of the 970 cases in the data, 247 were Democrats who voted, 206 were Republicans who voted, 142 were Independents who did not vote, and so forth.



PARTY: Political party self-identification * VOTE: Vote in last federal election Crosstabulation

Count

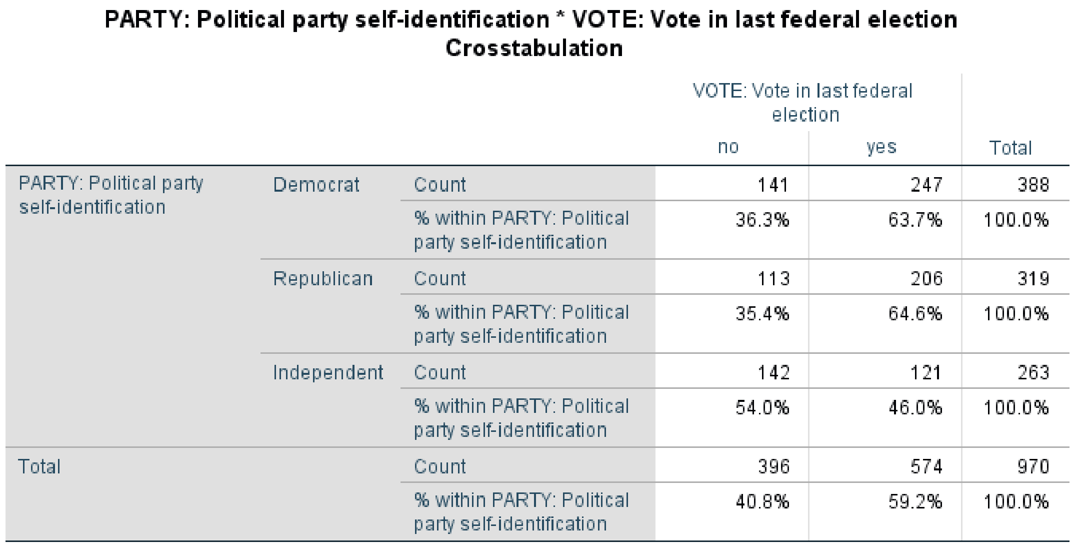
		VOTE: Vote in last federal election		
		no	yes	Total
PARTY: Political party self-identification	Democrat	141	247	388
	Republican	113	206	319
	Independent	142	121	263
Total		396	574	970

In the CROSSTABS command, the variable listed before the "by" defines the rows of the table, and the variable that comes after "by" defines the columns. If you want to flip the rows and columns just flip the order of the variables before and after "by".

It is difficult with the table in this form to determine whether Democrats were voting more than Republicans or Independents because there are more Democrats in the data. An option expresses the frequencies in percentage terms conditioned on the value of the variable defining either the row or column. Try

```
crosstabs tables = party by vote/cells count row.
```

which produces the output below. In addition to the frequencies in each cell of the table, SPSS provides percentages for the vote variable conditioned on row in the table.



PARTY: Political party self-identification * VOTE: Vote in last federal election Crosstabulation

			VOTE: Vote in last federal election		
			no	yes	Total
PARTY: Political party self-identification	Democrat	Count	141	247	388
		% within PARTY: Political party self-identification	36.3%	63.7%	100.0%
	Republican	Count	113	206	319
		% within PARTY: Political party self-identification	35.4%	64.6%	100.0%
	Independent	Count	142	121	263
		% within PARTY: Political party self-identification	54.0%	46.0%	100.0%
Total		Count	396	574	970
		% within PARTY: Political party self-identification	40.8%	59.2%	100.0%

So we can see here that 63.7% of Democrats voted (the other 36.3% did not), 64.6% of Republicans voted (35.4% of Republicans did not), and 46% of Independents voted (and the remaining 54% did not). You could also condition on the column if you desire. So by running

```
crosstabs tables = party by vote/cells count column.
```

you can easily discern (from the output below) that of those who voted, 43% were Democrats, 35.9% were Republicans, and 21.1% were Independents. Of those who did not vote, 35.6% were Democrats, 28.5% were Republicans, and 35.9% were Independents.

PARTY: Political party self-identification * VOTE: Vote in last federal election Crosstabulation

			VOTE: Vote in last federal election		
			no	yes	Total
PARTY: Political party self-identification	Democrat	Count	141	247	388
		% within VOTE: Vote in last federal election	35.6%	43.0%	40.0%
	Republican	Count	113	206	319
		% within VOTE: Vote in last federal election	28.5%	35.9%	32.9%
	Independent	Count	142	121	263
		% within VOTE: Vote in last federal election	35.9%	21.1%	27.1%
Total		Count	396	574	970
		% within VOTE: Vote in last federal election	100.0%	100.0%	100.0%

Tables and Graphs of Means and Variation

A variation on the crosstabulation is a table that contains various descriptive statistics for one or more variables for different groups defined by a second variable. For example, if you wanted the mean BMI and mean TV viewing frequency of Democrats, Republicans, and Independents, along with the standard deviations of these variables for each of the groups, you could use the MEANS command, as such:

```
means tables = bmi tv by party.
```

The resulting output below is fairly self-explanatory, although there many possible interpretations of the pattern. You can put lots of variables in the means command, and it will summarize the data for each group for each variable in one tidy output.

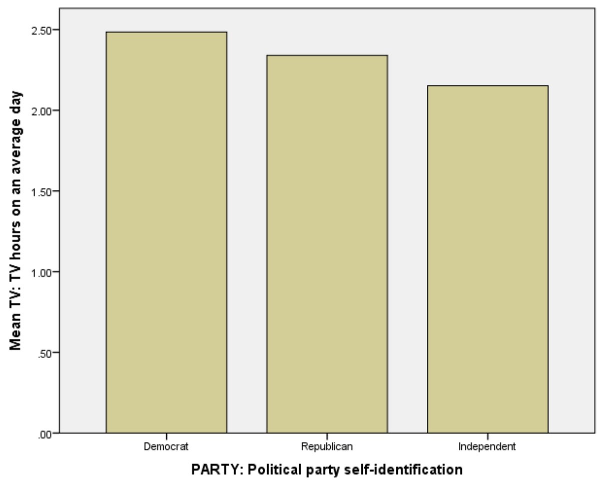
Report

PARTY: Political party self-identification		BMI: Body Mass Index	TV: TV hours on an average day
Democrat	Mean	23.7389	2.4845
	N	388	388
	Std. Deviation	4.20461	1.51129
Republican	Mean	23.7174	2.3401
	N	319	319
	Std. Deviation	4.14186	1.56298
Independent	Mean	23.1693	2.1521
	N	263	263
	Std. Deviation	4.10455	1.30825
Total	Mean	23.5774	2.3469
	N	970	970
	Std. Deviation	4.16025	1.48140

You could visually depict the group means using a bar graph in which the height of the bar represents the group mean. Try

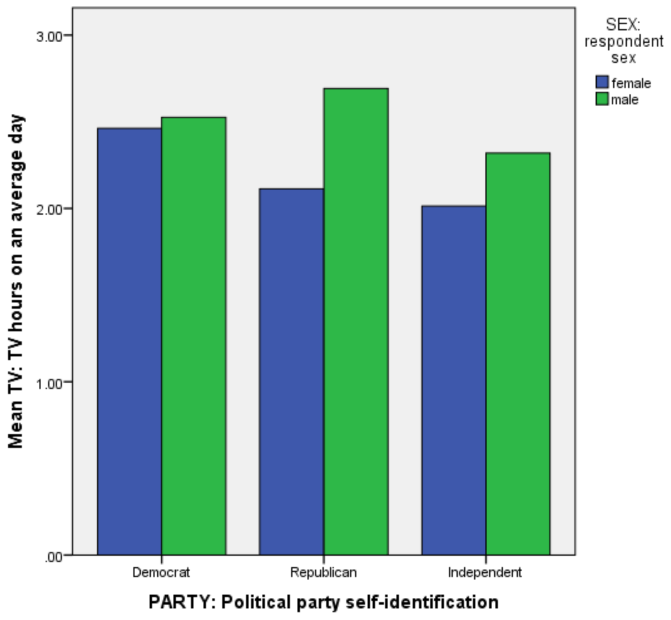
```
graph/bar=mean(tv) by party.
```

to produce a figure that looks like this:



You could add another variable to produce different bars for, say, men versus women:

```
graph/bar=mean(tv) by party by sex.
```



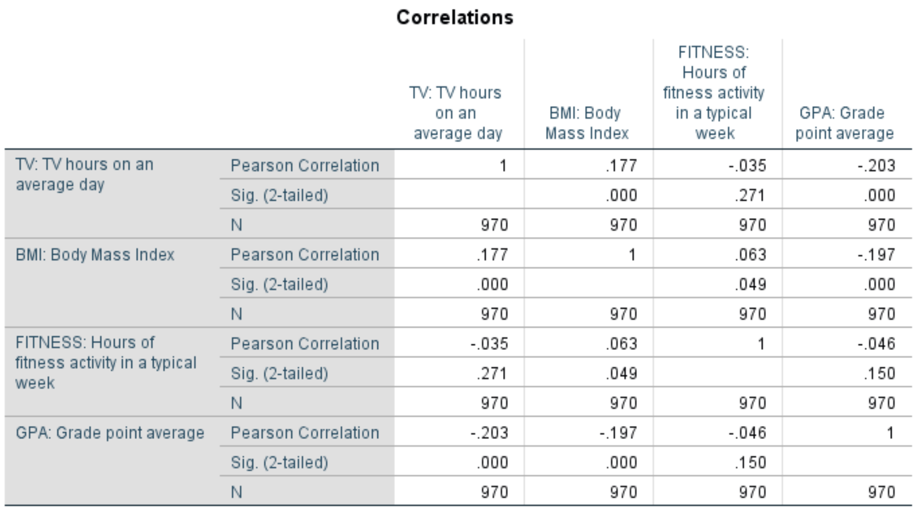
If you don't like the color or want to change some of the text, many things in SPSS graphs are editable. Put it into edit mode by double clicking the object in the output window, and fiddle with it until you get something you like.

Correlating Variables

There are many measures of association two variables. One of the most commonly-used measures is Pearson's coefficient of correlation, which is generated with the CORRELATIONS command. In its most basic form, the command requires you to list the variables you want to generate a correlation between. For instance,

```
correlations variables = tv bmi fitness gpa.
```

produces a 3 X 3 matrix of Pearson correlations, as below. The correlation between the variable listed in the row and the variable listed in the column is found at the intersection of the row and column in the table. The table is symmetrical, meaning that the values in the table below the diagonal of ones are the same as the values above the diagonal of ones.



Correlations

		TV: TV hours on an average day	BMI: Body Mass Index	FITNESS: Hours of fitness activity in a typical week	GPA: Grade point average
TV: TV hours on an average day	Pearson Correlation	1	.177	-.035	-.203
	Sig. (2-tailed)		.000	.271	.000
	N	970	970	970	970
BMI: Body Mass Index	Pearson Correlation	.177	1	.063	-.197
	Sig. (2-tailed)	.000		.049	.000
	N	970	970	970	970
FITNESS: Hours of fitness activity in a typical week	Pearson Correlation	-.035	.063	1	-.046
	Sig. (2-tailed)	.271	.049		.150
	N	970	970	970	970
GPA: Grade point average	Pearson Correlation	-.203	-.197	-.046	1
	Sig. (2-tailed)	.000	.000	.150	
	N	970	970	970	970

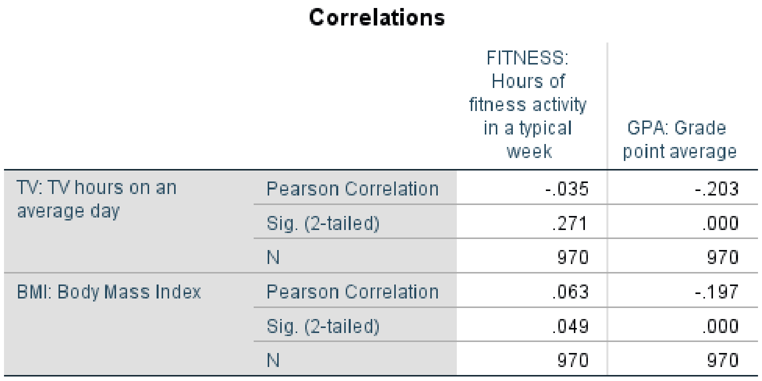
If the list of variables is a long one, the resulting matrix can be quite large, and it may very well include a bunch of correlations you really don't care about. A variation on the command allows you to reduce the size of the matrix by including only a subset of the possible correlations. The command format is

```
correlations variables = rowvars with colvars.
```

where `rowvars` and `colvars` are lists of variables that define the rows and columns of the matrix, respectively. For example, the command

```
correlations variables = tv bmi with fitness gpa.
```

produces a 2 X 2 matrix of correlations, as below:

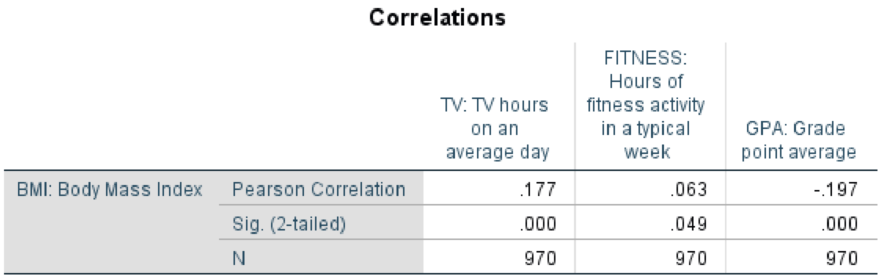


Correlations

		FITNESS: Hours of fitness activity in a typical week	GPA: Grade point average
TV: TV hours on an average day	Pearson Correlation	-.035	-.203
	Sig. (2-tailed)	.271	.000
	N	970	970
BMI: Body Mass Index	Pearson Correlation	.063	-.197
	Sig. (2-tailed)	.049	.000
	N	970	970

This can be particularly handy if you only care about the correlation between one of the variables in the list and all the others, as in

```
correlations variables = bmi with tv fitness gpa.
```



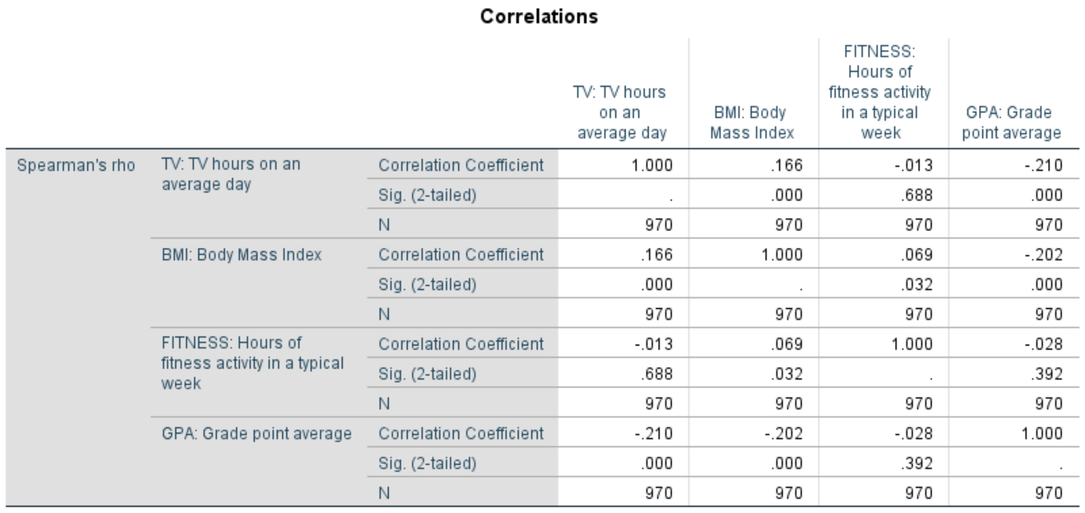
Correlations

		TV: TV hours on an average day	FITNESS: Hours of fitness activity in a typical week	GPA: Grade point average
BMI: Body Mass Index	Pearson Correlation	.177	.063	-.197
	Sig. (2-tailed)	.000	.049	.000
	N	970	970	970

A variant on Pearson's coefficient of correlation is Spearman's correlation. This measure of correlation can be generated using the CORRELATIONS command by correlating the variables as above but only after the measurements are first converted to ranks. Alternatively, you can use the NONPAR CORR command, which has an option built in for generating Spearman's correlation without having to manually rank each of the variables. Consider the command below, which looks similar to the CORRELATIONS command:

```
nonpar corr variables = tv bmi fitness gpa.
```

This command produces the output below, which looks much like the matrix of Pearson's correlations, but the cell entries are Spearman's correlation rather than Pearson's. The "WITH" function works with NONPAR CORR in the same way it does with CORRELATIONS.



Correlations

			TV: TV hours on an average day	BMI: Body Mass Index	FITNESS: Hours of fitness activity in a typical week	GPA: Grade point average
Spearman's rho	TV: TV hours on an average day	Correlation Coefficient	1.000	.166	-.013	-.210
		Sig. (2-tailed)	.	.000	.688	.000
		N	970	970	970	970
	BMI: Body Mass Index	Correlation Coefficient	.166	1.000	.069	-.202
		Sig. (2-tailed)	.000	.	.032	.000
		N	970	970	970	970
	FITNESS: Hours of fitness activity in a typical week	Correlation Coefficient	-.013	.069	1.000	-.028
		Sig. (2-tailed)	.688	.032	.	.392
		N	970	970	970	970
	GPA: Grade point average	Correlation Coefficient	-.210	-.202	-.028	1.000
		Sig. (2-tailed)	.000	.000	.392	.
		N	970	970	970	970

Selective Processing

There are times when you want to conduct an analysis only on certain cases in the data set. There is a variety of ways to tell SPSS to limit an analysis to only those cases that you want included. This can be done with the FILTER command or the SELECT IF command. Consider the command set below:

```
select if (bmi > 28).
frequencies variables = tv.
correlations variables = fitness tv.
```

This command will generate a frequency table of "tv" and calculate Pearson's coefficient of correlation between "bmi" and "fitness" but only for cases in the data file with a value of "bmi" greater than 28. In other words, SPSS will discard all cases that don't meet the condition of the argument in the SELECT IF command prior to continuing. You can include as many data analysis commands as you want following the SELECT IF, and SPSS will conduct the analysis only after discarding cases that don't meet the argument.

The problem with the SELECT IF command is that it is permanent, meaning that it will physically discard the cases that don't meet the condition of the SELECT IF command, meaning that they are gone from the data file being used. To recover them you must reload the data. That makes SELECT IF very dangerous, because if you accidently save the data file after running the SELECT IF command, forgetting that you have told SPSS to discard some of the cases, then those cases are gone from not only the data file in memory but also from the data file on your storage device. So use the SELECT IF option with caution.

There is one way of making the action of the SELECT IF command temporary, and that is by preceding it with the word TEMPORARY, as in

```
temporary.
select if (bmi > 28).
correlations variables = fitness tv.
```

This set of commands will calculate the correlation only among cases with "bmi" greater than 28 without permanently deleting the other cases from the data. The problem with TEMPORARY is that when it is used, the SELECT IF will apply only to the one command following the SELECT IF. That means that in order to generate, say, a frequency table and a correlation among cases that meet the condition, you'd have to use multiple TEMPORARY and SELECT IF commands, as below.

```
temporary.
select if (bmi > 28).
frequencies variables = tv.
temporary.
select if (bmi > 28).
correlations variables = fitness tv.
```

An alternative option is the FILTER command. Consider the set of commands below, which accomplishes the same analysis as the command set above but without the use of multiple SELECT IF and TEMPORARY commands.

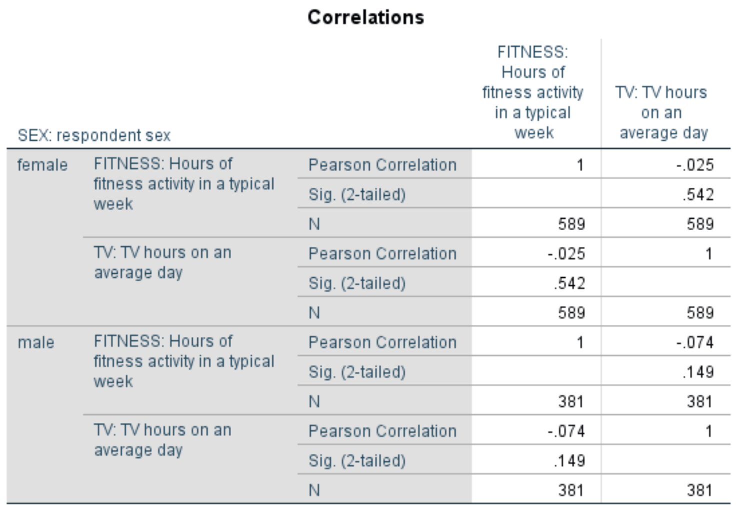
```
compute bigbmi = (bmi > 28).
filter by bigbmi.
frequencies variables = tv.
correlations variables = fitness tv.
filter off.
delete variables bigbmi.
```

The first line creates a new variable called “bigbmi” that is set to 1 for cases with “bmi” greater than 28. The next line turns on the filter, telling SPSS that for everything that follows and until the filter is turned off, conduct the operation only for cases with a value on the filter variable equal to 1. The following lines conduct the analysis. After all the analysis commands have been executed, the filter is turned off, and the filtering variable is deleted. When the filter is turned off, then all commands that follow will be conducted on all cases in the data. This would be much more efficient than TEMPORARY and SELECT IF when you have a lot of commands you want to execute only for those cases that meet the condition.

On occasion, you may want to conduct the same analysis repeatedly for subsets of the data. The FILTER or SELECT IF approaches described above could be used, but there is another approach based on the SPLIT FILE command. The SPLIT FILE command is used when you want to repeatedly execute one or more commands over and over again on subgroups of the data defined by a variable coding the groups. To do so, the file must first be sorted by the values used to code the groups. Once this is accomplished, the data set is split into subsets for the purpose of analysis using the SPLIT FILE command. Any commands that follow will be conducted separately in each of the groups used to define the split files. For example, the commands below correlate “fitness” and “tv” separately in both men and women and produces a two correlation matrices, one for males and one for females, stacked on top of each other

```
sort cases by sex.
split file by sex.
correlations variables = fitness tv.
split file off.
```

The resulting output is



Correlations

SEX: respondent sex			FITNESS: Hours of fitness activity in a typical week	TV: TV hours on an average day
female	FITNESS: Hours of fitness activity in a typical week	Pearson Correlation	1	-.025
		Sig. (2-tailed)		.542
		N	589	589
	TV: TV hours on an average day	Pearson Correlation	-.025	1
		Sig. (2-tailed)	.542	
		N	589	589
male	FITNESS: Hours of fitness activity in a typical week	Pearson Correlation	1	-.074
		Sig. (2-tailed)		.149
		N	381	381
	TV: TV hours on an average day	Pearson Correlation	-.074	1
		Sig. (2-tailed)	.149	
		N	381	381

It is important to first sort the data file by the variable defining the groups (using the SORT command, as above), or the analysis will not be conducted correctly.

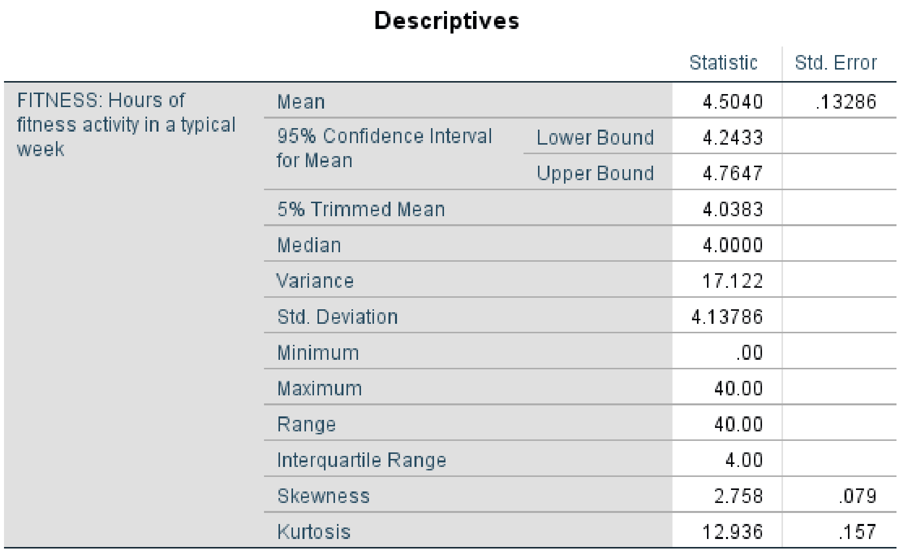
Estimation: Standard Errors and Confidence Intervals

Central to estimation theory is the *standard error* of a statistic. It quantifies how much you'd expect an estimate of a parameter to vary on average around the corresponding parameter when taking a random sample from the population of a given size. The standard error is also used in the construction of a confidence interval for a parameter.

SPSS has several procedures that produce standard errors and confidence intervals in its output either by default or as an option. We have already seen one such procedure—the EXAMINE procedure. If we think of the 970 students in the student data file as a random sample from a population of students, we might want to estimate the mean number of hours a student engages in some kind of fitness activity along with its standard error of the sample mean and a confidence interval for the population mean. Applying the command below to the student data file

```
examine variables = fitness/plot=none.
```

generates the output below. Notice that the output includes standard descriptive information such as the mean number of hours (4.504), as well as the standard error of the sample mean (0.133) along with a 95% confidence interval for the population mean (4.243 to 4.765).



Descriptives

			Statistic	Std. Error
FITNESS: Hours of fitness activity in a typical week	Mean		4.5040	.13286
	95% Confidence Interval for Mean	Lower Bound	4.2433	
		Upper Bound	4.7647	
	5% Trimmed Mean		4.0383	
	Median		4.0000	
	Variance		17.122	
	Std. Deviation		4.13786	
	Minimum		.00	
	Maximum		40.00	
	Range		40.00	
	Interquartile Range		4.00	
	Skewness		2.758	.079
	Kurtosis		12.936	.157

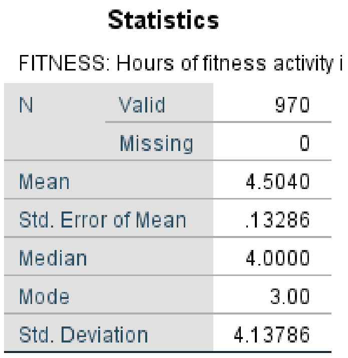
If you want more or less than 95% confidence, change the confidence desired by adding "cinterval" to the command and specifying the desired confidence. For instance, the command

```
examine variables = fitness/cinterval=90/plot=none.
```

will produce a 90% confidence interval rather than a 95% confidence interval. There are other procedures you could use. For example, if you want the standard error of a sample mean, just list it as one of the statistics you desire in the FREQUENCIES command, as in

```
frequencies variables = fitness/statistics mean semean median model stddev.
```

which produces



Statistics

FITNESS: Hours of fitness activity i

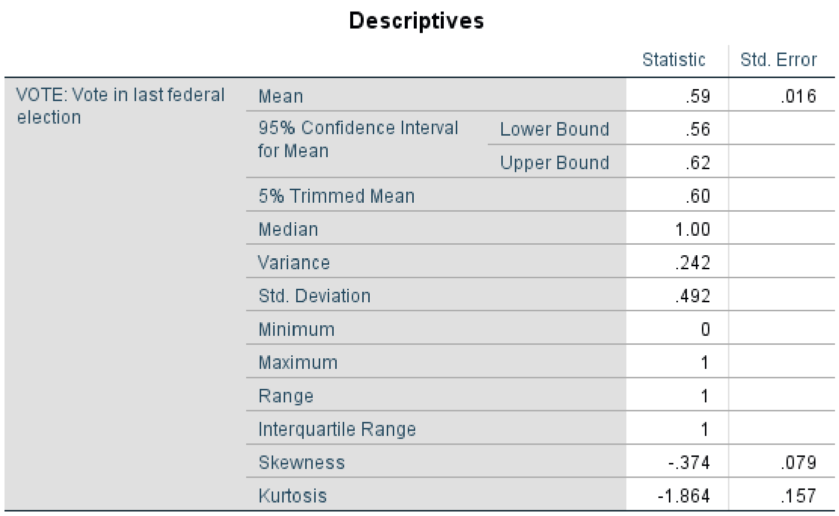
N	Valid	970
	Missing	0
Mean		4.5040
Std. Error of Mean		.13286
Median		4.0000
Mode		3.00
Std. Deviation		4.13786

If you wanted a table of means and standard deviations of a variable for different groups in the data as well as the estimated standard error of the sample mean for each group, try a command such as

```
means tables = fitness by sex/cells = mean stddev semean.
```

A confidence interval for a proportion can be constructed using the EXAMINE command by capitalizing on the fact that a proportion can be thought of as a mean of a variable coded zero and one. So suppose you ask 100 people if they intend to vote in the next federal election, coding a "no" response as vote = 0 and a "yes" response as vote = 1. Suppose in your data, 41 said no and 59 said yes, so the variable "vote" contains 41 zeros and 59 ones. A 95% confidence interval for the population proportion can be generated with

```
examine variables = vote.
```



Descriptives

			Statistic	Std. Error
VOTE: Vote in last federal election	Mean		.59	.016
	95% Confidence Interval for Mean	Lower Bound	.56	
		Upper Bound	.62	
	5% Trimmed Mean		.60	
	Median		1.00	
	Variance		.242	
	Std. Deviation		.492	
	Minimum		0	
	Maximum		1	
	Range		1	
	Interquartile Range		1	
	Skewness		-.374	.079
	Kurtosis		-1.864	.157

The “mean” displayed in the output is actually a proportion (0.59), and the 95% confidence interval for the “mean” is actually a confidence interval for the population proportion. It is 0.492 to 0.688.

Hypothesis Tests For a Single Mean or Proportion

Testing a hypothesis about a mean or a proportion is one of the staples in introductory statistics courses. Though such a test is not nearly as common in day-to-day research as tests comparing two or more means or proportions, you will eventually find an occasion to rely on one of these tests.

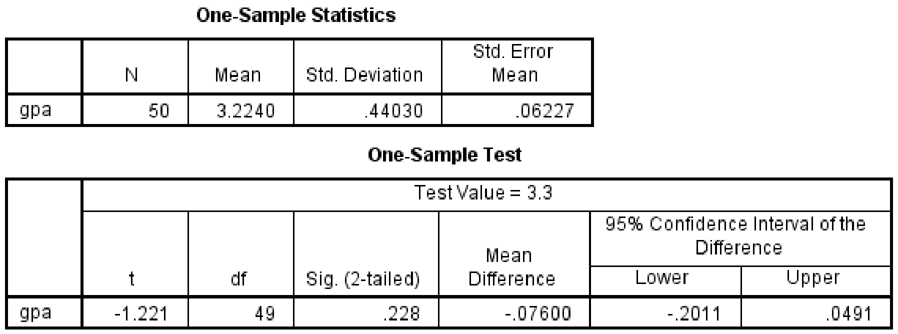
Null Hypothesis Test for a Single Mean

The one sample *t* test is most typically used to compare a single observed sample mean to a null hypothesized value to test a hypothesis about the value of the corresponding parameter. The T-TEST command in generic form is

```
t-test variable = var /testval = value.
```

where `var` is the name of the variable containing the data and `value` is the null hypothesized value of the population mean. For example, suppose you had the college grade point averages of 50 psychology majors randomly selected from your university and you wanted to test the null hypothesis that the mean GPA of psychology majors is 3.3. Assuming you had their 50 GPAs stored in the data file under a variable named “gpa”, the command below would conduct a corresponding one-sample t-test for this null hypothesis

```
t-test variable = gpa/testval = 3.3.
```



One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
gpa	50	3.2240	.44030	.06227

One-Sample Test

	Test Value = 3.3					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
gpa	-1.221	49	.228	-.07600	-.2011	.0491

Observe that the obtained mean of 3.224 corresponds to a t-ratio of 1.221 (which is 3.224 – 3.000 divided by the estimated standard error of 0.0623). Based on the t distribution with 49 degrees of freedom, this result has a two tailed p-value found under “Sig (2-tailed)” of 0.228.

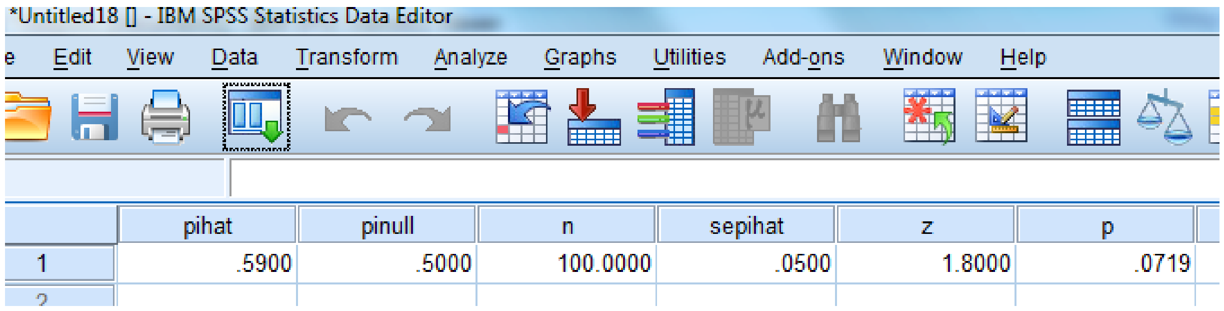
Null Hypothesis Test for a Single Proportion

Remarkably, SPSS does not have a procedure designed for testing a null hypothesis about a single proportion. However, this is a trick you can employ to get SPSS to function as a glorified calculator. You will use SPSS’s data window for displaying the results of the computations completed by syntax you write to conduct the test based on the formulas you’ll find in any introductory statistics test.

For instance, suppose that you asked 100 people whether they intend to vote in the next federal election. You want to test the null hypothesis that the population proportion is 0.50. Suppose that in the data, 59 said they intend to vote, and 41 said they did not. So your obtained proportion is 0.59. The program below conducts a test of the null hypothesis that the population proportion equals 0.5 (“pinull”) given an observed proportion (“pihat”) of 0.59 and a sample size (“n”) of 100:

```
data list free/pihat pinull n.
begin data.
0.59 0.50 100
end data.
compute sepihat=sqrt(pinull*(1-pinull))/sqrt(n).
compute z=(pihat-pinull)/sepihat.
compute p=2*(1-cdfnorm(abs(z))).
format all (F8.4).
execute.
```

When you run this program, the results of the computations and the test will be in the first row of the data window rather than in the output window, as such:


*Untitled18 [] - IBM SPSS Statistics Data Editor
e Edit View Data Transform Analyze Graphs Utilities Add-ons Window Help

	pihat	pinull	n	sepihat	z	p
1	.5900	.5000	100.0000	.0500	1.8000	.0719
2						

The number under “sepihat” is the standard error of the sample proportion assuming the population proportion is 0.50. The observed proportion of 0.59 based on 100 cases is Z = 1.80 standard errors above the null hypothesized value of 0.50. The two tailed *p*-value for such a result is 0.0719. This program is easily adapted to whatever observed proportion, null hypothesis, or sample size desired. Simply change the values in the line between “begin data” and “end data” accordingly. An example of this is given later.

A null hypothesis test for a proportion can also be construed in terms of the observed frequencies in the data, and a chi-squared goodness of fit test conducted comparing the observed frequencies to the expected frequencies if the null hypothesis is true. Suppose that your recorded each person’s answer as to whether or not she or he intended to vote as 0 = no, 1 = yes in a variable named “vote” 59 said they intend to vote (and so you have 59 1’s in the data), and 41 said they did not (meaning 41 0’s in the data). If the null hypothesis is true, then you’d expect that out of 100 people, 50 would say no and 50 would say yes. To compare the observed frequencies of 41 and 59 against their expected values of 50 and 50, use the command below:

```
npar tests/chisquare=vote.
```

The resulting output is

vote

	Observed N	Expected N	Residual
.00	41	50.0	-9.0
1.00	59	50.0	9.0
Total	100		

Test Statistics

	vote
Chi-Square	3.240[a]
df	1
Asymp. Sig.	.072

a. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 50.0.

The p-value (“Asymp. Sig”) of 0.072 for the obtained chi-square is the same as the p-value for the Z statistic calculated earlier. These are mathematically equivalent test. In fact, the chi-square statistic of 3.24 is the square of the Z statistic of 1.80.

You could conduct this test easily even if you didn’t have the 100 responses recorded but you knew 41 said no and 59 said yes. The program below will do it, relying on the WEIGHT command described earlier. It will produce the same output as the previous command.

```
data list free/vote freq.
begin data.
0 41
1 59
end data.
weight by freq.
npar tests/chisquare=vote.
weight off.
```

Using the chi-square test, if you want to test a null hypothesis for a population proportion other than 0.50, you have to figure out how many of each response you’d expect if the null hypothesis is true. For example, suppose you want to test the null hypothesis that two-thirds of the population intends to vote in the next election. Thus, out of 100 people, you would expect 66.67 people to say yes (i.e., vote = 1) and 33.33 people to say no (i.e., vote = 0). Just put these in as the expected frequencies in the NONPAR TESTS command, as below:

```
npar tests/chisquare=vote/expected = 33.33 66.67.
```

The syntax requires that the expected frequencies be listed in the order corresponding to the ordinal position of the category in the coding system. So in this example, “no” is coded vote = 0, and “yes” is coded vote = 1. So “no” is numerically lower in the coding system (0 is smaller than 1), so its expected frequency gets listed first. The resulting output is

vote

	Observed N	Expected N	Residual
.00	41	50.0	-9.0
1.00	59	50.0	9.0
Total	100		

Test Statistics

	vote
Chi-Square	2.647[a]
df	1
Asymp. Sig.	.104

a. 0 cells (0.0%) have expected frequencies less than 5. The minimum expected cell frequency is 33.3.

The equivalent code using the Z test would be

```
data list free/pihat pinull n.
begin data.
0.59 0.6667 100
end data.
compute sepihat=sqrt(pinull*(1-pinull))/sqrt(n).
compute z=(pihat-pinull)/sepihat.
compute p=2*(1-cdfnorm(abs(z))).
format all (F8.4).
execute.
```

If you execute this, you’ll find that Z =-1.627 (and observed that indeed, as noted above, the chi-square of 2.647 equals the square of Z) and p = 0.104.

Hypothesis Tests for Comparing Two Means

The *t* test introduced above for testing a hypothesis about a single mean is one of the staples of statistical inference. The *t* test has two other forms, used for comparing two means to each other.

The "Paired" or "Dependent Samples" t Test

One form is the "paired", "matched" or "dependent samples" or "dependent groups" t test, used when the same cases are measured on the same thing twice, two different things measured on the same scale are measured on the same people, or two people who are paired together in some fashion are measured on the same thing. Of interest is whether the two means are equal. For example, suppose you asked husbands and wives how satisfied they are with their marriage. If you asked them each this question, you'd have two measures of satisfaction, one from the man and one from the woman. If they were stored in variables name "satman" and "satwoman", a paired samples t test could be used to test the null hypothesis that husbands are as satisifed with their marriage, on average, as are their wives. Use the T-TEST PAIRS command, as below:

```
t-test pairs = satman satwoman.
```

If you had such data and conducted this test, the output might look something like this:

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	satman	48.7477	30	8.57046	1.56474
	satwoman	53.8910	30	10.87225	1.98499

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	satman & satwoman	30	.022	.908

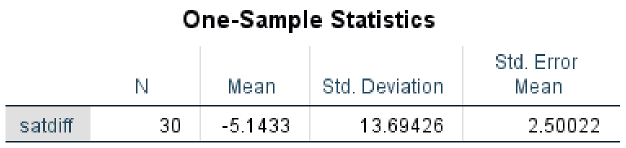
Paired Samples Test

		Paired Differences							
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper	t	df	Sig. (2-tailed)
Pair 1	satman - satwoman	-5.14334	13.69426	2.50022	-10.25686	-.02982	-2.057	29	.049

The paired t test is mathematicaly equivalent to a one-sample t test on the mean difference. In other words, rather than using the T-TEST PAIRS command, as above, you could instead construct the difference in satisfaction between the husband and wife and then test whether this difference is zero on average. The command would be

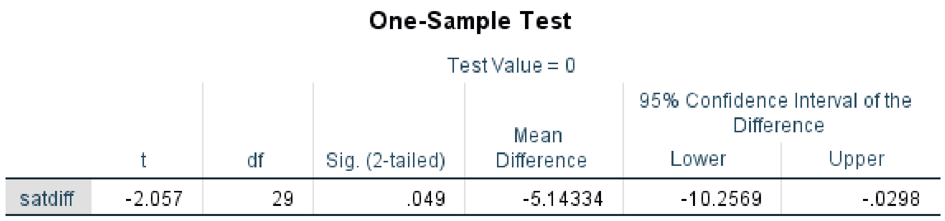
```
compute satdiff = satman-satwoman.
t-test variables = satdiff/testval = 0.
```

Compare the example output below to the output from the paired samples t test. Notice that the t statistic, degrees of freedom, and p-value are the same. Mathematically, these are the same test.



One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
satdiff	30	-5.1433	13.69426	2.50022



One-Sample Test

	Test Value = 0					
					95% Confidence Interval of the Difference	
	t	df	Sig. (2-tailed)	Mean Difference	Lower	Upper
satdiff	-2.057	29	.049	-5.14334	-10.2569	-.0298

The "Independent Groups" t Test

The occasion to compare two groups arises frequently in research, whether the groups are naturally existing (e.g., men versus women) or constructed artificially in an experiment (e.g., those assigned to a control condition and those assigned to an experimental treatment condition). The independent groups t test is used to test whether the two groups differ from each other on average. In SPSS, the syntax for the independent groups t test has the form

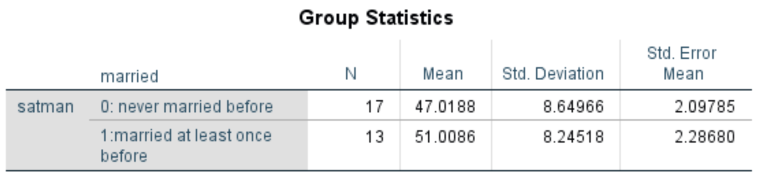
```
t-test groups=grpvar(g1,g2)/variables = varlist.
```

where *g1* and *g2* are the numerical codes stored in a variable named *grpvar* that code the two groups you want to compare, and *varlist* is a list of one or more variables containing the measurements that you wish to compare the groups on.

For example, perhaps 17 of the men in the example above were in their first marriage, whereas the other 13 had been married before. If a variable named "married" contained codes for the two groups of men, with 0 used to code never married before and 1 to code married once before, the command below would conduct an independent groups t test comparing the average marital satisfaction of men never married before to those married at least once before:

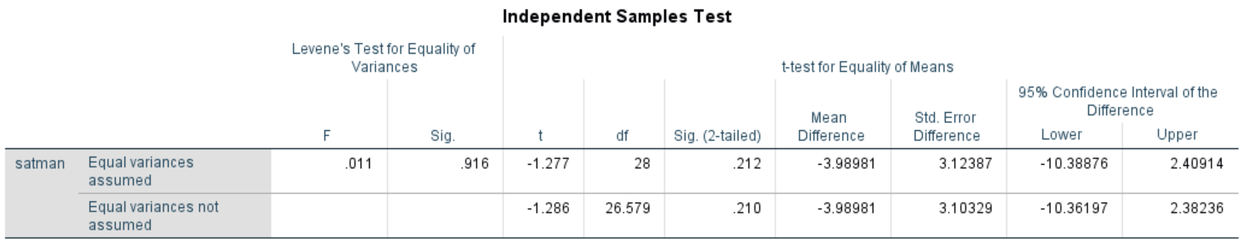
```
t-test groups=married(0,1)/variables = satman.
```

The output looks as below:



Group Statistics

	married	N	Mean	Std. Deviation	Std. Error Mean
satman	0: never married before	17	47.0188	8.64966	2.09785
	1:married at least once before	13	51.0086	8.24518	2.28680



Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
satman	Equal variances assumed	.011	.916	-1.277	28	.212	-3.98981	3.12387	-10.38876	2.40914
	Equal variances not assumed			-1.286	26.579	.210	-3.98981	3.10329	-10.36197	2.38236

The T-TEST command doesn't care how the two groups are coded, but they must be coded in the data with numbers and not string data. So, for example, you couldnt use strings such as "never married" and "married before" for `g1` and `g2` in the command above.

You could list more than one variable in the variables list if you wanted to compare the two groups on multiple variables. For example, the command

```
t-test groups=married(0,1)/variables = satman satwoman.
```

would conduct two t tests, one comparing the marital satisfaction of once married men to men who had never married before, and one comparing the average marital satisfaction of the wives of these two groups of men against each other. The two outputs would be stacked together, one on top of the other.

Hypothesis Tests Comparing Two (or More) Proportions

The χ^2 test of Independence

A test conceptually like the independent groups t test but for a dichotomous dependent variable is the chi-square test of independence. The chi-square test of independence has several uses, but one is to compare two independent proportions.

Earlier in this document, a command for generating a crosstabulation was described. Adding on the option "/statistics=chisquare" to the CROSSTABS command generates a chi-square test of independence. If both of the variables defining the rows and columns are dichotomous, this test can be interpreted as a test of equality of the two proportions that can be constructed by looking at the conditional distribution of one variable at values of the other.

For example, suppose each husband in the example above had been asked whether he ever, even once, felt any regret about marrying his current spouse. Imagine the responses are coded in a variable named "mregret", with responses coded 0 = no and 1 = yes. The CROSSTABS command below generates a crosstabulation of prior marriage and regret, while also producing the percentage who reported having felt some regret at least once. It also requests a chi-square test of independence.

```
crosstabs tables = married by mregret/cells count row/statistics=chisquare.
```

married * mregret Crosstabulation

			mregret		
			no	yes	Total
married	never married before	Count	13	4	17
		% within married	76.5%	23.5%	100.0%
	married at least once	Count	4	9	13
		% within married	30.8%	69.2%	100.0%
Total		Count	17	13	30
		% within married	56.7%	43.3%	100.0%

Chi-Square Tests

	Value	df	Asymptotic Significance (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	6.266[a]	1	.012		
Continuity Correction[b]	4.543	1	.033		
Likelihood Ratio	6.455	1	.011		
Fisher's Exact Test				.025	.016
Linear-by-Linear Association	6.057	1	.014		
N of Valid Cases	30				

a. 0 cells (0.0%) have expected count less than 5. The minimum expected count is 5.63.

b. Computed only for a 2x2 table

As can be seen in the crosstabulation, whereas only 23.5% of men who had never married before expressed having felt some regret marrying this spouse (4 out of 17), a full 69.2% of men who had married at least once before expressed having felt this (9 out of 13). These correspond to proportions of 0.235 and 0.692, respectively. The outcome of the chi-square test is found in the section of output that follows the crosstabulation. As can be seen, these two proportions are statistically different from each other, $\chi^2(1) = 6.266$, p= 0.012.

This test could be conducted with access only to the crosstabulation rather than the raw data. For instance, suppose someone delivered the crosstabulation above to you, but they did not provide the individual data coding each person's response. Using the weight command, you could entered the table in as data rather than the individual responses and conduct the test. The program below does this:

```
data list free/married mregret count.
begin data.
0 0 13
0 1 4
1 0 4
1 1 9
end data.
value labels married 0 'never married before' 1 'married at least once'.
value labels mregret 0 'no' 1 'yes'.
weight by count.
crosstabs tables = married by mregret/cells count row/statistics=chisquare.
```

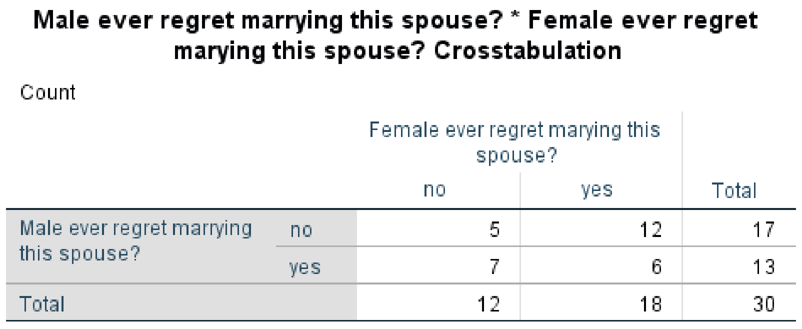
Comparing Two Dependent Proportions

Suppose each woman in each couple was also asked whether she had ever felt regret about marrying her current spouse. With responses from both husband and wife, it is possible to test whether the husbands are more or less likely than the wives to have reported feeling regret about their current marriage. We can't treat the data as if these are independent groups of men and women because they are coupled together. Each man is paired with one woman in the data. That makes the resulting proportions dependent.

The CROSSTABS command has an option for conducting a test of the difference between proportions when the data are paired in this fashion. The command below generates a crosstabulation of the husband's response against the wife's response (with the wife's response is stored in a variable names "fregret") and then conducts McNemar's test of equality of the proportions in the marginal of the table.

```
crosstabs tables = mregret by fregret/statistics=mcnemar.
```

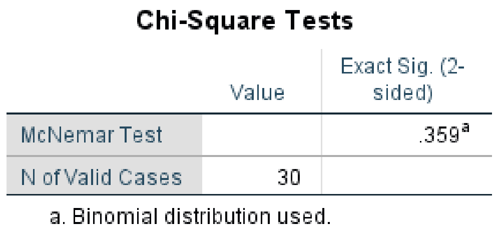
The output is below. As can be seen, 13 of the 30 husbands (or 0.433 in proportion terms) reported having felt regret at least once, whereas 18 of the 30 wives reporting having so felt (or 0.60). These two proportions are not statistically different according to McNemar's test, *p* = 0.359.



Male ever regret marrying this spouse? * Female ever regret marying this spouse? Crosstabulation

Count

		Female ever regret marying this spouse?		Total
		no	yes	
Male ever regret marrying this spouse?	no	5	12	17
	yes	7	6	13
Total		12	18	30



Chi-Square Tests

	Value	Exact Sig. (2-sided)
McNemar Test		.359[a]
N of Valid Cases	30	

a. Binomial distribution used.

Comparing More than Two Independent Means

The independent groups t test is used to compare two means to each other. When there are more than two means being compared, analysis of variance (ANOVA) is the method of choice for most researchers .

Single-Factor/One-Way ANOVA for Independent Means

When the groups are independent, meaning each case in the data is member of one and only one group, "between groups" analysis of variance is used. If the groups are defined by variation on a single dimension as opposed to more than one, a single-factor or "one-way" between-groups analysis of variance is used.

For instance, perhaps you want to compare the political knowledge of four groups of people who differ with respect to which political party they identify with---Democrats, Republicans, some other party, or "Independent." These four groups represent four levels on a single factor we might call "political party identification." Or maybe you have conducted an experiment to see if those who get cognitive-behavioral therapy to treat depression but without antidepressants differ on average in their depression six months after therapy from a group who receive the same therapy combined with antidepressant medication, or from a group that receives no therapy at all. It that case, the three therapies are levels of a single variable: therapy type (no therapy, with medication, and without mediation).

SPSS has three procedures that can conduct a single-factor between groups analysis of variance: MEANS, ONEWAY, and UNIANOVA. Suppose you have a variable named "pdiscuss" that quantifies how frequently a person reports engaging in political discussion with friends, families, neighbors, or workmates, in days per week. If you had an additional variable called "party" coding whether the person was a Republican (coded 1), Democrat (coded 2), or Independent (coded 3), the commands below would conduct a one-way between groups analysis of variance comparing the three groups in their average political discussion frequency. Using the MEANS procedure, the command to conduct the ANOVA that also produces the means in each of the groups is

```
means tables = pdiscuss by party/statistics anova.
```

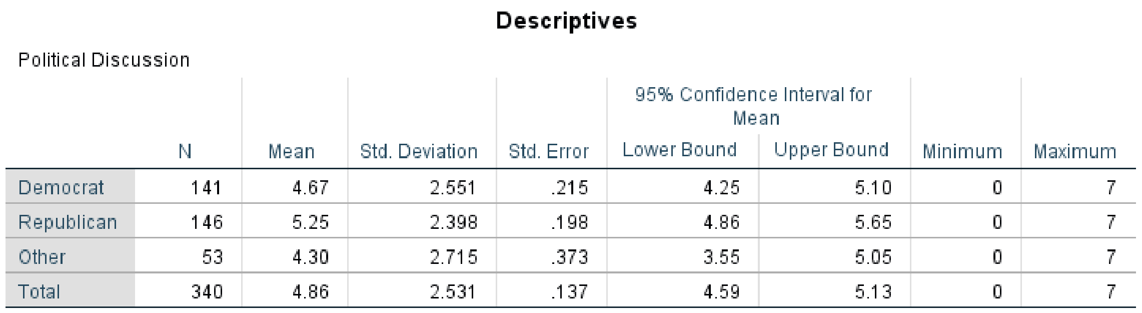
An equivalent command using the ONEWAY procedure is

```
oneway pdiscuss by party/statistics descriptive.
```

The syntax using the the UNIANOVA command would be

```
unianova pdiscuss by party/print descriptives.
```

An example output from the ONEWAY procedure is below. In this example, the evidence suggests no difference in average political discussion frequency between the three groups. The outputs from the MEANS and UNIANOVA command will look a bit different but will contain largely the same information, more or less.



Descriptives

Political Discussion

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Democrat	141	4.67	2.551	.215	4.25	5.10	0	7
Republican	146	5.25	2.398	.198	4.86	5.65	0	7
Other	53	4.30	2.715	.373	3.55	5.05	0	7
Total	340	4.86	2.531	.137	4.59	5.13	0	7

ANOVA

Political Discussion

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	43.990	2	21.995	3.484	.032
Within Groups	2127.786	337	6.314		
Total	2171.776	339			

All three of these procedures produce a test that assumes that the groups are equally variable on the dependent variable. A better test, Welch's F-test, is available that doesn't make this assumption. It is available in the ONEWAY procedure by adding "welch" as an option, as in

```
oneway pdiscuss by party/statistics descriptive welch.
```

The output looks as above but with the addition of

Robust Tests of Equality of Means

Political Discussion

	Statistic[a]	df1	df2	Sig.
Welch	3.410	2	140.364	.036

a. Asymptotically F distributed.

If you wanted to formally test equality of the variances of the groups in political discussion frequency, you can do so with the ONEWAY and UNIANOVA commands, but not with the MEANS command. For the ONEWAY command, adding the word "homogeneity" as an option in the statistics part of the command produces Levene's test, as in

```
oneway pdiscuss by party/statistics homogeneity descriptive.
```

which generates

Test of Homogeneity of Variances

Political Discussion

Levene Statistic	df1	df2	Sig.
1.384	2	337	.252

In the UNIANOVA command, add "homogeneity" to the list of print options

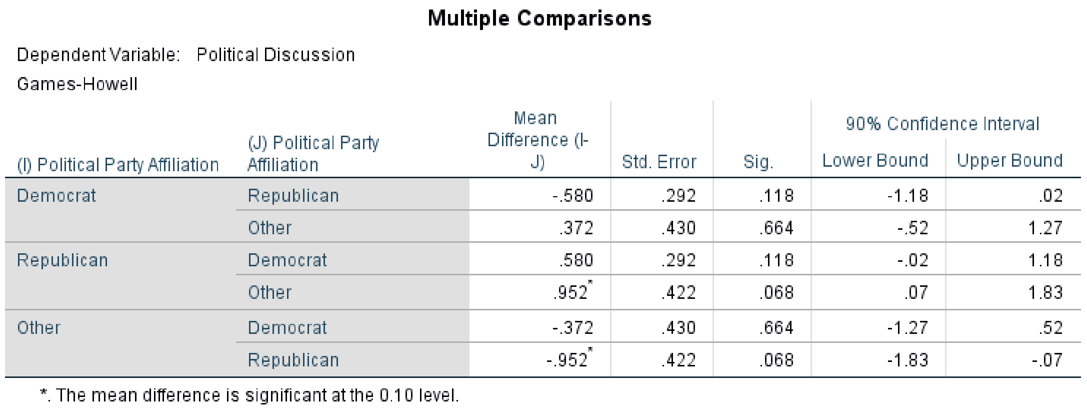
```
unianova pdiscuss by party/print homogeneity descriptives.
```

Pairwise Comparisons Between Means

It is common to follow up an analysis of variance with a comparison of subsets of the means. One approach when there are *k* groups is to conduct all of the possible *k*(*k* –1)/2 comparisons between pairs of means. There are many procedures for doing so that have been proposed that attempt to deal with the multiple test problem---inflation of the "familywise" Type I error rate that results when you conduct several hypothesis tests in an analysis. These methods go by various names---Tukey's least significant difference test, Duncan's multiple range test, Dunnett's test, Scheffe's test, the Games-Howell method, and so forth. Several of these methods are implemented in SPSS in the ONEWAY and UNIANOVA commands. For example, the "posthoc" option below adds output from the ONEWAY procedure that does all pairwise comparisons between means using the Games-Howell method, setting the familywise error rate to 0.10.

```
oneway pdiscuss by party/posthoc=gh alpha(0.10).
```

The output it generates looks as below:



Multiple Comparisons

Dependent Variable: Political Discussion
Games-Howell

(I) Political Party Affiliation	(J) Political Party Affiliation	Mean Difference (I-J)	Std. Error	Sig.	90% Confidence Interval	
					Lower Bound	Upper Bound
Democrat	Republican	-.580	.292	.118	-1.18	.02
	Other	.372	.430	.664	-.52	1.27
Republican	Democrat	.580	.292	.118	-.02	1.18
	Other	.952*	.422	.068	.07	1.83
Other	Democrat	-.372	.430	.664	-1.27	.52
	Republican	-.952*	.422	.068	-1.83	-.07

*. The mean difference is significant at the 0.10 level.

The familywise error rate defaults to 0.05 if you leave off the alpha(0.10) part of the command. Alternatively, you could set the familywise error rate to some other value if you wanted by changing the number in parentheses.

The equivalent command using the UNIANOVA procedure is

```
unianova pdiscuss by party/posthoc=party(gh)/criteria=alpha(0.10).
```

See the *Command Syntax Reference* available through the help menu for the code needed for the many different pairwise comparisons methods that SPSS offers.

Mean Comparisons Using Contrast Coefficients

Simple or complex contrasts between means in a single-factor analysis of variance can be specified using the contrast option in ONEWAY and specifying a set of contrast coefficients following the contrast option. For example, in a design with four groups coded in a variable named “web”, the command below produces two contrasts:

```
oneway time by web/contrast = -.5 -.5 .5 .5/contrast = 0 2 -1 -1/
   statistics descriptive.
```

The first contrast is the mean of the first two groups against the mean of the second two groups, and the second is the mean of the second group against the mean of the last two groups. In this description, by “first two” or “last two” means the groups with the two lowest and the two highest group codes on the variable identifying group (“web” in this example)

Comparing More than Two Dependent Means

A common research design includes the same person being measured on the same variable repeatedly, perhaps in different circumstances or in response to different stimuli. The paired samples t test was addressed earlier to compare two means that are dependent or matched, as they are when the same person is measured repeatedly on the same variable. When more than two such means are being compared, a “repeated measures” variation on the single-factor ANOVA is often used.

Single-Factor/One-Way Repeated Measures ANOVA

Suppose you have conducted a study in which you have asked each person to evaluate the appeal of three different advertisements that vary on some dimension that is being manipulated. For instance, perhaps one version contains only a single person voice-over describing the product, a second contains the same voice-over but includes classical music in the background, and a third contains the voice-over but with a modern, popular song playing in the background. Unlike in the single factor ANOVA for independent means, in this case the means are dependent because each person evaluates each of the advertisements. But the question is the same---is there a difference, on average, in how the three ads are evaluated?

In this study, perhaps you asked people to make an evaluation on a 0-100 scale, and so you have three measurements from each person scaled 0 to 100, one in response to each advertisement. So if you had 10 participants in the study, the data file would contain 10 rows and 3 columns, with each column containing the evaluation of one of the three advertisements from the person in that row.

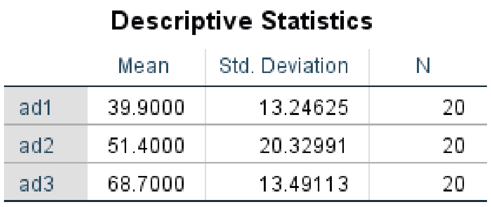
A one-way repeated measures ANOVA for *k* dependent means can be conducted using the GLM command. A generic and minimal command would look as

```
glm dv1 dv2 ... dvk/wsfactor name (k)/print=descriptive.
```

where `dv1, dv2, … dvk` are the names of the `k` variables containing the *k* repeated measurements, `name` is an arbitrary name you give to the factor you are measuring repeatedly, and *k* is the number of repeated measures. *k* should be the same as the number of variable names listed after GLM. For example, in the advertisement study, if the variables in the columns were labeled “ad1,” “ad2,” and “ad3”, the command below would conduct a single-factor repeated measures ANOVA comparing the mean evaluation of the three advertisements

```
glm ad1 ad2 ad3/wsfactor evaluate (3)/print=descriptive.
```

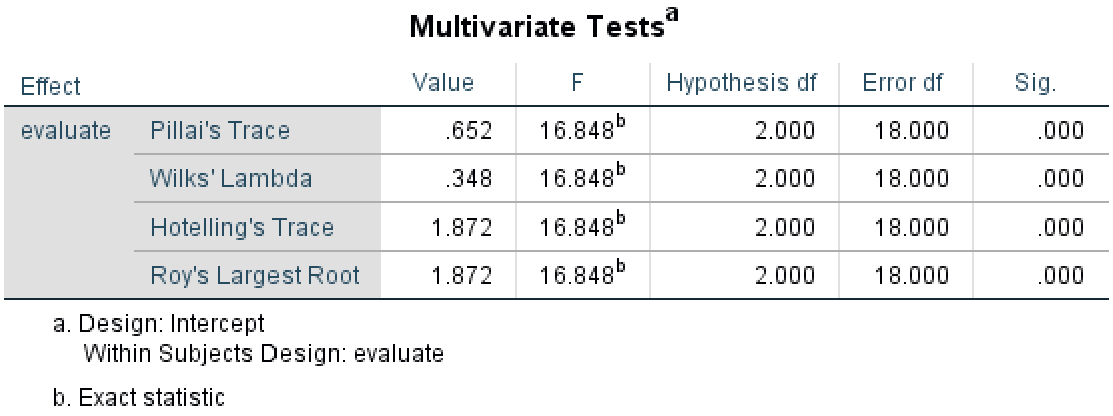
This command would produce a lot of output, only some of which is particularly important. “print=descriptives” produces a table of means, such as below



Descriptive Statistics

	Mean	Std. Deviation	N
ad1	39.9000	13.24625	20
ad2	51.4000	20.32991	20
ad3	68.7000	13.49113	20

You will also get two ANOVA summary tables, one for the multivariate approach to repeated measures:



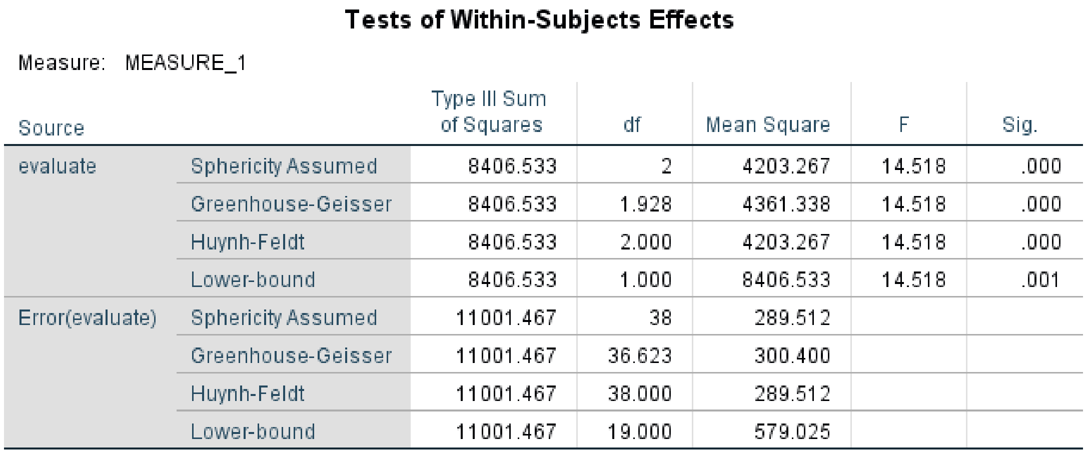
Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.
evaluate	Pillai's Trace	.652	16.848[b]	2.000	18.000	.000
	Wilks' Lambda	.348	16.848[b]	2.000	18.000	.000
	Hotelling's Trace	1.872	16.848[b]	2.000	18.000	.000
	Roy's Largest Root	1.872	16.848[b]	2.000	18.000	.000

a. Design: Intercept
Within Subjects Design: evaluate

b. Exact statistic

and one for the univariate approach:

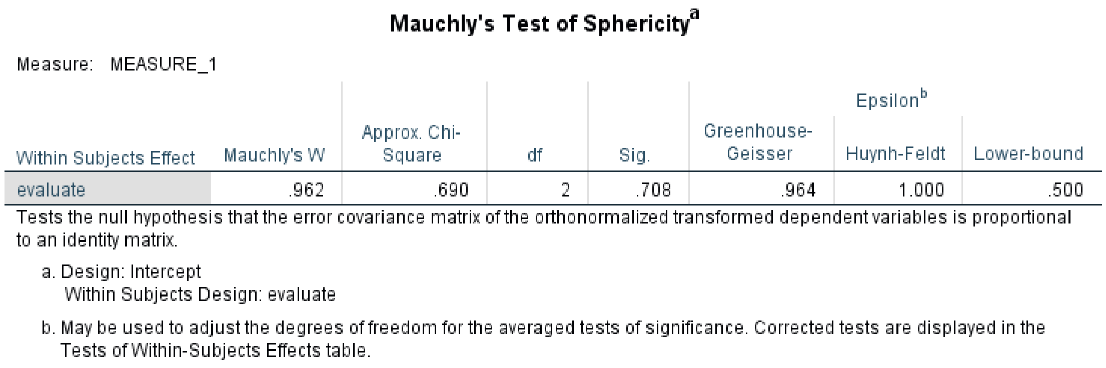


Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
evaluate	Sphericity Assumed	8406.533	2	4203.267	14.518	.000
	Greenhouse-Geisser	8406.533	1.928	4361.338	14.518	.000
	Huynh-Feldt	8406.533	2.000	4203.267	14.518	.000
	Lower-bound	8406.533	1.000	8406.533	14.518	.001
Error(evaluate)	Sphericity Assumed	11001.467	38	289.512		
	Greenhouse-Geisser	11001.467	36.623	300.400		
	Huynh-Feldt	11001.467	38.000	289.512		
	Lower-bound	11001.467	19.000	579.025		

Near the univariate approach summary table you will also find a test of sphericity:



Mauchly's Test of Sphericity[a]

Measure: MEASURE_1

					Epsilon[b]		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
evaluate	.962	.690	2	.708	.964	1.000	.500

Tests the null hypothesis that the error covariance matrix of the orthonormalized transformed dependent variables is proportional to an identity matrix.

a. Design: Intercept
Within Subjects Design: evaluate

b. May be used to adjust the degrees of freedom for the averaged tests of significance. Corrected tests are displayed in the Tests of Within-Subjects Effects table.

The rejection of the null hypothesis of equality of the means yields a vague conclusion. If you want to do specific comparisons between means, you can use a number of different coding systems for setting up these comparisons. For instance,

```
glm ad1 ad2 ad3/wsfactor evaluate (3) simple (2)/print=descriptive.
```

would add output comparing the first to the second mean (ad1 vs ad2) and the third to the second mean (ad3 vs ad2), as below.

Tests of Within-Subjects Contrasts

Measure: MEASURE_1

Source	evaluate	Type III Sum of Squares	df	Mean Square	F	Sig.
evaluate	Level 1 vs. Level 2	2645.000	1	2645.000	4.277	.053
	Level 3 vs. Level 2	5985.800	1	5985.800	9.203	.007
Error(evaluate)	Level 1 vs. Level 2	11749.000	19	618.368		
	Level 3 vs. Level 2	12358.200	19	650.432		

The number in parentheses following "simple" specifies the reference for comparison, with the number corresponding to the position in the list of variables following "GLM" that you are choosing as the reference. Other options include Helmert, Repeated, Difference, and a few others (see the *Command Syntax Reference* in the SPSS Help menu). A set of contrast coefficients can also be specified, as in

```
glm ad1 ad2 ad3/wsfactor evaluate (3) special (1 1 1 -1 .5 .5 0 -.5 .5)/
    print descriptive.
```

which would generate two contrasts, one comparing the mean of "ad1" to the combined mean of "ad2" and "ad3" (L1 in the output below), as well one comparing "ad2" to "ad3" (L2 in the output below).

Tests of Within-Subjects Contrasts

Measure: MEASURE_1

Source	evaluate	Type III Sum of Squares	df	Mean Square	F	Sig.
evaluate	L1	8120.450	1	8120.450	21.330	.000
	L2	1496.450	1	1496.450	9.203	.007
Error(evaluate)	L1	7233.550	19	380.713		
	L2	3089.550	19	162.608		

Designs that Include Dependent and Independent Means

Imagine a variation on the study design described in the prior section. Suppose that the three advertisements took two different forms each—audio only, or audio and video. Further, suppose that you randomly assigned people to the audio only condition or the audio and video condition. That is, half of the participants evaluate the three advertisements in audio form only, and the other half of the participants evaluate the three advertisements in video form with audio.

In such a design, the means are dependent within levels of audio or audio plus video (because the three responses come from the same person), but across these levels they are independent (because the people assigned to different conditions are independent of one other). This is a "mixed" design, where the evaluations are repeated or "within-subject" across levels of the music condition (no music, classical music, or popular music) but "between-subject" across levels of the format (audio, or audio plus video).

This kind of design allows for the simultaneous testing of difference between (1) the means of the three ads that vary as a function of music type, (2) the means of ads in audio versus audio and video format, and (3) the interaction between music type and format. When there is one within and one between subject factor as in this design, the generic format of the SPSS command for a "mixed model" analysis of variance is

```
glm dv1 dv2 ... dvk by iv/wsfactor name (k)/print=descriptive.
```

where all arguments are defined as in the prior example, and *iv* is the name of the variable coding the between-subjects factor. For example, if the variable in the data coding whether the person was assigned to the audio or the audio+video format was named "format", the command would be

```
glm ad1 ad2 ad3 by format/wsfactor evaluate (3)/print=descriptive.
```

This would produce a table of means containing the mean response to each ad in each of the two format conditions:

Descriptive Statistics

	format	Mean	Std. Deviation	N
ad1	audio	43.9000	15.07352	10
	audio plus video	35.9000	10.37572	10
	Total	39.9000	13.24625	20
ad2	audio	54.4000	19.58571	10
	audio plus video	48.4000	21.65487	10
	Total	51.4000	20.32991	20
ad3	audio	69.2000	12.33604	10
	audio plus video	68.2000	15.21549	10
	Total	68.7000	13.49113	20

an ANOVA summary table using the multivariate approach for factors involving the repeated measures variable (evaluate):

Multivariate Tests[a]

Effect		Value	F	Hypothesis df	Error df	Sig.
evaluate	Pillai's Trace	.658	16.371[b]	2.000	17.000	.000
	Wilks' Lambda	.342	16.371[b]	2.000	17.000	.000
	Hotelling's Trace	1.926	16.371[b]	2.000	17.000	.000
	Roy's Largest Root	1.926	16.371[b]	2.000	17.000	.000
evaluate * format	Pillai's Trace	.028	.248[b]	2.000	17.000	.783
	Wilks' Lambda	.972	.248[b]	2.000	17.000	.783
	Hotelling's Trace	.029	.248[b]	2.000	17.000	.783
	Roy's Largest Root	.029	.248[b]	2.000	17.000	.783

a. Design: Intercept + format
Within Subjects Design: evaluate

b. Exact statistic

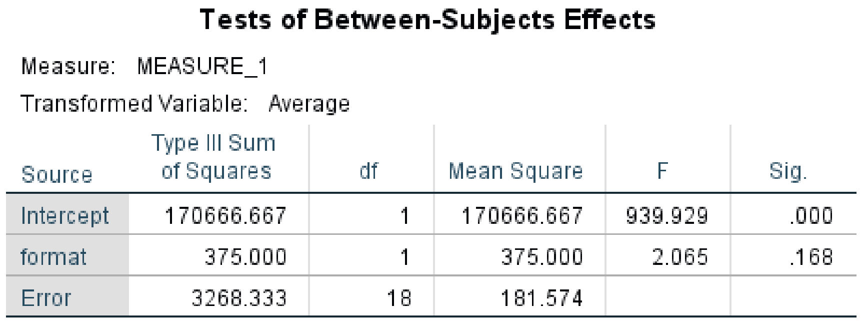
an ANOVA summary table based on the univariate approach to repeated measures:

Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
evaluate	Sphericity Assumed	8406.533	2	4203.267	13.919	.000
	Greenhouse-Geisser	8406.533	1.919	4381.438	13.919	.000
	Huynh-Feldt	8406.533	2.000	4203.267	13.919	.000
	Lower-bound	8406.533	1.000	8406.533	13.919	.002
evaluate * format	Sphericity Assumed	130.000	2	65.000	.215	.807
	Greenhouse-Geisser	130.000	1.919	67.755	.215	.799
	Huynh-Feldt	130.000	2.000	65.000	.215	.807
	Lower-bound	130.000	1.000	130.000	.215	.648
Error(evaluate)	Sphericity Assumed	10871.467	36	301.985		
	Greenhouse-Geisser	10871.467	34.536	314.786		
	Huynh-Feldt	10871.467	36.000	301.985		
	Lower-bound	10871.467	18.000	603.970		

and an ANOVA summary table for the factor that is between-subjects (format in this example):


Tests of Between-Subjects Effects

Measure: MEASURE_1
Transformed Variable: Average

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Intercept	170666.667	1	170666.667	939.929	.000
format	375.000	1	375.000	2.065	.168
Error	3268.333	18	181.574		

The GLM command contains many options for comparisons between "main effect" means involving only the between-subjects factor (in this case, with only levels of the between-subject factor, there are no comparisons to do), and the same options are available for comparisons between means involving the repeated factor that are described in the prior example. See the *Command Syntax Reference.*

Linear Regression Analysis

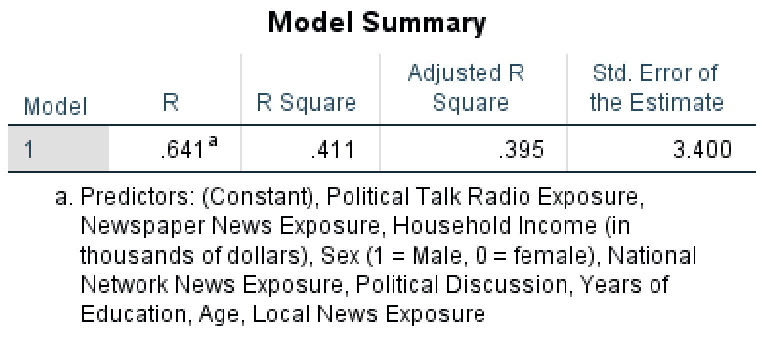
Linear regression analyis has many uses, all of which are based on the production of an equation to estimate one variable from another variable or set of variables. The most common form is ordinary least squares regression, which is programmed into SPSS's REGRESSION command. The regression command follows the structure

```
regression/dependent=dv/method=enter predictor variables.
```

where `dv` is the *outcome*, *criterion*, or *dependent* variable, and `predictor variables` is a list of one or more *independent*, *predictor*, or *regressor* variables. For instance, if you want to construct a model estimating scores on a test of political knowledge in your data (dependent variable "pknow") from a set of demographic predictor variables such as a person's sex, age in years, income, years of formal education ("educ"), and a set of variables quantifying media use such as frequency of newspaper reading ("npnews"), watching the local news on television ("locnews"), watching the national network news broadcast ("natnews") talking about politics with others ("pdiscuss"), and listening to political talk radio ("talkrad"), the command would be

```
regression/dependent=pknow/method=enter sex age income educ npnews locnews
   natnews pdiscuss talkrad.
```

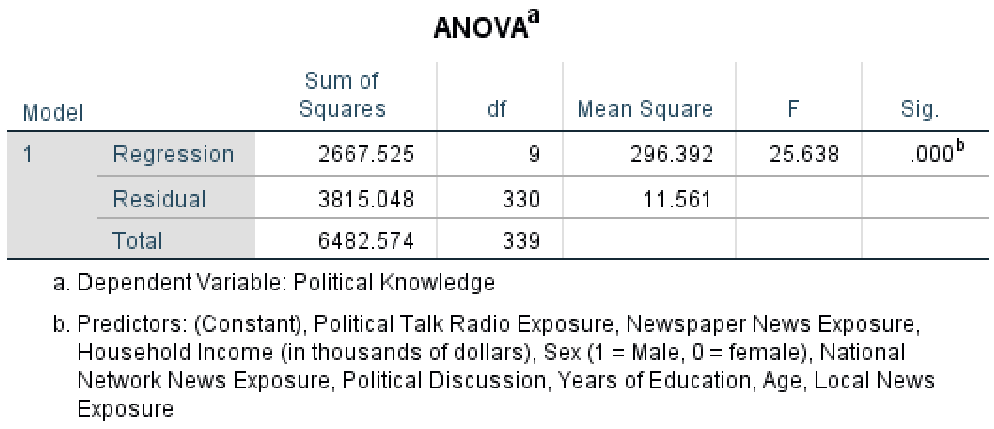
which generates three main sections of output containing information about the fit of the model, the model itself, and various things pertinent to hypothesis testing and inference:



Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.641[a]	.411	.395	3.400

a. Predictors: (Constant), Political Talk Radio Exposure, Newspaper News Exposure, Household Income (in thousands of dollars), Sex (1 = Male, 0 = female), National Network News Exposure, Political Discussion, Years of Education, Age, Local News Exposure

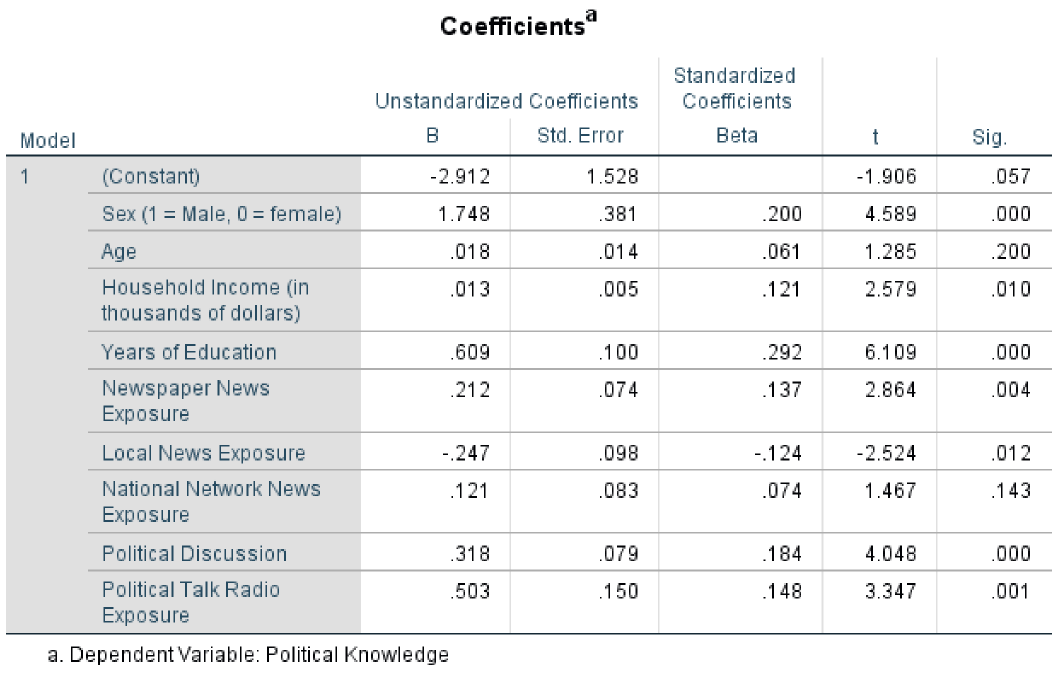


ANOVA[a]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	2667.525	9	296.392	25.638	.000[b]
	Residual	3815.048	330	11.561		
	Total	6482.574	339			

a. Dependent Variable: Political Knowledge

b. Predictors: (Constant), Political Talk Radio Exposure, Newspaper News Exposure, Household Income (in thousands of dollars), Sex (1 = Male, 0 = female), National Network News Exposure, Political Discussion, Years of Education, Age, Local News Exposure



Coefficients[a]

		Unstandardized Coefficients		Standardized Coefficients		
Model		B	Std. Error	Beta	t	Sig.
1	(Constant)	-2.912	1.528		-1.906	.057
	Sex (1 = Male, 0 = female)	1.748	.381	.200	4.589	.000
	Age	.018	.014	.061	1.285	.200
	Household Income (in thousands of dollars)	.013	.005	.121	2.579	.010
	Years of Education	.609	.100	.292	6.109	.000
	Newspaper News Exposure	.212	.074	.137	2.864	.004
	Local News Exposure	-.247	.098	-.124	-2.524	.012
	National Network News Exposure	.121	.083	.074	1.467	.143
	Political Discussion	.318	.079	.184	4.048	.000
	Political Talk Radio Exposure	.503	.150	.148	3.347	.001

a. Dependent Variable: Political Knowledge

The word “dependent” in the command can be shorted to “dep” if desired.

There are many options available in the REGRESSION command to add information to the output. One of these is the statistics option, *which must be placed between* “regression”and “dependent.” For example

```
regression/statistics defaults zpp tol bcov ci(95)/dep = pknow
  /method = enter sex age income educ npnews locnews natnews pdiscuss
   talkrad.
```

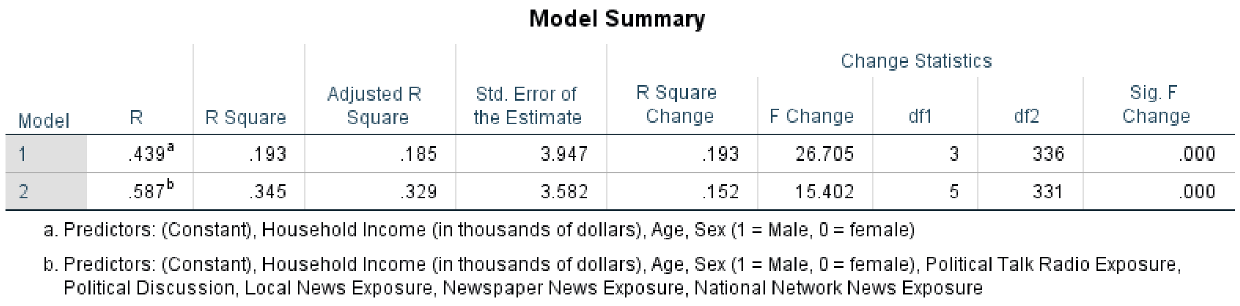
includes four sets of additional statistics in various places in the output. The “zpp” option adds the simple (aka “zero order”), partial, and semipartial correlations between the predictors and the outcome variable to the output; “tol” includes the tolerance and variance inflation factors for the predictor variables; “bcov” generates a matrix of correlations and covariances between regression coefficients, and “ci(95)” adds 95% confidence intervals for the regression coefficients (change the “95” to something else to chance the confidence interval width. 95 is the default, so if you leave it off you’ll get 95% confidence intervals). The word “defaults” is listed as well because as soon as you use the “statistics” option in a REGRESSION command, SPSS won’t print the things it would usually print without it. So by including “defaults” in the list of statistics options, SPSS generates all the stuff it would normally generate along with the additional things you are asking it to produce.

Hierarchical Entry or Removal of Predictors

Hierarchical entry of predictors in a regression model is primarily used to assess how well the model improves by when one or more (usually more than one) variables is added to an existing model. Increase in fit is often quantified using the change in the multiple correlation and corresponding test of significance. To tell SPSS to add variables to a model and provide output pertinent to improvement in fit, add these variables in an additional METHOD = ENTER command and include “change” in the “statistics” option. For example, the code below

```
regression/statistics = defaults change/dep = pknow/method = enter sex age
  income/method = enter npnews locnews natnews pdiscuss talkrad.
```

produces two regression models. The first includes sex, age, education, and income as predictors of pknow. The second includes sex, age, income, and education as the first model, but also adds npnews, locnews, natnews, pdiscuss, and talkrad as predictors. These are show in the output as "Model 1"and "Model 2" respectively. SPSS also generates a section of output as below that includes information about change in the fit of the model as the variables in the second "method = enter" list are added to the model including the variables in the prior "method = enter" list. In this example, adding the five variables in the second list increased the squared multiple correlation by 0.087, which is statistically significant, *F*(5,331) = 15.402, *p* < .001.



Model Summary

					Change Statistics				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	R Square Change	F Change	df1	df2	Sig. F Change
1	.439[a]	.193	.185	3.947	.193	26.705	3	336	.000
2	.587[b]	.345	.329	3.582	.152	15.402	5	331	.000

a. Predictors: (Constant), Household Income (in thousands of dollars), Age, Sex (1 = Male, 0 = female)

b. Predictors: (Constant), Household Income (in thousands of dollars), Age, Sex (1 = Male, 0 = female), Political Talk Radio Exposure, Political Discussion, Local News Exposure, Newspaper News Exposure, National Network News Exposure

If desired, another set of variables could be added to the second model by including an additional "method = enter" variable set following the first two.

This approach to building a model gives the data analyst complete control over the order in which the variables are added to the model. Another less common approach is *stepwise entry*, which lets a statistical algorithm figure out what variables to include in the model, with the goal of producing a model that has as few variables as possible from a list provided to the algorithm that maximizes the multiple correlation without including a bunch of variables that aren't really needed. "Forward selection" first starts with a model with no predictors and then adds predictors in the list one at a time until doing so no longer increases the multiple correlation to a statistically significant degree. Forward selection is requested by changing "enter" in the method option to "forward" as in

```
regression/dep = pknow/method = forward sex age income educ npnews locnews
   natnews pdiscuss talkrad.
```

An alternative is to first put all the predictor variables in the regression command into the model and then remove them one at a time until no more variables can be removed without significantly worsening the fit of the model. This is called "backward selection." It is specified by replacing "forward" with "backward" in the command above.

Although the forward and backward entry methods give a statistical algorithm rather than you control over what variables end up in the model, you can influence the criteria that SPSS uses for making the decision about what variables to include and exclude. See the *Command Syntax Reference* for details.

Saving Predicted Values, Residuals, and/or Casewise Diagnostics

The regression model can be used to produce estimates of the dependent variable from a weighted sum of the predictors. These estimates could be generated using a COMPUTE command, manually typing in the equation into a syntax line with variable names and regression weights. The residuals could also be generated easily by subtracting the actual values of the dependent variable for each case from each case's estimated value from the regression model. For example, the command

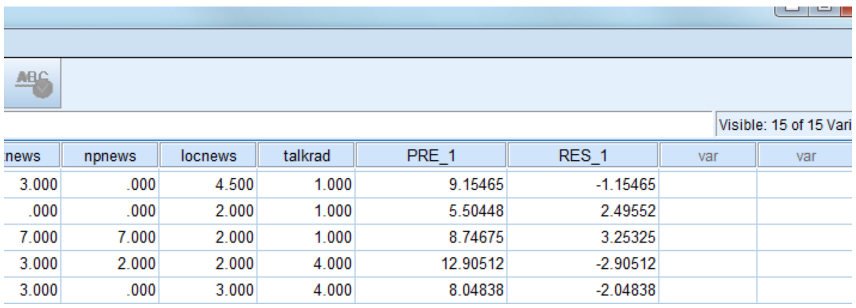
```
compute pknowest=2.912+1.748*sex+0.018*age+0.013*income+0.609*educ+
  0.212*npnews-0.247*locnews+0.121*natnews+0.318*pdiscuss+0.503*talkrad.
compute pknowres=pknow-pknowest.
```

would produce a new variable in the data called "pknowest" with values for each case representing that person's estimated political knowledge from the regression model shown in the earlier output, along with the residual—the difference between his or her actual political knowledge as measured and the estimate from the regression model.

Although this works, it is a bit tedious. SPSS has an option in the regression module to produce these automatically without having to go through the hassle of manually typing in the regression equation. To do so, add "pred" (for unstandardized predicted value) and "resid" (for unstandardized residual) to the save option, as in

```
regression/dependent=pknow/method=enter sex age income educ npnews locnews
   natnews pdiscuss talkrad/save pred resid.
```

When this is done, SPSS will add two variables to the data set containing the predicted values of the outcome variable (pre_1) and the residuals (res_1) for each case, as below.



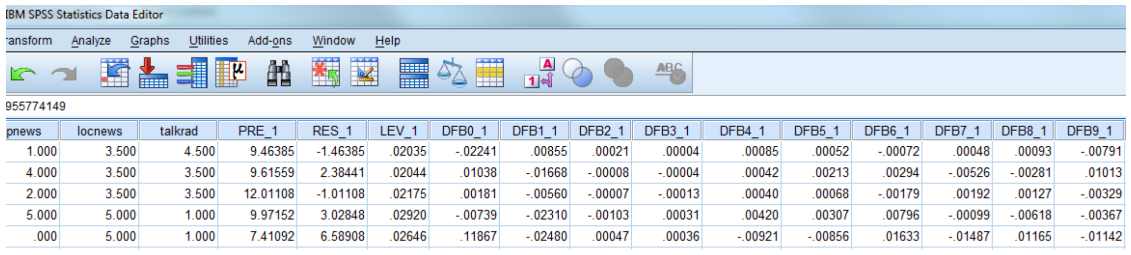
news	npnews	locnews	talkrad	PRE_1	RES_1	var	var
3.000	.000	4.500	1.000	9.15465	-1.15465		
.000	.000	2.000	1.000	5.50448	2.49552		
7.000	7.000	2.000	1.000	8.74675	3.25325		
3.000	2.000	2.000	4.000	12.90512	-2.90512		
3.000	.000	3.000	4.000	8.04838	-2.04838		

If "pred_1" and "res_1" already exist in the data, SPSS will pick different names for the predicted values and residuals.

The residuals are often used for checking whether various assumptions of linear regression are met. Another useful set of statistics SPSS can generate pertain to how much influence or potential influence a case in the data has on the resulting regression model. SPSS can produce a case's "leverage," as well as statistics called "dfbeta", which quantify how much each regression coefficient changes by including the case in the analysis. For example, adding "leverage" and "dfbeta" to the list of options in save, as in

```
regression/dependent=pknow/method=enter sex age income educ npnews locnews
   natnews pdiscuss talkrad/save pred resid leverage dfbeta.
```

produces each case's leverage in the data, as well as 9 dfbeta variables (one for the regression constant and one for each regression coefficient for each predictor) that quantifies how much each regression coefficient changes when a case is in the analysis compared to when it is not:


IBM SPSS Statistics Data Editor
ransform Analyze Graphs Utilities Add-ons Window Help

955774149

pnews	locnews	talkrad	PRE_1	RES_1	LEV_1	DFB0_1	DFB1_1	DFB2_1	DFB3_1	DFB4_1	DFB5_1	DFB6_1	DFB7_1	DFB8_1	DFB9_1
1.000	3.500	4.500	9.46385	-1.46385	.02035	-.02241	.00855	.00021	.00004	.00085	.00052	-.00072	.00048	.00093	-.00791
4.000	3.500	3.500	9.61559	2.38441	.02044	.01038	-.01668	-.00008	-.00004	.00042	.00213	.00294	-.00526	-.00281	.01013
2.000	3.500	3.500	12.01108	-1.01108	.02175	.00181	-.00560	-.00007	-.00013	.00040	.00068	-.00179	.00192	.00127	-.00329
5.000	5.000	1.000	9.97152	3.02848	.02920	-.00739	-.02310	-.00103	.00031	.00420	.00307	.00796	-.00099	-.00618	-.00367
.000	5.000	1.000	7.41092	6.58908	.02646	.11867	-.02480	.00047	.00036	-.00921	-.00856	.01633	-.01487	.01165	-.01142

There are some other statistics for each case that can be generated using the save option, such as standardized residuals, standardized predicted values, and so forth. See the *Command Syntax Reference* for detail.

Generating Indicator ("Dummy") Variables

The variables in the regression command used as predictors must be either dichotomous or a quantitative dimension of some kind. To put categorical variables with more than two categories into a regression model, the categorical variable must be represented with a set of new variables representing membership in the group. There are many ways of representing multicategorical variables in this fashion. Indicator or "dummy" coding is the most commonly used method. For instance, if you have a variable in your data set called "religion" with four arbitrary numerical values representing a person's religious affiliation (1 = Protestant, 2 = Catholic, 3 = Jewish, 4 = Other), an indicator coding system with "Other" as the reference group could be constructed with the code below:

```
compute d1 = 0.
compute d2 = 0.
compute d3 = 0.
if (religion = 1) d1 = 1.
if (religion = 2) d2 = 1.
if (religion = 3) d3 = 1.
```

The result of this syntax is three new variables d1, d2, and d3 that code a person's religion. These can be entered into the regression model as a representation of a religion. Note that although it appears that "Others" are not represented in this system, they are. Anyone who identifies with some other religion is represented with zeros on d1, d2, and d3.

Although the example code above works, it is potentially very dangerous to produce indicator variables in this fashion if there are any missing data on the variable you are coding. In this example, the first three lines of codes will default all cases to be coded as Protestants. But if a case is missing on "religion," none of the indicator codes get changed, making cases missing on religion coded as if they are Protestants. A safer approach is

```
compute d1 = (religion = 1).
compute d2 = (religion = 2).
compute d3 = (religion = 3).
```

Here, SPSS evaluates the argument in parentheses as true or false and assigns a zero to cases for which the argument is false, and a one to cases for which the argument is true. Importantly, if a case is missing on any variable in the argument, the argument is not evaluated and so the indicator is set to missing. Therefore, any case that is missing on religion ends up missing on all three indicators rather than all zeros.

Commenting Your Code

At the beginning of this book, I mentioned that one of the reasons for learning syntax is that the code you have written functions as natural documention of your thinking when you are preparing for and conducting an analysis. The code itself serves as a means of communicating with yourself and others what you are doing. But sometimes it is useful to include comments to yourself and others to more clearly convey what you are doing with a particular command or set of commands. This can be done by adding comments in various parts in your code. In SPSS syntax, comments are denoted in one of three ways. The simplest is to start the comment with an asterick (*), as in

```
* This conducts the main regression analysis we care about.
regression/dep=pknow/method=enter age income pdiscuss.
```

Or you can just start the comment with the word “comment”:

```
comment This conducts the main regression analysis we care about.
regression/dep=pknow/method=enter age income pdiscuss.
```

When using an asterick or the comment command, the comment can extend across multiple lines. End the comment with a period (.) to tell SPSS that what follows is not a part of the comment. For example:

```
* This conducts the main regression analysis we care about, predicting
knowledge from age, income, and discussion. The results are important.
regression/dep=pknow/method=enter age income pdiscuss.
```

Notice that the period in the second line of the comment is acceptable because it is not at the end of the line. Only the period at the end of a line will be interpreted as denoting the end of the comment. So this would produce an error

```
* This conducts the main regression analysis we care about, predicting
knowledge from age, income, and discussion.
The results are important.
regression/dep=pknow/method=enter age income pdiscuss.
```

because SPSS interprets the first period on the second line as the end of the comment (as the period occurs at the end of the line), meaning that “The results are important” is interpreted as SPSS command rather than a comment. There is no “The” command in SPSS, so an error is produced. Instead, you would need to write

```
* This conducts the main regression analysis we care about, predicting
knowledge from age, income, and discussion.
* The results are important.
regression/dep=pknow/method=enter age income pdiscuss.
```

Another approach is to place your comments between the symbols /* and */. The text between these symbols will not be treated as instructions for SPSS and will be ignored. But such a comment must be no more than one line long. For example

```
/* This conducts the main regression analysis we care about */.
regression/dep=pknow/method=enter age income pdiscuss.
```

The */ is actually optional if the comment is on its own line. So you can just say

```
/* This conducts the main regression analysis we care about.
regression/dep=pknow/method=enter age income pdiscuss.
```

The benefit of using /* */ is that you can place comments within SPSS commands, but when you do this, */ is not optional:

```
regression/dep=pknow/ /* the dependent variable is knowledge */
  method=enter age income pdiscuss.
```

However, when you comment within an SPSS command like this, the comment can't be set on its own line. Doing so will produce an error because SPSS will interpret */ as the end of the regression command. So this is not acceptable:

```
 regression/dep=pknow/
 /* the dependent variable is knowledge */
method=enter age income pdiscuss.
```

The */ remains optional if the comment is at the end of an SPSS command but on the same line as that command. So this is acceptable:

```
regression/dep=pknow/method=enter age income pdiscuss.  /* good results!
```