R: downloading and installing by Robert Stinerock

What is R? R is an extremely powerful and widely-used statistical package which is made available free of charge. It can be used on a variety of platforms such as Windows XP (and above), Linux, Unix, and Apple Mac Macintosh OS X. Its particular strengths are in statistical analysis, graphical representations, and (even) basic computations.

Why are we using R rather than some other statistical software package? We use R because it is more powerful that Microsoft Excel and cheaper-by-far than SAS or SPSS. Moreover, as Joseph Adler points out in his book (*R in a Nutshell: A Desktop Quick Reference* O'Reilly Media, Inc. Cambridge. 2010): "R is used at the world's largest technology companies (including Google, Microsoft, and Facebook), the largest pharmaceutical companies (including Johnson & Johnson, Merck, and Pfizer), and at hundreds of other companies. It's used in statistics classes at universities around the world and by statistics researchers to try new techniques and algorithms." Writing in the New York Times, Ashlee Vance (New York Times, January 6, 2009) adds to this: "R…is used by a growing number of data analysts inside corporations and academia. It is becoming their lingua franca partly because data mining has entered a golden age, whether being used to set ad prices, find new drugs more quickly or fine-tune financial models. Companies as diverse as Google, Pfizer, Merck, Bank of America, the Intercontinental Hotels Group and Shell use it. ….The financial community has demonstrated a particular affinity for R; dozens of packages exist for derivatives analyses alone."

To see the entire article, "Data Analysts Captivated by R's Power" by Ms. Vance: http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html

How do I get my own free version of R?

To obtain your version, simply click on the link (below) and follow the directions:

http://www.r-project.org/

Once you get to this location, note the area in the upper left-hand corner named "Download, Packages." Just below that heading, click on the CRAN link. You are presented with the list of hosts called "CRAN Mirrors" (arranged by geographic location) which contain identical R content. You should click on the location closest to your own because the download will typically be faster. For example, since I live in New York City, I use the link for:

http://software.rc.fas.harvard.edu/mirrors/R/

If you are a Windows user, click on “Windows” and then on “base.” You should be presented with a “Download R for Windows” link. If you are a Mac user, click on the “MacOS X” link, and then on the latest pkg file.

Click on the link, and then save to your hard drive. Once the download is complete you should double click on the executable file. You can safely accept all installation defaults.

The installation process should create an R icon on your desktop. After double clicking on the R icon, you will be presented with the following blurb and the R environment prompt, “>,” below at the very bottom.

```
R version 2.9.1 (2009-06-26)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

>
```

Whenever you see the R environment prompt, '>,' you are ready to start writing your R commands.

To exit R, simply type 'q' at the prompt, '>,' and select 'n' if you don't wish to save your work, or files; select 'y' if you do wish to save them. They will reappear the next time you call up R for another session.

R: entering data

I. Basic File Management: **Windows**

As a first step, you should create a directory to hold all your R data, R files, and R documents. (I have simply placed mine in "My Documents," and I have named it "Rfiles.") You should name this directory anything that is easy to remember, and is evocative of its purpose (i.e., to serve as a storage place for all R related materials).

Before leaving this directory, you should copy and paste the entire path to your R directory. (This can be found whenever you are in the R directory itself: just go to the 'address' line above, and copy-and-paste from that.) The reason why you will want this path address copied-and-pasted is that it will make life easier when you want to read and write data files to-and-from the R environment. Here is what we need to do:

- Locate your R shortcut, either in your Start Menu or on your Desktop
- Right-click on it and wait until a Menu appears
- When it does appear, find the "Properties" entry and left-click on it
- When the dialog appears, find the "Start in" line
- When you do, paste the path address to your R file right into the text box next to the words "Start in."
- Click "Apply" and "OK"

How to read a data set (or 'data frame') into the R environment:

There are several ways of getting data into R but one of the most convenient is to (1) enter the data into an Excel Workbook; (2) save the Excel data file in the 'common separated variable' format; and (3) 'read' this file directly into R. Here is an example of how this is done:

Method 1

- Create your Excel data file and save it under the name 'trial.xls.' (I picked the filename 'trial' at random; there is nothing special about it.) When saving it,

however (File>Save As…), enter 'trial' in the 'Save As' area, click the down arrow in 'Format,' and then select either 'Windows Comma Separated (.csv)' or 'MS-DOS Comma Separated (.csv).' Note that the file name in the 'Save As' area is now 'trial.csv.' Finally, click the down arrow in 'Where' and select 'Rfiles.' (I am assuming that you have completed the steps described above.) When you open the directory 'Rfiles,' you should see the document 'trial.csv.' If you click on it, it will come up in the Excel format in which it was created.

- To open the data file in R, click on the R icon. As the R prompt, ">," write the following command:

  >trial<-read.csv('trial.csv')

  If this is correctly executed, you will get back the R prompt: ">." To see the filename, enter (at the R prompt): 'ls().' All the data files currently loaded into R will appear (although at this point, the only filename that should show up is 'trial.') To see the actual data set, simply type (again, at the R prompt) the data set name, 'trial' (but without the quotes). The entire data set should appear on the screen. You are now ready to perform your basic analysis.

Method 2

Another, more cumbersome and (therefore) less often used, method of entering data is to simply enter it while in R. This works particularly well when we wish to create a small data set. For example, we might want to create a vector of numbers, or a vector variable, x, which consists of the simple array of the following five numbers: 57, 61, 38, 16, and 84.

x<-c(57,61,38,16,84)

We have here instructed R to create a vector, called "x," which consists of the above five numbers. To see the filename, x, we simply enter (at the R prompt): 'ls().' Now we will see 'x' listed, along with 'trial.' And when we enter "x" at the R prompt, we get back the row of numbers assigned to 'x.'

How to write a data set (or 'data frame') from the R environment:

In order to write your data set (from the R environment) to your R folder:

> write.csv(trial,file='hope.csv')

What we have done here is write the data set 'trial' from the R environment to the folder named Rfiles, and we have named it 'hope.csv.' If we have done this successfully, we will now be able to find this data set in the Rfiles folder.

II. Basic File Management: **Apple Macintosh OS X**

As a first step, you should create a directory to hold all your R data, R files, and R documents. (I have simply placed mine in "Documents," and I have named it "R.") You should name this directory anything that is easy to remember, and is evocative of its purpose (i.e., to serve as a storage place for all R related materials).

How to read a data set (or 'data frame') into the R environment:

If you are using **Numbers for Mac**, you may enter your data as normal. Once all your data are entered, however, you will want to save your data set in the ".csv" format to the R directory you just created. This can be done if you follow these simple six steps: (1) select 'File > Export,' (2) of the three alternative formats (PDF, Excel, and CSV), select 'CSV,' (3) click on 'Next,' (4) type the new name for the data set, say 'trial,' to choose a name at random (though any name will do),(5) choose where (i.e., your R directory name) to save the document, and (6) click on 'Export.' If you check your R directory, you should find the 'trial.csv' file you just created.

If you prefer to use **Microsoft Office for Mac** instead, and thus have **Excel for Mac** installed on your Mac, you would enter data into the Excel Workbook in the normal way. Once all your data are entered, however, you will want to save your data in the ".csv" format in the R directory you just created. This can be done if you follow these steps: (1) select 'File > Save As,' (2) in the 'Format' area, select either 'Windows Comma Separated (.csv)' or 'MS-DOS Comma Separated (.csv),' (3) type in the new name for the data set, again using the name 'trial,' (in the 'Save As' area), (4) choose where (i.e., your R directory name) to save the document, and (5) click on 'Save.' If you check your R directory, you should find the 'trial.csv' file you just created.

Once we have created our '.csv' data set, we 'read' this file directly into R. Here is an example of how this is done:

Method 1

- To open the .csv data file in R, click on the R icon. (Remember that my R directory has been named simply R, and I have placed it in the Documents area.) At the R prompt, ">," write the following command. (Remember that this is what we would do if we are using a Mac rather than a Windows-based operating system)

  >trial<-read.csv('/Users/rstinero/Documents/R/trial.csv')

  If this is correctly executed, you will get back the R prompt: ">." To see the filename, enter (at the R prompt): 'ls().' All the data files currently loaded into R will appear (although at this point, the only filename that should show up is 'trial.') To see the actual data set, simply type (again, at the R prompt) the data set name, 'trial' (but without the quotes). The entire data set should appear on the screen. You are now ready to perform your basic analyses.

Method 2

Another, more cumbersome and (therefore) less often used, method of entering data is to simply enter it while in R. This works particularly well when we wish to create a small data set. For example, we might want to create a vector of numbers, or a vector variable, x, which consists of the simple array of the following five numbers: 57, 61, 38, 16, and 84.

x<-c(57,61,38,16,84)

We have here instructed R to create a vector, called "x," which consists of the above five numbers. To see the filename, x, we simply enter (at the R prompt): 'ls().' Now we will see 'x' listed, along with 'trial.' And when we enter "x" at the R prompt, we get back the row of numbers assigned to 'x.'

How to write a data set (or 'data frame') from the R environment:

In order to write your data set (from the R environment) to your R folder:

> write.csv(trial,file='/Users/rstinero/Documents/R/hope.csv')

What we have done here is write the data set 'trial' from the R environment to the folder named R, and we have named it 'hope.csv.' If we have done this successfully, we will now be able to find this data set in the R folder.

R: basic uses

One possible, but by no means the only, use of R involves basic arithmetic calculations. That is, R can be used like a hand-held calculator.

To see this, let's start at the R prompt, ">," and add two numbers, 6 and 8. If we simply write 6+8 and hit the "enter" key, R provides our answer interactively.

```
> 6+8

[1] 14
```

When we request R to give us the sum of 6 and 8, it "responds" with the answer, 14. We also see R provides a bracketed number 1---that is, a [1]---which is the "index" of the initial number on that particular line. We will return to this later when we have become more familiar with how R provides answer to questions we might put to it.

Let's try a few more calculations below.

Subtracting 10 from 32, we get the answer of 22 along with the index number, [1].

```
> 32-10

[1] 22
```

Multiplying 9 times 6, we get the answer of 54. (Note that we denote 'multiplication' with the symbol "*" or the asterisk.)

```
> 9*6

[1] 54
```

Finally, dividing 48 by 4, R provides the answer of 12.

```
> 48/4

[1] 12
```

R is also capable of providing the answer to slightly more complicated mathematical expressions.

If we wish to find 5 raised to the 3rd power:

```
> 5^(3)

[1] 125
```

For the square root of 16:

```
> 16^(.5)

[1] 4
```

We should note that the use of the parenthesis, ( ), around the exponent above is not normally necessary. We have added it only for purposes of clarity.

Here are some other examples of the basic calculations that R can accomplish with ease.

Raise the base of the natural logarithm, e, to some power:

exp(-3) =

> exp(-3)

[1] 0.04978707

If we want logarithms:

ln(10) =

> log(10)

[1] 2.302585

for logarithms to the base e.

If we want logarithms to the base 10:

log(10)=

> log(10,10)

[1] 1

>

For the constants, e and π:

> exp(1)

[1] 2.718282

> pi

[1] 3.141593

>

We can even derive trigonometric values (sin, cos, tan, etc.):

> sin(pi/2)

[1] 1

Some basic data set management methods

We will often want to work with our data set, or data frame, for the purposes of identifying mistakes, finding statistical outliers, correcting problems, rearranging observations, and sub-setting the data frame for the purposes of performing more in-depth analysis.

To consider this issue in the most practical way, let's work with a very simple data frame. The data consist of the following six pieces of information on each of the eight starting players on the 2010 New York Yankees as of the first week of September 2010: number on uniform; last name; position in the field; year of birth; handedness of batting (where '1' denotes 'right-handed'; '2' indicates 'left-handed'; and '3' represents 'switch-hitter'), and batting average. The data frame is named 'nyyankees.'

> nyyankees

	number	name	position	born	bats	avg
1	20	posada	c	1971	3	999
2	25	teixeira	1b	1980	3	266
3	24	cano	2b	1982	2	319
4	2	jeter	ss	1974	1	264
5	13	rodriguez	3b	1975	4	266
6	33	swisher	rf	1980	3	294
7	14	granderson	cf	1981	2	249
8	11	gardner	lf	1983	2	887

>

We will now perform several simple editorial (and janitorial) tasks that are very common to basic data analysis. Even though the data set involved here is trivially small, and very simple, the methods performed here will also work on data sets considerably larger, and considerably more complex.

First, let's consider how to identify mistaken data entries. Since I know that the variable 'bats' can legally assume on 3 values (namely, '1,' '2,' or '3'), I want to check to be sure that nothing outside that range of values has been recorded. (Clearly, upon visual inspection we can see that an entry of '4' has been made for 'rodriquez.)

To identify this mistake:

```
> table(nyyankees$bats)

1 2 3 4

1 3 3 1
```

By directing R to provide a 'table,' we have gotten a frequency distribution of all the values in the data frame called 'nyyankees' for the variable named 'bats.' We see that 'bats' takes the value of '1' one time, '2' three times, '3' three times, and '4' (an illegal value) one time. To find out which observation, or player, has this mistaken entry, we simply request R to find "which" one with the following command :

```
> which(nyyankees$bats==4)

[1] 5
```

This tells us that the fifth observation, or 'rodriguez,' has the mistaken entry for '4' for bats.

To correct this mistake, once we know that the player Rodriguez bats 'right,' and therefore should have a value of '1' for 'bats' (rather than '4'), we instruct R to carry out this amendment in the following way:

```
> nyyankees$bats[5]<-1
```

Here we have informed R to insert the value of '1' in the 5th observation of the variable 'nyyankees$bats.' Note that we use square brackets to denote the exact location of the destination of '1.' The result is listed below.

```
> nyyankees
```

	number	name	position	born	bats	avg
1	20	posada	c	1971	3	999
2	25	teixeira	1b	1980	3	266
3	24	cano	2b	1982	2	319
4	2	jeter	ss	1974	1	264
5	13	rodriguez	3b	1975	1	266
6	33	swisher	rf	1980	3	294
7	14	granderson	cf	1981	2	249
8	11	gardner	lf	1983	2	887

```
>
```

Suppose now we wish to derive the mean batting average, 'avg,' across all 8 starting players. We can inform R to provide the mean of the variable with the following command:

```
> mean(nyyankees$avg)

[1] 443
```

While 443 is indeed the average of the 8 players, we know (if we know anything about the game of baseball) that this cannot be correct: it is much too high. When we inspect the original data frame to see why this might have happened, we note the following two anomalies: the first observation, for 'posada,' has an average of '999,' and the last observation, for 'gardner,' has an average of '887.' No doubt, both these entries have artificially inflated the mean team average. After looking into this further, we learn that the '999' for the first observation has been entered as a 'missing data value.' This is a common practice when the data items we want are simply unavailable or suspect. We have here simply decided to use the value of '999' to denote this; we just don't know what the actual value is, and so we enter '999,' a value so out-of-range that we would never be fooled into thinking it is accurate.

The other data item, the '887' for the last observation, is not a missing data value. It is a mistake. Sometimes (more often than you might think) data entry is so tedious, mind numbing, and repetitive, we enter a value that is plainly incorrect. In this instance, we can simply go back to the original source of the data (the New York Times Sports Page website, in this case), find the correct value, and enter it.

Let's now make both alterations and recalculate the mean value for the team's batting average.

First, we will correct the value for the last observation. Checking the source of the original data set, we find that the correct batting average for 'gardner' is actually 287 rather than 887. Let's make the correction the same way we did above when we changed 'rodriguez' value for 'bats' from '4' to '1.'

```
> nyyankees$avg[8]<-287
```

Here we have informed R to insert the value of '287' in the 8th observation of the variable 'nyyankees$avg.' Note that we use square brackets to denote the exact location of the destination of '287.' The result of this instruction is listed below in the new, amended data set.

> nyyankees

	number	name	position	born	bats	avg
1	20	posada	c	1971	3	999
2	25	teixeira	1b	1980	3	266
3	24	cano	2b	1982	2	319
4	2	jeter	ss	1974	1	264
5	13	rodriguez	3b	1975	1	266
6	33	swisher	rf	1980	3	294
7	14	granderson	cf	1981	2	249
8	11	gardner	lf	1983	2	287

>

Second, we will want to instruct R to ignore the value of '999' when it recalculates the mean of the variable. After all, the '999' has no real meaning apart from reminding us, and R, that we just don't have the particular piece of information, and accordingly, it should not take '999' into consideration when deriving the mean. If we request R to calculate the mean average once again, it is now lower (which is better because it is more realistic) but it is still much too high because R is still "seeing" the value of '999' when it derives the mean value. That is,

> mean(nyyankees$avg)

[1] 368

Thus, the new calculation lowers the mean from 443 to 368. Let's now direct R to ignore the 999 value for the first observation, and then request it to recalculate the mean a third (and final) time.

We do this with the following R command:

```
> nyyankees$avg[nyyankees$avg==999]<-NA
```

We have here directed R to change the missing data value entry from '999' to 'NA.' (In fact, this command would convert all values of '999' to 'NA,' if there were more than one.) To see what this looks like, here is the newly altered data frame.

```
> nyyankees
```

	number	name	position	born	bats	avg
1	20	posada	c	1971	3	NA
2	25	teixeira	1b	1980	3	266
3	24	cano	2b	1982	2	319
4	2	jeter	ss	1974	1	264
5	13	rodriguez	3b	1975	1	266
6	33	swisher	rf	1980	3	294
7	14	granderson	cf	1981	2	249
8	11	gardner	lf	1983	2	287

```
>
```

Now if we recalculate the mean (with a slight alteration to one of the arguments), we have a mean batting average of (about) 278, or

```
> mean(nyyankees$avg,na.rm=TRUE)

[1] 277.8571
```

Where na.rm is "a logical value indicating whether NA values should be stripped before the computation proceeds." (This is copied-and-pasted directly from the R help files.) If na.rm = TRUE, the observation assigned the value "NA" is not included in the calculation of the mean. Clearly, we would use this stipulation, in the same way, when asking R to provide other values as well (such as the variance, the standard deviation, or graphical representations.)

Should we wish to compare the mean average of right-handed, left-handed, and switch hitters, we need only direct R to 'subset' our observations (first), and then develop the mean on each subgroup. Here are the commands, and results:

```
> mean(nyyankees$avg[nyyankees$bats==1],na.rm=TRUE)

[1] 265

> mean(nyyankees$avg[nyyankees$bats==2],na.rm=TRUE)

[1] 285

> mean(nyyankees$avg[nyyankees$bats==3],na.rm=TRUE)

[1] 280

>
```

This conforms with what we can see from the data frame on the previous page, and we would use the same approach to developing any kind of statistics (variance, standard deviation, etc.) on the subgroups.

Finally, if we wish to reorder the data by, say, the names in alphabetical order:

```
> nyyankees[order(nyyankees$name),]
```

	number	name	position	born	bats	avg
3	24	cano	2b	1982	2	319
8	11	gardner	lf	1983	2	287
7	14	granderson	cf	1981	2	249
4	2	jeter	ss	1974	1	264
1	20	posada	c	1971	3	NA
5	13	rodriguez	3b	1975	1	266
6	33	swisher	rf	1980	3	294
2	25	teixeira	1b	1980	3	266

And if we would like to reorder the observations from the highest to the lowest batting averages, we need to direct R in the following way:

> nyyankees[order(nyyankees$avg,decreasing=TRUE),]

	number	name	position	born	bats	avg
3	24	cano	2b	1982	2	319
6	33	swisher	rf	1980	3	294
8	11	gardner	lf	1983	2	287
2	25	teixeira	1b	1980	3	266
5	13	rodriguez	3b	1975	1	266
4	2	jeter	ss	1974	1	264
7	14	granderson	cf	1981	2	249
1	20	posada	c	1971	3	NA

Clearly, if we wished to rank the data items from lowest to highest batting averages, we could do so simply by setting 'decreasing' equal to "FALSE."

CHAPTER 2

R: tabular and graphical methods of presenting data

There are three basic approaches to summarizing and presenting data, both qualitative as well as quantitative, for the purposes of interpretation: tabular, graphical, and numerical methods. Fortunately, R provides us with some handy tools for accomplishing such summarization methods. Chapter 2 introduces us to a few *tabular* methods that we use to organize our data in a *table*; and *graphical* methods that are *graphs*, or *pictures*, of the information contained in those tables. Let us consider these methods within the context of a couple of practical problems.

Methods of summarizing and presenting qualitative data

Suppose over the course of the most recent semester we have collected data on the *region of origin* for graduate students (masters, MBA, and doctoral) who have enrolled in a statistics course. If we are to classify the countries from which the students have come, and group those countries (crudely) into regions like Latin America and Europe----as well as listing the nations of China, India, and the US---we have compiled the following set of data. Note that we have n = 41 observations in this particular data collection exercise. Reading this data set into R, naming it 'std.origin,' and printing it out below, we have:

> std.origin

1 US

2 US

3 Latin America

4 India

5 China

6 China

7 China

8 US

9 China

10 China

11 China

12 India

13 India

14 US

15 China

16 India

17 China

18 China

19 India

20 China

21 Latin America

22 US

23 US

24 Latin America

25 India

26 China

27 Europe

28 China

29 US

30 China

31 Europe

32 Europe

33 India

34 India

35 US

36 China

37 India

38 China

39 Europe

40 India

41 Europe

Our objective is to summarize and present the information in this data set, and so we will request R to provide the frequencies, or counts, of each observation in that data set. We do this by requesting a 'table,' and naming it 'std.counts.'

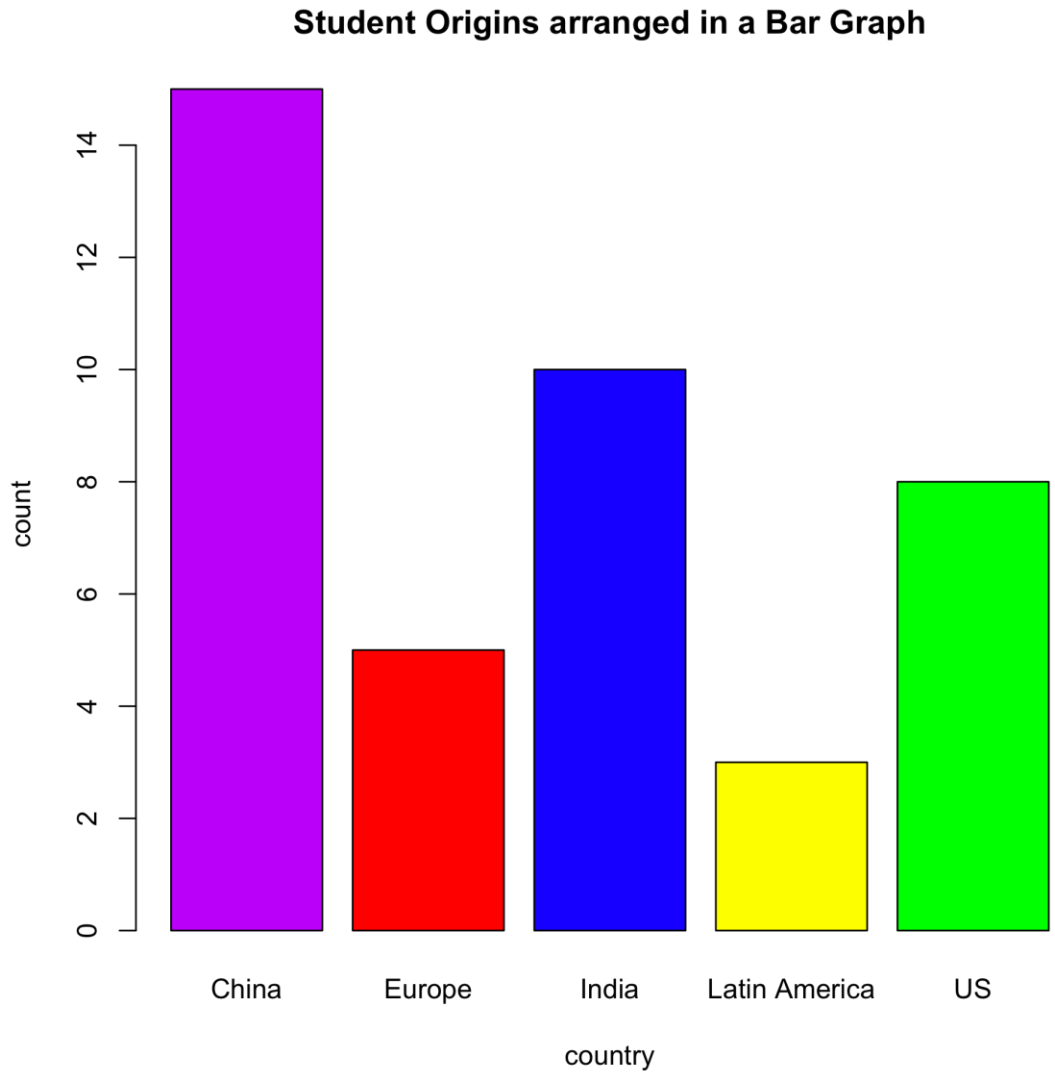
```
> std.counts<-table(std.origin)

> std.counts

std.origin
      China      Europe       India      Latin America          US
         15           5          10                  3           8
```

As we can see, of the 41 students in the class, 15 are from China, 5 from Europe, 10 from India, 3 from Latin America, and 8 from the US. That's nice to know, but perhaps we would like to organize this information into pictures that display the information in a more intuitively (and eye) appealing way. To do this, we request R

to provide a bar graph with nice colored rectangles, the heights of which correspond to the extent that each region is represented in the class. Here is the command.

> barplot(std.counts,col=c("purple","red","blue","yellow","green"),main="Student Origins arranged in a Bar Graph",xlab='country',ylab='count')

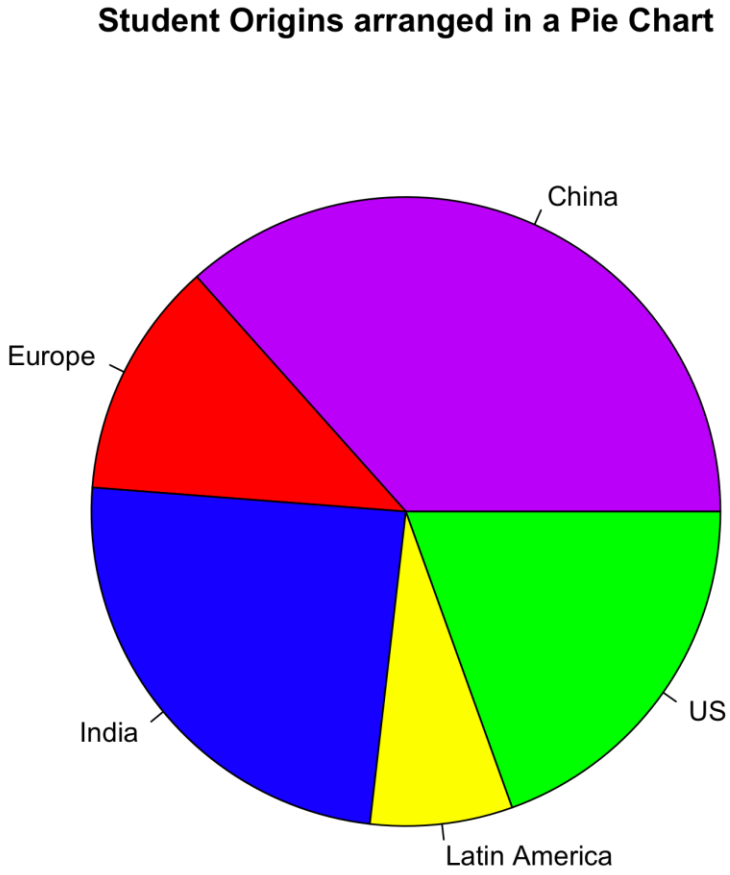
We have asked R to provide a bar graph (R responds to the term 'barplot') of the categorical data contained in the data set named 'std.counts.' We have also requested a title, "Student Origins arranged in a Bar Graph". And we have directed that the x axis be called 'country,' and the vertical axis to be called 'count.' Finally, we have specified that the bars be colored, in alphabetical order, purple,red, blue, yellow, and green. We may change the order of these colors, or even the color itself, with the last argument of the command
above.



The bar graph is not the only means by which we may represent categorical data; we may also request the pie chart. Here is the way we do this.

> pie(std.counts,col=c("purple","red","blue","yellow","green"),main="Student Origins arranged in a Pie Chart")

In this case, we have requested a pie chart (R understands the term 'pie') organizing the nominally-scaled information contained in the data set named 'std.counts,' and coloring each slice of the pie alphabetically; that is, purple is China, red is Europe, blue is India, yellow is Latin America, and green is the US. If we want to rearrange the colors, we need only reorder them in the statement above, remembering that whatever color we list first will go to China (since China is first, alphabetically), what ever color we name second will go to Europe (since Europe is second, alphabetically), and so on.



Methods of summarizing and presenting quantitative data

As we can see from Chapter 2, we also have descriptive methods for quantitative, or metric, data. Suppose we have data on the amount of television viewing (in hours) occurring in a sample of n = 100 households during a recent two-day holiday period (New Year's Eve and New Year's Day). Reading this data set into R, naming it 'tv.hours,' and printing it out below, we have:

> tv.hours

4.9 5.2 5.4 6.6 6.9 7.0 7.2 7.5 7.5 7.6 7.6 7.7 7.9 8.4 8.4 8.6 8.9 9.1

9.1 9.2 9.2 9.3 9.4 9.5 9.5 9.5 9.7 9.8 9.8 9.8 10.0 10.0 10.0 10.3 10.3

10.4 10.5 10.7 10.7 10.7 10.7 10.8 10.8 10.8 11.0 11.1 11.1 11.1 11.3

11.4 11.4 11.5 11.7 11.9 12.1 12.1 12.1 12.2 12.5 12.6 12.7 12.7 12.7

12.8 12.8 12.9 12.9 13.0 13.0 13.2 13.2 13.2 13.2 13.4 13.5 13.5 13.5

13.7 13.8 13.9 14.2 14.3 14.4 14.4 14.4 14.5 14.5 14.7 14.8 14.8 15.3

15.5 15.7 16.0 16.4 16.9 17.1 17.3 19.0 22.9

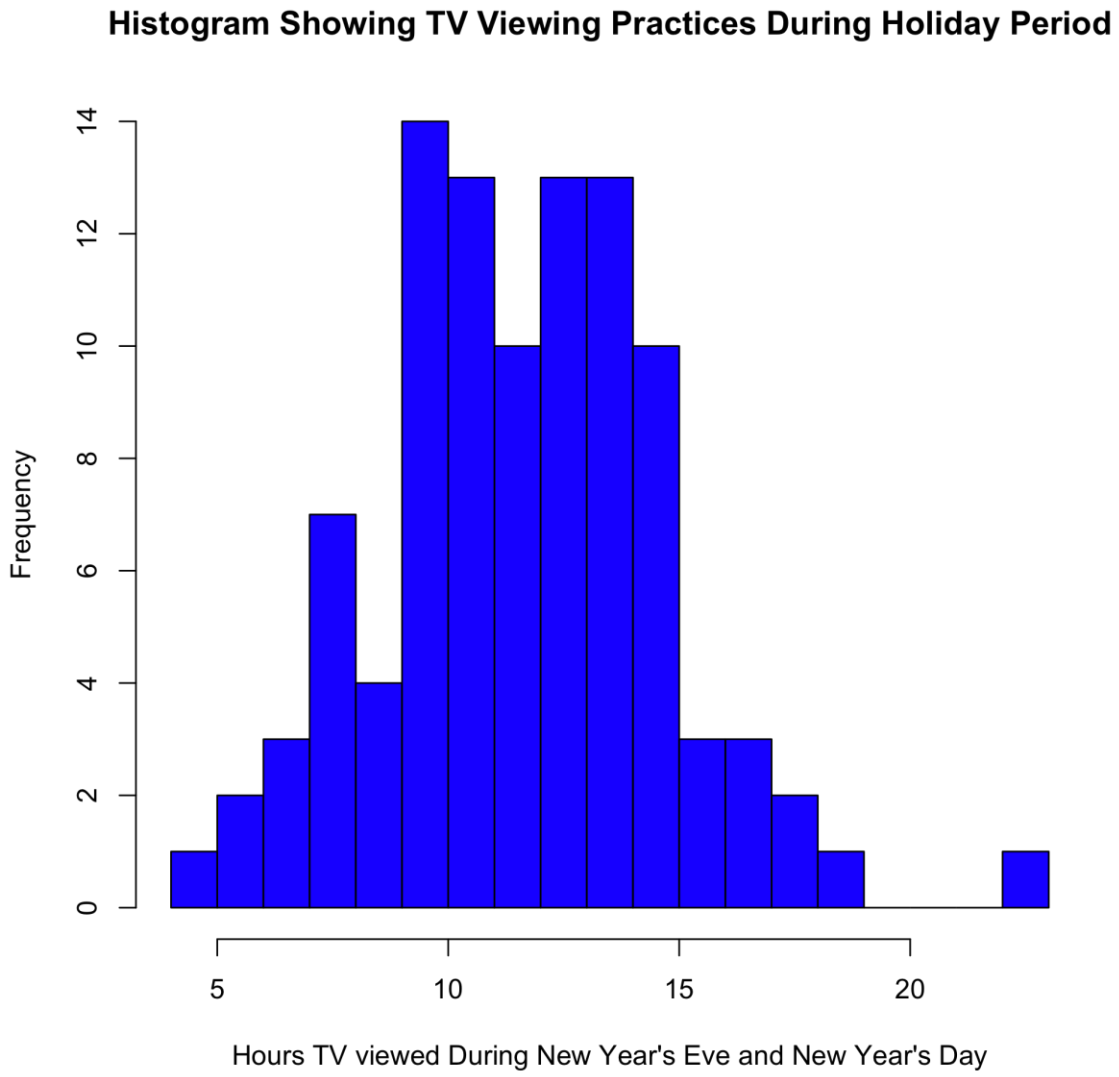
A cursory inspection reveals that among this particular sample of n = 100 households, the lightest amount of television viewing is 4.9 hours, the heaviest amount 22.9 hours. (The data are rank ordered from the smallest to the greatest values.)

This is interesting, but we will gain additional insight into how the data are distributed (its central tendency, it dispersion from the measure of central tendency, and so on) if we can make use of a few very simple graphical methods such as the histogram and the stem-and-leaf display.

Let us direct R to provide a simple histogram.

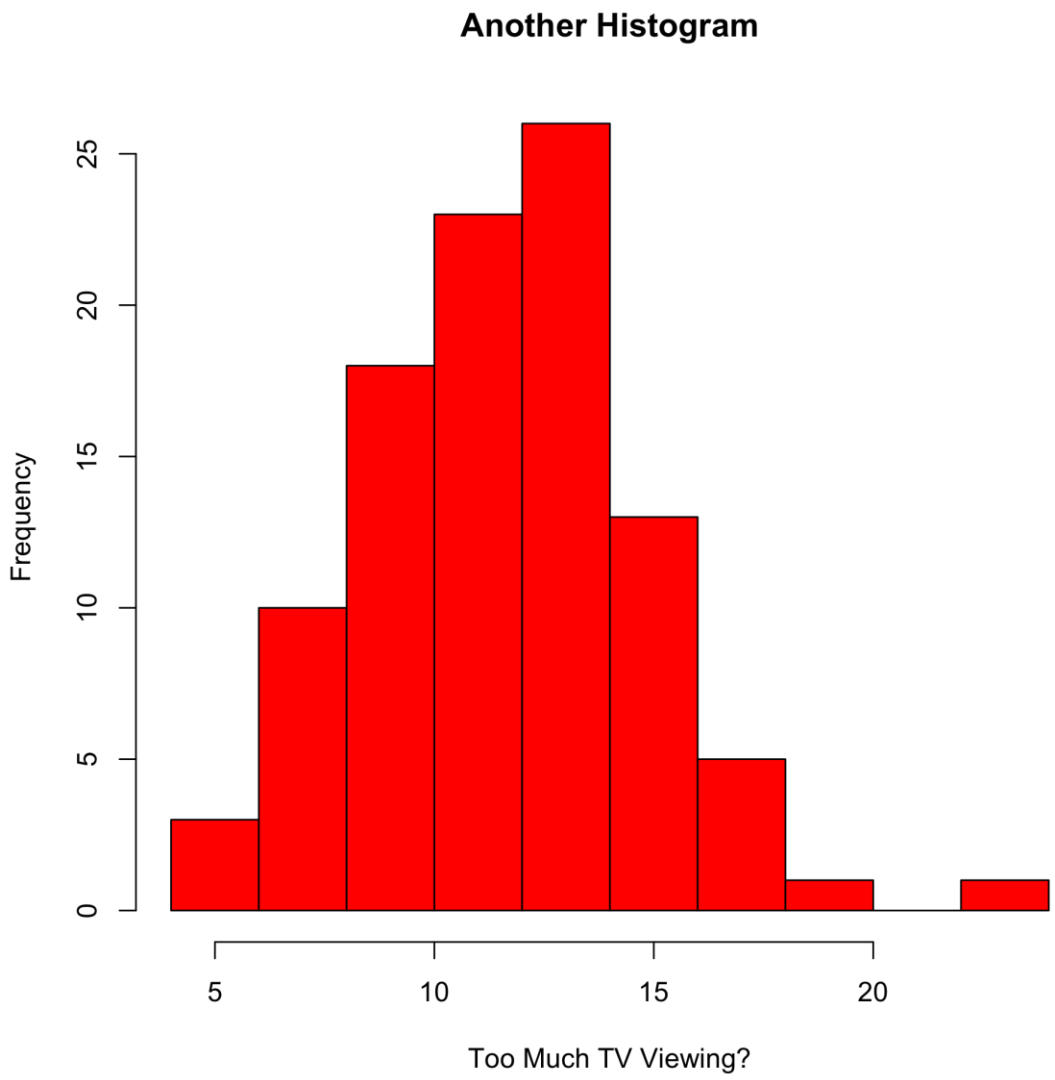
> >hist(tv.hours,br=c(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23),xlab= "Hours TV viewed During New Year's Eve and New Year's Day",main="Histogram Showing TV Viewing Practices During Holiday Period",col="blue")

We have here instructed R to develop a histogram of the quantitative data named 'tv.hours,' and to arrange it into classes or categories which begin with '4,' the '5,' then '6,' and so on until '23.' We have also requested that it provide a description of the horizontal axis, "Hours TV viewed During New Year's Eve and New Year's Day," and we have further asked that the illustration be named "Histogram Showing TV Viewing Practices During Holiday Period." The color of the bars will be "blue."



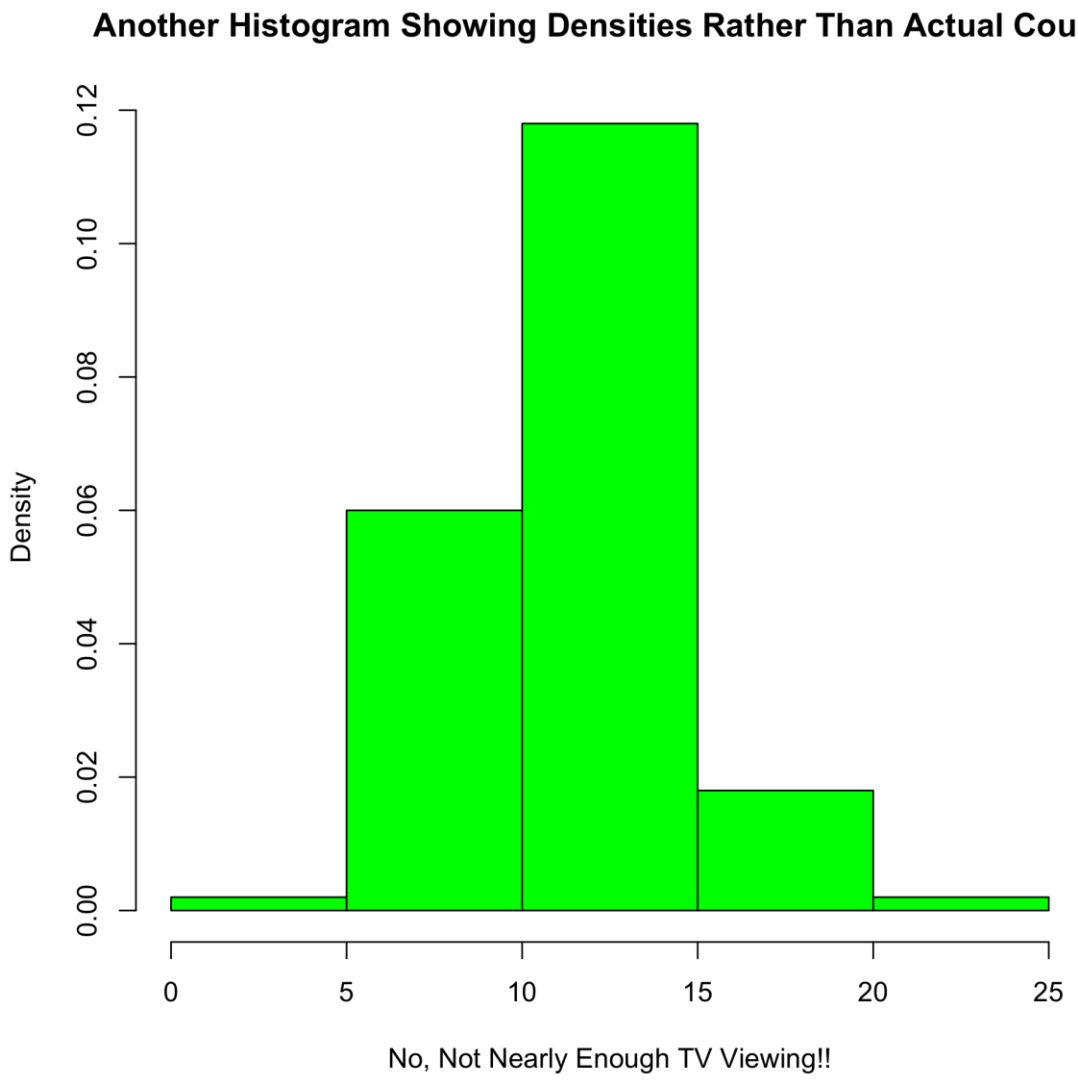
We can also request R to simply arrange all data values into, say, 10 equal-width categories (with the argument "breaks=10") of the histogram where we now color the bars red. Note that we have also renamed the main title of the illustration, "Another Histogram," and we have described the horizontal axis differently, "Too Much TV Viewing?"

> hist(tv.hours,breaks=10,main="Another Histogram", xlab="Too Much TV Viewing?",col="red")



Finally, we can make a few other cosmetic changes to how our data are presented in the histogram format. Here we now will (1) request '5' equally-wide categories into which the data will be arranged (using "breaks=5"), (2) display 'densities' rather than actual 'counts' of observations on the vertical axis (using "prob=TRUE"), (3) change (once again) both the main title of the illustration as well as the description of the horizontal axis, and (4) request a new color, this time green.

> hist(tv.hours,breaks=5,prob=TRUE,main="Another Histogram Showing Densities Rather Than Actual Counts", xlab="No, Not Nearly Enough TV Viewing!!",col="green")



Finally, let us consider another method of displaying quantitative data, the useful (though somewhat less well-known) stem-and-leaf display. See Chapter 2 for a brief description of this approach. Recall that this approach shows the data (in a pictorial sense) in rank-order form; the shape of the distribution of the data is also displayed. It is similar to a histogram rotated ninety degrees in a clockwise direction; that is, a histogram turned on its side.

When constructing a Stem-and-Leaf Display, it is helpful to first arrange the data in ascending order, from the smallest value to the largest. The first digits of each data value are arranged to the left of a vertical line. To the right of this vertical line, we record the last digit for each item in rank order. Each line in this display is referred to as a stem; each digit on the stem is referred to as a leaf. The R command we use:

```
> stem(tv.hours)

  The decimal point is at the |

   4 | 924
   6 | 6902556679
   8 | 44691122345557888
  10 | 000334577778880111344579
  12 | 11125677788990022224555789
  14 | 2344455788357
  16 | 04913
  18 | 0
  20 |
  22 | 9
```

Here we note that the 'default' stem-and-leaf display provided by R is probably a little too compressed. That is, there are too many 'leaves' on each 'stem' to provide the insight we may desire. It is possible to request R to 'stretch' the stem-and-leaf display by including (in this instance) the argument "scale=2." Note the difference in the display, and how the 'leaves' are now separated more widely.

```
> stem(tv.hours,scale=2)

  The decimal point is at the |

   4 | 9
   5 | 24
   6 | 69
   7 | 02556679
   8 | 4469
   9 | 1122345557888
```

10 | 00033457777888

11 | 0111344579

12 | 1112567778899

13 | 0022224555789

14 | 2344455788

15 | 357

16 | 049

17 | 13

18 |

19 | 0

20 |

21 |

22 | 9

Note that we can exactly recover the original data when it is arranged in this way. We can also gain insights into the range, central tendency, and degree of dispersion (around the mean) of the data values immediately. Note that this display does indeed appear similar to a histogram turned on its side; it has the advantage over the histogram, however, of showing the actual data values.

Additionally, this display technique has flexibility in that it can be "stretched" even further if we believe that the original stem-and-leaf representation has condensed the data too much. How can this be done? Again in the case of the 'tv.hours' data set, we can stretch the display by using two stems for each leading digit. Whenever the stem value is stated twice, the first value corresponds to leaf values of 0 to 4, and the second value corresponds to leaf values of 5 to 9. Let us consider how this might be done.

> stem(tv.hours,scale=4)

The decimal point is at the |

4 | 9

5 | 24

5 |

6 |

6 | 69

7 | 02

7 | 556679

8 | 44

8 | 69

9 | 112234

9 | 5557888

10 | 000334

10 | 57777888

11 | 0111344

11 | 579

12 | 1112

12 | 567778899

13 | 0022224

13 | 555789

14 | 23444

14 | 55788

15 | 3

15 | 57

16 | 04

16 | 9

17 | 13

17 |

18 |

18 |

19 | 0

19 |

20 |

20 |

21 |

21 |

22 |

22 | 9

CHAPTER 3

R: descriptive statistics

Here are some basic commands which, when applied to a data set, will provide the most commonly used measures of central tendency and location: the mean, median, mode, and percentiles; and the best known measures of dispersion: the range, the interquartile range, the variance, and the standard deviation. We will also see how the coefficient of variation can be found. Using the data vector defined above, x:

```
x<-c(57,61,38,16,84)
```

we can now find the mean:

```
> mean(x)
[1] 51.2
```

The median:

```
> median(x)
[1] 57
```

The mode:

Percentiles: we are able to find percentiles (referred to as "quantiles" in R) in the following way. Once again, using the data vector x as defined above

```
> quantile(x,probs=c(0.0,0.25,0.5,0.75,1.0))
  0%  25%  50%  75% 100%
  16   38   57   61   8
```

Here we have the results of the 0^{th}, 25^{th}, 50^{th}, 75^{th}, and 100^{th} percentiles. Note that the 50^{th} percentile is, by definition, the same thing as the median. Accordingly, it also always has the same value. We should also note we are free to define any percentiles we like, but that we have (in this example) asked for only the first, second, and third quartiles.

The range: To find the range, we can look at it two ways: the first approach provides the minimum and maximum values; the second, the difference between the two values.

```
> range(x)
[1] 16 84
> max(x)-min(x)
[1] 68
```

The interquartile range: We recall that the interquartile range is simply the difference between the third and first quartiles; that is, it is the range of the middle fifty percent of the data when it is ranked in ascending order from lowest to highest. In this case, it is the difference between 61 and 38.

```
> IQR(x)
[1] 23
```

The variance:

```
> var(x)
[1] 654.7
```

The standard deviation:

```
> sd(x)

[1] 25.58711
```

The coefficient of variation:

The coefficient of variation is defined as the ratio of the standard deviation to the mean; that is, it is a relative measure of the variability of a data set. For the data set x defined above

```
> sd(x)/mean(x)

[1] 0.4997482
```

This indicates that the standard deviation is nearly 50% of the mean.

For data sets with more than one variable, we are able to derive the *covariance* and the *correlation*. Suppose we have a simple data set consisting of n = 6 observations (6 high school graduating seniors) on two variables, GPA, their grade point average, and SAT, their Math SAT score. We can summarize the linear association of these two variables in the following way.

First, let's take a look at the data set itself. Here we see that student 1 has a grade point average of 2.7 and a Math SAT score of 450; student 2 has a grade point average of 3.5 and a Math SAT score of 560; and so on.

```
> gpa.sat
```

	GPA	SAT
1	2.7	450
2	3.5	560
3	3.7	700
4	3.3	620
5	3.6	640
6	3.0	570

Second, we can derive the covariance with the following expression. In the parenthetical statement, we specify the data set name ('gpa.sat'), then add '$,' and

finally include the variable name itself. That is, the command below requests R to provide us with the covariance of two variables: GPA which is in the data set titled gpa.sat, and SAT which is also in the data set gpa.sat.

```
> cov(gpa.sat$GPA,gpa.sat$SAT)

[1] 28.6
```

Third, we can ask R to provide the standard deviation of the two variables, GPA and SAT, from the data set titled gpa.sat.

```
> sd(gpa.sat$GPA)

[1] 0.3847077

> sd(gpa.sat$SAT)

[1] 85.32292
```

Fourth, we can request R to give us the correlation of these two variables. If we write out the expression for the correlation coefficient (that is, the ratio of the covariance of the two variables to the product of the standard deviation of each variable), we have

```
> cov(gpa.sat$GPA,gpa.sat$SAT)/(sd(gpa.sat$GPA)*sd(gpa.sat$SAT))

[1] 0.8713036
```

It is easier, however, to simply use the 'cor' command. As we see, the result is exactly the same. Note that the information specified in the parenthetical expression is the same in the 'cor' command as it is in the 'cov' command.

```
> cor(gpa.sat$GPA,gpa.sat$SAT)

[1] 0.8713036

>
```

CHAPTER 4

R: combinations, factorials, and permutations

Combinations: how many ways may we take 2 from a total of 3 items?

> choose(3,2)

[1] 3

How many ways may we draw 1 item from 10?

> choose(10,1)

[1] 10

Factorials: a factorial is defined as n!=(n)*(n-1)*(n-2)*….(2)*(1). That is, if n=5, then n!=5!=5*4*3*2*1=120. Clearly, when n is large, this can be a cumbersome calculation, unless of course we are using R.

> prod(5:1)

[1] 120

If n=10, then

> prod(10:1)

[1] 3628800

If we wish to calculate the product of 25*24*23*22*21*20 (which is =127,512,000), then

> prod(25:20)

[1] 127512000

Permutations: This puts us in position to derive permutations which, we recall, equal the product of the number of combinations of "N objects taken n at a time" and n!; that is, a permutation is the same thing as a combination except that, in the case of the permutation, order "counts."

> choose(3,2)*prod(2:1)

[1] 6

How many permutations of N=6 objects may be taken n=3 at a time? Since there are 20 combinations of 6 objects taken 3 at a time, and since each "triple" may be taken 6 different ways, we find the product of 20 and 6, or 120. Using R, we have

```
> choose(6,3)
[1] 20
```

And

```
> prod(3:1)
[1] 6
```

The product of these two values can be found easily in the follow way:

```
> choose(6,3)*prod(3:1)
[1] 120
```

CHAPTER 5

R: discrete prob. dist.: Binomial, Poisson, Hypergeometric

R provides a means to determine probabilities for a number of distributions. In other words, we might think of this capability as providing the same function as do tables of probabilities in statistics textbooks. R, however, does this for more types of probability distributions, and it does it far more accurately. Among the discrete probability distributions, the most important for our purposes are the binomial, Poisson, hypergeometric; among the continuous, the uniform, normal, and exponential. It either case, there are four different commands: 'd' which provides us with the probability mass values (for the discrete distributions) and the probability density values (for the continuous distributions); 'p' which provides the cumulative probabilities for both discrete as well as continuous distributions; 'q' which provide the quantiles; and 'r' which generates numbers from any one of the distributions.

Binomial Probability Distribution: when we have a binomial experiment, R will calculate the binomial probabilities for us with ease.

If we flip an unbiased coin 5 times, what is the probability of getting exactly 3 heads in the 5 flips? In this instance, n = 5 trials (or 'flips'), x = 3 successes (or 'heads'), and p = .5 (that is, the probability of getting a 'success,' or heads, on any given trial is .5, or fifty-fifty). Thus, we are looking for p(x = 3|n = 5, p = .5) = .3125. Using R:

> dbinom(3,5,.5)

[1] 0.3125

In this connection, what is the probability of getting *no more than* 3 heads in the 5 flips? Now we are looking for the "cumulative" probability:

p(x = 0) + p(x = 1) + p(x = 2) + p(x = 3) =

= (.03125) + (.15625) + (.3125) + (.3125)

= .8125

Using R, we find this directly:

> pbinom(3,5,.5)

[1] 0.8125

Note that we can enter the probability as a fraction, if we prefer, rather than as a decimal. In some cases, we have no choice. For example, what is the probability that if we cast a single die three times that we will have exactly one 'three'?

$f(x) = P(x|n, p) = C_x^n \; p^x (1 - p)^{n-x}$

$f(1) = P(x = 1|n = 3, p = 1 / 6) = C_1^3 \; (1/6)^1(5/6)^2$

$f(1) = P(x = 1|n = 3, p = 1 / 6) = (3)\; (1/6)^1(5/6)^2 = 0.3472$

In this instance, n = 3 trials (or 'tosses'), x = 1 success (or one 'three'), and p = 1/6 (that is, the probability of getting a 'success,' or a 'three,' on any given trial is 1/6). Thus, we are looking for p(x = 1|n = 3, p = 1/6) = .3472. Using R:

> dbinom(1,3,1/6)

[1] 0.3472222

Poisson Probability Distribution: Since the number of arrivals at the drive-in teller window of a bank during a 15 minute period is Poisson distributed with a mean rate of 10, what is the probability of exactly 5 arrivals in 15 minutes?

The Poisson Probability Function is:

$f(x) = P(x \mid \mu) = \frac{\mu^x e^{-\mu}}{x!}$ = the probability of x occurrences in an interval

μ = expected value or mean number of occurrences in an interval

$e \approx 2.7183\ldots$ the base of the natural logarithm

Since in our example, $\mu = \frac{10\ arrivals}{15\ \text{min}\ utes}$, we need only substitute and solve.

$$f(x) = P(x \mid \mu) = \frac{\mu^x e^{-\mu}}{x!} = \frac{10^5 e^{-10}}{5!} = 0.0378$$

Using R,

> dpois(5,10)

[1] 0.03783327

We can also find the "cumulative" probability for values of a Poisson distributed random variable. For example, since the number of arrivals at the drive-in teller window of a bank during a 15 minute period is Poisson distributed with a mean rate of 10, what is the probability of 5 or fewer arrivals in 15 minutes?

Deriving this by adding the individual probability mass values, we have

> dpois(0,10)+dpois(1,10)+dpois(2,10)+dpois(3,10)+dpois(4,10)+dpois(5,10)

[1] 0.06708596

But it is much easier to simply use the cumulative probability expression for the Poisson distributed random variable

> ppois(5,10)

[1] 0.06708596

Clearly, the answers are exactly the same.

Hypergeometric Probability Distribution: Suppose we are interested in developing the probability that a sample will have certain characteristics when drawn without replacement. In this case, we know we do not have a binomial experiment since the probability of 'success' changes from trial-to-trial. For example, suppose we wish to draw a sample of size n = 3 from an urn which contains five marbles, N = 5, three black ('success') and two white ('failure'), and compute the probability that we will draw 2 black and one white marbles. In R, we can perform this calculation using combinations as defined above:

```
> choose(3,2)*choose(2,1)/choose(5,3)

[1] 0.6
```

Alternatively, we can use the following expression to derive the probability directly

```
> dhyper(1,2,3,3,log=FALSE)

[1] 0.6
```

In this instance, we have five elements, or 'arguments,' contained in the parentheses. They are, in order:

```
dhyper(x, m, n, k, log = FALSE)
```

Arguments

`x`	the number of white balls drawn without replacement from an urn which contains both black and white balls.
`m`	the number of white balls in the urn.
`n`	the number of black balls in the urn.
`k`	the number of balls drawn from the urn.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p); we will normally assign the value of 'FALSE' to this

Suppose that a hand of 12 cards is dealt from a normal deck of 52 playing cards. What is the probability that of the 12 cards, 3 will be diamonds, 3 will be clubs, 3 will be spades, and 3 will be hearts?

> choose(13,3)^4/choose(52,12)

[1] 0.03241886

CHAPTERs 6, 7, and 8

R: continuous prob. dist.: Uniform, Normal, Exponential, Student's t

Uniform Probability Distribution: The first continuous probability distribution to be considered is the uniform which is defined to exist between a finite lower value and a finite upper value. For example, suppose that the flight times of commercial aircraft traveling from Chicago to New York are uniformly distributed between 120 minutes and 140 minutes. What is the probability that a flight will take between 120 minutes and 130 minutes?

$$P(120 \le x \le 130) = \int_{120}^{130} \frac{1}{20}\,dx = (1/20)\,(130 - 120) = (1/20)\,(10) = 0.50$$

Using R, we find the answer in the following way

> punif(130, min=120, max=140, lower.tail = TRUE, log.p = FALSE)

[1] 0.5

where

```
punif(x, min=0, max=1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`x`	Value of random variable
`min,max`	lower and upper limits of the distribution. Must be finite.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are *P[X <= x]*, otherwise, *P[X > x]*.

What is the probability that a flight from Chicago to New York will take between 135 and 140 minutes? That is, P (135 ≤ x ≤ 140)?

$$\int_{135}^{140} \frac{1}{20}\,dx = (1/20)\,(140 - 135) = (1/20)\,(5) = 0.25$$

Again, using R, we find

```
> punif(140, min=120, max=140, lower.tail = TRUE, log.p = FALSE)-
punif(135,min=120,max=140,lower.tail=TRUE,log.p=FALSE)

[1] 0.25
```

Normal Probability Distribution: As in the case of the uniform distribution, we usually will not find the probability density of the normal distribution particularly useful. After all, it only provides the "height" of the curve, or the functional value, at a particular value of the random variable. It therefore does not provide probabilities. The most useful commands associated with the normal distribution are the 'p' and 'q' commands. For example, if we want to know the probability that the normally distributed random variable will take on a value of -1.96 or less, R provides the answer according to the following command

```
> pnorm(-1.96)

[1] 0.02499790
```

We note here that a table in a statistics book will typically provide the answer of .025 which of course is simply the rounded version of the answer above. We also point out that the 'p' command gives the area under the standard normal curve from negative infinity to the value of z in question.

Some other examples of how to use R to find the probabilities of the standard normal random variable:

P(-1.00 < Z < 1.96) = .8163 (from a table of probabilities)

```
> pnorm(1.96)-pnorm(-1.00) (from R)

[1] 0.8163469
```

P(Z > 1.09) = 1 – p(Z < 1.09) = .1379 (from a table)

> 1 - pnorm(1.09) (from R)

[1] 0.1378566

P(Z < -.85) = .1977 (from a table)

> pnorm(-.85)

[1] 0.1976625 (from R)

P(.53 < Z < 2.42) = .2903 (from a table)

> pnorm(2.42)-pnorm(.53)

[1] 0.2902957

Finally, P(Z > -.36) = .6406 (from a table)

> 1-pnorm(-.36)

[1] 0.6405764

Very often in statistics, we are interested in determining the value of Z which cuts off an area, or probability, in the tail of the distribution. R finds this value by way of the 'q' commands.

For example, suppose we want to know the value of Z which provides an area of .025 in the upper tail, or (what is the same thing) the value of Z which provides an area of .975 in the lower tail of the same distribution.

A table in a textbook will typically provide a value of approximately Z = -1.96. Using R

> qnorm(.975)

[1] 1.959964

Once again, we note that while a table will usually provide a rounded value, R gives us something a bit more accurate, and the value is carried out to additional places.

As another example, suppose we want to know the value of Z which provides an area of .05 in the upper tail, or (what is the same thing) the value of Z which provides an area of .95 in the lower tail of the same distribution. Using a table of probabilities from a textbook, we will usually get a value of roughly Z = 1.645, but using R

> qnorm(.95)

[1] 1.644854

Finally, suppose we want to know the value of Z which provides an area of .01 in the upper tail, or (what is the same thing) the value of Z which provides an area of .99 in the lower tail of the same distribution. Using a table of probabilities from a textbook, we will usually get a value of roughly Z = 2.33, but using R

> qnorm(.99)

[1] 2.326348

As before, we note that the value provided by R is carried out additional places beyond that which is usually given by the tables in the textbooks.

Exponential Probability Distribution: Another continuous probability distribution we consider is the exponential probability distribution. We will recall that this distribution describes phenomena such as the time between occurrences (arrivals at a drive-up teller window, for example) and the distance between occurrences (such as the number of bubbles found in plate glass windows produced by a certain process). The exponential cumulative distribution is what we are concerned with here since that what we use to compute actual probabilities.

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`rate`	vector of rates.

`log, log.p` logical; if TRUE, probabilities p are given as log(p).
`lower.tail` logical; if TRUE (default), probabilities are *P[X <= x]*, otherwise, *P[X > x]*.

Details

If `rate` is not specified, it assumes the default value of `1`.

The exponential distribution with rate λ has density

f(x) = lambda e^(- lambda x)

for *x >= 0*.

If we let the random variable x be the time it takes to load a truck, and if we assume that the mean, or average, time to load a truck (according to company records) is 15 minutes, then the expression we use to find cumulative probabilities is:

$$P(x \leq x_o) = 1 - \exp\left(-\frac{x_o}{\mu}\right)$$

$$P(x \leq x_o) = 1 - \exp\left(-\frac{x_o}{15}\right)$$

where $\mu = 15$.

We can now ask our questions.

What is the probability that loading a truck will take 6 minutes or less?

$$P(x \leq x_o) = 1 - \exp\left(-\frac{x_o}{15}\right)$$

$$P(x \leq 6) = 1 - \exp\left(-\frac{6}{15}\right) = 1 - 0.6703 = 0.3297$$

What is the probability that loading a truck will take 18 minutes or less?

$P(x \leq 18) = 1 - \exp(-\frac{18}{15}) = 1 - 0.3012 = 0.6988$

What is the probability that loading a truck will take between 6 and 18 minutes?

$P(6 \leq x \leq 18) = 0.6988 - 0.3297 = 0.3691.$

Student's t Probability Distribution: Finally, let us consider how we might find probabilities and their associated t values for t distributions with different degrees of freedom, df. Suppose we have a sample size of n = 13 and df = (n – 1) = 12. As in the case of the uniform and standard normal probability distributions, we usually will not find the probability density of the t distribution particularly useful. (After all, it only provides the "height" of the curve, or the functional value, at a particular value of t, and it therefore does not provide probabilities.) The most useful commands associated with the t distribution are the 'p' and 'q' commands; the 'd' command will not be so helpful to us.

For example, for a t distribution with df = 12, what is the probability that t will take on a value of 1.782 or less? That is,

P(t < 1.782, df = 12) = .95 (from a table)

> pt(1.782,12)

[1] 0.9499756 (from R)

What is the probability that t will take on a value of 2.681 or greater? That is,

P(t > 2.681, df = 12) = 1 – P(t < 2.681, df = 12) = .01 (from a table)

> 1 - pt(2.681,12)

[1] 0.009999963 (from R)

Finally, what is the probability that t will take on a value between -1.356 and 1.782?

P(-1.356 < t < 1.782, df = 12) = .85 (from table)

> pt(1.782,12)-pt(-1.356,12)

[1] 0.849942 (from R)

Finally, suppose we want to know the value of t which cuts off an area of .05 in the upper tail, or (what is the same thing) the value of t which provides an area of .95 in the lower tail of the same distribution with, say, df = 69. Using a table of probabilities from a textbook, we will get a value of roughly t = 1.667, but using R we have

> qt(.95,69)

[1] 1.667239

Or alternatively

> -qt(.05,69)

[1] 1.667239

What is the value of t which cuts off an area of .025 in the upper tail, or (what is the same thing) the value of t which provides an area of .975 in the lower tail of the same distribution with df = 69. Using a table of probabilities from a textbook, we will usually get a value of roughly t = 1.995, but using R we have

> qt(.975,69)

[1] 1.994945

Or

> -qt(.025,69)

[1] 1.994945

As a final example, what is the value of t which cuts off an area of .01 in the upper tail, or (what is the same thing) the value of t which provides an area of .99 in the lower tail of the same distribution with df = 69. Using a table of probabilities from a textbook, we will usually get a value of roughly t = 2.382, but using R we have

> qt(.99,69)

[1] 2.381615

Or

> -qt(.01,69)

[1] 2.381615

R: confidence interval estimates

R has the ability to provide confidence interval estimates of population parameters when we are working with samples. Suppose we are using the small data set (tiny, actually) consisting of the annual salaries of five recent high school graduates who have not gone on to college. Expressed in thousands of dollars per year, the sample looks like this: 18, 20, 21, 21, 25. As an easy way to handle this, we might want to create a vector variable, 'salary,' which consists of the simple array of the above five numbers.

> salary<-c(18,20,21,21,25)

If we want to check to make sure that the vector was in fact created, we simply enter the name of the vector, and R should respond with the elements which comprise it

> salary

[1] 18 20 21 21 25

Once we are satisfied that we have created the vector variable we want, we are in a position to perform the basic analysis required. For a confidence interval estimate, one of the easiest things we can do is

> t.test(salary)

(R responds with a wealth of information, including the p value associated with a hypothesis test which tests the null hypothesis that 'salary' is equal to zero against the alternative hypothesis that the "true mean is not equal to 0'; the 95% confidence interval estimate of the population mean; and the sample mean itself.)

One Sample t-test

data: salary

t = 18.4182, df = 4, p-value = 5.113e-05

alternative hypothesis: true mean is not equal to 0

95 percent confidence interval:

17.83437 24.16563

sample estimates:

mean of x

21

Alternatively, we can set the confidence level to something other than the default of 95%, say, 90% and 99%

```
> t.test(salary,conf.level=.90)

        One Sample t-test

data:  salary
t = 18.4182, df = 4, p-value = 5.113e-05
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 18.56932 23.43068
sample estimates:
mean of x
       21
> t.test(salary,conf.level=.99)

        One Sample t-test

data:  salary
```

t = 18.4182, df = 4, p-value = 5.113e-05

alternative hypothesis: true mean is not equal to 0

99 percent confidence interval:

15.75052 26.24948

sample estimates:

mean of x

21

To find the sample size required to control both level of confidence as well as the level of precision, we can use a simple command in R. For example, suppose we wish to find the size of the sample we need to set the confidence level at 95% with the desired margin of error, or precision, at 1. Assuming the "planning value" for the standard deviation is 20:

> (qnorm(.975)*20/1)^2

[1] 1536.584

Rounding up, we see that the size of the sample we need is n = 1,537.

If we wish to increase the level of confidence from 95% to 99%,

> (qnorm(.995)*20/1)^2

[1] 2653.959

And rounding up once again, we see that the sample size required to achieve this higher level of confidence (99% as opposed to 95%) has increased from n = 1,537 to n = 2,654

If we want a confidence level of 99% and a desired margin of error of 0.5 (that is, more precision), then

> (qnorm(.995)*20/.5)^2

[1] 10615.83

The sample size has now increased to n = 10,616

CHAPTER 9

R and hypothesis testing

When we want to conduct a hypothesis test concerning the population mean or proportion, R can handle the challenge quite easily. Let's revisit the problem involving the issue of whether London Heathrow should be designated a superior service airport: *"a business travel magazine wants to classify transatlantic gateway airports according to the mean rating for the population of business travelers. A rating scale with a low score of 0 and a high score of 10 will be used, and airports with a population mean rating greater than 7 will be designated as superior service airports."* We are then told that the magazine staff collected a sample of 60 respondents who provide data concerning London's Heathrow Airport with a mean rating of 7.25 and a sample standard deviation of 1.052. Should London Heathrow be designated as a superior service airport? This calls for a one-tail hypothesis test using our six-step procedure at the 95% confidence level. In R, this is all we need to do

```
> t.test(heathrow$Rating,conf.level=.95,mu=7,alternative='g')


        One Sample t-test


data:  heathrow$Rating

t = 1.8414, df = 59, p-value = 0.03529

alternative hypothesis: true mean is greater than 7

95 percent confidence interval:

 7.023124      Inf

sample estimates:

mean of x

    7.25
```

Note that the fourth argument is represented as "alternative='g'. This is how we inform R that we are looking for an upper-tail test; that is, a test where we reject the null hypothesis when the test statistic falls in the upper tail. Thus, the 'g' is used for 'greater than.' If we are interested in a lower-tail test (that is, a test where we reject the null hypothesis when the test statistic falls in the lower tail), we would represent this by rewriting the fourth argument as "alternative='l', where 'l' is the letter and not the number one, "1."

For example, if we were to redo the Hilltop Coffee example using the sample standard deviation (instead of the population standard deviation) and the t distribution rather than the standard normal, we can see this in practice:

"*the Federal Trade Commission (FTC) periodically conducts statistical studies designed to test the claims that manufacturers make about their products. For example, the label on a large can of Hilltop Coffee states that the can contains 3 pounds of coffee. The FTC knows that Hilltop's production process cannot place exactly 3 pounds of coffee in each can, even if the mean filling weight for the population of all cans filled is 3 pounds per can. However, as long as the population mean filling weight is at least 3 pounds per can, the rights of consumers will be protected. Thus, the FTC interprets the label information on a large can of coffee as a claim by Hilltop that the population mean filling weight is at least 3 pounds per can. We will show how the FTC can check Hilltop's claim by conducting a lower tail hypothesis test.*"

```
> t.test(coffee$Weight,conf.level=.95,mu=3.00,alternative='l')


      One Sample t-test


data:  coffee$Weight

t = -2.8237, df = 35, p-value = 0.003891

alternative hypothesis: true mean is less than 3

95 percent confidence interval:

   -Inf 2.967869

sample estimates:

mean of x

   2.92
```

Let's us now consider how a two-tail test might be conducted in the instance of the Holiday Toys example. *"Holiday Toys manufacturers and distributes its products through more than 1,000 retail outlets. In planning production levels for the coming winter season, Holiday must decide how many units of each product to produce prior to knowing the actual demand at the retail level. For this year's most important new toy, Holiday's marketing director is expecting demand to average 40 units per retail outlet. Prior to making the final production decision based upon this estimate, Holiday*

decided to survey a sample of 25 retailers in order to develop more information about the demand for the new product. Each retailer was provided with information about the features of the new toy along with the cost and the suggested selling price. Then each retailer was asked to specify an anticipated order quantity."

Instead of spelling out formally the six-step hypothesis method, we could have R do the work for us in the following way

"Holiday Toys manufacturers and distributes its products through more than 1,000 retail outlets. In planning production levels for the coming winter season, Holiday must decide how many units of each product to produce prior to knowing the actual demand at the retail level. For this year's most important new toy, Holiday's marketing director is expecting demand to average 40 units per retail outlet. Prior to making the final production decision based upon this estimate, Holiday decided to survey a sample of 25 retailers in order to develop more information about the demand for the new product. Each retailer was provided with information about the features of the new toy along with the cost and the suggested selling price. Then each retailer was asked to specify an anticipated order quantity."

```
> t.test(holidaytoys$Units,conf.level=0.95,mu=40)


      One Sample t-test


data:  holidaytoys$Units

t = -1.1026, df = 24, p-value = 0.2811

alternative hypothesis: true mean is not equal to 40

95 percent confidence interval:

 32.5334 42.2666

sample estimates:

mean of x

   37.4
```

Note that in this instance we don't need to specify the fourth argument as we did in the case of the one-tail tests because the default assumes that we are conducting a two-tail test.

CHAPTER 12

R: Chi Square Distribution

Just as we could use R to find probabilities for the various probability distributions encountered in the earlier chapters (binomial, Poisson, hypergeometric, uniform, normal, and t distribution), we can also use it to find probabilities with the Chi Square distribution.Suppose we would like to find the chi square value which cuts off an area of .05 in the upper tail of the chi square distribution having df = 2:

```
> qchisq(.95,2)

[1] 5.991465
```

We again note that R defines the "area of .05 in the upper tail of the chi square distribution" with .95, the area to the left of the value we want. For the df = 2, this might be from a contingency table test in which we have n = 2 rows and m = 3 columns. Since df = (n – 1)(m – 1), (2 -1)(3 – 1) = (1)(2) = 2.

CHAPTERs 14 and 15

R: Regression analysis

Suppose that we have the following data set (described in detail in the class Discussion Notes).

```
> cityraw
```

	attitude	duration	weather
1	2	2	4
2	2	2	5
3	3	4	1
4	4	6	1
5	5	8	7
6	6	10	3
7	8	12	4
8	9	12	11
9	9	9	10

10	10	12	11
11	10	17	8
12	11	18	8

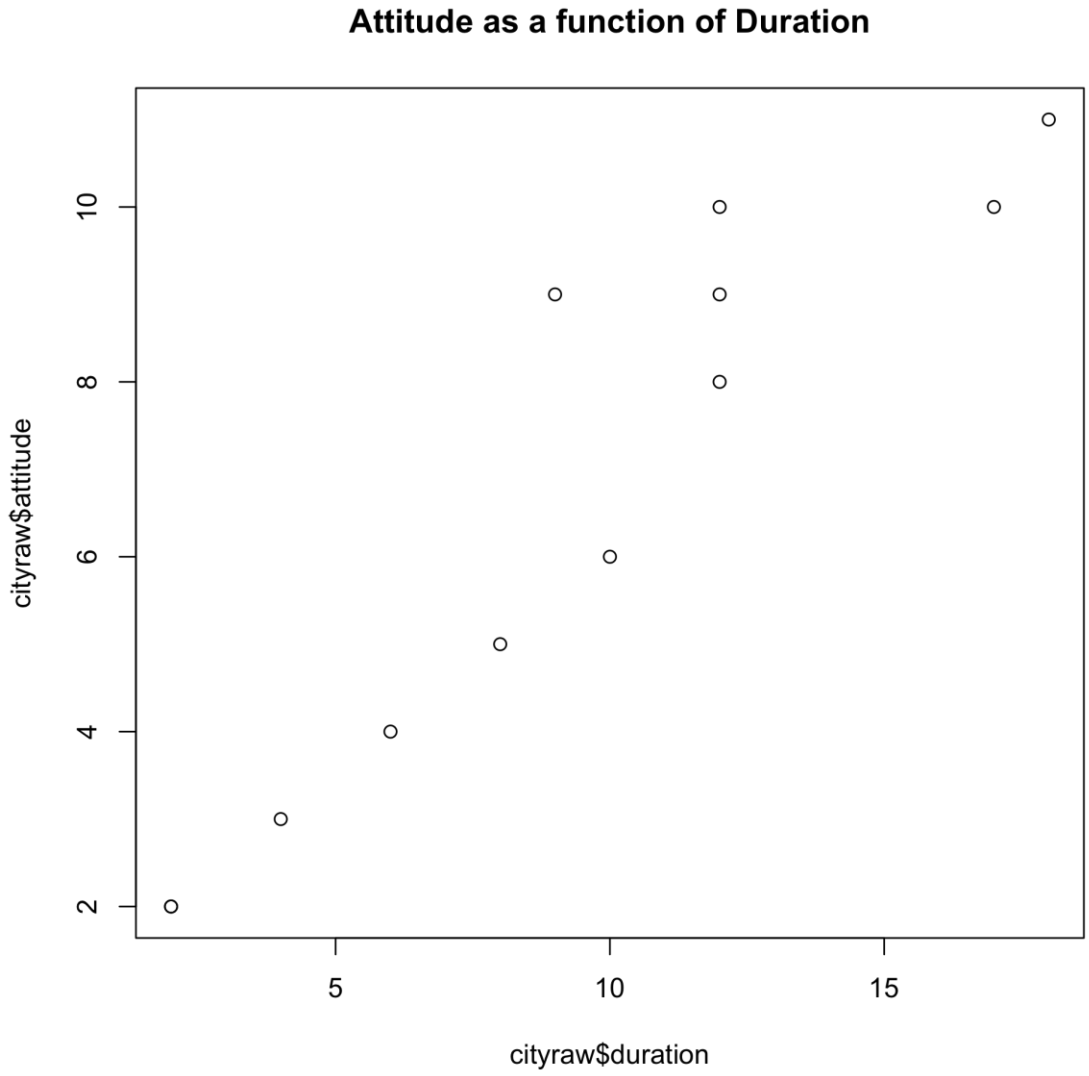
>

The data set, titled 'cityraw,' clearly consists of n = 12 observations on 3 variables, 'attitude,' 'duration,' and 'weather.' In what ways can R help us analysis this data using the powerful tools of regression analysis?

First, if we would like to 'eye ball' the relationships of the variables to one another, a simple and intuitive approach would be to create, and inspect, a scatterplot. Here is how we would do this:

```
> plot(cityraw$duration,cityraw$attitude,main="Attitude as a function of Duration")
```

The first variable in the parenthetical expression is displayed along the horizontal axis; the second along the vertical. We have also added the title "Attitude as a function of Duration."



Since we see that these two variables are systematically related---that is, related in a positive and linear fashion, in this case---we may wish to summarize this relationship with a simple linear regression equation.

A handy way of doing this is to define the regression model, using the 'lm' (for linear model) command, and assigning the model to an expression. Suppose we use the expression 'x,' to take an expression at random. (Any variable will do.) Then

```
> x<-(lm(cityraw$attitude~cityraw$duration))
```

In the terminology of R, we would say that 'x' is now an *model object*; the tilde symbol, "~," which is normally found in the upper left-hand corner of our keyboard,

is interpreted as "is defined by." If we type 'x,' we get the most basic regression printed output:

> x

Call:

lm(formula = cityraw$attitude ~ cityraw$duration)

Coefficients:

(Intercept) cityraw$duration

1.0793 0.5897

The only results that appear are (1) the regression model as defined by ourselves, and the regression equation itself. Thus we can see from the printout that the model itself is

cityraw$attitude = 1.0793 + (0.5897)*(cityraw$duration)

That's it. No diagnostics; no r-squared value; no significance tests; no ANOVA table. We will typically want at least some of these regression results, and, in this regard, R doesn't disappoint. If we return to our model object x, there are several easy steps we can take.

First, if we want the above results plus tests of significance, t statistics, and p-values, we need only type "summary(x)."

>

> summary(x)

Call:

lm(formula = cityraw$attitude ~ cityraw$duration)

Residuals:

Min 1Q Median 3Q Max

-1.1045 -0.7199 -0.3485 0.0941 2.6132

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.07932	0.74335	1.452	0.177
cityraw$duration	0.58972	0.07008	8.414	7.55e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.223 on 10 degrees of freedom

Multiple R-squared: 0.8762, Adjusted R-squared: 0.8639

F-statistic: 70.8 on 1 and 10 DF, p-value: 7.545e-06

>

As long as we know what we are looking for, everything is easily identified.

The r-square value is .8762 (which means that 87.62% of variation in the dependent variable is 'explained by' variation in the independent variable). Also importantly, the t statistic associated with the independent variable is t = 8.414 (with df = 10), and the p-value is 0.000007545 (which means that we would reject the null hypothesis that the independent variable is NOT linearly related to the dependent variable because alpha, the probability of a Type I error, is set at 0.05: remember that we reject the null hypothesis whenever the p-value is less than alpha.)

And should we want the ANOVA table, we need only ask be entering 'anova(x)':

> anova(x)

Analysis of Variance Table

Response: cityraw$attitude

	Df	Sum Sq	Mean Sq	F value	Pr(>F)

cityraw$duration	1	105.952	105.952	70.803	7.545e-06 ***
Residuals	10	14.964	1.496		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

As we see, this command provides the usual information found in the ANOVA table: Sums of Squares, degrees of freedom, F statistics, and significances or p-values.

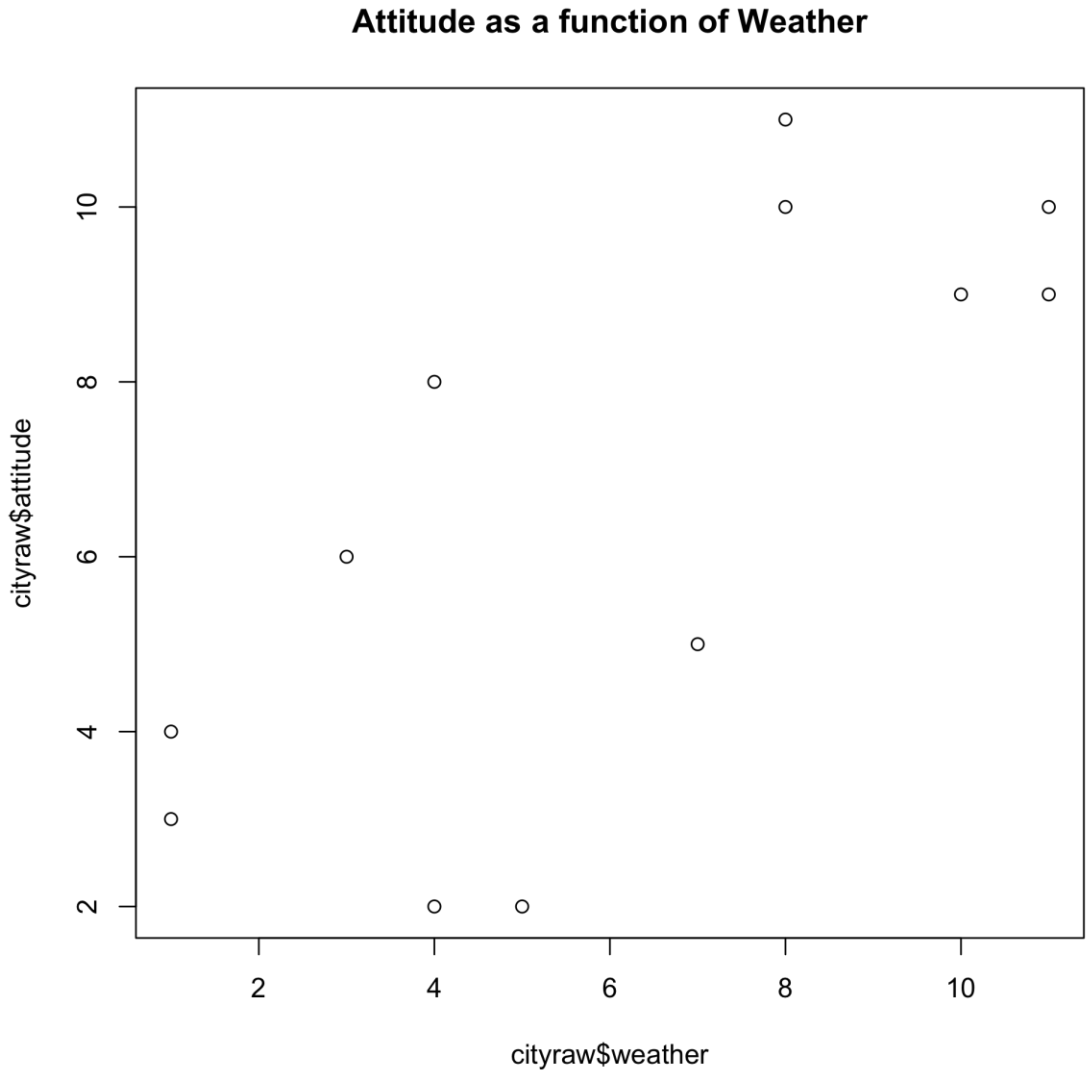
As we know, in the case of simple linear regression (where we have only one independent variable), the p-value associated with the F statistic in the ANOVA table will exactly equal the p-value associated with the t statistic in the coefficient table. In both cases,

p-value = .000007545

Let us now move on to the more interesting, more powerful, and more widely used multiple regression approach. The departure from the case of the simple linear regression model is that we now add additional independent variables. In our case, we will now add (to the analysis) the variable 'cityraw$weather.' Please see the Discussion Notes for more explanation of the meaning of this variable.

As before, let's begin (a good practice) by 'eye balling' the relationship, if any, between the new independent variable, cityraw$weather, and the dependent variable, cityraw$attitude.

```
> plot(cityraw$weather,cityraw$attitude,main="Attitude as a function of Weather")
```



We note that although the band of points seem to move from the lower left corner to the upper right, the relationship does not seem quite as strong as with the first independent variable. Still, there does appear to be a relationship that we might capture if we were to drop this new independent variable into the regression model and run the analysis.

Let's recycle our variable 'x' as we redefine our regression model, or *model object*.

```
> x<-lm(cityraw$attitude~cityraw$duration+cityraw$weather)
```

We note that we have now incorporated the second independent variable, cityraw$weather, and we have done so simply by adding it to the first independent

variable. Apart from that, everything is the same as before. We when type 'x' at the R prompt, we again recover only the most basic model information, the intercept term and the two partial regression coefficients:

```
> x

Call:

lm(formula = cityraw$attitude ~ cityraw$duration + cityraw$weather)

Coefficients:

   (Intercept)        cityraw$duration      cityraw$weather

        0.3373                  0.4811               0.2887
```

The new regression model (which can now be characterized as the multiple regression model because of the presence of *multiple* independent variable) can be expressed as

cityraw$attitude = 0.3373 + (0.4811)*(cityraw$duration) + (.2887)*(weather)

As before, if we want the above results plus tests of significance, t statistics, and p-values, we need only type "summary(x)."

```
> summary(x)

Call:

lm(formula = cityraw$attitude ~ cityraw$duration + cityraw$weather)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.2065	-0.5262	-0.1497	0.5443	1.4465

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.33732	0.56736	0.595	0.56679
cityraw$duration	0.48108	0.05895	8.160	1.89e-05 ***
cityraw$weather	0.28865	0.08608	3.353	0.00848 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8597 on 9 degrees of freedom

Multiple R-squared: 0.945, Adjusted R-squared: 0.9328

F-statistic: 77.29 on 2 and 9 DF, p-value: 2.149e-06

Here we have a wealth of useful information.

The r-square is now .945; the adjusted-r-square is .9328.

The F statistic associated with the overall regression model is F = 77.29; the associated p-value = .000002149.

And the t values for the two independent variables are:

For cityraw$duration: t = 8.160 with p-value = .0000189

And for cityraw$weather: t = 3.353 with p-value = .00848.

We should pause for a moment to see that we can also look up values of the F distribution for the purposes of defining our 'rejection region' in the hypothesis test concerning the overall regression model. To find the value of F which cuts off an area of .05 in the upper tail of the F distribution with df(numerator) = k = 2, and df(denominator) = n – k – 1 = 12 – 2 – 1 = 9, we simply request from R

```
> qf(.95,2,9)

[1] 4.256495
```

We can also ask R to provide us with some additional information about our regression equation.

For example, just as it is possible to derive confidence interval estimates for population means and population proportions (see the Chapter 8 Discussion Notes), it is also possible to derive confidence interval estimates for our regression coefficients. After all, just as our sample mean (and sample proportion) are 'sample statistics' used to estimate 'population parameters,' so can our regression coefficients be seen in this way: if the regression equation is calibrated on a sample, rather than a population, the regression coefficients are in fact 'sample statistics' too. Indeed, regression coefficients calibrated on a sample (rather than on a population) are random variables; they have their own sampling distributions; they have their own expected values, and they have their own standard deviations. Accordingly, each (sample) regression coefficient can be characterized with its own confidence interval estimate as well.

Using our *model object* x, all we need to do to obtain, say, the 95% confidence interval estimates of the (population) regression coefficients is write out the following expression:

```
 > confint(x,level=.95)
                        2.5 %          97.5 %
(Intercept)         -0.94613861  1.6207785
cityraw$duration     0.34771091  0.6144411
cityraw$weather      0.09393154  0.4833739
>
```

Thus for the variable, cityraw$duration, we can say that we are 95% confident that the true population partial regression coefficient (capturing the linear relationship between 'duration' and 'attitude') is no less than .34771091 and no more than .6144411. Put another way, and recalling that (in general) a confidence interval estimate can be expressed as *point estimate ± margin of error*, we can apply that general expression to our confidence interval estimates here

For cityraw$duration: .4811 ± .133

For cityraw$weather: .2887 ± .195

Another way in which we can use R to help us understand our data analytic problem is to have it provide *predicted values.* To understand what we are describing here, let's consider (1) one of our observations from the sample, and (2) our regression equation.

Let's select observation no. 6: this person, whoever she/he is, has provided the following values

Attitude = 6

Duration = 10 years

Weather = 3

Plugging this person's values on 'duration' and 'weather' into the regression equation, we can get a 'predicted value' on the dependent variable, 'attitude.' That is, we can use the regression equation to predict what a person's attitude toward life in San Diego might be (on a scale from 1 to 11) based entirely on how long they have lived in San Diego ('duration') and how much importance they place on having good weather ('weather'). Let's do this.

Predicted attitude = (.337) + (.481)*(10) + (.289)*(3) = 6.014

Clearly, a regression equation with strong predictive strength should give us values on the *predicted dependent variable* that are "close" to the values of the *actual dependent variable*. At least in the case of observation no. 6, we see that the prediction is rather good:

Actual dependent variable (actual value on 'attitude') = 6

Predicted attitude = (.337) + (.481)*(10) + (.289)*(3) = 6.014

Obviously, we would like to see how "close" all the predicted values on the dependent variable are to the actual values on the dependent values. Put another way, what is the correlation between the *actual dependent variable* and the *predicted dependent variable* across the entire data set?

To answer this question, let's (1) create predicted dependent variables for all observations in the sample, (2) combine this new information, as a new variable, into the original dataset, cityraw, and (3) correlation the two variables described above.

First, to derive all the predicted dependent variable values in R, and 'assign' those values to a new variable, say, 'y' (no reason we need 'y'; it could be anything we like)

```
> predict(x)->y
```

```
> y
       1        2        3        4        5        6        7        8        9       10       11
 2.454083  2.742736  2.550277  3.512429  6.206497  6.014038  7.264843  9.285412
7.553531  9.285412 10.824834
       12
11.305910
>
```

We note here that the predicted value for observation no. 6 (i.e., the sixth number in the row) is 6.014038, or just about exactly what we calculated above.

Second, let's 'bind' this new variable, y, into the original data set, cityraw, and name it a 'new' data set, 'newcityraw.'

```
>
> cbind(cityraw,y)->newcityraw
```

Here we have told R that we want it to 'bind' the new vector of information, y, into the original data set, cityraw, and to name this new data set, 'newcityraw.' Moreover, we have described specifically that we want it combined as a 'column variable' with the command 'cbind.' (If we want to add a row to the data set, we would use 'rbind' rather than 'cbind.')

Now let's take a look at the new data set, newcityraw:

```
> newcityraw
```

	attitude	duration	weather	y
1	2	2	4	2.454083
2	2	2	5	2.742736
3	3	4	1	2.550277
4	4	6	1	3.512429
5	5	8	7	6.206497

6	6	10	3	6.014038
7	8	12	4	7.264843
8	9	12	11	9.285412
9	9	9	10	7.553531
10	10	12	11	9.285412
11	10	17	8	10.824834
12	11	18	8	11.305910

>

As we can see, the new variable has been 'bound in' with the original data set, and it is named 'y.' Everything else is exactly the same.

To get an idea of how "close" our predicted dependent variable is to our actual dependent variable, we can of course simply look at the correlation coefficient. This diagnostic statistic, r, provides us with an intuitively appealing insight into the "goodness of fit." Values of r close to zero mean that the regression model is not predicting well; values of r close to 1 mean that the regression model is predicting very well. Let's see what r reveals:

> cor(newcityraw$attitude,newcityraw$y)

[1] 0.9721026

>

As we see, the correlation between the actual and predicted dependent variables is extremely high at .972, and so we would conclude that the 'data fit the model' very well, and the data points are "close" to the regression plane described by the regression equation.

One final point is that r is simply the positive square root of the multiple r-square. That is, since

r-square = .945

r = + $\sqrt{r-square}$ = + $\sqrt{.945}$ = + .972

We can see that r, the correlation coefficient between the actual and predicted dependent variables, is simply the positive square root of the multiple r-square.

Finally, let us return briefly to something mentioned a few lines above: adding a row to a data set. Suppose we wish to add an additional observation, or row of data, to the cityraw data set: the person in question reports having an attitude of '1' toward

life in San Diego, has been a resident for only 1 year, and assigns an importance of 1 (out of a possible 11) to the importance of having good weather. Thus this observation would be completely characterized by 1, 1, and 1.

```
> x<-c(1,1,1)

> x

[1] 1 1 1

> rbind(cityraw,x)->r
```

	attitude	duration	weather
1	2	2	4
2	2	2	5
3	3	4	1
4	4	6	1
5	5	8	7
6	6	10	3
7	8	12	4
8	9	12	11
9	9	9	10
10	10	12	11
11	10	17	8
12	11	18	8
13	1	1	1

Here we have told R that we want it to 'bind' the new vector of information, x, into the original data set, cityraw. Moreover, we have described specifically that we want it combined as a 'row variable' (and not a 'column variable') with the command 'rbind' (and not 'cbind'). As we can see from the amended data set above, we now have a new observation, the 13th, which has the values of '1' on each of the three variables. We have also named the new data set 'r.'

As an addition useful issue to consider in multiple regression, we sometimes like to know which independent variables are most powerful in predicting, explaining, or describing the relationship between them and the dependent variable. We might (naively) think that all we need to consider is the relative size of the partial

regression coefficient. After all, it stands to reason that larger regression coefficients should have associated with them greater predictive/descriptive/explanatory power than do the smaller ones. However, this is not the case.

To understand this issue better, let us consider a different data set, 'butler,' which is drawn from our textbook, and which consists of n = 10 observations on three variables. The data set consists of a trucking company which wishes to develop a better understanding of its work schedules for its drivers and trucks. The dependent variable is 'total daily travel time in hours.' The two independent variables are 'number of miles traveled' and 'number of deliveries.'

Here is the data set

> butler

	Assignment	Miles	Deliveries	Time
1	1	100	4	9.3
2	2	50	3	4.8
3	3	100	4	8.9
4	4	100	2	6.5
5	5	50	2	4.2
6	6	80	2	6.2
7	7	75	3	7.4
8	8	65	4	6.0
9	9	90	3	7.6
10	10	90	2	6.1

>

As we can see, the first observation consists of a 'driving assignment' in which the driver traveled 100 miles, made 4 deliveries, and was out for 9.3 hours. The ninth observation involves an 'assignment' in which the driver traveled 90 miles, made 3 deliveries, and was out for 7.6 hours. And so on.

To summarize the relationship, if any, between the dependent variable 'time' and the two independent variables, 'miles' and 'deliveries,' let's submit the data to a simple multiple regression analysis.

> t<-lm(butler$Time~butler$Miles+butler$Deliveries)

Here we can regressed the dependent variable on the two independent variables, and named, or titled, the results 't.'

> t

Call:

lm(formula = butler$Time ~ butler$Miles + butler$Deliveries)

Coefficients:

(Intercept)	butler$Miles	butler$Deliveries
-0.86870	0.06113	0.92343

From this, we can see that our regression equation is:

butler$Time = -.86870 + (.06113)*(butler$Miles) + (.92343)*(butler$Deliveries)

To get more information in determining which, if any, of the independent variables really predicts/describes/explains anything in terms of the variation in the dependent variable, we can submit 't' to 'summary.'

> summary(t)

Call:

lm(formula = butler$Time ~ butler$Miles + butler$Deliveries)

Residuals:

```
   Min      1Q  Median      3Q     Max
-0.79875 -0.32477  0.06333  0.29739  0.91333


Coefficients:
                    Estimate   Std. Error  t value  Pr(>|t|)
(Intercept)         -0.868701  0.951548    -0.913   0.391634
butler$Miles        0.061135   0.009888    6.182    0.000453 ***
butler$Deliveries   0.923425   0.221113    4.176    0.004157 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.5731 on 7 degrees of freedom
Multiple R-squared: 0.9038, Adjusted R-squared: 0.8763
F-statistic: 32.88 on 2 and 7 DF,  p-value: 0.0002762
```

What we note immediately is:

First, the adjusted-r-square = .8763

Second, the p-value associated with the regression model F statistic = .0002762. This indicates that at least one of the independent variables is significant, and that the regression model does indeed capture the relationship between the dependent variable and at least one of the independent variables rather well.

Third, the p-values associated with the t statistics of the two independent variables are .000453 and .004157 for 'miles' and 'deliveries,' respectively. This suggests that both the independent variables are significant, and should be retained in the regression model.

Thus, we know that the regression model itself is significant, it explains nearly 90% of variation in the dependent variable, and that both independent variables contribute to the model's power.

What we don't know is which independent variable is most influential. Here is how we can consider this issue. We note that the magnitude of the partial regression coefficient on the variable 'deliveries' is much larger than the magnitude of the other

partial regression coefficient----in fact, just over 15 times larger. However, this is a function of how the two variables are scaled, and it doesn't necessarily mean that 'deliveries' is in reality the most important of the two variables. To get more insight into this particular issue, we can standardize all three of the variables, dependent as well as independent, and run the regression again.

We recall that standardization involves transforming the original data into standardized scores by (from each observation) subtracting the mean of that particular variable, and then dividing the result by the standard deviation of that variable. In R, we are able to standardize every observation of each variable, one at a time, and assign it---that is, the entire, new vector of standardized values---to a variable. In this case, we assign the standardized values for the variable 'time' (and 'miles' and 'deliveries') to a new vector named 'ztime' (and 'zmiles' and 'zdeliveries').

```
> ztime<-(butler$Time-mean(butler$Time))/sd(butler$Time)

> zdeliveries<-(butler$Deliveries-mean(butler$Deliveries))/sd(butler$Deliveries)

> zmiles<-(butler$Miles-mean(butler$Miles))/sd(butler$Miles)
```

Next, we 'bind' together all three new variables into a new data set named 'std.butler.'

```
> std.butler<-cbind(ztime,zmiles,zdeliveries)
```

Before we print out the new data set, std.butler, let's derive the set of standardized data values for one of the observations, the 10^{th}. Work assignment 10 is completely characterized by

Time = 6.1 hours (mean of Time = 6.7 hrs; sd of Time = 1.629588 hrs)

Deliveries = 2 (mean of Deliveries = 2.9; sd of Deliveries = 0.875595 deliveries)

Miles = 90 (mean of Miles = 80 miles; sd of Miles = 19.578900 miles)

Then to standardize the variables values for observation 10:

Ztime = (6.1 – 6.7)/(1.629588) = -0.36822

Zdeliveries = (2 – 2.9)/(0.875595) = -1.0279

Zmiles = (90 – 80)/(19.578900) = 0.5108

To see if our results conform with those provided by R, we need only call up the new data set, 'std.butler.'

```
> std.butler
        ztime      zmiles     zdeliveries
 [1,]  1.5954958  1.0215078  1.2562885
 [2,] -1.1659392 -1.5322618  0.1142080
 [3,]  1.3500349  1.0215078  1.2562885
 [4,] -0.1227304  1.0215078 -1.0278724
 [5,] -1.5341305 -1.5322618 -1.0278724
 [6,] -0.3068261  0.0000000 -1.0278724
 [7,]  0.4295565 -0.2553770  0.1142080
 [8,] -0.4295565 -0.7661309  1.2562885
 [9,]  0.5522870  0.5107539  0.1142080
[10,] -0.3681913  0.5107539 -1.0278724
```

If we now conduct our multiple regression on the standardized set of data values from 'butler,' we will get results that will more clearly identify which of the independent variables is most important in accounting for the variation in the depndent variable, time.

```
> tz<-lm(ztime~zmiles+zdeliveries)

> summary(tz)

Call:

lm(formula = ztime ~ zmiles + zdeliveries)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.49015	-0.19930	0.03886	0.18249	0.56047

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.077e-18	1.112e-01	9.68e-18	1.000000
zmiles	7.345e-01	1.188e-01	6.182	0.000453 ***
zdeliveries	4.962e-01	1.188e-01	4.176	0.004157 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3517 on 7 degrees of freedom

Multiple R-squared: 0.9038, Adjusted R-squared: 0.8763

F-statistic: 32.88 on 2 and 7 DF, p-value: 0.0002762

>

We note that regardless of whether we run the multiple regression analysis on the raw, unstandardized data (from data set 'butler') or the standardized data (from std.butler), our basic statistics are exactly the same. In particular, note that, for the overall regression model, we have the following results from both regression analyses:

r-square = .9038

adj-r-square = .8763

F = 32.88 with a p-value = .0002762

And for the partial regression coefficients:

Miles: t = 6.182 with p-value = .000453

Deliveries t = 4.176 with p-value = .004157

For our purposes, however, we should note the difference in the actual regression model itself:

ztime = .0000 + (.7345)*(zmiles) + (.4962)*(zdeliveries)

What we note here is that (1) the intercept is now zero (in fact, whenever we conduct our multiple regression analysis on standardized data, the intercept term is forced through the origin: it will always be zero), (2) the coefficient on 'zmiles' is .7345, (3) the coefficient on 'zdeliveries' is .4962.

In fact, since the two independent variables are now expressed in terms of the same units----their own standard deviations, and not something as different as 'miles' and 'deliveries'---we can now see that the independent variable 'miles' is the most important of the two independent variables in terms of explaining the variation in the dependent variable, 'time.'

This is exactly the opposite conclusion of the one we would have reached had we compared the magnitude of the partial regression coefficients obtained by using raw, unstandardized data in our regression analysis. In that case, when we were attempting to compare independent variables with very different units (miles and deliveries) it seemed as is 'deliveries' was (by far) the most important of the independent variables.

The point is that, by standardizing our data, we can get a better insight into which are the most influential independent variables.