MATH1280 Notes

Robert Hoot

May 19, 2015

Contents

Chapter 1

Chapter 1 Notes

1.1 About These Notes

These notes might be useful to help explain difficult concepts or to explain concepts that assume that you have specific knowledge from other academic disciplines. The quizzes in this class do NOT contain anything that was derived from these notes, but these notes might help you to understand the textbook better. These notes are entirely optional. If you have any comments, questions, or suggestions about these notes, you can email robert.hoot@uopeople.edu, or, if you think other students might benefit from the question or comment, you can post to the Course Forum.

1.2 Read the Chapter

The link for the textbook is in the Learning Guide for Week 1. The Learning guide for Week 1 is long, so be sure to read all of the pages. The textbook is: *Introduction to Statistical Thinking (With R, Without Calculus)* by Benjamin Yakir.

A good way to read computer books is to read the full chapter straight through without touching the computer and without spending too much time on difficult parts. After you read the full chapter, go back and read it again, study it carefully, and write example programs as you go. Practice answering the questions at the end of the chapter using R.

The R commands for this Week 1 are not complicated, the main challenge might be to install R and learn how to execute the most basic commands. If you are able to finish Chapter 1 early and read to Chapter 2, you will be in better shape when we get to Chapter 4, which is more difficult than the first few chapters.

1.3 Thousands Separator and Decimal Separator in R

The textbook uses the period (or 'dot') to separate the integer part of the number from the decimal. In the United States, one thousand is sometimes written 1,000. When writing R code, do not add the comma when writing big numbers. The product of "one million and one" times 7 would be written:

```
1000001*7
```

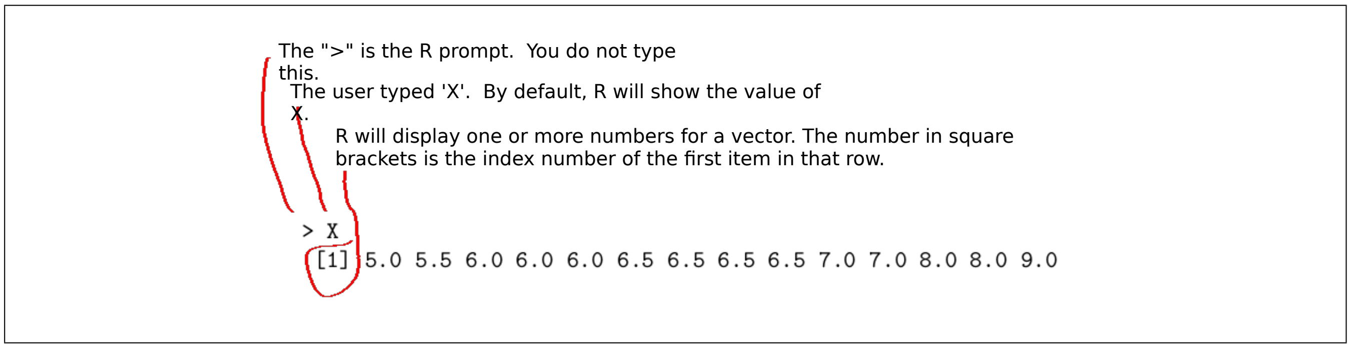
(You can also add space around the '*' character if you want to).

In some countries, the solution to 1 divided by 100 would be written `0,01`, but do not do that in this class—use the '.' character as the decimal separate and write: 0.01 or just .01. In R, it looks like this:

```
> 1 / 100
[1] 0.01
```

1.4 Reading R Output

When you run R commands, there is often something that is displayed on the screen, but the meaning of the output is not always obvious. Here are some notes on reading the output from R



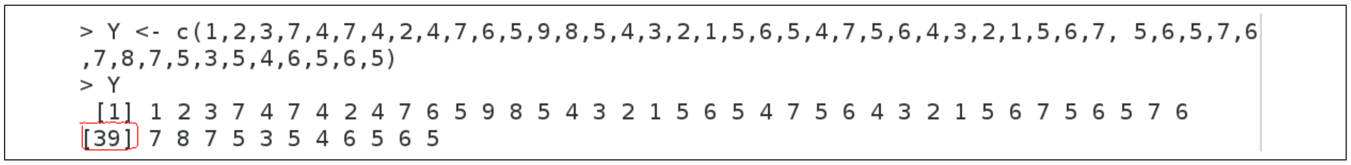
The ">" is the R prompt. You do not type this.

The user typed 'X'. By default, R will show the value of X.

R will display one or more numbers for a vector. The number in square brackets is the index number of the first item in that row.

```
> X
[1] 5.0 5.5 6.0 6.0 6.0 6.5 6.5 6.5 6.5 7.0 7.0 8.0 8.0 9.0
```

Figure 1.1: Page 9 example: If R is displaying a vector that has more than one line, it will start each line with the index number of the first value that is shown on that line.

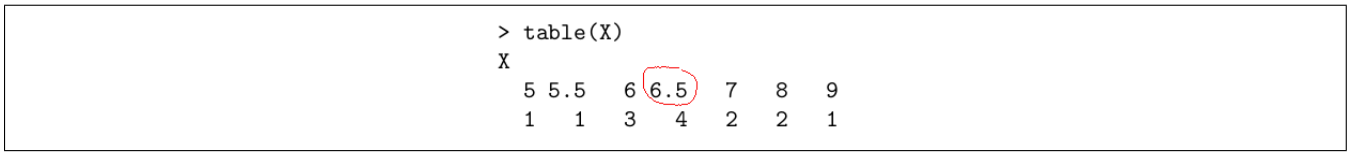


```
> Y <- c(1,2,3,7,4,7,4,2,4,7,6,5,9,8,5,4,3,2,1,5,6,5,4,7,5,6,4,3,2,1,5,6,7, 5,6,5,7,6
,7,8,7,5,3,5,4,6,5,6,5)
> Y
 [1] 1 2 3 7 4 7 4 2 4 7 6 5 9 8 5 4 3 2 1 5 6 5 4 7 5 6 4 3 2 1 5 6 7 5 6 5 7 6
[39] 7 8 7 5 3 5 4 6 5 6 5
```

Figure 1.2: Example of multi-line output from R. The numbers in brackets at the start of each line represent the index number of the entry that follows.

for the first chapter:

Figure 1.1 shows some R code that appears in the textbook (Yakir, 2009, p. 9). The figure shows how to interpret some simple R output. In the Figure 1.2, the second line of R output starts with `[39]`, which means that the number that follows is the 39th in the vector.



```
> table(X)
X
  5 5.5   6 6.5   7   8   9
  1   1   3   4   2   2   1
```

Figure 1.3: R output from the textbook (Yakir, 2011, p. 10). The first row of numbers serves as the header for the table. The value of 6.5 with the 4 below it means that the value 6.5 appears four times in the data that was sent to the `table()` command.

1.5 `table(round(X))`: What is a Function?

The textbook shows R commands like `table(X)`, where `table()` is a *function.* You can also think of functions in R as a type of computer command. Functions are sets of instructions that the computer can follow to do useful things. I will try to explain the arguments to functions and return values of functions so that you can understand what the computer is doing when it executes a command like this: `table(round(X))`. If you are already familiar with computer programming, you can probably skip this section.

When you type `table()` into R, and then put either a vector (a list of numbers or characters) or an *object* name (like X) into the parentheses, you are telling R to read the data and create a frequency table using that data. R does this by reading some commands that have been saved inside the R program (or in libraries). Those commands are associated with the word *table* so that R knows what to do with the data. When the `table()` command runs, it verifies that there is at least one value, then it starts to count the unique values in the data and build the table.

Most functions have a *return value.* That means a function that reads data might return a single number, a list of numbers, some text strings, or perhaps even a formatted report. If I use the `round()` function, R will take whatever numbers I give to it and round them to the number of digits that I specify. Consider this R code (do not type the `>` character):

```
> X <- c( 1.111, 1.093, 2.987, 3.033, 1.499, 2.030)
> round(X, 1)
[1] 1.1 1.1 3.0 3.0 1.5 2.0
```

In the first line of code above, the R *object* called X is defined as a vector of numbers: 1.111, 1.093, 2.987, 3.033, 1.499, 2.030. In other programming languages, you might call X a *variable*, but the textbook uses the more generic term *object.* You could say that the "input values" (or *arguments*) to the `round(X, 1)` command consist of the object X and the number 1. Those two values are "sent to" the `round()` function. The return value is the vector of numbers 1.1, 1.1, 3.0, 3.0, 1.5, 2.0. If you look at the individual numbers in the vector that was *returned*, you can see that each value has been rounded to one decimal place and printed on the computer screen.

Here is why all of this is important: When you execute R commands, you can type all the numbers the long way (like `c(1.111, 1.093, 2.987, 3.033, 1.499, 2.030)`), or you can enter an object name (like X), or you can process numbers using a function, in which case the "return value" from the function will be treated just like you typed all the numbers (or characters). There are some details that will be in next week's notes—the "return value" from the `table()` function is actually an object that contains the set of numbers in the second row of the report along with some extra information about how to display the report.

If I put one function inside the other, the inner-most function is executed first and the results are sent to the enclosing function. In the example below, the input data are first rounded to one decimal place, then the list of rounded numbers is sent to the `table()` command, which produces a formatted frequency table. Note that the output of the `table()` command does not start rows with `[1]`—the `table()` function shows a formatted report as opposed to producing an ordinary vector.

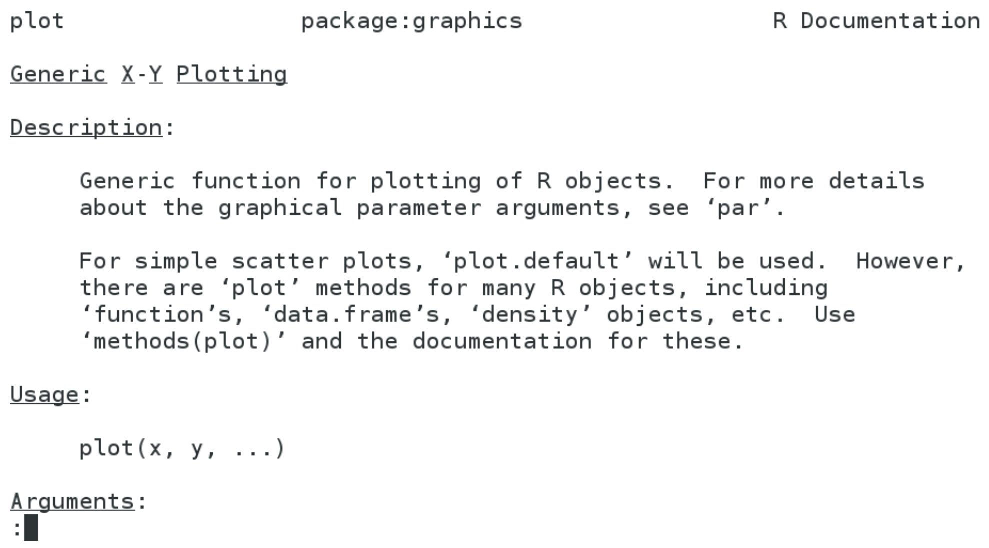
```
> X <- c(1.111, 1.093, 2.987, 3.033, 1.499, 2.030)
> table(round(X, 1))

1.1 1.5   2   3
  2   1   1   2
```

1.6 Getting Help for R commands

1.7 Help in Interactive R

There are many ways to get help for R commands, including searching the Internet and using the interactive help. If you wanted some information about the `plot` command, you could type `?plot` into R and you might see something like this:



```
plot                 package:graphics                  R Documentation

Generic X-Y Plotting

Description:

     Generic function for plotting of R objects.  For more details
     about the graphical parameter arguments, see ‘par’.

     For simple scatter plots, ‘plot.default’ will be used.  However,
     there are ‘plot’ methods for many R objects, including
     ‘function’s, ‘data.frame’s, ‘density’ objects, etc.  Use
     ‘methods(plot)’ and the documentation for these.

Usage:

     plot(x, y, ...)

Arguments:
:
```

Navigation of the help page might differ for each operating system, but try pressing page up and page down keys to navigate, and try pressing `q` to quit the help page.

Near the top of the help page is a description of the command, then it says that the basic usage is `plot(x, y, ...)`. The `x` and `y` are two variables that will be plotted, and the three dots means that there can be many other arguments. Lower in the help page, other options are listed. The options listed there include `xlab` and `ylab`. Those arguments control labels that are printed on the x-axis and y-axis of the plot. Try running this in R:

```
> X <- c(5,5.5,6,6,6,6.5,6.5,6.5,6.5,7,7,8,8,9)
> plot(table(X), xlab="My X-axis Label", ylab="My Y-axis Label")
```

If you want to search the help files for a key word, enter it with two question marks. In the next example, I wanted information about titles for the plot. I entered `??titles`, then a list of R *libraries* and functions appeared with short descriptions about what the function does. I noticed one called `graphics::title`, so I then pressed `q` to quit the first help page, and then ran the another command to get help on `graphics::title`:

```
> ??titles
> ?graphics::title
```

The word "graphics::" above means that the function called `title` is in the library called `graphics`. In some cases, the library is automatically loaded, so you can refer to the function using only the simple name: "title." I executed one more command to add a title to the plot command above: `title("My Title")`.

1.8 Online Sources of Help

There are many online tutorials for R. Some pages help with the programming aspect of R and some help with specific types of statistical analysis. The first couple sections from the R-project tutorial might be helpful `http://cran.r-project.org/doc/manuals/R-intro.html`. That tutorial contains more detailed programming information than we will cover in the class.

If you look for help online, remember that some sources will assume that you are using MS Windows and some will assume that you are using a UNIX-based operating system. The biggest difference is in how you specify file names (such as when you read or write a data file). In MS Windows, you might have to enter filenames in a odd format (with forward slashs) like this:

```
source('C:/Documents and Settings/user/My Documents/IntroStat/Week1.r')
```

The `source` command allows you to read an R program that was written in a regular file. In Mac OS X, Linux, or BSD you might have a filename like:

```
source('~/IntroStat/Week1.r')
```

Other commands should be the same on all operating systems.

1.9 Email your Instructor

If you are stuck, email your instructor. Your instructor might be able to save you time.

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 2

Chapter 2 Notes

2.1 Setting the Working Directory

The textbook (Yakir, 2011) provides instructions for setting the working directory in R[1]. The working directory is the directory where R looks for files that you read or write in R. The example in the textbook shows how to set the working directory in MS Windows. That is the best way to configure R for this class so that you do not have to deal the odd formatting of directory names in R. If you have already set the default working directories and you are able to read the data files as shown in Chapter 2 of the text book, then you can skip the next few paragraphs. If you are using a different operating system, check the notes from Week 1 for some additional tips.

If your working directory is not set correctly, you can run the `setwd` command to set it. The tricky part is that in MS Windows, you might have to enter directory names in a odd format (with forward slashes) like this: `setwd('C:/Documents and Settings/user/My Documents/IntroStat')`. In Mac OS X, Linux, or BSD you might have enter directory names like `setwd('~/IntroStat')`. Once you set the working directory, you can read and write files by specifying only the file name and not the full path.

To check your working directory in a couple ways. One is to run this command in R:

[1]These notes were prepared by Robert Hoot for MATH1280 at UoPeople. robert.hoot@uopeople.edu

```
> getwd()
```

That command shows the name of the working directory. You can double-check it by listing the files that are in that directory:

```
> dir()
```

2.2 Using R as a Calculator

2.2.1 Basic Math

Chapter 1 of the textbook (Yakir, 2011) introduced some basic math using R. You can use R as a calculator, and doing so might help you to understand some of the R commands later.

Here is simple R program that you can paste into R to show basic math operations:

```
# Addition"
1+2

# Subtraction
10-2

# Multiplication:
3*7

# Division
21/5

# Exponentiation (2 raised to the 8th power)
2^8
```

Some operations are performed with functions or can be calculated in several ways:

```
# Square root
# These two example return the principal/positive square root.
sqrt(81)
81^.5

# cosine (in radians)
cos(2*pi)


# you can use parentheses as expected
(1 + 9) / 2
```

2.2.2 Vector Math

Unlike other programming languages, many commands in R work the same way for single numbers as they do for vectors (ordered lists of numbers). If I set X equal to 5 in R, X would be a vector of length 1

```
> X <- 5
> X
[1] 5
```

If I set X equal to many numbers it looks like this:

```
> X <- c(5,5.5,6,6,6,6.5,6.5,6.5,6.5,7,7,8,8,9)
> X
[1] 5.0 5.5 6.0 6.0 6.0 6.5 6.5 6.5 6.5 7.0 7.0 8.0 8.0 9.0
```

What happens if I multiply X by 1.25?

```
> X <- c(5,5.5,6,6,6,6.5,6.5,6.5,6.5,7,7,8,8,9)
> X * 1.25
```

```
[1]  6.250  6.875  7.500  7.500  7.500  8.125  8.125  8.125  8.125  8.750
[11]  8.750 10.000 10.000 11.250
> X / 3
[1] 1.666667 1.833333 2.000000 2.000000 2.000000 2.166667 2.166667 2.166667
[9] 2.166667 2.333333 2.333333 2.666667 2.666667 3.000000
>
```

A single number is sometimes called a *scalar*. If you multiply a vector by a scalar in R, R will mutiply each number in the vector by the scalar. If you divide a vector by a scalar, each number in the vector will be divided by the scalar. In the example above, I divided the vector Y by the scalar 3. The result shows the answers rounded to six decimal places.

2.3 Rounding

If you want to round a single number or a vector of numbers, you can use the `round()` function.

```
> height <- c(171.0482, 202.9851, 180.9481, 190.1809,
+ 176.3147, 163.6866, 182.9220, 200.7493, 180.6477, 166.0683)
>
> round(height, 2)
 [1] 171.05 202.99 180.95 190.18 176.31 163.69 182.92 200.75 180.65 166.07
> round(height, 0)
 [1] 171 203 181 190 176 164 183 201 181 166
```

2.4 `freq/sum(freq)` Explained

Page 17 of the textbook shows this:

```
> freq <- table(work.hours)
> freq
```

```
work.hours
2 3 4 5 6 7
3 5 3 6 2 1
> sum(freq)
[1] 20
> freq/sum(freq)
work.hours
   2    3    4    5    6    7
0.15 0.25 0.15 0.30 0.10 0.05
```

The `freq/sum(freq)` command above might be confusing. When `freq` is displayed above, it looks like there are two rows of data, but what you are seeing is a *formatted* display of the data. There are several R commands that can help you to understand what is in an R object. One commands is `dim()`.

```
> dim(freq)
[1] 6
```

The `dim()` command shows information about the dimensions of an R object. In this case, `dim()` tells us that there is one dimension to `freq`, and its length is 6. That means that it is a vector of length six. If you run the `dim()` command on a data frame with 100 rows and 2 columns, you would see `[1] 100 2` in the output from the `dim()` command.

I can double check which numbers are stored in `freq` using the `as.vector` command (that command works on vectors, not data frames):

```
> freq
work.hours
2 3 4 5 6 7
3 5 3 6 2 1
> as.vector(freq)
[1] 3 5 3 6 2 1
```

Using the `as.vector` command above, I was able to verify that the numbers 3, 5, 3, 6, 2, 1 are the true values stored in `freq`, and I might guess that the other numbers displayed earlier were some kind of heading. I can use the `attributes()` command to get more information about `freq` (the `attributes()` command provides information about fancy data objects that are created by R functions):

```
> attributes(freq)
$dim
[1] 6

$dimnames
$dimnames$work.hours
[1] "2" "3" "4" "5" "6" "7"


$class
[1] "table"
```

The first piece of output from the `attributes` command describes the dimensions (just like we saw from the `dim()` command), but it also describes some "dimnames." The dimnames correspond to the column headers when we type `freq` into R:

```
> freq
work.hours
2 3 4 5 6 7
3 5 3 6 2 1
```

So, getting back to the example from page 17 of the textbook:

```
> freq/sum(freq)
work.hours
   2    3    4    5    6    7
0.15 0.25 0.15 0.30 0.10 0.05
```

R is actually calculating

```
> as.vector(freq) / sum(freq)
[1] 0.15 0.25 0.15 0.30 0.10 0.05
```

and adding some column titles to the report. We can see the pieces of the calculation like this:

```
> sum(freq)
[1] 20
> as.vector(freq)
[1] 3 5 3 6 2 1
> as.vector(freq) / sum(freq)
[1] 0.15 0.25 0.15 0.30 0.10 0.05
```

I could have typed the numbers into R like this to get the same result:

```
> c(3/20, 5/20, 3/20, 6/20, 2/20, 1/20)
[1] 0.15 0.25 0.15 0.30 0.10 0.05
```

2.5 More Tips on Reading Data Files

Based on some early results from the Unit 2 Assignment, I can see that some people could use some tips for using functions in R (this is mainly for non-programmers). I will give a full example that also shows some steps that help you to ensure that you are reading data from the directory where your data file is.

First use the `getwd()` function to check your directory (your input directory will be different from mine), and be sure that your input file is there. You can also run the `dir()` command to list files in that directory. I use the `names()` or `colnames()` command to verify the column names that were read into the data frame. If you see error messages, then stop what you are doing and fix it—you might have to move your data file into the directory that was reported by the `getwd()` function.

```
> getwd()
```

```
[1] "/home/S1/Documents/UOP/MATH1280/data"
> flower.data <- read.csv("flowers.csv")
> names(flower.data)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
> colnames(flower.data)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

You might get errors if you open the `.csv` files in a spreadsheet program and then same them again. The problem is that Excel might not save the file in the correct format, so your good file will be overwritten with one that has a bad format.

It is important to understand how to identify an input error when you use commands like `names()`, `colnames()`, or `objects()`. If you run one of these commands and see `NULL` as the answer, then your data object is empty. Maybe you need to read the input file or enter the values to put something in that object. If you see something like the following, it means that there was a parse error:

```
> names(ex1.data)
[1] "id..sex...height."
```

Note that there is only one set of double quotes, and instead of spaces between `id` and `sex` there are dots. This means that the computer did not recognize the column separator character (which is usually a comma). If you see this error there are two things to consider: (a) if you opened the `.csv` file in a spreadsheet program and then save it (or the computer auto-saved it), the spreadsheet program might have ruined the formatting of the file, or (b) for a file that you created, you probably forgot to put double quotes around the column names or forgot to put commas between the columns. You can email your instructor if you have a problem. If you are working on a project outside the class, you might need to read the instructions for the `read.table()` function (you do not need this command for this class, but you could enter `?read.table` into R and read the help file for it, then set the values for `header`, `sep`, `dec` and whatever else is needed to match your input file).

You can access a column in the flower.data data frame using the `$` notation:

```
> flower.data$Petal.Width
```

```
  [1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 0.2 0.2 0.1 0.1 0.2 0.4 0.4 0.3
 [19] 0.3 0.3 0.2 0.4 0.2 0.5 0.2 0.2 0.4 0.2 0.2 0.2 0.2 0.4 0.1 0.2 0.2 0.2
 [37] 0.2 0.1 0.2 0.2 0.3 0.3 0.2 0.6 0.4 0.3 0.2 0.2 0.2 0.2 1.4 1.5 1.5 1.3
 [55] 1.5 1.3 1.6 1.0 1.3 1.4 1.0 1.5 1.0 1.4 1.3 1.4 1.5 1.0 1.5 1.1 1.8 1.3
 [73] 1.5 1.2 1.3 1.4 1.4 1.7 1.5 1.0 1.1 1.0 1.2 1.6 1.5 1.6 1.5 1.3 1.3 1.3
 [91] 1.2 1.4 1.2 1.0 1.3 1.2 1.3 1.3 1.1 1.3 2.5 1.9 2.1 1.8 2.2 2.1 1.7 1.8
[109] 1.8 2.5 2.0 1.9 2.1 2.0 2.4 2.3 1.8 2.2 2.3 1.5 2.3 2.0 2.0 1.8 2.1 1.8
[127] 1.8 1.8 2.1 1.6 1.9 2.0 2.2 1.5 1.4 2.3 2.4 1.8 1.8 2.1 2.4 2.3 1.9 2.3
[145] 2.5 2.3 1.9 2.0 2.3 1.8
```

If you want to see rounded data, you can put any numeric object name inside the `round()` function:

```
> round(flower.data$Petal.Width, 0)
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [38] 0 0 0 0 0 0 1 0 0 0 0 0 0 1 2 2 1 2 1 2 1 1 1 1 2 1 1 1 1 2 1 2 1 2 1 2 1
 [75] 1 1 1 2 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2
```

Here is a tip for new programmers: the `round()` function does not change the original data object. If we look at the original data object again, it is unchanged:

```
> flower.data$Petal.Width
  [1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 0.2 0.2 0.1 0.1 0.2 0.4 0.4 0.3
 [19] 0.3 0.3 0.2 0.4 0.2 0.5 0.2 0.2 0.4 0.2 0.2 0.2 0.2 0.4 0.1 0.2 0.2 0.2
 [37] 0.2 0.1 0.2 0.2 0.3 0.3 0.2 0.6 0.4 0.3 0.2 0.2 0.2 0.2 1.4 1.5 1.5 1.3
 [55] 1.5 1.3 1.6 1.0 1.3 1.4 1.0 1.5 1.0 1.4 1.3 1.4 1.5 1.0 1.5 1.1 1.8 1.3
 [73] 1.5 1.2 1.3 1.4 1.4 1.7 1.5 1.0 1.1 1.0 1.2 1.6 1.5 1.6 1.5 1.3 1.3 1.3
 [91] 1.2 1.4 1.2 1.0 1.3 1.2 1.3 1.3 1.1 1.3 2.5 1.9 2.1 1.8 2.2 2.1 1.7 1.8
[109] 1.8 2.5 2.0 1.9 2.1 2.0 2.4 2.3 1.8 2.2 2.3 1.5 2.3 2.0 2.0 1.8 2.1 1.8
[127] 1.8 1.8 2.1 1.6 1.9 2.0 2.2 1.5 1.4 2.3 2.4 1.8 1.8 2.1 2.4 2.3 1.9 2.3
[145] 2.5 2.3 1.9 2.0 2.3 1.8
```

Here is another tip for new programmers: when you need to perform a calculation on rounded data, insert the whole expression with `round()`:

```
> table(round(flower.data$Petal.Length, 0))

 1  2  3  4  5  6  7
24 26  3 34 35 24  4
```

If you wanted a table of square roots of petal lengths, rounded to one decimal place, you would use the sqrt function (or you could use `x^.5` notation).

```
> table(round(sqrt(flower.data$Petal.Length), 1))

  1 1.1 1.2 1.3 1.4 1.7 1.8 1.9   2 2.1 2.2 2.3 2.4 2.5 2.6
  2   9  26  11   2   1   2   5  15  17  18  17  16   5   4
```

2.6 The Data Are...

Here is some trivia for you. You might have noticed this sentence in Chapter 2 of your textbook: "Qualitative data are generally described by words or letters" (Yakir, 2001, pg. 24). Professional statisticians consider the word *data* to be plural, so they say "the data are..." instead of "the data is...." If there is one piece of data, you might use the singular version: *datum*, but you will rarely see the word *datum*. Students, and even instructors, sometimes say "The data is...." You should attempt to use the word *data* with plural verbs, but do not subtract points on peer evaluation if a student uses *data* with a singular verb (unless you are specifically instructed to do so).

If you think it sounds strange to say "the data are," you could use the word *dataset* instead. Here are some examples: "The dataset contains four variables." "The dataset is stored in a text file."

2.7 Email your Instructor

If you have trouble with the example problems or are not sure about other parts of the chapter, you can email your instructor.

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 3

Chapter 3 Notes

3.1 Histograms

Histograms are graphical displays of data that have a few important features. Yakir (2011) described histograms as follow:

> A histogram consists of contiguous boxes. It has both a horizontal axis and a vertical axis. The horizontal axis is labeled with what the data represents (the height, in this example). The vertical axis presents frequencies and is labeled Frequency. By the examination of the histogram one can appreciate the shape of the data, the center, and the spread of the data. (Yakir, 2011, p. 31)

It might be tempting to create a plot (like the plot in Yakir, 2011, p. 12), and call it a histogram, but that would be a mistake. The bars on a histogram are usually *contiguous*. Things that are contiguous are physically touching one another (or represented as touching one another in a diagram). It is possible for a bar in a histogram to not touch another bar, but that would usually happen only if there are outliers or data that is draw from an irregular distribution. Outliers are discussed on page 33.

3.2 Data Classes in R

Yakir (2011) focuses on the distinction between *factors* and *numeric* variables (p. 24). This distinction will become more important later because it will help you to determine the kind of statistical analysis you can use with your data. You can use the `class()` command in R to determine how R will interpret simple data objects. Sometimes fancy R functions will return data objects that have custom classes, but for regular data, the `class` command often shows one of these four values:

1. numeric
2. integer (a specific type of numeric – it counts as numeric)
3. character (regular text strings)
4. factor (an official factor as described in your textbook)

If you enter data directly in to R, the classes will typically look like this:

```
> a <- 1.234
> b <- 7
> c <- c(1.123, 4.2, 5.92, 4.33)
> d <- c(1.123, 4.2, 5.92, 4.33, 100)
> e <- c( 1, 2, 3, 33, 49, 55)
> class(a)
[1] "numeric"
> class(b)
[1] "numeric"
> class(c)
[1] "numeric"
> class(d)
[1] "numeric"
> class(e)
[1] "numeric"
```

Classes for factors or character strings look like this:

```
> f <- c('apple', 'orange', 'pear', 'pear', 'orange')
> class(f)
[1] "character"
>
> g <- as.factor(c('apple', 'orange', 'pear', 'pear', 'orange'))
> class(g)
[1] "factor"
```

Note that if you create your own *factor*, you might need to tell R that the data should be interpreted as a factor by using the `as.factor()` command as shown above.

The other common type of R data class is `integer`. R will identify integers when you use the `read.csv()` command to read a column of data that is exclusively in integer format. There is no need in this class to use the as.integer command–if all the numbers are integers and are entered accurately, R will produce the same results regardless if whether the `class()` command says that the data are `integer` or `numeric` format.

```
> e <- c( 1, 2, 3, 33, 49, 55)
> e.int <- as.integer(e)
> e.int
[1]  1  2  3 33 49 55
> d <- c(1.123, 4.2, 5.92, 4.33, 100)
> d.int <- as.integer(d)
> d.int
[1]   1   4   5   4 100
> d
[1]   1.123   4.200   5.920   4.330 100.000
```

Note that the `as.integer()` command will truncate (not round) data, so the third value in the `d` vector, `5.920`, became `5` in the `d.int` vector.

If you examine the `ex.1` data frame (Yakir, 2011, p. 23), you should see something like this:

```
> class(ex.1$sex)
```

```
[1] "factor"
> class(ex.1$height)
[1] "integer"
```

In this case, the researchers entered height data in integer format, meaning that there were no decimal places in the input file for `height`.

3.3 Examining Data

3.3.1 Column Names

There is an example in the book using this command (Yakir, 2011, p. 30)

```
> ex.1 <- read.csv("ex1.csv")
```

Remember that the first > character is the R prompt—you do not type it:

The ex.1 data frame is small and has only three columns, but it might be useful to learn how to examine the data frame so that you know the methods that can be used to examine more complicated data. If you wanted to display all the data, you could try this:

```
> ex.1
```

You should see 100 rows of data scroll across the screen. You can scroll up to see the first few rows. Each row represents one *observation*.

The column called id is a numeric identifier. The record identifier might be used to validate the data. For example, if height data were recorded with the student number and there was a problem with one of the entries (a height of 2000 centimeters) then the researchers could look at the id number and go get a correct reading for that student. There is typically no need from a strictly statistics viewpoint to add a record ID, but in practice it is very useful to ensure data integrity. If data is collected across time, then some kind of record ID would be needed so that the researcher could link the observation for the first time period to the observation in the second time period.

Another useful command for interrogating a data frame is the colnames() command:

```
> colnames(ex.1)
[1] "id"     "sex"    "height"
```

That command is useful when the data frame has many columns or you need to check the exact spelling of the column name.

3.3.2 Array Notation in R

The text book does not introduce array notation until Chapter 8, so this is optional for now. Perhaps it will be helpful to know how to use array notation in R so that you can look at data in .csv data files that we read.

If you read a `.csv` file into an R object named `ex.1` and you wanted to see the first 10 rows of data without having to scroll up, you could try this command

```
> ex.1[1:10, ]
        id    sex height
1  5696379 FEMALE    182
2  3019088   MALE    168
3  2038883   MALE    172
4  1920587 FEMALE    154
5  6006813   MALE    174
6  4055945 FEMALE    176
7  9263269 FEMALE    193
8  4742356 FEMALE    156
9  6012071   MALE    157
10 4588777   MALE    186
>
```

You can now see the column names and some example data. The `1:10` part of the `ex.1[1:10, ]` command above means to create a data object that contains all the integers from 1 to 10, inclusive. It is a quicker way of saying this:

```
> ex.1[ c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), ]
```

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 4

Chapter 4 Notes

4.1 Introduction

This chapter, and the next will require more study time that the prior chapters took. You can do yourself a favor by dedicating as much time as you can to studying these chapters. You will be in good shape if you can master the next two chapters.

In this chapter, the main thing that we we will be discussing is variability of data, and we will do so in the context of samples, populations, and random variables. The notes below are my attempt to review the main ideas so that you might understand the detail in the book better. Random variables might be confusing for some of you: "A random variable is the future outcome of a measurement, before the measurement is taken" (Yakir, 2011, p. 53). If you have a data file with real numbers in it, it is not a random variable, because a random variable is an abstract concept. There is a section below with more information.

In Chapter 4, you might be tempted to skip some ideas that sound familiar, but do not skip anything. For example, you probably already understand what the mean is, and when you see the thing called *expectation*, you might be tempted to skip it because it sounds the same as the mean. You will do better in this class if you really understand the various equations for expectation (e.g., Yakir, 2011, p. 57–58). We also introduce means for two different things: $\bar{x}$ represents the sample mean and μ represents the population mean.

4.2 Sample Mean vs Population Mean

There are two equations for the mean: one is for the mean of a sample ($\bar{x}$) and one is for the mean of a population (μ). The mean of a sample is a statistic and the mean of a population is a parameter (see Yakir, 2011, p. 52 for the definition of *parameter*). When you read $\bar{x}$, say "x-bar." When you read μ, say "me-you" quickly.

Here is one way to calculate the mean of a SAMPLE:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

(Yakir, 2011, p. 50)

It means: add all the values of x in the sample and divide by the number of observations in the sample. If your sample includes 100 observations, then add all the values and divide by 100. When you read the symbols in the equation above, you can read it in different ways. One way to read it is: "x-bar equals the sum from i equals 1 to n of x sub i." You could also say: "the sample mean of x equals the sum of all observed values of x divided by the numbers of observations for x." We can also look at this equation another way.

$$\begin{aligned} \bar{x} &= \big(\sum_{i=1}^{n} x_i\big)\frac{1}{n} \\ \bar{x} &= \big(\sum_{i=1}^{n} x_i\big) \times \text{“probability of selecting observation } i\text{”} \end{aligned} \tag{4.1}$$

Here are some example numbers from the book: "1, 1, 1, 2, 2, 3, 4, 4, 4, 4, 4" (Yakir, 2011, p. 57). There are 11 observations. The probability of selecting the first observation is 1/11. The probability of selecting the second observation is 1/11. The probability of selecting the third observation is 1/11, and so on. Note that I am not referring to the probability of selecting a "1"—I am referring to the probability of selecting one of the 11 observations. The book uses P(x) to denote the probability of selecting a given observation (Yakir, 2011, p. 57). It might be more precise to say P(x_i), to denote the probability of selecting the ith observation, and in our example, i can be any integer from 1 to 11.

The equation for the entire population is similar, but a different letter is used and the equation applies to calculations that are based on the entire population (not a sample from a population):

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} \tag{4.2}$$

Note that the capital letter N usually refers to the number of observations in the entire population—lower case n usually refers to the number of observations in a sample.

4.3 Random Variable

Many statistical equations and concepts are based on random variables. A random variable can be constructed using different distributions (introduced in Chapter 5), but here is one definition:

> A random variable is the future outcome of a measurement, before the measurement is taken. It does not have a specific value, but rather a collection of potential values with a distribution over these values. After the measurement is taken and the specific value is revealed then the random variable ceases to be a random variable! Instead, it becomes data. Yakir, 2011, p. 53.

Here is a simple example. We can think of rolling dice—or a single die (a cube with six sides). Let's say that I am planning on rolling a single die 20 times tomorrow, and I want to get an idea what to expect. I know that the die as six sides, and that the sides are labeled 1 through 6. Assuming that the die is fair, the probability that I will roll a 1 on any given roll is 1/6. The probability that I will roll a 2 is also 1/6. The probability that I will roll a 3 is also 1/6, and so forth. I can describe the population of numbers associated with rolling a die by listing the six possible values and their associated probabilities.

When we talk about more complicated data, like the height of every person on the planet, the numbers are more varied, but over the next few chapters we will learn how we can describe that distribution of data using mathematical models. Sometimes the models are not perfect, but they help us to simplify analysis and still get very reliable results.

Here is an example of what we can do when reasoning about a random variable. If we know that the height of students varies in a particular way (the histogram looks similar to a bell curve), then we can make some educated guesses about the probability of the mean height of a group of students who are selected randomly. If we know that the mean height of male students is 2 meters with a standard deviation of 10 centimeters, then we can use the techniques that we will learn in this class to say that it is unlikely that the mean height of a group of randomly-selected students would be 2.5 meters. If we did NOT make a random sample of students, but instead measured the height of people on the basketball team, the result might be 2.5 meters, but it would be exceedingly unlikely that a *random* selection of students would be that tall.

4.4 Expectation

The expectation of a random variable is a more precise way to talk about the mean: "The expectation marks the center of the distribution of a random variable" (Yakir, 2011, p. 57). Perhaps more precisely, the expectation is the weighted mean of the distribution. *Expectation* is a useful concept because when speaking about a hypothetical variable that does not exist in reality, it is a bit awkward to talk about the mean: I can not find the mean of a vector when it does not have any specific values in it.

Here is an important equation for expectation:

$$\mathrm{E}(X) = \sum \big(x \times \mathrm{P}(X = x)\big) \tag{4.3}$$

(Yakir, 2011, p. 57).

This says that the expectation of random variable X is equal to the sum of the products of each observed value of x times the probability that the observation will occur. This equation can be translated into R. In the next example, I created an x variable and a p variable that represents the probability of each observation of x. I multiplied the two variables and found the sum. I assume that the values of x and p represent the known probabilities for a given population.

You will need to understand how to find the expectation of a variable when you are given the values that can occur and the probabilities that those values will occur. In the following example, there is a distribution of numbers described by the data in x and p. The only values that can exist in this imaginary distribution (random variable) are lised in the R object that I called x. If I selected randomly from this random variable, the probabilities associated with getting those values are shown in the R object called p:

```
> x = c(  4,   6,   7,  8,  9, 10, 12,  13,  14, 16)
> p = c( .2, .05, .05, .1, .1, .1, .2, .05, .05, .1)
> x*p
 [1] 0.80 0.30 0.35 0.80 0.90 1.00 2.40 0.65 0.70 1.60
> sum(x*p)
[1] 9.5
```

The expected value of a group of numbers drawn from this random variable would be 9.5.

I will try to show this a different way by using a trick (see Yakir, 2011, p. 44). I will generate a *sample* of data that has the characteristics of the random variable that is described by a list of values in the variable and their associated probabilities. I will do this by converting p into frequencies (instead of relative frequencies):

```
> x = c(  4,   6,   7,  8,  9, 10, 12,  13,  14, 16)
> p = c( .2, .05, .05, .1, .1, .1, .2, .05, .05, .1)
> # Because my p values above have only 2 decimal places,
> # I can multiple them by 100 to get integers:
> p.2 <- p * 100
> p.2
 [1] 20  5  5 10 10 10 20  5  5 10
```

The first "20" above means that the first value in x (which is "4") occurs 20 times in my sample that I am creating.

The next command will take each value of x and make redundant copies of each value according to the corresponding value in the R object called $p.2$. In other words, I will create 20 fours, and 5 sixes, and 5 eights and 10 tens....

```
> x.2 <- rep(x, p.2)
> x.2
  [1]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  6  6  6  6  6
 [26]  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9  9
 [51] 10 10 10 10 10 10 10 10 10 10 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 [76] 12 12 12 12 12 13 13 13 13 13 14 14 14 14 14 16 16 16 16 16 16 16 16 16 1
```

```
> length (x.2)
[1] 100
```

I now have an R object that I called $x.2$, and it contains a sample that has the same distribution of the random variable described by the values in x and the corresponding probabilities in p.

```
> mean(x.2)
[1] 9.5
```

so the mean here is the same as the calculated expectation above.

4.5 Variance and Standard Deviation

There are two calculations for standard deviation: one is for a sample and one is for the population. Likewise, there are two calculations for variance: one for a sample (Yakir, 2011, p. 58) and one for a population (Yakir, 2011, p. 52). The standard deviation is the square root of the variance. The equation for the POPULATION variance is:

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N} \tag{4.4}$$

This equation says that the POPULATION variance (σ^2) equals the sum of each difference between the value of x and the population mean of x (where the population mean is μ), all divided by the number of observations in the POPULATION. The variance for a population is written σ^2 (say "sigma squared"). Note that $(x_i - \mu)$ is the difference between one of the x values and the mean of the population.

There are several (equivalent) ways to express the sample variance. The sample variance is calculated using real data that you draw from a population. The result represents the estimate of the population variance that is made by looking at data from a sample. The equation that I usually use is this:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} \tag{4.5}$$

Note that the denominator for the sample variance is $n - 1$ whereas for the population the denominator is N. For the sample, n is the number of observations (number of rows of data). Using algebra, the equation can be written in a different form.

$$s^2 = \frac{n}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2\frac{1}{n} \tag{4.6}$$

In this format (similar to the second equation in Yakir, 2011, p. 58) the first part $(\frac{n}{n-1})$ is an adjustment to account for small sample sizes (degrees of freedom). We will not cover the details of that adjustment in this course other than to say if the sample size is very small, the resulting estimate of the variance will increase. The second part is the sum of squared deviations, and the third part $(\frac{1}{n})$ is the probability of selecting a given observation from the sample (if there are 11 observations, the probability of selecting any one of them would be 1/11).

The math that is needed to get to the next equation is not covered in this class, but I will try to give you an idea of how it is obtained. You can pass this course without understanding the details in this paragraph, although it would be good to recognize the final equation. If we call the $\bar{x}$ from the previous equation the *expectation of x*, written $\mathrm{E}(X)$, and if we multiply the $\frac{n}{n-1}$ and the $\frac{1}{n}$ and cancel the red n values, we get $\frac{1}{n-1}$, which we can call the adjusted probability of a given observation of x, written $\mathrm{P}(x)$, and if we use some techniques of algebra that apply to the summation symbol $(\sum)$, then we can rewrite the equation again like this (equivalent to the third equation on page 58):

$$s^2 = \sum_{i=1}^{n}\left[\left(x_i - \mathrm{E}(X)\right)^2 \times \mathrm{P}(x)\right] \tag{4.7}$$

Remember that when the notes in this chapter refer to the probability of x, it is a reference to the probability of randomly selecting a given observation from the sample (or population). It could be the case that in a sample of 100 people, 10 people are the same height, but in this chapter we are referring to the probability of selecting data for a given person when we randomly select from the sample. The probability of selecting an observation when the sample size is 100 is always 1/100 (for what we cover in this class).

Practice calculating the values in Table 4.3 on page 59 by using information from Table 4.2 on page 58. Note that $\mathrm{E}(X) = .85$ as shown on the bottom-right corner of Table 4.2.

4.6 Expectation and Standard Deviation

Section 4.4.2 of the textbook is called *Expectation and Standard Deviation.* The notes above address expectation and standard deviation separately, but there is a subtle point in the next chapter that relates the two concepts and also integrates the idea of a random variable: "In a similar way, the variance of a random variable may be defined via the probability of the values that make the sample space" (Yakir 2011, p. 68). In other words, if we have enough data to calculate the expectation of a random variable, we have enough information to calculate the standard deviation (or variance). Remember that the variance is just the square of the standard deviation.

In this course and the next statistics course, we will learn that we can use just the expectation and either the standard deviation or distribution of a random variable and make some precise statement about the probability of obtaining particular results. This is the key that enables researchers to make statements about their confidence that observed results represent the universe the way that it really is.

4.7 Selecting Subsets of Data in R

You might have some questions in which you have to select subsets of data from an R data frame. For example, you might want to get the mean height of people who live in a particular city—you would select people from the CITY column of your data and then calculate the `mean(height)` for those people.

One way to select subsets is to use the `subset()` command. In the next example, I read the `pop3` data file, then I create a smaller data frame called *popa* that contains only those observations where the `type` column equals *a*. Notice the `colnames` command tells me that there are two columns, one with the name *type* and one with the name *time.* I use the `subset` command and specify `pop3$type == 'a'` to indicate that I want the records where `type` is equal to *a* (Note that there are two equal signs there). My `popa` data frame has 49,949 rows.

```
> pop3 <- read.csv('pop3.csv')
> colnames(pop3)
[1] "type" "time"
>
> popa <- subset(pop3, pop3$type == 'a')
> nrow(popa)
[1] 49949
```

Here is another example using the `subset()` command to select only male participants from the `ex1` file:

```
> ex.1 <- read.csv('ex1.csv')
> colnames(ex.1)
[1] "id"     "sex"    "height"
> ex.1[1:5, ]
       id    sex height
1 5696379 FEMALE    182
2 3019088   MALE    168
3 2038883   MALE    172
4 1920587 FEMALE    154
5 6006813   MALE    174
> M <- subset(ex.1, ex.1$sex == 'MALE')
> summary(M)
       id               sex         height
 Min.   :1620128   FEMALE: 0   Min.   :117.0
 1st Qu.:3565594   MALE  :46   1st Qu.:164.2
 Median :5438082               Median :172.0
 Mean   :5609951               Mean   :173.8
 3rd Qu.:7608068               3rd Qu.:187.0
 Max.   :9878130               Max.   :208.0
>
```

Another way to do this involves *boolean operations* on vectors. That means you tell the computer to test the values and identify when the test is TRUE versus when it is FALSE.

I let x be the set of integers from 1 to 10. I then ran a test that evaluates to a Boolean (meaning that when I say `x < 5`, the computer is going to give me answers in terms of TRUE and FALSE):

```
> x <- c(1, 2, 3, 4,5, 6, 7, 8, 9, 10)
> x < 5
 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> x > 3
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

If you look at the values of TRUE and FALSE, you can see that the TRUE values for the first test identify the values of x that are less than 5. The second test produces TRUE values when x is greater than 3.

I can use these TRUE and FALSE values to select subsets of numbers from x by using **square bracket notation.** I might read the first command below as "give me all the values of x for which x is less than 5." I might read the second command below as "give me all the values of x for which x is greater than 3."

```
> x[ x < 5 ]
[1] 1 2 3 4
>
> x[ x > 3]
[1]  4  5  6  7  8  9 10
```

You can put those calculations inside other commands, like this:

```
> mean(x[ x > 3])
[1] 7
> mean(x[ x < 5])
[1] 2.5
>
> small.values <- x[ x < 5]
> mean(small.values)
[1] 2.5
```

If you have a data file with thousands of observations, you don't want to count things by hand, so sometimes these *selection techniques* can save you lots of work. In the next example, I read the `pop3` file and show the first five rows with the command `> x[1:5]`. The `1:5` is a way to specify the first 5 items, then I put a comma after that because the data frame has rows and columns and the square brackets would take two values: the first for the desired rows and the second for the desired column. If you leave the column part blank, R gives you all the columns:

```
> pop3 <- read.csv('pop3.csv')
> colnames(pop3)
[1] "type" "time"
> pop3[1:5, ]
  type       time
1    b  0.1022744
2    b 22.1781342
3    c  9.5181055
4    a  4.0313741
5    b 24.0751296
```

I now use a test that looks fancier, but logically it is the same as the test for `x > 5`. This time I point to the R object called `pop3$type`. I can see that there are 49949 observations of type *a*, 33410 observations of type *b*, and so on:

```
> length(pop3$type[ pop3$type == 'a'])
[1] 49949
> length(pop3$type[ pop3$type == 'b'])
[1] 33410
> length(pop3$type[ pop3$type == 'c'])
[1] 16641
> length(pop3$type[ pop3$type == 'd'])
[1] 0
```

I will now prepare to do something a bit fancier. I will get the mean time for observations where the type is *a*. First, I show summary information about all the time data so that you can get an idea of how the numbers look.

```
> summary(pop3$time)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
  0.00003   1.85100   4.47300   6.84400   9.21800 135.00000 
```

Now I show the time values for *a.* Inside the square brackets, I said `pop3$type == 'a'` (notice the double equal signs) to create TRUE values for the observations where the type is *a.* I used those TRUE/FASE values in square brackets to select values of `pop3$time`, then I put that result inside the `summary()` command.

```
> summary(pop3$time[ pop3$type == 'a'])
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
 0.00011  1.46600  3.47300  5.00300  6.94900 60.66000 
```

Here is the equivalent information for *b, c,* and *d* (there are no observations of type *d*):

```
> summary(pop3$time[ pop3$type == 'b'])
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
 0.00003  2.32000  5.47600  7.99500 11.00000 92.46000 
> summary(pop3$time[ pop3$type == 'c'])
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
  0.00198   2.85900   6.94600  10.06000  14.02000 135.00000 
> summary(pop3$time[ pop3$type == 'd'])
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
```

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 5

Chapter 05 Notes

5.1 Word to the Wise

Be sure to review the solved problems at the end of the chapter.

5.2 Discrete Random Variables

A *discrete* variable is a vector of numbers (an ordered list of numbers) in which all of the numbers are integers. There are some naturally occurring situations that can be described using discrete random variables. One example is the outcome of an (ideal) coin toss (or coin flip). The textbook assumes that people are familiar with coins that have a picture of head on one side and the other side of the coin is called *tails.* For the sake of simplicity, we ignore the possibility that a person can flip a coin that balances on the edge, or we can define the rules of the game such that anything other than a *heads* or *tails* in a coin toss has to be repeated. We can arbitrarily count *heads* as a 1 and *tails* as a 0 (or vice-versa, as long as we score everything consistently). Another use for discrete data is to analyze *count data.* As long as an event can be clearly defined such that it either occurs or doesn't occur, then we can count the events using integers. An example would be the number of people who attend a football game. We ignore or disallow the possibility that half a person will be in the stadium, so we count the number of people using integers.

5.3 Binomial Random Variables

If an event has only two outcomes (like *heads* or *tails*) then we can say that it is a *binomial* event. Sometimes an event that is scored as either success or failure (or *heads* or *tails*) is said to be a *dichotomous event* or *dichotomously-scored*, but our textbook does not use that term. Binomial random variables are theoretical variables that are defined by the probability of "success" for an event. The traditional example is of tossing a coin. If the coin is not biased, then there is a 50% chance (probability of .50) that each toss will result in *heads*. Other binomial variables can have a different probability of success, like the probability that I can make a 3-point shot in basketball.

If I plan on flipping a coin 11 times, I can identify the probability that I will obtain EXACTLY five heads:

```
> dbinom(5, 11, .5)
[1] 0.2255859
```

If I forget what the numbers passed to the `dbinom()` function mean, I can issue this R command and read the help page: `?dbinom`. The 11 means that I plan on flipping the coin 11 times, and the .5 means that the probability of "success" (*heads*) is .5.

```
> ?dbinom
> dbinom(x=5, size=11, prob=.5)
[1] 0.2255859
```

It is possible that I could flip a coin 11 times and get no heads, and it is possible that I could get 11 heads. I can find the probability of getting exactly 0 heads, 1 head, 2 heads, etc., using R commands like this:

```
> dbinom(x=0, size=11, prob=.5)
[1] 0.0004882812
> dbinom(x=1, size=11, prob=.5)
[1] 0.005371094
> dbinom(x=2, size=11, prob=.5)
```

```
[1] 0.02685547
> dbinom(x=3, size=11, prob=.5)
[1] 0.08056641
> dbinom(x=4, size=11, prob=.5)
[1] 0.1611328
> dbinom(x=5, size=11, prob=.5)
[1] 0.2255859
```

I could have saved some typing by passing the values 0 through 5 to the `dbinom` function all at once, like this:

```
> dbinom(x=c(0,1,2,3,4,5), size=11, prob=.5)
[1] 0.0004882812 0.0053710938 0.0268554688 0.0805664062 0.1611328125
[6] 0.2255859375
```

There is an even shorter version using this notation in R: `0:5`, which tells R to generate a sequence of integers that starts with 0 and continues through 5:

```
> 0:5
[1] 0 1 2 3 4 5
> dbinom(x=0:5, size=11, prob=.5)
[1] 0.0004882812 0.0053710938 0.0268554688 0.0805664062 0.1611328125
[6] 0.2255859375
```

If I wanted to know the probability of getting something between 0 and 5 heads (inclusive) I could manually add the six results above, or I could use the `sum` command like this (similar to what is done in Yakir, 2011, p. 68):

```
> sum(dbinom(x=c(0,1,2,3,4,5), size=11, prob=.5))
[1] 0.5
```

The 0.5 above means that there is a 50% chance that I will get between 0 and 5 heads (inclusive) if I toss a fair coin 11 times. Note that real coins often have a slight bias so that the real probability of heads might be a tiny bit different from .5.

There is yet another way to determine the probability of getting between 0 ad 5 heads. The `pbinom` function gives the cumulative probability of getting up to the specified number of successes (see also, Yakir, 2011, p. 68):

```
pbinom(q=5, size=11, prob=.5)
```

The expectation of a binomial distribution is derived the same way as expectation was described in the previous chapter, but it can be simplified to this: $n \times p$ (Yakir, 2011, p. 69). Note that n is the number of trials (coin flips), and p is the probability of success (I arbitrarily defined *heads* to represent success).

$$\begin{aligned} E(X) &= \sum \big(x \times \mathrm{P}(X = x)\big) \\ &= np \end{aligned} \tag{5.1}$$

The variance simplifies to this:

$$Var(X) = np(1 - p) \tag{5.2}$$

Assume that X is distributed according to a binomial distribution: Binomial(100, .6). Answer the next few questions.

The " Binomial(100, .6)" part above means that the random variable can be described with a binomial distribution in which 100 experiments are conducted, and the probability of success for each experiment is .6.

Find the probability $\mathrm{P}(50 < X \leq 58)$

We want to find the red region in Figure 5.1 on page 49. We have a tool that would help us to answer this question. We know how to find the cumulative probability that a randomly

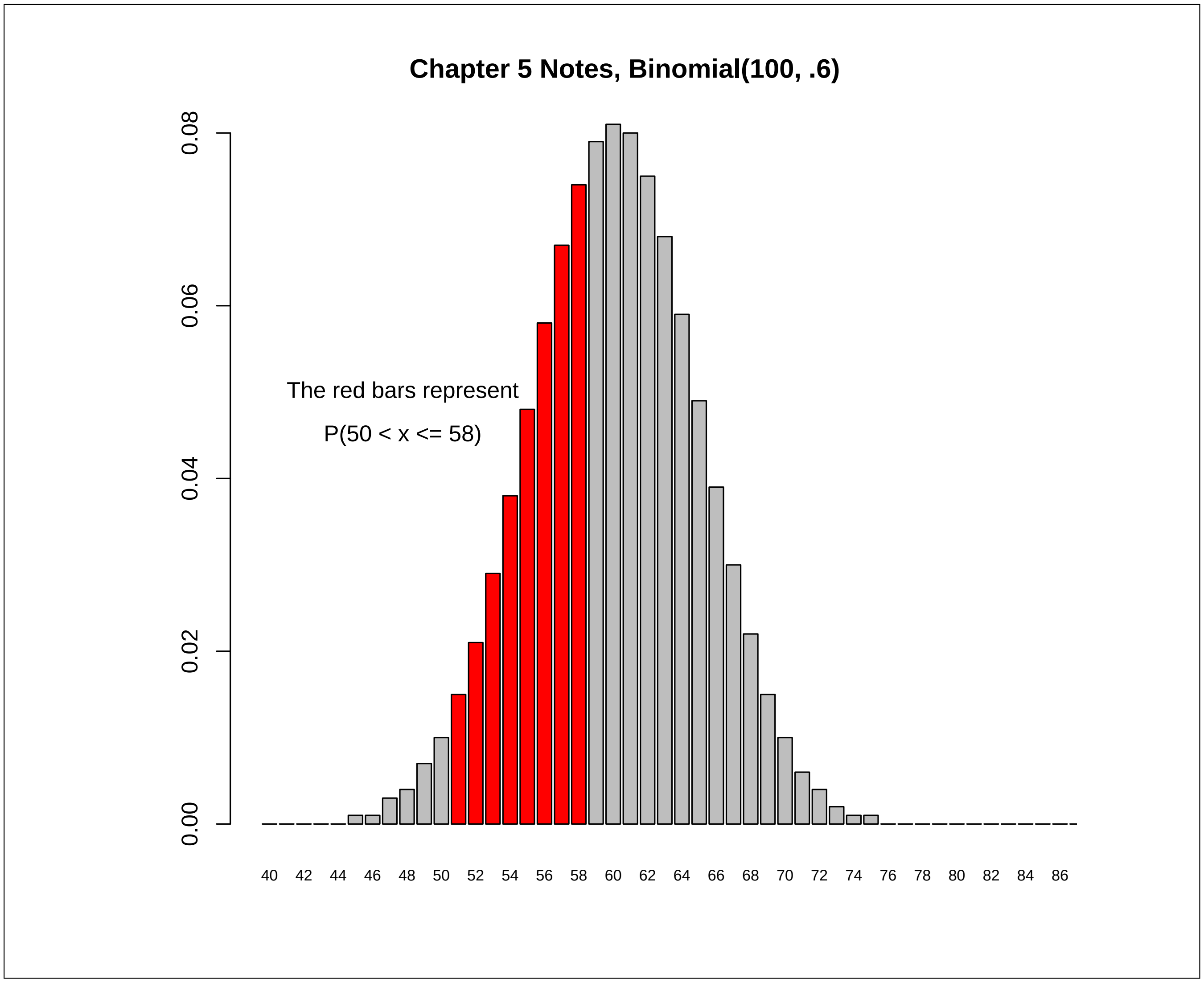


Figure 5.1: Binomial Example 1

selected value of X will be less than or equal to a specific value (using the `pbinom()` function). We can run it twice with carefully chosen input values.

First think about what you want to find. You could run the `pbinom(58, 100, .6)` command once to find the probability that a randomly selected value called x is less than or equal to 58—that is the right-hand side of our problem above. We want to KEEP the probability that a randomly selected value x is greater than 50, meaning that we want to remove tha part that were x is less than or equal to 50. The `pbinom(50, 100, .6)` command will tell us exactly what we want to remove: the probability that x is less than or equal to 50.

(note: some books use a capital letter X to refer to the distribution and a lower letter x to refer to a specific observation that is drawn from that random variable or drawn from a population)

Be very careful to distinguish among "less than" "less than or equal to" "greater than" and "greater than or equal to." when working with a **discrete** distribution. *When you work with continuous distributions, there is no distinction between these equalites because any differences are infinitely small.* The `pbinom` functions finds the probability that x is "less than or equal to" a given value and it only works on integer values (because the binomial distribution is discrete).

The following shows the final answer by finding the probability that x is less than or equal to 58, and then removing the stuff that we do not want (the probability that x is less than or equal to 50):

```
> pbinom(58, size=100, p=.6) - pbinom(50, size=100, p=.6)
[1] 0.3503681
>
> # Double check by using dbinom for the probabilities of
> # getting exaclty 51 success, 52.... 58
> dbinom(51, size=100, p=.6) +
+ dbinom(52, size=100, p=.6) +
+ dbinom(53, size=100, p=.6) +
+ dbinom(54, size=100, p=.6) +
+ dbinom(55, size=100, p=.6) +
+ dbinom(56, size=100, p=.6) +
+ dbinom(57, size=100, p=.6) +
+ dbinom(58, size=100, p=.6)
[1] 0.3503681
```

Find the probability P(60 <= X < 68)

We want to find the green region in Figure 5.1 on page 53.

We are going to use the `pbinom` function to answer this question, its default behavior is to show us the probability that the selected value is less than or equal to a given number. We want to find (keep the numbers) if x is less than 68, which is the same as x being less than or equal to 67 because the binomial distribution is discrete, but we need to remove the portion that is less than 60, which is the same as the removing the portion that is less than or equal to 59 because the binomial distribution is a discrete distribution.

answer:

```
> pbinom(67, size=100, prob=.6) - pbinom(59, size=100, prob=.6)
[1] 0.4817906
>
> # double check
> dbinom(60, size=100, p=.6) +
+ dbinom(61, size=100, p=.6) +
+ dbinom(62, size=100, p=.6) +
+ dbinom(63, size=100, p=.6) +
+ dbinom(64, size=100, p=.6) +
+ dbinom(65, size=100, p=.6) +
+ dbinom(66, size=100, p=.6) +
+ dbinom(67, size=100, p=.6)
[1] 0.4817906
```

5.4 Poisson Distribution

Similar to the binomial distribution is the Poisson distribution. The first part of *Poisson* is pronounced "pwa-" and the last part is pronounced as in French. Both the binomial distribution

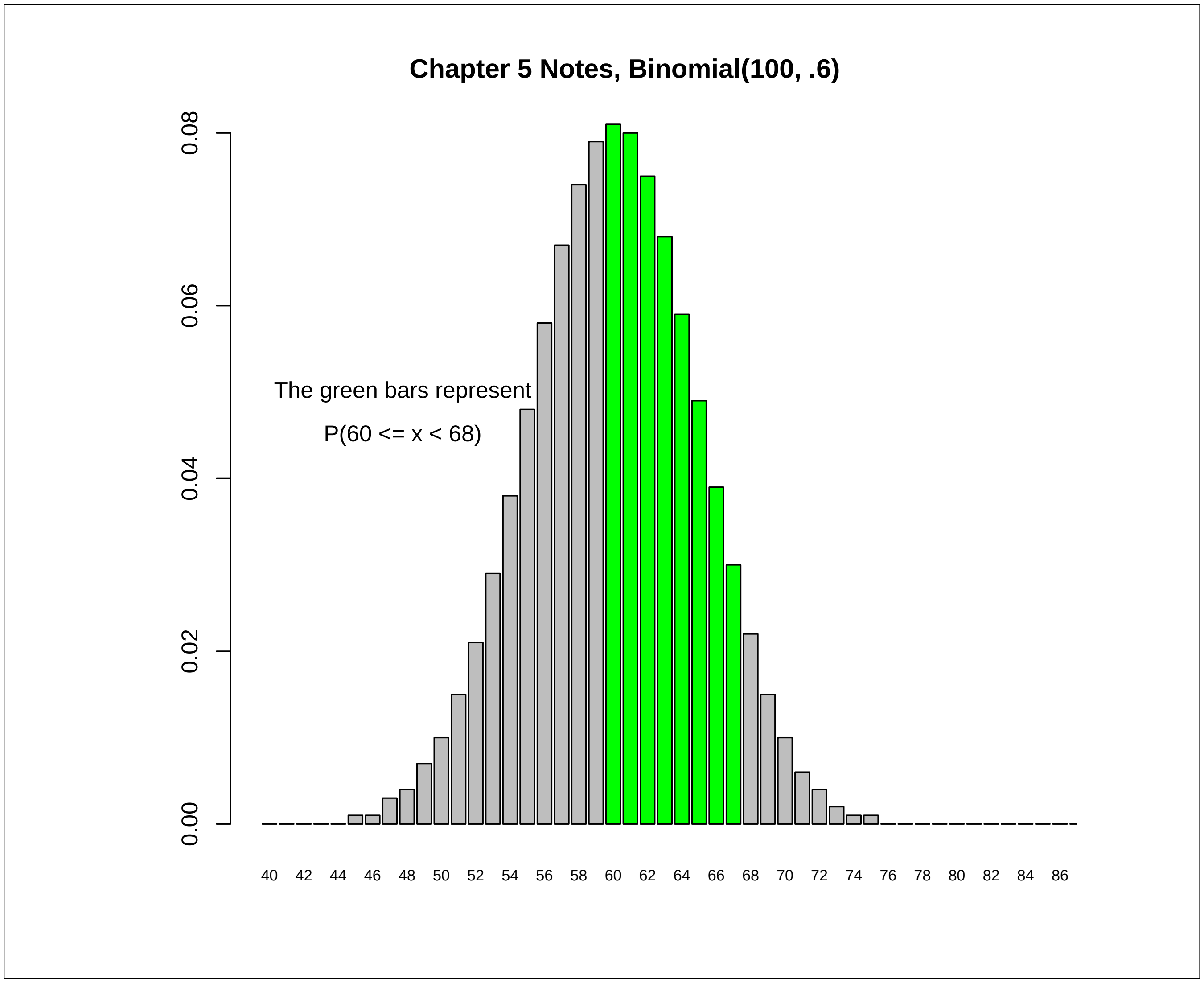


Figure 5.2: Binomial Example 2, n=10,000

and the Poisson distribution are derived from a sample space of non-negative integers. The Poisson distribution is discrete.

> The Poisson distribution is used as an approximation of the total number of occurrences of rare events. Consider, for example, the Binomial setting that involves n trials with p as the probability of success of each trial. Then, if p is small but n is large then the number of successes X has, approximately, the Poisson distribution. Yakir, 2011, p. 71.

The expectation for a Poisson distribution is represented using the Greek letter lambda (λ) as compared to the binomial distribution where the expectation was $n \times p$. Lambda would be the expected number of events within the specified period of time. If you usually get around 2 phone calls per minute (with the actual number of calls varying according to the Poisson distribution), then lambda would be 2. When you are talking about a Poisson distribution, you might ask yourself a question like this: "If I randomly select an observation from a population (of integers) that is distributed according to a Poisson distribution with a lambda of 2, what would be the probability of selecting a 5?" The answer would be:

```
> dpois(x=5, lambda=2)
[1] 0.03608941
```

Here are a few comparisons so that you can judge how close the Poisson distribution is to the binomial distribution as n grows and p becomes smaller. Remember that the expectation for a binomial distribution is $n \times p$, so I used that to represent λ when I calculated the Poisson probability. Note that the n values increase in each block of code:

```
> n=10
> p=.5
> x=4
> dbinom(x, n, p)
[1] 0.2050781
> dpois(x, lambda=n*p)
[1] 0.1754674
>
```

```
>
> n=100
> p=.5
> x=40
> dbinom(x, n, p)
[1] 0.01084387
> dpois(x, lambda=n*p)
[1] 0.02149963
>
>
> n=1000
> p=.05
> x=30
> dbinom(x, n, p)
[1] 0.0005578706
> dpois(x, lambda=n*p)
[1] 0.0006771985
>
>
> n = 10000
> p=.005
> x=30
> dbinom(x, n, p)
[1] 0.0006647291
> dpois(x, n*p)
[1] 0.0006771985
```

Yakir (2011, p. 73) shows a chart of Poisson distributions as lambda changes.

The equation that describes the probability of a getting a given value of x in a Poisson distribution is listed in a footnote (Yakir, 2011, p. 73). That equation can be expressed in R. The next example shows a long calculation for the probability of a observing $x = 4$ when $\lambda = 2$ and also shows the result from the `dpois()` function in R:

```
> lambda=2
> x=4
> exp(-1*lambda)*lambda^x / factorial(x)
```

```
[1] 0.09022352
> dpois(x=4, lambda=2)
[1] 0.09022352
```

5.5 The Great Calculus Scare

Yakir (2011, p. 75) shows some calculus for defining the variance of a population using expectation and expected probability. You do not need to know calculus for this class, but it might be nice to know that the summation operations that we used earlier (for discrete variables) are quite similar to the calculus shown on page 75, which applies to continuous variables. The big integration symbol ($\int$) was originally a modified letter S for *sum*.

5.6 The Uniform Distribution

This distribution implies that every value in the sample space is equally likely. The expectation of a truly uniform distribution (i.e., the POPULATION) is the sum of the smallest value and the largest value, all divided by 2.

$$\mathrm{E}(X) = \frac{a+b}{2} \tag{5.3}$$

so if you have a uniform distribution where the lowest possible value is 12 and the highest possible value is 16, the expectation is (12 + 16) / 2 = 14. Yakir, 2011, p. 78).

The variance for the POPULATION of a truly uniform distribution is easy to calculate:

$$\mathrm{Var}(X) = \frac{(b-a)^2}{12} \tag{5.4}$$

If you take a sample from a uniform distribution, the actual probabilities will fluctuate a bit, but will be close to the theoretical values. You can generate data for a uniform distribution using the `runif` command:

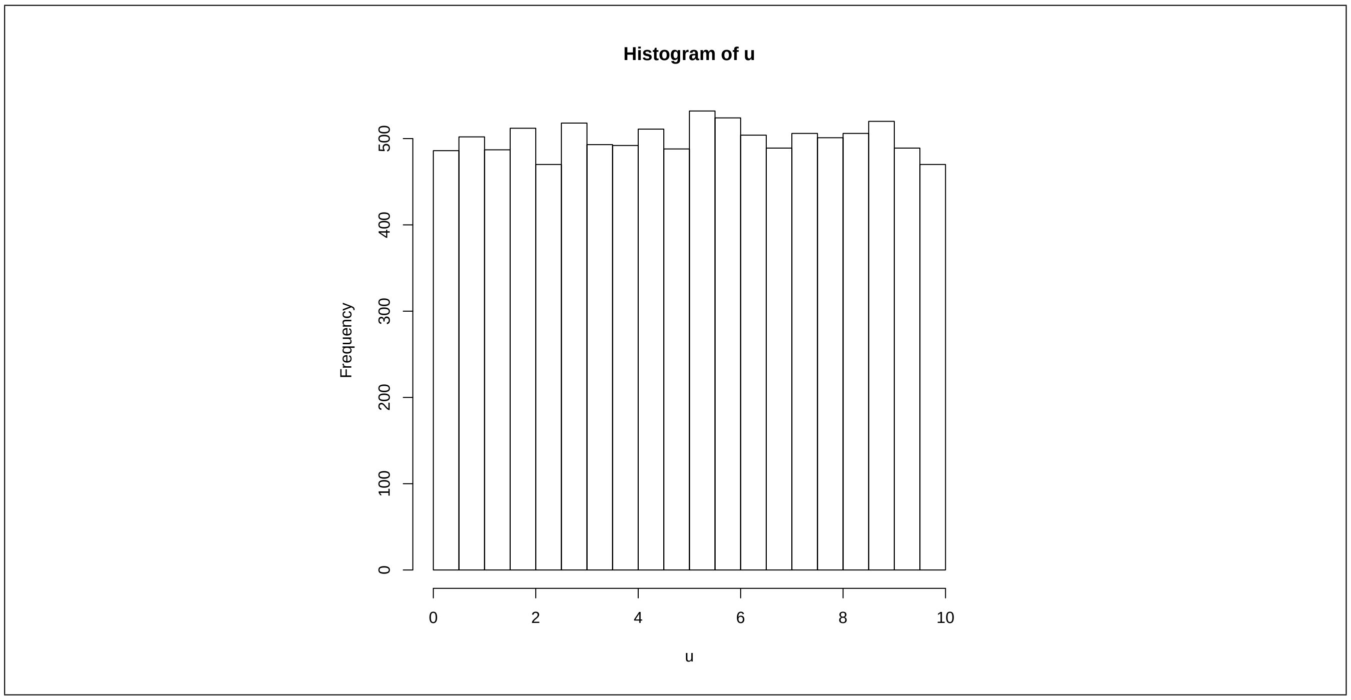


Figure 5.3: Histogram of a sample taken from a uniform distribution, n=10,000

```
> set.seed(12345)
> u <- runif(10000, 0, 10)
> var(u)
[1] 8.240007
> (10 - 0)^2/12
[1] 8.333333
> hist(u)
```

See Figure 5.3 for the histogram.

5.7 Exponential Distribution

This is similar to the Poisson distribution except the sample space is non-negative real numbers (not just integers) and the distribution is sometimes used to model the time between occurrences of an event that is known to have an average *rate* (probability of occurrence). Remember that the Poisson distribution models the total number of occurrences of an event that has an average rate, whereas the exponential distribution models the time between events. The *rate* (λ; see Yakir, 2011, p. 79) for an exponential distribution is only an average of some fluctuating probability. This is different than the ticking of a watch, which might tick once per second but it does so with great reliability, so an exponential distribution probably would not be good for modeling the time between ticks of a watch. Maybe the time between leaves falling from a tree in autumn would be modelled by an exponential distribution—over some period of time, perhaps one day, we might know how many leaves will fall, but the interval between falling leaves could vary.

Example: If 20 leaves fall from a tree per hour, then lambda (the rate) could be expressed in terms of leaves per hour, leaves per minute, or other such units:

$$\begin{aligned}\lambda &= \frac{20 \text{ events}}{\text{hour}} \\ &= \frac{20 \text{ events}}{\text{hour}} \times \frac{1 \text{ hour}}{60 \text{ minutes}} \\ &= \frac{20 \text{ events}}{60 \text{ minutes}} \\ &= \frac{1 \text{ event}}{3 \text{ minutes}}\end{aligned} \tag{5.5}$$

When you are talking about an exponential distribution, you might ask yourself a question like this: "If I randomly select a time period to collect data, what is the probability that the time between events will be 3 minutes or less if the lambda (rate) is 20 per hour (equals 1/3 events per minute)?" The answer would be:

```
> pexp(3, 1/3) - pexp(0, 1/3)
[1] 0.6321206
```

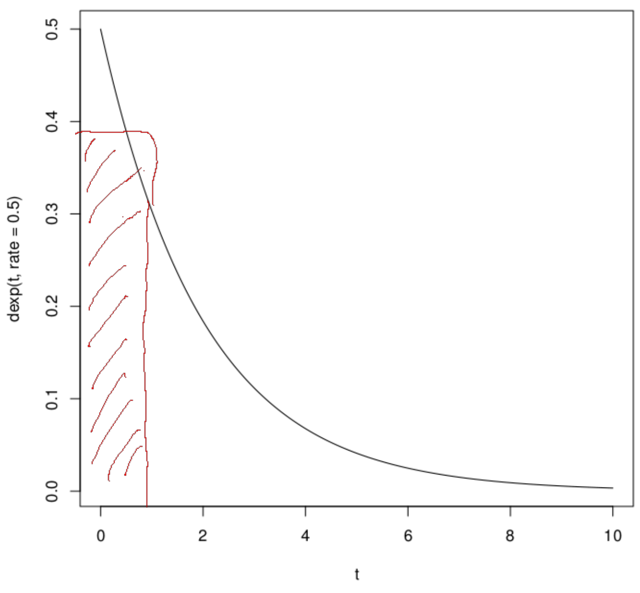
Note that I used the `pexp()` command and subtracted one from the other. I did this because when you deal with continuous distributions, you generally need to look between two points and estimate the probability that something is between the two points (or outside that range). If I asked what was that probability of observing a time between events of exactly 3 minutes, the answer would approach zero because 3.000000000001 minutes is different from exactly 3 minutes, and I could add an infinite number of decimal places to describe outcomes that are not exacly zero. It makes more sense to ask about the probability that the time between events is between 2.5 minutes and 3.5 minutes:

```
> pexp(3.5, 1/3) - pexp(2.5, 1/3)
[1] 0.123195
```

Why would an exponential distribution not describe the time between ticks of a watch when we know that the average time between ticks is 1 second (the λ rate is 1 tick per second)? The answer is that the exponential distribution describes an event process only if the time between events can be described by the *probability density function* described in a footnote (Yakir, 2011, p. 79). An example of the time curve is shown in Figure 5.4. The time between ticks of a watch does not follow anything like the curve shown in the figure. The figure (based on $\lambda = .5$) shows that there is about a .08 probability that the lag between events will be 4 seconds (by looking at 4 on the x-axis and looking up to see where the curve is at that point). The time between ticks of a (idealized) mechanical watch would be super close to 1 second and the probability that the lag between ticks will be 4 seconds is super close to zero.

Figure 5.4 was created using this R code (and then annotated):

```
> t <- c(0:10000)/1000
> plot(t, dexp(t, rate=.5), type='l')
> pdf()
> plot(t, dexp(t, rate=.5), type='l')
```



This is a plot of an exponential distribution with a
rate of .5 per second, showing time intervals on the x-axis
from zero to 10 seconds.

This plot can be used to estimate the probability that the
interval between events will be in a specific range. I can
make a rough estimate by shading the region of interest.

If I want to estimate the probability that the time
between events will be 1 second or less, I could shade
that region and estimate the area.

My eyeball estimate is .38. The computer says;

```
> pexp(1, rate=.5)
[1] 0.3934693
```

I drew the red rectangle to estimate the area under the
curve between t=0 and t=1.

Figure 5.4: Plot of Probability Density for Exponential Distribution with Rate=.5

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 6

Chapter 06 Notes

6.1 The Normal Distribution

6.1.1 Introduction

This chapter is similar to the previous chapter, but this time we focus on one distribution; the normal distribution (or the normal random variable). If we take a moderately large sample of data from a normal distribution (say 300+ observations), the plot should look approximately symmetric. This distribution is used so widely, that some statistics courses focus almost exclusively on it. One problem with taking the narrow approach can be described using an old aphorism: when you have a hammer, everything looks like a nail. In other words, people who study only the normal distribution tend to assume that everything can be described as a normal distributions, and sometimes that leads to errors.

If you are studying a population, then the expectation (mean) of a normal distribution is usually written as μ and its standard deviation is written σ (Yakir, 2011. p. 88). If you are studying a random variable, you would typically refer to it as a population and use μ and σ to refer to its properties. If you take a sample from a normal distribution or a sample from a normally distributed random variable, then you would usually call the mean $\bar{x}$ and the standard deviation s (Yakir, 2011. p. 45).

6.1.2 Find the probability given x-values

I'll describe a problem from this chapter (Yakir, 2011, pp. 88-89), but alter the numbers a bit and add some details to make it more concrete. Let's say that you manufacture shoes, and you want to know how accurately the shoes are being manufactured. You have studied the manufacturing process and you have found that shoes are usually a tiny bit longer or shorter than the ideal size—this is the *size-error.* The data for size-error can be described using a normal distribution with mean μ of 0 and a population standard deviation (σ) of 1 mm. If the error value is negative, that means that the shoe is too short. If the error is positive, then the shoe is too long. Small deviations are acceptable.

After you collect data from many shoes, you might be able to estimate the mean and standard deviation of all the "errors" in the manufacturing process (mean of zero and standard deviation of 1 mm). You could then estimate the probability that a shoe is between 0 and 3 mm too big by running this:

```
> pnorm(3, mean=0, sd=1) - pnorm(0, mean=0, sd=1)
[1] 0.4986501
```

That probability is very close to .50, so approximately half the shoes have a size-error of 0 – 3 mm (and approximately half would have the corresponding negative size-error).

Now you want to estimate the percentage of shoes that would be rejected if your inspectors rejected a shoe that had an error of greater than 2 mm (i.e., 2 or more mm too short or 2 or more mm too long). There are two plots in Figure 6.1 on page 64. The first has the region between -2 and 2 shaded, and the second has the regions smaller than -2 and larger than 2 shaded. We want to find the probability of being in the shaded tails in the second plot.

First let's look at what the `pnorm()` function in R is doing.

```
> pnorm(2, mean=0, sd=1)
[1] 0.9772499
```

That command said that the cumulative probability that an observation is less than 2 mm is .9772499, as shown in Figure 6.2 on page 65.

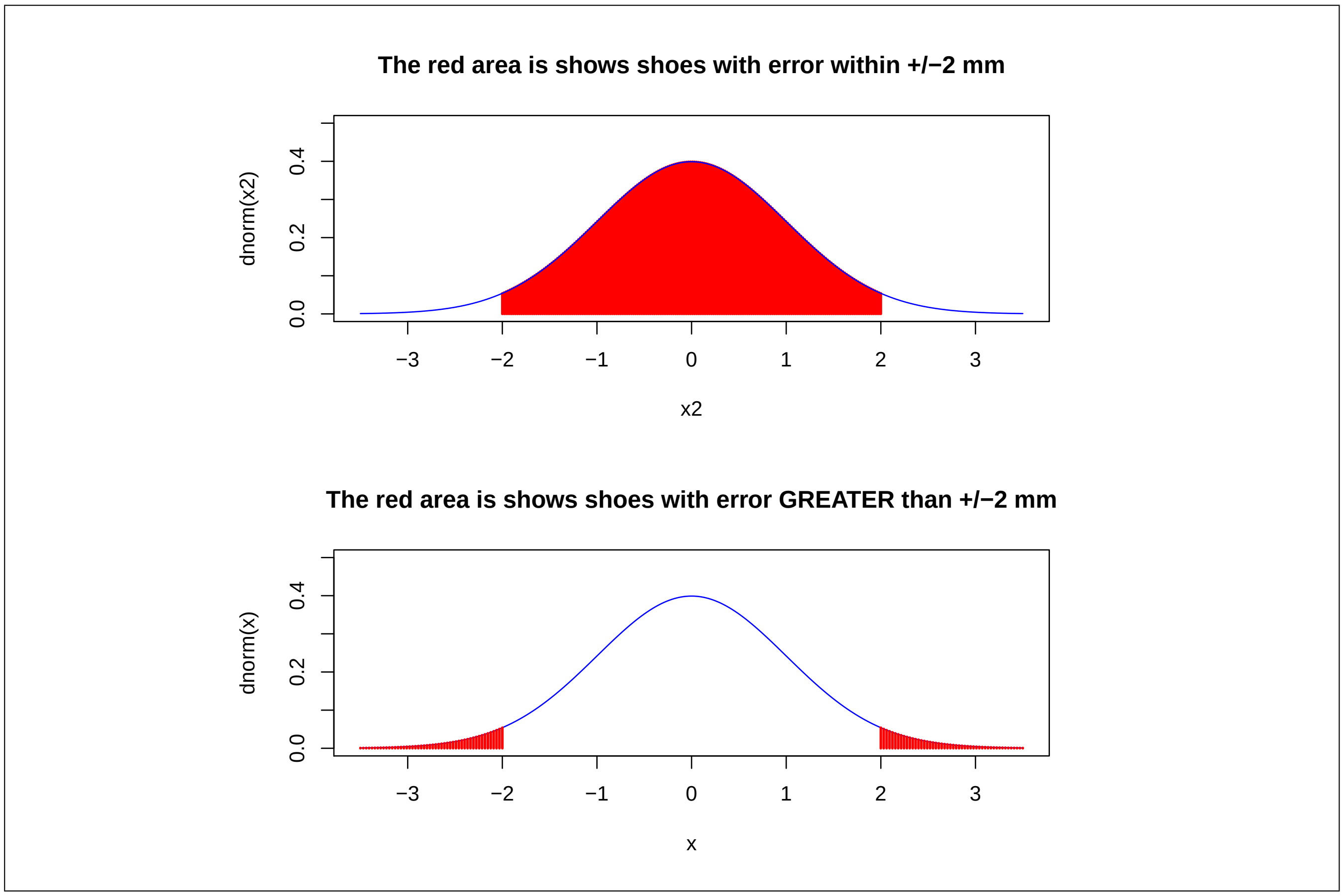


Figure 6.1: Normal Distribution for the Shoe Example.

I want to know the probability of being higher than 2. I know that probabilities add to 1, so I can calculate 1 - .9772499, or do it like this:

```
> 1 - pnorm(2, mean=0, sd=1)
[1] 0.02275013
```

There is a .02275013 probability that a randomly selected shoe will be more than 2 mm too big.

The probability of being in the lower tail in the lower figure on page 64 is .022750 (this is basically the same as the other value, but with a small difference in rounding):

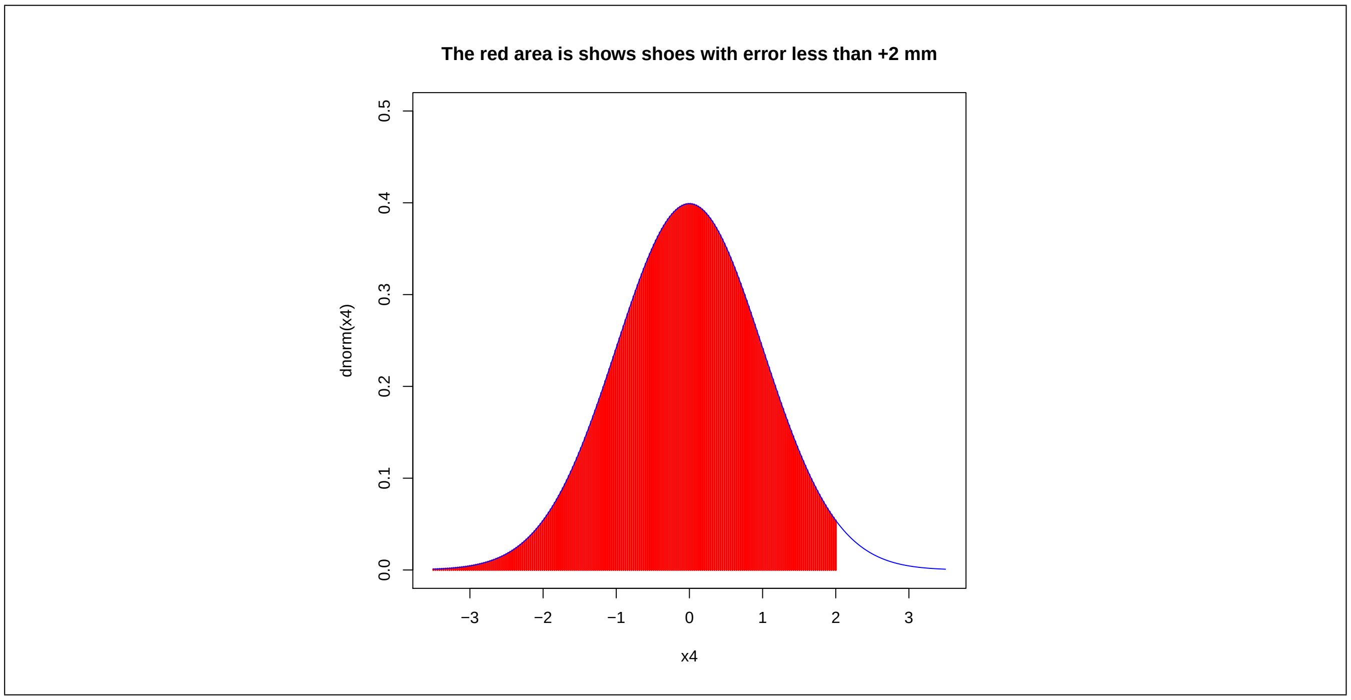


Figure 6.2: Normal Distribution for pnorm(2, mean=0, sd=1)

```
> pnorm(-2, mean=0, sd=1)
[1] 0.022750
```

I could add the two probabilities together to get the answer,

```
> 0.022750 + 0.02275013
[1] 0.04550013
```

6.1.3 Find the x-values given probability

What range of shoe size-errors would include 95% of all shoes? If the question does not say otherwise, it is assumed that the region is symmetric about the expectation (mean). In other words, the plots above (Figure 6.1 on page 64) show a big red area in the center. We are looking to make the region a slightly different size so that it includes 95% of the shoes. The center of that region is the expectation (mean). The task is to find where the x-values associated with the lower and upper edge of the big red area would be so that the red region includes 95% of the shoes.

First, be sure to understand the probabilities. If 95% of the shoes fall within the central region, as in the first plot in Figure 6.1, then the other 5% is in the tails (corresponding to the second plot). Statisticians sometimes call that extra 5% *alpha* (the Greek letter α). We split that 5% into two tails of 2.5% each, as shown in the second plot in Figure 6.1. If 2.5% is in the right-hand tip of the curve and 2.5% is in the left hand curve, then the middle range would be between 2.5% and 97.5%. To double check: .975 - .025 = .95, which is what we wanted to find.

Sometimes it is neater to write the program as shown below... then you can change the alpha value to answer different questions and copy and paste the commands to find the answer:

```
> alpha <- .05
> tails <- 2
> qnorm(1 - alpha/tails, mean=0, sd=1)
[1] 1.959964
```

If we put .975 (which is 1 - alpha/tails) into the `qnorm()` function, the answer tells us which value of x is bigger than 97.5% of the observations in the distribution. The value 1.959964 is bigger than 97.5% of the observations. If our units are millimeters, then 1.95966 means 1.95966 mm. If 97.5% of the observations are less than 1.959964, then 2.5% are bigger.

Why did I use $1 - \text{alpha/tails}$ above? Because alpha / tails by itself is a tiny number (.025), which corresponds to the area under the far left corner of the plots above. The total area under the probability curve is 1, and the errors that are far above the mean will be be bigger than almost 100% of the other errors—or more precisely—bigger than 97.5% of the errors (1 - .025). So the "one minus" part of the `qnorm()` calculation above is just a way to identify the .975 cutoff point in the far right part of the curve in Figure 6.1 on page 64.

Now find the lower bound (the size-error value associated with the left-hand side of the big red curve in Figure 6.1).

```
> alpha <- .05
> tails <- 2
> qnorm(alpha/tails, mean=0, sd=1)
```

```
[1] -1.959964
```

This time we put just "`alpha / tails`" in the `qnorm` function, and that is the same as putting .025 in the `qnorm` function. The shoes that have a *size-error* between -1.959964 and +1.959964 mm from the ideal size comprise 95% of all shoes.

If the question changes and you want to find the amount of error that would identify the middle 90% of observations, just change `alpha <- .10` and rerun the commands. If you have a question with a different mean or standard deviation, then change the mean and standard deviation in the `qnorm()` function and run it. Sometimes it helps to draw the normal curve and label the mean and standard deviation to see if you answer looks reasonable.

The following tells us that the value 129.3995 is bigger than 97.5% of the observations in a population that has a mean of 100 and a standard deviation of 15:

```
> qnorm(.975, mean=100, sd=15)
[1] 129.3995
```

The following tells us that the value 87.37568 is bigger than 20% of the observations in a population that has a mean of 100 and a standard deviation of 15 (it also means that 87.37568 is smaller than 80% of the observations):

```
> qnorm(.20, mean=100, sd=15)
[1] 87.37568
```

Try experimenting with the calculations to see if you can interpret the results.

6.1.4 Continuity Correction

Some distributions, such as the binomial and Poisson distributions, have a sample space comprised of integers. You can not get half a tail when you flip a coin (or, more precisely, the experiment is defined to reject any odd events like the coin balancing on the edge). The normal distribution is continuous, so there are artifacts when converting between the two.

Here is an example of a question about the binomial distribution:

> Let the distribution of X be Binomial(37, 0.4). The next 4 questions correspond to this information. The answer may be rounded up to 3 decimal places of the actual value:

Note that the 0.4 above means that the probability of success for a single trial is .4, and the 37 means that we will conduct 37 experiments (37 games of chance in which your probability of winning a single game is .40, which is 40%).

Find the binomial probability that P($8 < X \leq 16$) when p = .4.

This one is a regular binomial probability question. In this example, the *less than or equal* and *less than* comparisons are set to make it easy to deal with the continuity problem. The `pbinom()` function returns the probability of x being *less than or equal to* the first value passed to the function (e.g. the number 8 in `pbinom(8, size=37, p=.4)`). The first part of the calculation below finds the probability of x being less than or equal to 16, which is part of what the original question asked. For the second part of the calculation, we remove "the probability of x being less than or equal to 8." If you think about that, this means that we leave the part where $x > 8$, which is what was requested in the original question:

You can answer this in R with:

```
pbinom(16, size=37, p=.4) - pbinom(8, size=37, p=.4)
```

Find the binomial probability that P($8 < x < 16$) when p = .4.

This has a small change compared to the last one (< 16 instead of ≤ 16), but it is an important difference. Because the `pbinom()` command finds the probability of being less than or equal to a given value, if we ran `pbinom(16, size=37, prob=.4)` it would include the probability of $x = 16$, which we do not want. We would want to run this (remember to NOT make this "minus one" adjust for continuous distributions like the exponential, uniform, or normal distributions):

```
pbinom(15, size=37, p=.4) - pbinom(8, size=37, p=.4)
```

Find the **normal approximation** (WITHOUT continuity correction) of the probability P($8 < X \leq 16$) when p = .4.

Now we use `pnorm()` to approximate the binomial distribution, but we do not make the continuity correction because that is what the instructions said. See Yakir (2011) p. 98 for some ideas (without continuity correction). For the `pnorm()` command, we need mean and standard deviation, and we can get those from the original binomial probability data (Yakir 2011, p 69) and from the example on page 98: $\mu = n \times p$ and std dev is $\sqrt{n \times p \times (1-p)}$ (in the R code below, `sigma` represents the population standard deviation).

To get a rough idea of what the calculations are doing, see Figure 6.3 on page 70 in these notes.

```
mu <- 37 * .4
sigma <- sqrt(37 * .4 * (1 - .4))
pnorm(16, mean=mu, sd=sigma) -  pnorm(8, mean=mu, sd=sigma)
```

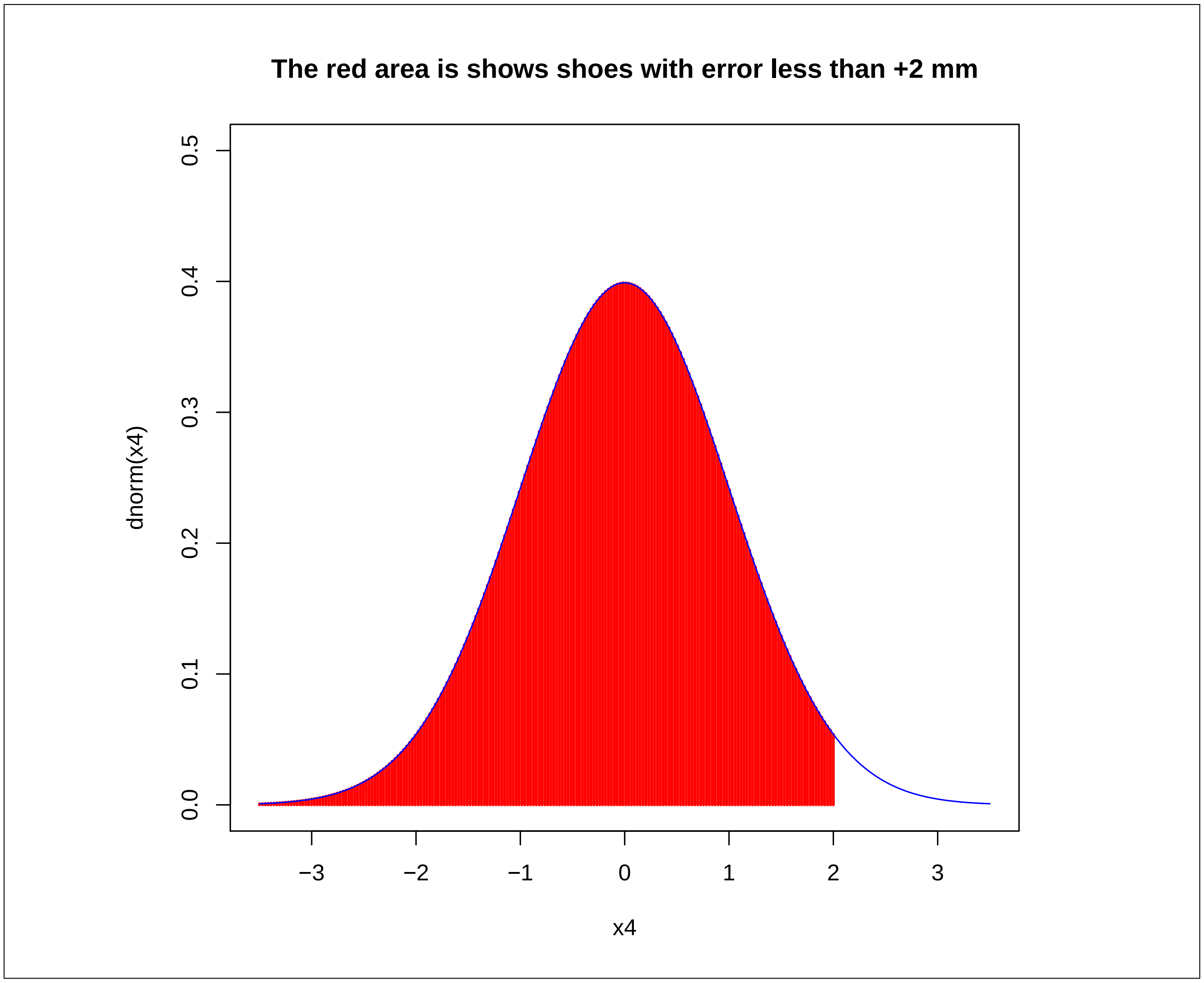


Figure 6.3: Using `pnorm()` to Solve Probability Problems.

Find the normal approximation (WITH continuity correction) of the probability
P($8 < X \leq 16$) when p = .4.

See Yakir (2011) pp. 98–100. It is probably best to draw a picture like the one in Figure 6.3. Shade the region of interest and determine from the $<$ or $\leq$ if you need to round up or down. In binomial probability, think of "16" as being a bar that has a width of 1 that extends from 15.5 to 16.5. We adjust the `pnorm()` commands below to include the data where that "1-unit wide" bar would be:

```
mu <- 37 * .4
sigma <- sqrt(37 * .4 * (1 - .4))
pnorm(16.5, mean=mu, sd=sigma) -  pnorm(8.5, mean=mu, sd=sigma)
```

If the ≤ 16 were < 16 then the solution would be to change the 16.5 in the R code to 15.5 (to translate the discrete binomial probability problem to the continuous normal distribution). You would make that change because in "the binomial world" the bar at 16 extends from 15.5 to 16.5, and you want to be "less than" whatever is in that bar, meaning that you would want to be less than 15.5 in the "normal distribution world."

Reference

Yakir, B. (2011). *Introduction to Statistical Thinking (With R, Without Calculus).* Retrieved from `http://pluto.huji.ac.il/~msby/StatThink/IntroStat.pdf`

Chapter 7

Chapter 7 Notes

7.1 Intro

The main challenge in Chapter 7 was to distinguish between the mean and standard deviation of a sample of data (as we first studied in Chapter 3) and the mean and standard deviation of a *sampling distribution.* A sampling distribution is a distribution of the attributes of many samples. Also be sure that you are thoroughly familiar with the difference between the Law of Large Numbers and the Central Limit Theorem. It helps to understand sampling distributions if you want to understand the Central Limit Theorem.

7.2 Sampling Distributions

A *sampling distribution* is a set of numbers that contains information about other samples. Let's say that we are going to conduct 30 studies in which we sample 100 people in each study and measure IQ. That means that we will survey 3,000 people in total, and our plot of the sampling distribution will have 30 observations (one for each study). Yakir (2011, p. 114) says that the expectation (mean) of the sampling distribution is the same as the expectation of the original data. In our example, we would expect the mean to be very close to 100. Yakir (2011, p. 114) also says that the variance of the **sampling distribution of the mean** can be calculated by taking the variance of the original data and dividing by n. This can be written as $\text{Var}(\bar{X}) = \text{Var}(X)/n$. Remember that the equation applies to the sampling distribution of the **mean**. If you find a different statistic for each sample (like the minimum, maximum, 32nd percentile...) the variance of the sampling distribution might be different (use a simulation to check it). In this case $n = 100$ to represent the number of observations in each sample.

Remember that $\text{Var}(\bar{X})$ is not the variance of the original population but the variance of a bunch of means that would be obtained from many samples from that population. If we take one sample of 100 people and the mean is 97, that that is the first value in the sampling distribution. If we take another sample of 100 people and the mean is 101, then that is the second variable in the sampling distribution. Notice that the *unit of measurement* is the mean of a sample that has potentially many observations.

7.3 Example 8.3.3

This is a review of Example 8.3.3 (Yakir, 2011, p. 130). Some parts of the example in the book might be unclear, but if you read the first paragraph of section 8.3.3 and the two numbered items, that helps to put the example into context. The two numbered items say this:

1. The probability that the average excess time used by the 80 customers in the sample is longer than 20 minutes.
2. The 95th percentile for the average excess time for samples of 80 customers who exceed their basic contract time allowances. (Yakir, 2011, p. 130)

I will write the R code in a different format to show the statistics that come from the original data (the sample of 80 customers) and the calculations that use those statistics to make statements about the sampling distribution. The following is modified from Yakir (2011, p. 130):

```
> # Statistics about the original sample
> # ('lamb' represents the Greek letter 'lambda').
> lamb <- 1 / 22
> n <- 80
> x.bar <- 1 / lamb
> # The extra parenthesis below are for clarity:
> var.x <-  1 / (lamb^2)
> sd.x <- sqrt( var.x)
>
> #
> # Now use the statistics above to find the expectation
> # and standard deviation of the sampling distribution:
> #
> # The mean does not change:
> mu.bar <- x.bar
> mu.bar
[1] 22
# The standard deviation of the sampling distribution
# is the standard deviation of the population times
# the square root of 1/n
> sig.bar <- sd.x * sqrt(1/n)
> sig.bar
[1] 2.459675
```

Now we are going to use the mean (`mu.bar`) and standard deviation (`sig.bar`) of the SAMPLING DISTRIBUTION to answer some questions about our expectation of the sample. The top paragraph on page 131 is a bit vague, but in the context of the entire example, we are being told to focus on the normal distribution when we are dealing with the sampling distribution—regardless of the distribution of the original data! The original data was a sample of the excess cell phone time used by 80 people, and our question is about the probability that THE MEAN OF THAT SAMPLE is longer than 20 minutes. The answer is:

```
> 1-pnorm(20,mu.bar,sig.bar)
[1] 0.7919241
```

where `pnorm(20,mu.bar,sig.bar)` is the cumulative probability of observing a score less than or equal to 20 in a population that has a mean of `mu.bar = 22` and a standard deviation of `sig.bar = 2.459675`. When you take one minus that value, you get the probability that the SAMPLE MEAN is greater than .20, which is what the original question asked. So the probability of getting greater than .20 is .79. For continuous distributions, the answers are the same for "less than" and ”less than or equal to” because the probability of being EXACTLY 20 approaches zero (e.g., 20.000000000000000001 is not exactly 20).

You can look at other values for `pnorm` by plotting this:

```
x = 0:35
plot(x, pnorm(x, mean=mu.bar, sd=sig.bar), type='l')
```

It might be easier to understand the original density plots. The R code below will create the plot shown in Figure 7.1). Look at the plot of the data (a couple pages below) and notice that the spread of the blue line (the sampling distribution) is a lot skinnier than the plot of the red line (the original data). Note also that the red line is not even remotely similar to the normal distribution, but the distribution of the sample means is normal.

```
x=c(-1000:8000) / 100
plot(x, dexp(x, lamb),type="l",
  col='red',  ylim=c(0, .18),
  main="Dist. for the Sample (red) and Samp Dist (blue)")
lines(x, dnorm(x, mu.bar,sig.bar),type="l",
  col='blue',
  main="Density Plot for Sampling Distribution")
text(30, .12, '< distribution of sample means', adj=0)
text(30, .11, '  (normally distributed)', adj=0)
text(38, .02, 'red=distribution of individual', adj=0)
text(38, .01, 'customers (exp distribution)', adj=0)
```

For the second part of Example 8.3.3, the goal was to find the 95th percentile of the “average excess time for samples of 80 customers” (Yakir, 2011, p. 130). The question is asking about percentages related to the sampling distribution… the distribution of sample means.

```
> qnorm(0.95,mu.bar,sig.bar)
[1] 26.04580
```

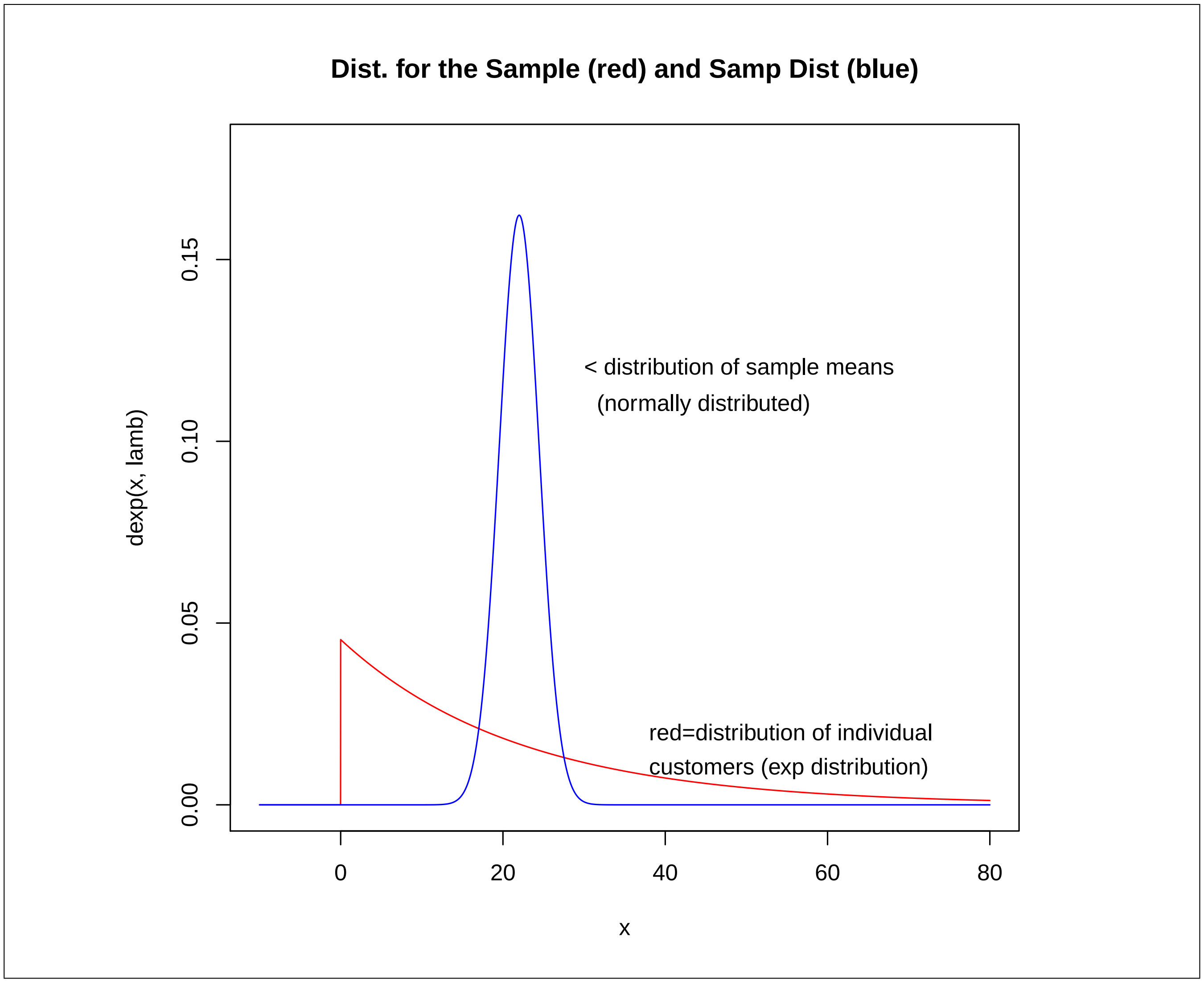


Figure 7.1: Original distribution and Sampling Distriubtion for Example 8.3.3.

The answer of 26.04580 means that we would expect 95% of the SAMPLE MEANS of size $n = 80$ to be less than 26.04580. This makes no statement about the individual customers. See if you can look at the plot with the blue and red lines and see if it looks like 95% of the area under the blue curve is less than 26.04580.

7.4 Sampling Distributions and the Central Limit Theorem

7.4.1 Sampling Distributions

A sampling distribution is a collection of numbers that represent statistics for many samples. The "statistic" for each sample can be something like the mean, standard deviation, minimum, maximum, 32nd percentile, or any other thing that you can calculate from a sample of data. You collect many samples of data, then you make the calculation and save one number (the statistic) for each sample and put that one number from each sample into the *sampling distribution.*

The table of data that follows represents 20 samples of data, each of which contains 12 observations. The last column represents the mean of each of the 20 samples, and that **last column is the sampling distribution of the mean** for that study. The data, hypothetically, could have come from 20 days of data collection in which we sampled the amount of time between customers entering a store on 20 different days (assume that the population can be represented with exponential distribution with a rate [lambda] of 3).

I generated the data for test purposes by running the R code shown below, and the `hist` command at the end shows the exponential distribution of data. Try running this code:

```
set.seed(12345)
# change the next two values to experiment
sample.size=12
sample.count=20
x <- array(round(rexp(sample.count * sample.size, rate=3), 2),
  dim=c(sample.count, sample.size))
```

```
hist(x)
```

The blue column at the end represents the mean of each sample (each row of data). That blue column is the sampling distribution of the mean. If we obtain many more samples that contain 12 observations each, and we keep extending that blue column so that it is longer, then that blue column will eventually have an approximately normal distribution even though the raw data in columns 1–12 has an exponential distribution. The tendency of the blue column to be normally distributed follows from the *central limit theorem*.

1	2	3	4	5	6	7	8	9	10	11	12	mean
0.15	0.38	1.11	0.08	0.95	0.16	0.57	0.39	0.54	0.10	0.15	0.19	0.40
0.18	0.14	0.08	0.12	0.11	0.06	0.81	0.01	0.01	0.05	0.36	0.14	0.17
0.27	0.49	0.12	0.09	0.21	0.09	0.85	0.09	0.35	0.12	0.12	0.33	0.26
0.01	0.19	0.53	0.05	0.11	0.27	0.72	0.21	0.03	0.69	0.08	0.31	0.27
0.15	0.41	0.00	0.07	0.16	0.43	0.04	0.19	0.88	0.54	0.20	0.39	0.29
0.01	0.24	1.23	0.41	0.62	0.04	0.10	0.08	1.23	0.43	0.01	0.31	0.39
2.13	0.33	0.51	0.17	0.88	0.40	0.29	0.12	0.27	0.28	0.12	0.12	0.47
0.42	0.93	0.30	0.40	0.21	0.17	0.13	0.08	0.45	0.38	0.08	0.19	0.31
0.32	0.08	0.22	0.20	0.01	0.14	0.40	0.19	0.72	0.12	0.15	0.04	0.22
0.61	0.07	0.00	0.14	0.13	0.18	0.25	0.70	0.20	0.24	0.06	0.01	0.22
0.08	0.58	0.20	1.46	0.50	0.58	0.08	0.00	0.00	0.07	0.05	0.04	0.30
0.15	0.15	0.94	0.33	0.11	0.44	0.02	0.62	0.01	0.08	0.96	0.34	0.35
0.42	0.79	0.07	0.04	0.09	0.65	0.08	0.80	0.14	0.44	0.19	0.66	0.36
0.13	0.29	0.14	0.01	0.04	0.24	0.58	0.22	0.22	0.44	0.61	0.33	0.27
0.03	0.24	0.01	0.84	0.20	0.08	0.71	0.21	0.99	0.00	0.58	0.41	0.36
0.57	0.05	0.15	0.12	0.37	0.56	0.02	0.02	0.21	0.56	0.29	0.00	0.24
1.79	0.03	0.17	0.14	0.68	0.55	0.16	0.88	0.47	0.55	0.14	0.46	0.50
0.63	0.12	0.87	0.20	0.19	0.17	0.34	0.13	1.15	0.16	0.06	0.76	0.40
0.12	0.39	0.43	0.04	0.12	0.07	0.31	0.52	0.07	0.12	0.05	0.28	0.21
0.39	0.35	0.29	0.18	0.37	0.00	0.05	0.32	0.14	0.71	0.02	0.06	0.24

Instead of finding the mean of each row, we could have created a different sampling distribution by calculating the minimum value, maximum value, skew, first quartile, third quartile, cutoff point for the top 1%, or any other statistic that we can calculate for one row of data.

7.5 Simulation of Sampling Distributions

7.5.1 Normal Distribution Example

In the next example, I want to examine the sampling distribution of the 3rd quartile for a normal distribution that has a mean of 30, a population standard deviation of 2, and a sample size of 100. What does that mean? It means that we are going to take 100 samples of data and find the 3rd quartile for each one. We will then have 100 numbers (each of the 100 numbers represents the third quartile for one sample). Because I have 100 numbers that were calculated from randomly selected data, those 100 numbers will have some degree of variation. I want to find the mean and standard deviation of those 100 numbers that represent the 3rd quartile of each sample. When I have the result, I can say something like: "The typical 3rd quartile from my samples was 31.3 with a standard deviation of .001." I now have a good idea of what the 3rd quartile will be from any samples that I take (samples of size 100 with mean of 30 and standard deviation of 2).

In the middle of the R code below is the `quantile` command, but if you wanted to find the sampling distribution for some other statistics, you could change that line of code. You can also change the `sample.size`, `mu`, or `s` settings to explore different sample sizes, population mean values, or population standard deviations.

```
# You can change the first few numbers to test how this works.
# with different settings:

# Lines that start with "#" are comments in R code.

# sample.size is the number of observations in each sample.
sample.size = 100
# population mean for a normal distribution
mu = 30
# standard deviation for the population
s  = 2

# 'replicates" is the number of simulations to run:
replicates=50000
```

```
# Modified from Yakir, 2011, p. 117
sample.data <- rep(0, sample.size)

# After running each simulation, save one number,
# such as mean, mode, quantile, or any other statistic
# that you can calculate... save it in "sample.stats".
sample.stats <-  rep(0, replicates)
for(i in 1:replicates)
{
  # Each time this loop runs, it generates
  # some random/simulated data and then finds the 3rd quartile...

  sample.data <- rnorm(sample.size, mean=mu, sd=s)

  # You can change the "quantile()" command below to get
  # other statistics like skew, max, min, median...
  # (the "[i]" part says to save the 3rd quartile
  # for the current simulation in the list of
  # answers that are kept in the object called  "sample.stats".)
  sample.stats[i] <- quantile(sample.data, .75)

}
# Find the mean of the sampling distribution:
mean(sample.stats)

# Find the standard deviation of the sampling distribution:
sd(sample.stats)

# Find the variance of the sampling distribution:
var(sample.stats)

# show a histogram of the sampling distribution
hist(sample.stats)
```

The mean above should be close to the mean of the population. The standard deviation of this sampling distribution does not follow the equation in the book, because the equation in the book is for the special case of the sampling distribution of the mean (Yakir, 2011, p. 114):

$$\mathrm{Var}(\bar{X}) = \frac{\mathrm{Var}(X)}{n}$$

Where $\mathrm{Var}(\bar{X})$ is the variance of the sampling distribution and $\mathrm{Var}(X)$ is the variance of the original population. The equation in the book holds true if we are finding the mean of each sample, but in this example we looked at the 75^{th} quantile.

Exponential Distribution Example

In the next example, I want to examine the sampling distribution of the mean for a exponential distribution that has a rate of 3 and sample size of 100. In the middle of the R code below is the `mean` command, but if you wanted to find the sampling distribution for some other statistics, you could change that line of code. You can also change the sample size and lambda values near the top of the code. The mean and standard deviation at the bottom should be super close to the mean and standard deviation of the sampling distribution using the equation (Yakir, 2011, p. 114, second equation) using the variance for an exponential distribution (Yakir, 2011, p. 80).

```
# You can change the first few numbers to test how this works.
# with different settings:

# Lines that start with "#" are comments in R code.

# sample.size is the number of observations in each sample.
sample.size = 100
# population mean for a normal distribution
lamb = 3

# 'replicates" is the number of simulations to run:
replicates=50000

# Modified from Yakir, 2011, p. 117
sample.data <- rep(0, sample.size)

# After running each simulation, save one number,
# such as mean, mode, quantile, or any other statistic
# that you can calculate...save it in "sample.stats".
sample.stats <-  rep(0, replicates)
for(i in 1:replicates)
{
  # Each time this loop runs, it generates
  # some random/simulated data and then finds the mean...

  sample.data <- rexp(sample.size, rate=lamb)

  # You can change the "mean()" command below to get
  # other statistics like skew, max, min,
  # median, 3rd quartile...
  # (the "[i]" part says to save the mean
  # for the current simulation in the list of
  # answers that are kept in the object called  "sample.stats".)
  sample.stats[i] <- mean(sample.data)

}
# Find the mean of the sampling distribution:
mean(sample.stats)

# Find the standard deviation of the sampling distribution:
sd(sample.stats)

# Find the variance of the sampling distribution:
var(sample.stats)

# show a histogram of the sampling distribution
hist(sample.stats)
```

The mean and standard deviation from the simulation above should be close to the values that you get from the questions on pages 114 and 80 (where 'lamb' below equals 3 for the rate parameter and each sample in the simulation was of size 100):

```
> # Compare these "expected" results to the results
> # of your simulation.
> v = 1 / (lamb^2)
> # Variance of the original distribution:
> v
[1] 0.1111111
> v.bar = v / 100
> # We predict that the variance of our sampling distribution will be this:
> v.bar
[1] 0.001111111
>
> sd.bar = sqrt(v.bar)
> sd.bar
[1] 0.03333333
```

Chapter 8

Chapter 8 Notes (Review)

8.1 Expectation, Mean, and Frequency Tables

For the final exam, you will need to know how to calculate the expectation of a random variable. You would need to do this as part of the calculation for finding the variance or standard deviation of a random variable (theoretical distribution). When you collect a sample of real data, add the values and divide by the number of observations, you have the mean. If you are dealing with a random variable (theoretical distribution), there might not be any real data to collect, so we refer to the expectation. The expectation can be considered a more abstract way to think about the mean, and the expectation might reflect different "weights" for each possible value in the distribution.

We have covered many examples of finding the expectation when you are given a random variable that is defined by a table of values and probabilities. The values represent the sample space and the probabilities represent the chance of drawing that value if you select randomly from the population.

The equations for finding the expectation of a random variable (from a probability table) are in Yakir (2011), p. 57, and an example is at the top of p. 58. That example continues and shows how to calculate the population variance from a probability table (p. 59).

Question 3.2.1 is a similar example of finding the mean from a frequency table (Yakir, 2011, p. 43). Note that it is an example of finding the mean when you are given a frequency table that reflects sample data as opposed to population probabilities. The second part of the problem, Question 3.2.2, refers to the **sample variance** as opposed to the population variance. The equation for sample variance contains $n-1$ in the denominator (shown as `sum(freq) -1` in the Question 3.2.2), unlike the population standard deviation, which uses N.

8.2 Standard Deviation

We have covered standard deviation of a sample, of a population, of a random variable, and of a sampling distribution. When you want to find the standard deviation, first ensure that you know what you are calculating. The sample standard deviation (s) contain $n-1$ in the denominator versus the population standard deviation (σ), which typically contains N in the denominator.

- The standard deviation is the square root of the variance. This applies regardless of what the distribution is.
- See the file called MATH128Ch5-6QuickNotes.pdf (in the Announcements section of the MATH1280 classroom) for calculations for standard deviation for different distributions.
- See Yakir (2011) p. 46 and 63 for a list of symbols and what they mean.
- Sample Standard Deviation (s): Use this when you have sample data (a full list of numbers as opposed to having a random variable or something that represents a population). See Yakir (2011), pp. 38–39 for the manual calculation. The R function is at the top of Yakir (2011) p. 40.
- Population Standard Deviation (sigma, σ). Use this when you are dealing with an entire population. See Yakir (2011), p. 52. The is the same idea as the examples on page 38–39, except the denominator contains N instead of $n-1$ (the capital letter N usually refers to the number of observations in the population as opposed to lower case n which typically refers to the number of observations in a sample).
- Sample Standard Deviation when you have only a frequency table: Be sure to not confuse a sample of data with a definition of a population. If you are given a random variable,

it represents a population. If the frequency table represents a data sample that had been collected, then see Example 3.2.1 and 3.2.2 in the Solved Exercises.

- Frequency Table and Random Variable tip: if you are given a frequency table (for either a population or a sample), you can not type values from that table into the `sd()` function in R to get the standard deviation because doing so would not reflect the "weight" of each number (some numbers are more likely than others)—use the manual calculations or another technique if you know it.

- Standard Deviation when you have only the sample space (a list of values) and their probabilities (such as the definition of a random variable): Be sure that you know if you want the sample standard deviation or the population standard deviation. For sample standard deviation, look for s or s^2 (for variance) in Yakir (2011) p. 57–58 for equations and example calculations. For population standard deviation, look for the line in the book that shows square root of $\mathrm{Var}(X)$ in Yakir (2011) p. 58–59. See Example 4.1.7 and 4.1.8 for the population variance and standard deviation of a random variable.

- Standard Deviation of a Sampling Distribution: A sampling distribution is when you collect a sample (let's say the sample contains 100 observations), find a statistic (such as mean or standard deviation), then repeat that process many times then look at the statistics. If you conduct 30 studies, each of which contained 100 observations, you would have 30 observations of "sample means." The distribution of sample means is the sampling distribution of the sample mean, and that standard deviation is smaller than the standard deviation of the original sample. The variance of the sampling distribution **of the mean** is $\mathrm{Var}(\bar{X}) = \mathrm{Var}(X)/n$ (Yakir, 2011, p. 114). If you are finding the sampline distribution of some other statistics (like the 'minimal' value in each sample), the variance of the sampling distribution might be different. Note that on the left side of that equation, there is bar over the X, which means that it is the variance of the sample means. The right side means that you can start with the variance of the original population and divide by n to get the variance of the sampling distribution. For standard deviation, you would take the square root of that: $\sqrt{\mathrm{Var}(X)/n}$ where n is the number of observations in the sample. Also see the Chapter 7 notes below.

8.3 Random Variables

For Chapter 5, see the MATH1280Chapter5-6Quicknotes file in the Announcements section and the Chapter 5 notes that are also in the Announcements section.

8.4 Normal Distribution

8.4.1 Intro

Chapter 6 reflects cumulative knowledge from the first five chapters. Review the whole chapter and do the example problems. If you are familiar with finding probabilities when you are given values of observations, then it will be easier to do similar tasks with other observations. You should also be able to take a probability (such as the top 1%, bottom 1% or middle X%) and identify the cut-off points (criteria values that mark the top 1% or whatever you are trying to find). The notes for Chapter 6 that are posted in the Announcements forum of MATH1280 address many of the main topics.

8.4.2 Standard Normal Distribution

If I have a distribution with a mean of 100 and a standard deviation of 7, I could say that a value of 114 is two standard deviations from the mean (because 2 times the standard deviation of 7 is 14). In this case, the value 114 has a z-score of 2 (because it is two standard deviations above the mean; see Yakir, 2011, p. 90). In some cases, it is useful to view all the values in the distribution in terms of their z-score.

In the example below, I have a set of numbers in x. If I find the deviations (differences between each value and the mean of the data) and then I divide by the standard deviation, I can show all the values in terms of the z-scores:

```
> x <- c(23, 28, 20, 25, 30, 26, 23, 28, 27, 26, 23, 25)
> x2 <- (x - mean(x)) / sd(x)
> # The mean of the new version is 0 and the SD is 1 (within a small rounding error)...
> # That is "standardized" form:
> round(mean(x2), 6)
[1] 0
> sd(x2)
[1] 1
>
> # Show a list of the original numbers next
> # to their standard scores
> data.frame(x, x2)
    x         x2
1  23 -0.8410405
2  28  0.9611891
3  20 -1.9223782
4  25 -0.1201486
5  30  1.6820810
6  26  0.2402973
7  23 -0.8410405
8  28  0.9611891
9  27  0.6007432
10 26  0.2402973
11 23 -0.8410405
12 25 -0.1201486
>
> # values that are below the mean of x have negative z-scores
> mean(x)
[1] 25.33333
```

8.4.3 Normal Approximations

One part of Chapter 6 that was new was the normal approximation of other distributions and the use of continuity correction. Here is the general idea: if we model a coin-flipping experiment using a binomial distribution, then we use information about n (number of coin flips) and p (probability of success) to predict the answer.

If we use the normal approximation, then we consider n coin tosses to be "one experiment," and the outcome of a binomial probability experiment is expected to be $E(X) = np$. In terms of the normal distribution, this means that the population mean would be $\mu = np$, and the standard deviation would be whatever we calculated from the binomial distribution ($s = np(1-p)$). If $n = 100$ and $p = .5$, then $E(X) = 100 \times .5 = 50$ and $s = \sqrt{100 \times .5 \times (1 - .5)} = 5$. See Yakir (2011), pp. 96–97. You might also want to do the continuity correction because the binomial distribution is discrete (allows only integers) and the normal distribution is continuous.

To convert a Poisson distribution to a normal approximation, follow the same general approach: choose a subjectively long period of time and determine the expectation for that period of time using whatever you know about the Poisson distribution (see the MATH1280Ch5-6QuickNotes.pdf file in the Announcements section). There is an example in Yakir (2011), p. 99.

8.4.4 Continuity Correction

If you are puzzled by the need to "add 1" or "add .5" when you work with the normal approximation of a Poisson distribution, see the Chapter 5-6 "quick" notes that are posted in the Announcement forum, or email your instructor. There are detailed examples of problems in those notes.

8.5 Law of Large Numbers and Central Limit Theorem

Review the Chapter 7 notes from last week. There might be a simulation on the final. You can refer to the Chapter 7 notes for detailed examples of simulations. If you struggle with the programming aspect of simulations, you can survive the final by doing well on other topics.