R for Psych Handbook

David John Baker

2018-04-09

Contents

Chapter 1

Preface

This book serves as a collection of resources used in the LSU psychological statistics courses. It contains some lecture notes, as well as R code and data to run each of the examples in R.

The book is still under construction and if you would like to help out with it, please get in touch!

The majority of the content comes from having taken and then TA's for the LSU psychology department's stats courses. Was given oppertunity to work for psych department as music student, partially because of R, and writing this handbook was way of saying thank you for the great experience that I had. Content was generated by both Jason Hicks and Jason Harman, I just provided R as the glue to hold it all together. Idea is that anyone going through the LSU stats rotation would be able to if they wanted put in the extra work and learn R while they are doing the assigments in SPSS. Data and content are from previous years, form same, content different.

As noted in next chapter, it's hard to learn R, but worth doing it. Personally found that R has opened up more oppertunities than anything else I have invested in. See it as investing in self. Got started with it because of guy who advised my Masters thesis. If you feel like you can't do this, know that reason I started R is I knew so little about computers, Daniel had to show me how to do an IFELSE() statement in Excel. At that point figured if I was going to learn, might as well be frustrated and get a better product out of it.

One thing to learn a lot by running these examples, though the best way forward is to find your own dataset and start to set up project with it.

Beauty of R is that you are trying to become a researcher. R is cullinary school of statistics. SPSS is a microwave. Saw that on twitter.

Let's get cooking.

Chapter 2

Introduction to R

Before starting out, it's worth mentioning that R has a steep learning curve compared to other statistical softwares. While there are tons of blog posts as to why you should learn R, I will keep my list quick so if you get discouraged at any point, you can come back to this list and get reinspired before R starts paying you back.

1. The R community is fantastic, check out #rstats on Twitter (this is really my favorite reason)
2. R will always be free because the people behind it believe in open source principles LINK.
3. Time spent learning R is time spent learning how computers work, if you learn about R, you are also learning computer programming. Time spent in something like SPSS or SAS does not easily tranfer to other programs.
4. On r-jobs.com the way they decide to split jobs is jobs that make above and below $100,000.
5. R is your ticket out of academia, if you need it.

2.1 Getting Started

Since this book is about statistics and R, the introduction to all things R is a bit shorter than other guides. If you need help, find a friend (or email the author of this book) and they will get you started. The first things you need to do is download R from CRAN, then get the latest version of RStudio. RStudio is an IDE that makes working in R a lot easier. Do not use R without it.

Once you have that, you can start to play around with this.

2.2 Setting The Working Directory

At the end a long writing session you normally have to find a place to save that journal article you are working on. You do this by clicking 'Save As' then finding where to stick it and what to call it. If you are programming you need to do this right away by saving your script (the what) and setting your working directory (the where).

Once you have a new script set up, now we can start to run the next commands. This chapter covers a few basic things

- The rationale behind to writing scripts and code
- R as Calculator
- Data Structures
- Manipulating Data
- Tons of functions

- Whirlwind tour of R for psychologists

The goals of this chapter are to:

- Learn Basics of R and RStudio
- Practice concepts with simple examples
- Familiarize with vast array of functions
- Have vocabulary and "R" way to think about problem solving

2.3 Rationale

Easily reproducible Less Human Error in cleaning Time invested in programming transfers Pretty Graphs Open Science Huge integration

2.4 R as Calculator

The Console of R is where all the action happens. You can use it just like you would use a calculator. Try to do some basic math operations in it.

```
2 + 2
```

```
## [1] 4
```

```
5 - 2
```

```
## [1] 3
```

```
10 / 4
```

```
## [1] 2.5
```

```
9 * 200
```

```
## [1] 1800
```

```
sqrt(81)
```

```
## [1] 9
```

```
10 > 4
```

```
## [1] TRUE
```

```
2 < -1
```

```
## [1] FALSE
```

You don't always want to print your output and retype it in. Idea is to be very lazy (efficient).

Save some math to an object with the <- operator, then manipulate that.

```
foo <- 2 * 3
foo * 6
```

```
## [1] 36
```

Notice what has popped up in your environment in RStudio!

Let's get more efficient.

```
yearsInGradSchool <- c(2,1,4,5,6,7,3,2,4,5,3)
```

talk about the c function.

```
yearsInGradSchool * 3
```

```
##  [1]  6  3 12 15 18 21  9  6 12 15  9
```

Or this

```
yearsInGradSchool - 2
```

```
##  [1]  0 -1  2  3  4  5  1  0  2  3  1
```

```
yearsInGradSchool < 2
```

```
##  [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

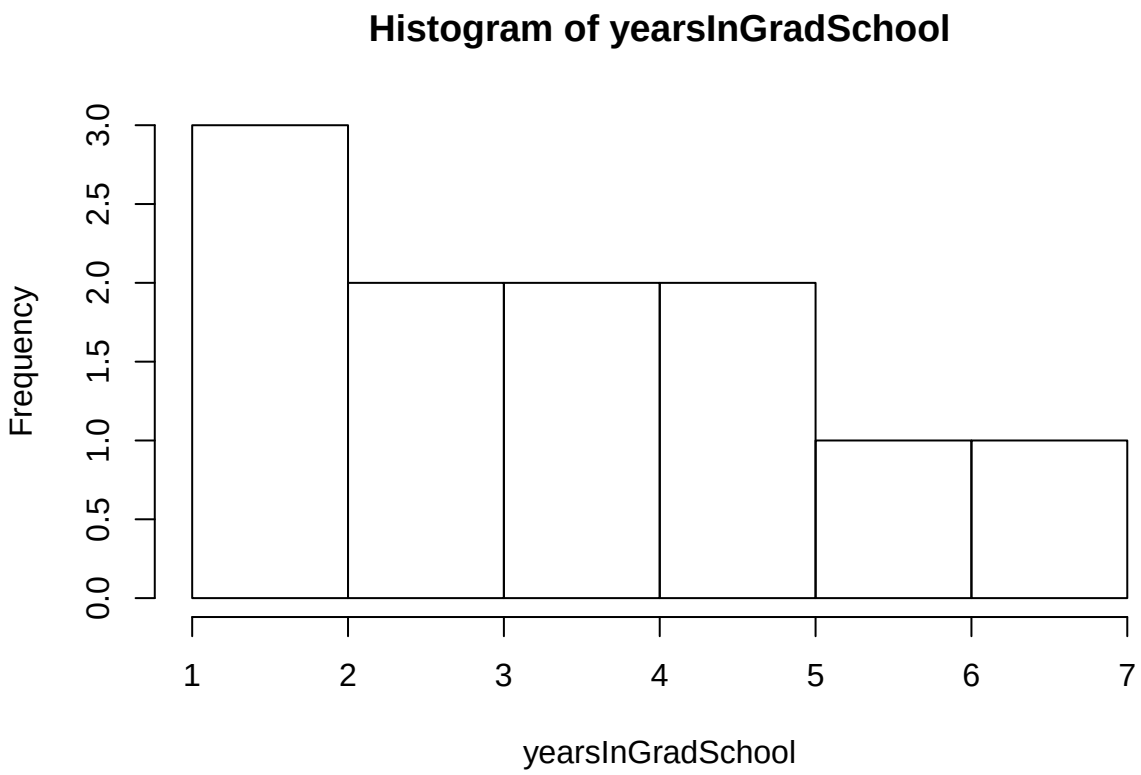
```
mean(yearsInGradSchool)
```

```
## [1] 3.818182
```

```
sd(yearsInGradSchool)
```

```
## [1] 1.834022
```

```
hist(yearsInGradSchool)
```




```
scale(yearsInGradSchool)
```

```
##              [,1]
##  [1,] -0.99136319
##  [2,] -1.53661295
##  [3,]  0.09913632
##  [4,]  0.64438608
##  [5,]  1.18963583
##  [6,]  1.73488559
```

```
##  [7,] -0.44611344
##  [8,] -0.99136319
##  [9,]  0.09913632
## [10,]  0.64438608
## [11,] -0.44611344
## attr(,"scaled:center")
## [1] 3.818182
## attr(,"scaled:scale")
## [1] 1.834022
```

```
range(yearsInGradSchool)
```

```
## [1] 1 7
```

```
min(yearsInGradSchool)
```

```
## [1] 1
```

```
class(yearsInGradSchool)
```

```
## [1] "numeric"
```

```
str(yearsInGradSchool)
```

```
##  num [1:11] 2 1 4 5 6 7 3 2 4 5 ...
```

```
summary(yearsInGradSchool)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.500   4.000   3.818   5.000   7.000
```

```
yearsInGradSchool <- c(2,1,4,5,6,7,3,2,4,5,3)
classesTaken <- c(5,2,5,7,9,9,2,8,4,7,2)
gradData <- data.frame(yearsInGradSchool,classesTaken)
```

```
cor(yearsInGradSchool,classesTaken)
```

```
## [1] 0.6763509
```

Basic plots, arguments. ggplot and libraries

```
plot(yearsInGradSchool,classesTaken,
     data = gradData,
     main = "My Plot",
     xlab = "Years in Grad School",
     ylab = "Classes Taken")
```

```
## Warning in plot.window(...): "data" is not a graphical parameter
```

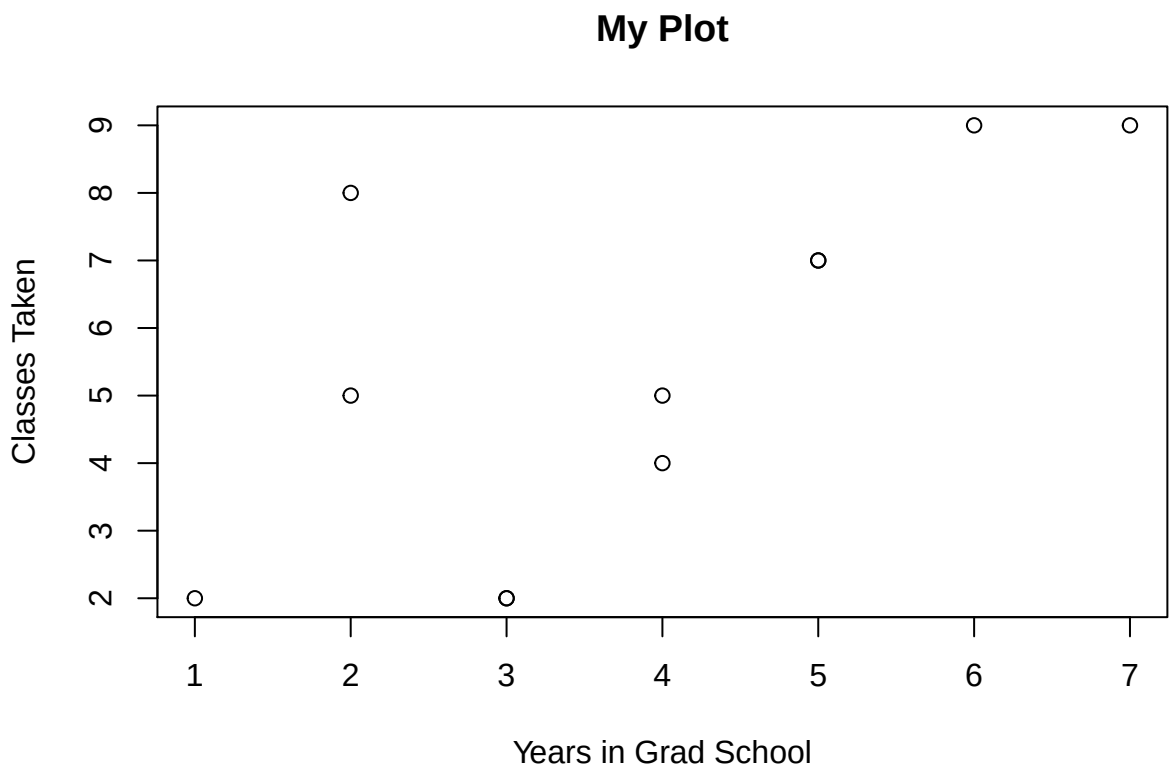
```
## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not
## a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "data" is not
## a graphical parameter
```

```
## Warning in box(...): "data" is not a graphical parameter
```

```
## Warning in title(...): "data" is not a graphical parameter
```



2.5 Data Exploration

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
class(iris)
```

```
## [1] "data.frame"
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
```

```
##
##
```

Accessing individual 'columns' is done with the $ operator

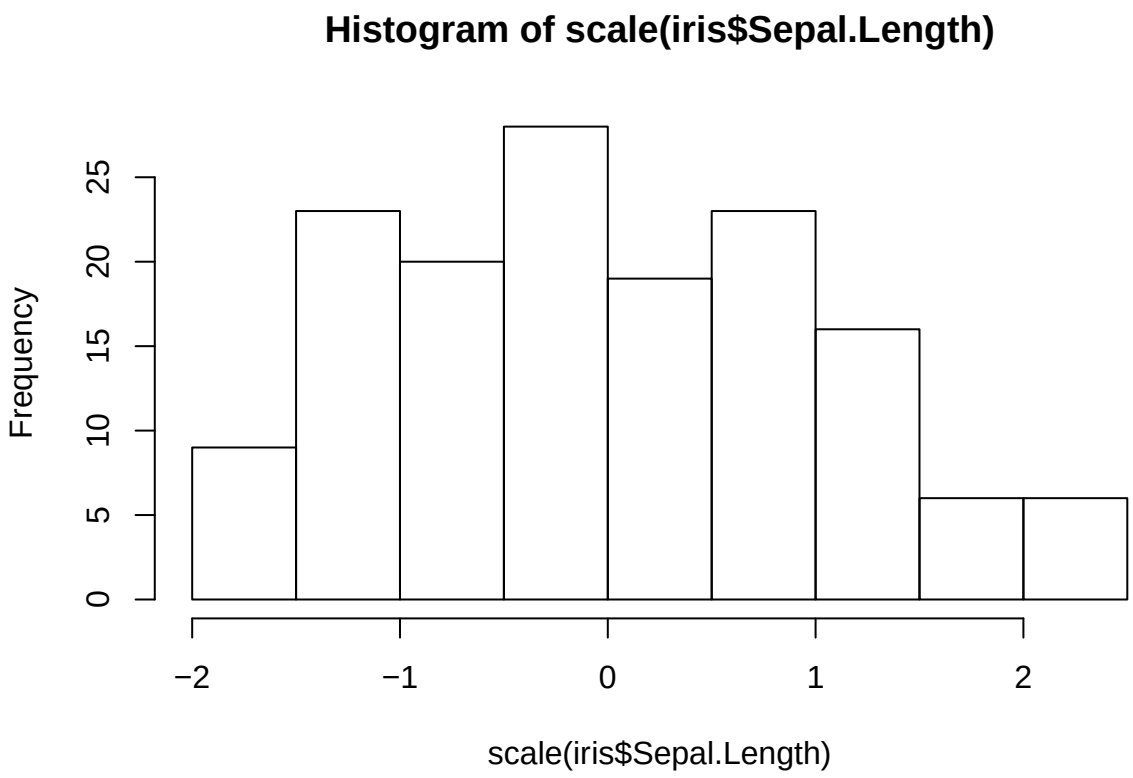
```
iris$Sepal.Length
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4
##  [18] 5.1 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5
##  [35] 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0
##  [52] 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8
##  [69] 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4
##  [86] 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8
## [103] 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7
## [120] 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7
## [137] 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
```

Can you use this to plot the different numeric values against each other?

What would the follow commands do?

```
hist(scale(iris$Sepal.Length))
```




```
iris$Sepal.Length.scale <- scale(iris$Sepal.Length)
```

2.6 Indexing

Let's combine logical indexing with creating new objects.

What do the follow commands do? Why?

```
iris[1,1]
```

```
## [1] 5.1
```

```
iris[2,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 2          4.9           3          1.4         0.2  setosa
##   Sepal.Length.scale
## 2            -1.1392
```

```
iris[,5]
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

```
iris[iris$Sepal.Length < 5,]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
```

```
## 35           4.9         3.1          1.5         0.2     setosa
## 38           4.9         3.6          1.4         0.1     setosa
## 39           4.4         3.0          1.3         0.2     setosa
## 42           4.5         2.3          1.3         0.3     setosa
## 43           4.4         3.2          1.3         0.2     setosa
## 46           4.8         3.0          1.4         0.3     setosa
## 48           4.6         3.2          1.4         0.2     setosa
## 58           4.9         2.4          3.3         1.0 versicolor
## 107          4.9         2.5          4.5         1.7  virginica
##     Sepal.Length.scale
## 2           -1.139200
## 3           -1.380727
## 4           -1.501490
## 7           -1.501490
## 9           -1.743017
## 10          -1.139200
## 12          -1.259964
## 13          -1.259964
## 14          -1.863780
## 23          -1.501490
## 25          -1.259964
## 30          -1.380727
## 31          -1.259964
## 35          -1.139200
## 38          -1.139200
## 39          -1.743017
## 42          -1.622254
## 43          -1.743017
## 46          -1.259964
## 48          -1.501490
## 58          -1.139200
## 107         -1.139200
```

```
iris[,c(1:4)]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1            5.1         3.5          1.4         0.2
## 2            4.9         3.0          1.4         0.2
## 3            4.7         3.2          1.3         0.2
## 4            4.6         3.1          1.5         0.2
## 5            5.0         3.6          1.4         0.2
## 6            5.4         3.9          1.7         0.4
## 7            4.6         3.4          1.4         0.3
## 8            5.0         3.4          1.5         0.2
## 9            4.4         2.9          1.4         0.2
## 10           4.9         3.1          1.5         0.1
## 11           5.4         3.7          1.5         0.2
## 12           4.8         3.4          1.6         0.2
## 13           4.8         3.0          1.4         0.1
## 14           4.3         3.0          1.1         0.1
## 15           5.8         4.0          1.2         0.2
## 16           5.7         4.4          1.5         0.4
## 17           5.4         3.9          1.3         0.4
## 18           5.1         3.5          1.4         0.3
## 19           5.7         3.8          1.7         0.3
```

```
## 20          5.1         3.8          1.5         0.3
## 21          5.4         3.4          1.7         0.2
## 22          5.1         3.7          1.5         0.4
## 23          4.6         3.6          1.0         0.2
## 24          5.1         3.3          1.7         0.5
## 25          4.8         3.4          1.9         0.2
## 26          5.0         3.0          1.6         0.2
## 27          5.0         3.4          1.6         0.4
## 28          5.2         3.5          1.5         0.2
## 29          5.2         3.4          1.4         0.2
## 30          4.7         3.2          1.6         0.2
## 31          4.8         3.1          1.6         0.2
## 32          5.4         3.4          1.5         0.4
## 33          5.2         4.1          1.5         0.1
## 34          5.5         4.2          1.4         0.2
## 35          4.9         3.1          1.5         0.2
## 36          5.0         3.2          1.2         0.2
## 37          5.5         3.5          1.3         0.2
## 38          4.9         3.6          1.4         0.1
## 39          4.4         3.0          1.3         0.2
## 40          5.1         3.4          1.5         0.2
## 41          5.0         3.5          1.3         0.3
## 42          4.5         2.3          1.3         0.3
## 43          4.4         3.2          1.3         0.2
## 44          5.0         3.5          1.6         0.6
## 45          5.1         3.8          1.9         0.4
## 46          4.8         3.0          1.4         0.3
## 47          5.1         3.8          1.6         0.2
## 48          4.6         3.2          1.4         0.2
## 49          5.3         3.7          1.5         0.2
## 50          5.0         3.3          1.4         0.2
## 51          7.0         3.2          4.7         1.4
## 52          6.4         3.2          4.5         1.5
## 53          6.9         3.1          4.9         1.5
## 54          5.5         2.3          4.0         1.3
## 55          6.5         2.8          4.6         1.5
## 56          5.7         2.8          4.5         1.3
## 57          6.3         3.3          4.7         1.6
## 58          4.9         2.4          3.3         1.0
## 59          6.6         2.9          4.6         1.3
## 60          5.2         2.7          3.9         1.4
## 61          5.0         2.0          3.5         1.0
## 62          5.9         3.0          4.2         1.5
## 63          6.0         2.2          4.0         1.0
## 64          6.1         2.9          4.7         1.4
## 65          5.6         2.9          3.6         1.3
## 66          6.7         3.1          4.4         1.4
## 67          5.6         3.0          4.5         1.5
## 68          5.8         2.7          4.1         1.0
## 69          6.2         2.2          4.5         1.5
## 70          5.6         2.5          3.9         1.1
## 71          5.9         3.2          4.8         1.8
## 72          6.1         2.8          4.0         1.3
## 73          6.3         2.5          4.9         1.5
```

```
## 74      6.1         2.8         4.7         1.2
## 75      6.4         2.9         4.3         1.3
## 76      6.6         3.0         4.4         1.4
## 77      6.8         2.8         4.8         1.4
## 78      6.7         3.0         5.0         1.7
## 79      6.0         2.9         4.5         1.5
## 80      5.7         2.6         3.5         1.0
## 81      5.5         2.4         3.8         1.1
## 82      5.5         2.4         3.7         1.0
## 83      5.8         2.7         3.9         1.2
## 84      6.0         2.7         5.1         1.6
## 85      5.4         3.0         4.5         1.5
## 86      6.0         3.4         4.5         1.6
## 87      6.7         3.1         4.7         1.5
## 88      6.3         2.3         4.4         1.3
## 89      5.6         3.0         4.1         1.3
## 90      5.5         2.5         4.0         1.3
## 91      5.5         2.6         4.4         1.2
## 92      6.1         3.0         4.6         1.4
## 93      5.8         2.6         4.0         1.2
## 94      5.0         2.3         3.3         1.0
## 95      5.6         2.7         4.2         1.3
## 96      5.7         3.0         4.2         1.2
## 97      5.7         2.9         4.2         1.3
## 98      6.2         2.9         4.3         1.3
## 99      5.1         2.5         3.0         1.1
## 100     5.7         2.8         4.1         1.3
## 101     6.3         3.3         6.0         2.5
## 102     5.8         2.7         5.1         1.9
## 103     7.1         3.0         5.9         2.1
## 104     6.3         2.9         5.6         1.8
## 105     6.5         3.0         5.8         2.2
## 106     7.6         3.0         6.6         2.1
## 107     4.9         2.5         4.5         1.7
## 108     7.3         2.9         6.3         1.8
## 109     6.7         2.5         5.8         1.8
## 110     7.2         3.6         6.1         2.5
## 111     6.5         3.2         5.1         2.0
## 112     6.4         2.7         5.3         1.9
## 113     6.8         3.0         5.5         2.1
## 114     5.7         2.5         5.0         2.0
## 115     5.8         2.8         5.1         2.4
## 116     6.4         3.2         5.3         2.3
## 117     6.5         3.0         5.5         1.8
## 118     7.7         3.8         6.7         2.2
## 119     7.7         2.6         6.9         2.3
## 120     6.0         2.2         5.0         1.5
## 121     6.9         3.2         5.7         2.3
## 122     5.6         2.8         4.9         2.0
## 123     7.7         2.8         6.7         2.0
## 124     6.3         2.7         4.9         1.8
## 125     6.7         3.3         5.7         2.1
## 126     7.2         3.2         6.0         1.8
## 127     6.2         2.8         4.8         1.8
```

```
## 128          6.1         3.0          4.9         1.8
## 129          6.4         2.8          5.6         2.1
## 130          7.2         3.0          5.8         1.6
## 131          7.4         2.8          6.1         1.9
## 132          7.9         3.8          6.4         2.0
## 133          6.4         2.8          5.6         2.2
## 134          6.3         2.8          5.1         1.5
## 135          6.1         2.6          5.6         1.4
## 136          7.7         3.0          6.1         2.3
## 137          6.3         3.4          5.6         2.4
## 138          6.4         3.1          5.5         1.8
## 139          6.0         3.0          4.8         1.8
## 140          6.9         3.1          5.4         2.1
## 141          6.7         3.1          5.6         2.4
## 142          6.9         3.1          5.1         2.3
## 143          5.8         2.7          5.1         1.9
## 144          6.8         3.2          5.9         2.3
## 145          6.7         3.3          5.7         2.5
## 146          6.7         3.0          5.2         2.3
## 147          6.3         2.5          5.0         1.9
## 148          6.5         3.0          5.2         2.0
## 149          6.2         3.4          5.4         2.3
## 150          5.9         3.0          5.1         1.8
```

```
iris[c(1,2,3,4,5,6,8),c(1:3,5)]
```

```
##   Sepal.Length Sepal.Width Petal.Length Species
## 1          5.1         3.5          1.4  setosa
## 2          4.9         3.0          1.4  setosa
## 3          4.7         3.2          1.3  setosa
## 4          4.6         3.1          1.5  setosa
## 5          5.0         3.6          1.4  setosa
## 6          5.4         3.9          1.7  setosa
## 8          5.0         3.4          1.5  setosa
```

```
Setosas <- iris[iris$Species == "setosa",]
```

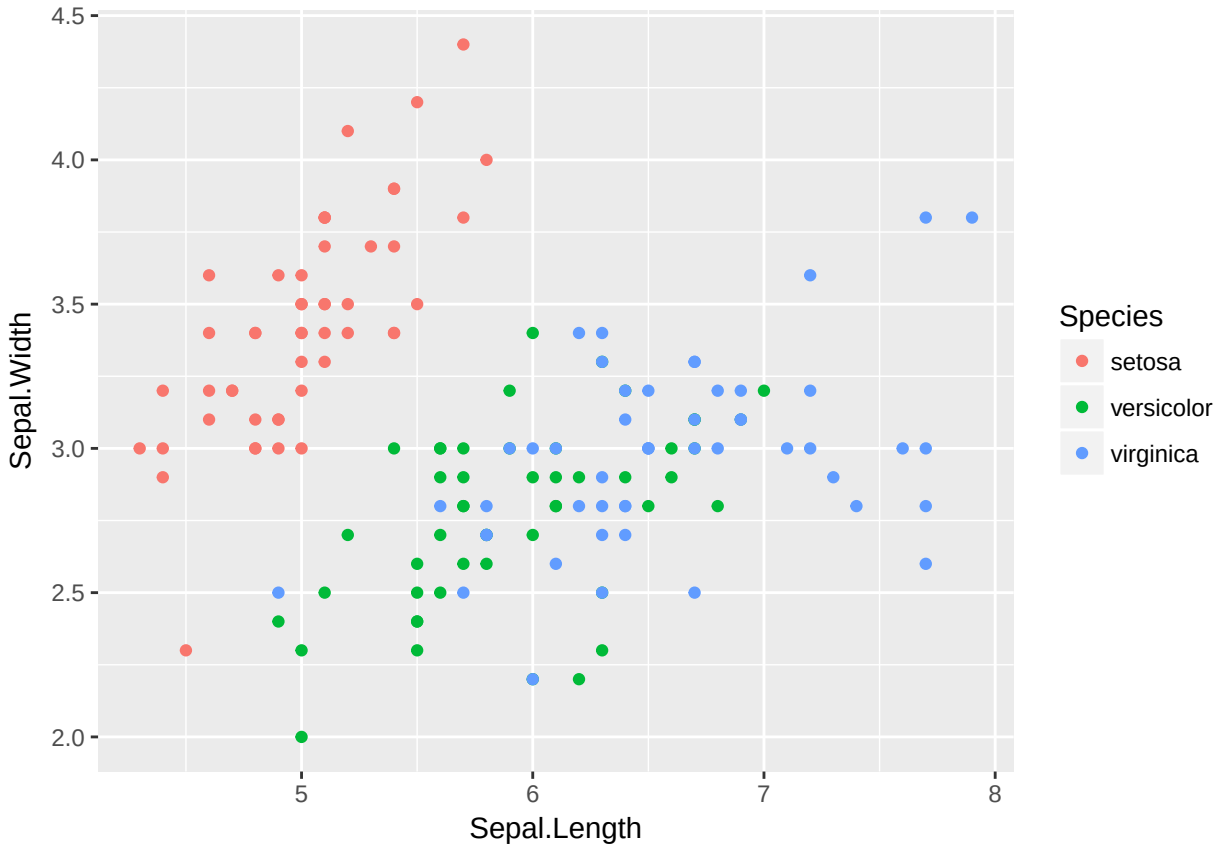
This could be an entire lecture by itself!!!

2.7 Whirlwind Tour of R

R's power comes in the fact that you download packages to put on top of base R

- Turning Data tables into already formatted APA Latex Tables (stargazer, xtable)
- Creating Publication Quality Graphs (ggplot2)
- Text manipulation (stringr)
- Exploring data and not making chart after chart after chart (psych)
- Every statistical test you could want (psych, cars, ezanova, lavaan)
- Software to plot not so normal output (SEMplots)
- Making Websites in R
- These slides were written in R
- Quickly processing huge datasets (data.table, dplyr)
- Tons of Machine Learning

```
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width,
                 color = Species),
       xlab = "Sepal Length",
       ylab = "Sepal Width",
       main = "My Plot") + geom_point()
```



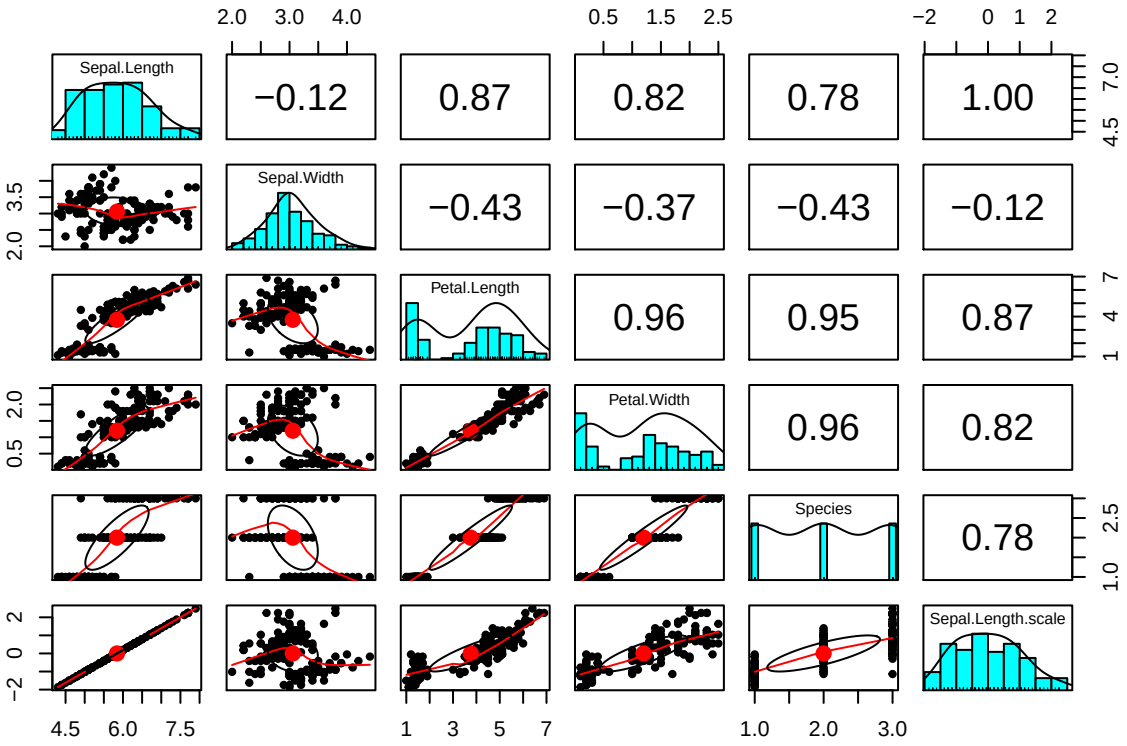
Or stuff for data exploration

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(iris)
```



2.8 Functions for Psychologists

- nlme() and lme4() for Multilevel Modeling
- lavaan() for Latent Variable Analysis
- ezAnova() for ANOVA based testing; the anova() function does model comparisons
- profileR for Repeated Measures MANOVA
- glm() and lm() for linear models
- caret() for Machine Learning

2.9 Resources

LINK THESE IN

- swirl
- stackoverflow.com
- Twitter
- Your peers
- R Community is fantastic (tidyverse!!!)

2.9.1 Template Stuff

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

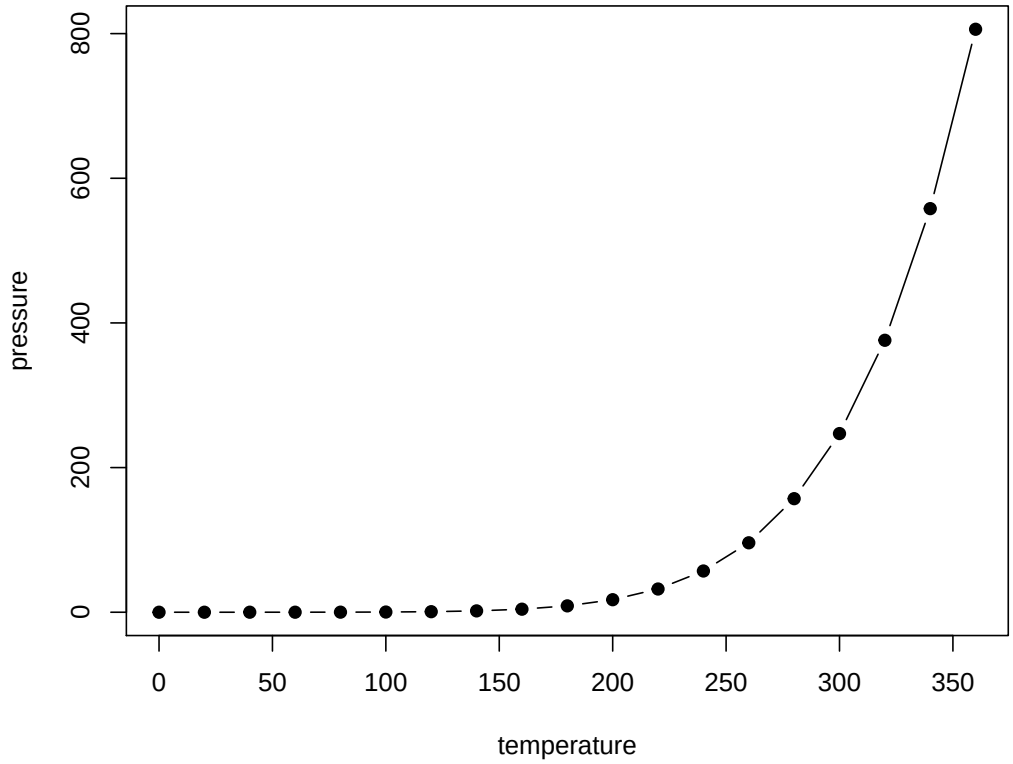


Figure 2.1: Here is a nice figure!

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (**?**) in this sample book, which was built on top of R Markdown and **knitr** (**?**).

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Sepal.Length.scale
5.1	3.5	1.4	0.2	setosa	-0.89767388
4.9	3.0	1.4	0.2	setosa	-1.13920048
4.7	3.2	1.3	0.2	setosa	-1.38072709
4.6	3.1	1.5	0.2	setosa	-1.50149039
5.0	3.6	1.4	0.2	setosa	-1.01843718
5.4	3.9	1.7	0.4	setosa	-0.53538397
4.6	3.4	1.4	0.3	setosa	-1.50149039
5.0	3.4	1.5	0.2	setosa	-1.01843718
4.4	2.9	1.4	0.2	setosa	-1.74301699
4.9	3.1	1.5	0.1	setosa	-1.13920048
5.4	3.7	1.5	0.2	setosa	-0.53538397
4.8	3.4	1.6	0.2	setosa	-1.25996379
4.8	3.0	1.4	0.1	setosa	-1.25996379
4.3	3.0	1.1	0.1	setosa	-1.86378030
5.8	4.0	1.2	0.2	setosa	-0.05233076
5.7	4.4	1.5	0.4	setosa	-0.17309407
5.4	3.9	1.3	0.4	setosa	-0.53538397
5.1	3.5	1.4	0.3	setosa	-0.89767388
5.7	3.8	1.7	0.3	setosa	-0.17309407
5.1	3.8	1.5	0.3	setosa	-0.89767388

Chapter 3

Data Manipulation in R

```
# #============================================================================================
# # LSUserRs Data Cleaning Template (Title Your Script)
# # Say a couple of the things that it does here. Who wrote it?
# # When was the last edit? What does it do? Does it work with any data type?
# # Rubber duck this as much as possible because you won't remember
# # what you did in 6 months. Especially if you come up with something clever.
# #============================================================================================
# # TRY YOUR BEST TO NOT JUST COPY AND PASTE CODE, GOOGLE WHAT YOU WANT, GET FAMILIAR WITH A FUNCTION'S
# # ARGUMENTS AND EMBRACE YOUR INNER NERD AND READ THE DOCUMENTATION!!
# #============================================================================================
# # Import Libraries
# #--------------------------------------------------
# # Load all libraries at the start of your script, remember they have to be installed first!
# library(stringr)
# library(data.table)
# library(psych)
#
# #============================================================================================
# # Set up your working directory
# #--------------------------------------------------
#
# #============================================================================================
# # Load in your data
# #--------------------------------------------------
# # After telling R where to look in your computer/dropbox/google drive/R Project grab what you need!
# # Make sure to load in both datasets as we will want them both in our analysis.
# # We also want to make sure to clean both datasets as we are going.
# experiment.data <- read.csv("datasets/Demographic_Data.csv")
# item.level.data <- read.csv("datasets/ItemLevelData.csv")
#
#
# #============================================================================================
# # Inspect the structure of your data
# #--------------------------------------------------
# # R is going to do its best to guess what kind of data each of the columns of your spreadsheet are.
# # Sometimes it thinks things are lists (especially if importing from SPSS)
# # Go through each variable as you would with other programs and make sure you set it to what
```

```
# # you think will be most useful later. The big thing to check here is categorical variables, and
# # if you see any sort of string/character variables.
#
# # View(experiment.data) # Looks OK on surface levels....
# str(experiment.data) # Check to see if R guessed correctly on data types
#
# # Using read.csv() R had a couple of bad guesses on variables we might need.
# # We will have to reassign the variable types or else we'll run into trouble later.
# # In R we use Factor for grouping and analysis, best practice is to not set it as that
# # until you are OK with the format. It's easist to manipulate a character or string.
#
# experiment.data$inst <- as.character(experiment.data$inst)
# experiment.data$Gender <- as.character(experiment.data$Gender)
# experiment.data$Major <- as.character(experiment.data$Major)
# experiment.data$Minor <- as.character(experiment.data$Minor)
# experiment.data$BeginTrain <- as.character(experiment.data$BeginTrain)
# experiment.data$AbsPitch <- as.character(experiment.data$AbsPitch)
# experiment.data$Live12Mo <- as.numeric(experiment.data$Live12Mo)
# experiment.data$ActListenMin <- as.character(experiment.data$ActListenMin)
#
# str(experiment.data) # Notice how that our character columns now have " " around them.
#
#
# #=============================================================================================
# # Check for Import Errors
# #-----------------------------------------------
# # Use a combination of the names(), View(), table(), is.na(), and complete.cases() to get a brief sum
# # going on in your data set to be sure there were minimal import errors and your data looks like
# # you want it to. It might also be worth plottting some variables and use some common sense to find m
# # Are there any participants with 999 as their subject number? Negative values where there shouldn't
# # If there are, note them and fix these before starting any sort of statistical screening!
#
# table(complete.cases(experiment.data)) # Not all observations have everything!
# complete.cases(experiment.data)
# table(is.na(experiment.data))
#
# # Gotta decide what to do about it!!
#
# #=============================================================================================
# # Cleaning Free Text Response Data
# #-----------------------------------------------
# # In your data cleaning before you might have noticed that participants were able to freely respond
# # with whatever gender they wanted. Most data look to fall within the normal binary, but the computer
# # needs things to be exactly the same before making an easy split?
# # What would be the laziest, most effecient way to fix the gender column? What format does the variab
# # have to be in order to make the changes that you need?
# # When you have it figured out, make sure to run the code from top to bottom to make sure things go i
# # the right order!!! As we are not dealing with huge amounts of data, the table() function will help
#
# # Let's now take a look at some of these problem ones
# # Why, for example is Begin Training not working? Print the variable to see.
#
# experiment.data$BeginTrain
```

```
#
# # Some people didn't respond, one person decided to tell us what grade they started.
# # There are 2 ways to go about fixing this. We could "hardcode" the problem if this
# # is the only time we will do this analysis on this dataset or we could try to write a
# # line of code that doesn't care what exact position the error is.
# # On line 250 in this object is the thing that needs swapping out.
# # We can access it with R's indexing. Counting from index we see it's in line 250.
#
# experiment.data$BeginTrain[250]
#
# # Quick, ask yourself why we don't use the comma here?!
# # If you were set on using the comma, what would you change?
# # Ok, now let's swap in the value we want with <-
# # Remember we are putting a value into a character operator so it has to have ""
#
# experiment.data$BeginTrain[250] <- "12"
# experiment.data$BeginTrain
#
# # Nice, no more text data, but what if it's not always in 250?
# # For example, what do we do with all these blank spaces?
# # Let's use R's inbuilt ifelse() function to go through this vector and swap
# # out what we want!
#
# ifelse(experiment.data$BeginTrain == "","0",experiment.data$BeginTrain)
#
# # This works by going through each entry and doing the conditional on the value!
# # Let's now write over our old column and in the same step make everything a number.
# experiment.data$BeginTrain <- as.numeric(ifelse(experiment.data$BeginTrain == "","0",experiment.data$
#
# #Tah Dah!!
# experiment.data$BeginTrain
#
# # Let's now clean up the Gender column, first let's look at it
# experiment.data$Gender # Are there any common trends?
#
# table(experiment.data$Gender)
#
# # Pretty much two answers, how do we make them all say one thing?
# # Let's use the stringr package for this. Import it up top.
#
# # Clean Gender
#
# experiment.data$Gender <- str_to_lower(experiment.data$Gender)
# experiment.data$Gender <- str_replace(experiment.data$Gender,"^.*f.*$","Female")
# experiment.data$Gender <- str_replace(experiment.data$Gender,"^m.*$","Male")
# experiment.data$Gender <- str_replace(experiment.data$Gender,"^country$","No Response")
# experiment.data$Gender <- str_replace(experiment.data$Gender,"","No Response")
# experiment.data$Gender[30] <- "No Response" #Something Might Be Up w this datapoint?
#
# experiment.data$Gender <- as.factor(experiment.data$Gender)
#
# #-------------------------------------------------
# # Can we do same thing for AP?
```

```
# experiment.data$AbsPitch
# experiment.data$AbsPitch <- str_to_lower(experiment.data$AbsPitch)
# experiment.data$AbsPitch <- str_replace(experiment.data$AbsPitch,"^.*n.*$","no")
# experiment.data$AbsPitch[30] <- "no"
# experiment.data$AbsPitch <- as.factor(experiment.data$AbsPitch)
# table(experiment.data$AbsPitch)
#
#
# #=============================================================================================
# # Merging Data
# #-----------------------------------------------
# # Often we will have data from other spreadsheets we want to attach such as demographic data
# # to behavioral responses. Using the data.table functionality, let's merge our two csv
# # files together so that we have every variable accessible to us for this analysis.
# # Note I like to work with the data.table package, though there are other ways to do this!
#
# # In order to do this, we need 1 shared column between the two datasets.
# # For most psychology cases, this is probably going to be a participant ID number.
# # Note that for this to work, you need the columns to have an exact match of name!
# # First let's check that they are the same!!
# names(experiment.data)
# names(item.level.data)
#
# # First off our subject ID columns are not the same. Let's swap that.
# setnames(item.level.data,"tmp.dat.subject.1.","SubjectNo")
# setnames(experiment.data,"Sub","SubjectNo") # Make this clearer!!!
#
# # If you need to do more than 1, use the c() operator!
# # Now if we look at this column, it's all messe up.
# # The code below fixes it, if you want to learn more about regex, check it out
# # if not, just skip below.
#
# item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = ".csv", re
# item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "C", repla
# item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "M", repla
# item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "CM", repl
# item.level.data$SubjectNo <- as.numeric(item.level.data$SubjectNo)
#
# # Let's just quickly check to see if all the subject numbers make sense
# hist(experiment.data$SubjectNo) # Cause for alarm! Negative values and placeholders!
#
# # Drop those
# experiment.data <- experiment.data[experiment.data$SubjectNo > 0 & experiment.data$SubjectNo < 1000,]
#
# # Note this works because the SubjectNo variable is numeric
# hist(experiment.data$SubjectNo)
# hist(item.level.data$SubjectNo)
#
# # Ok, finally we merge our datasets. What we are doing here is called an "inner join"
# # Here we willkeep all of the ROWS of the dataset in the middle of the command
# # Note we need to swap over our key to be a character value.
# item.level.data <- data.table(item.level.data)
# experiment.data <- data.table(experiment.data)
```

```
#
# item.level.data
#
# exp.data <- item.level.data[experiment.data, on="SubjectNo"]
#
# exp.data
#
# # View(exp.data) # Use View to hover over column number
#
# # Let's reorganize our columns so individual stuff is at the back
# # We could do this with data.table, but it's a different syntax so let's swap back
# # Normally you try to stick to minimal switching, but we're just taking
# # a big tour du R right now and learning to think
# exp.data <- data.frame(exp.data)
# exp.data <- exp.data[,c(1,40:100,2:39)]
# View(exp.data)
#
# #=============================================================================================================
# # Checking for Univariate Outliers
# #-----------------------------------------------
# # For this example, let's imagine a univariate outlier is one with a zscore
# # greater than 3. While we could write a bit of code to look for this, let's use
# # the pairs.panels() function in the psych pacakge to just get used to looking at our data
# # The function is not the biggest fan of huge datasets, so let's index our
# # dataset to only grab what we need. Try to change the values and look
# # at variables of interest.
#
# pairs.panels(exp.data[,2:7], lm = TRUE)
#
# # But of course we need to look at numbers in terms of their zscores!
# # Let's first standardize our entire dataset using the apply function
# # Note we only can do this on numeric values!
#
# # The apply function takes 3 argument
# # The first is what you want to manipulate, the second is if it's rows 1 or columns 2
# # (remeber this because it's always rows then columns!), and the function.
# # You can even write your own (though we'll get to functions later)
# gmsi.z.scores <- apply(exp.data[2:7],2,scale)
#
# exp.data.with.z <- cbind(exp.data, gmsi.z.scores)
#
# # Now we can index this to find values above whatever theshold we want!
#
# table(gmsi.z.scores > 2)
# gmsi.z.indexer <- gmsi.z.scores > 2
# gmsi.z.scores[gmsi.z.indexer] # See what they are, find them , decide to get rid of
#
#
# #=============================================================================================================
# # Checking for Multivariate Outliers
# #-----------------------------------------------
# # A bit tricker, I leanred how to do this off a blog post.
# gmsi.responses <- exp.data[,c(63:100)]
```

```
#
# mahal <- mahalanobis(gmsi.responses,
#                      colMeans(gmsi.responses, na.rm = TRUE),
#                      cov(gmsi.responses,
#                          use = "pairwise.complete.obs"))                    ## Create Distance Measur
#
# cutoff <- qchisq(.999, ncol(gmsi.responses))                                ## Create cutoff object .
# summary(mahal < cutoff)                                                     ## 11 Subjects greater th
#
# # Add On Variables
# exp.data$mahal <- mahal
# exp.data <- data.table(exp.data) # To use easier indexer, needs data.table
# exp.data[exp.data$mahal < cutoff]
#
# #=============================================================================================================
# # Checking for Skew and Kurtosis
# #-----------------------------------------------
# apply(gmsi.responses, 2, skew)
# apply(gmsi.responses, 2 , kurtosi)
#
# #=============================================================================================================
# # Exporting Data
# #-----------------------------------------------
# # It's best practice to separate your cleaning and your analysis into separate scripts.
# # Export the dataset you have into a new csv file into a directory that would make sense to
# # someone who has never seen your project before.
#
# write.csv(exp.data,"My_Experiment_Data.csv")
# #=============================================================================================================
```

Chapter 4

Episotomology of Statistics

We describe our methods in this chapter.

Chapter 5

Descriptive Statistics, z Scores, Central Limit

5.1 Descriptive Statistics and the Normal Distribution

5.1.1 Organizing Data

Descriptive statistics are traditionally used to summarize data from observed samples. Most often, sample data are organized into distributions of information based on ascending scores. For example, we might have a table with some SAT-Verbal scores from a few different students. Before going on to think about this, also take note that the shape of this data (though very minimal) is in the tidy format. According to Hadley Wickham on the tidyr CRAN page for data to be tidy it must have the following properties:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

This isn't very important right now, but once you get to more complex designs, it will be good to have had thought about this before.

```
student <- c(1,2,3,4,5,6)
SAT <- c(480,530,560,650,720,760)
satData <- data.frame(student,SAT)
satData
```

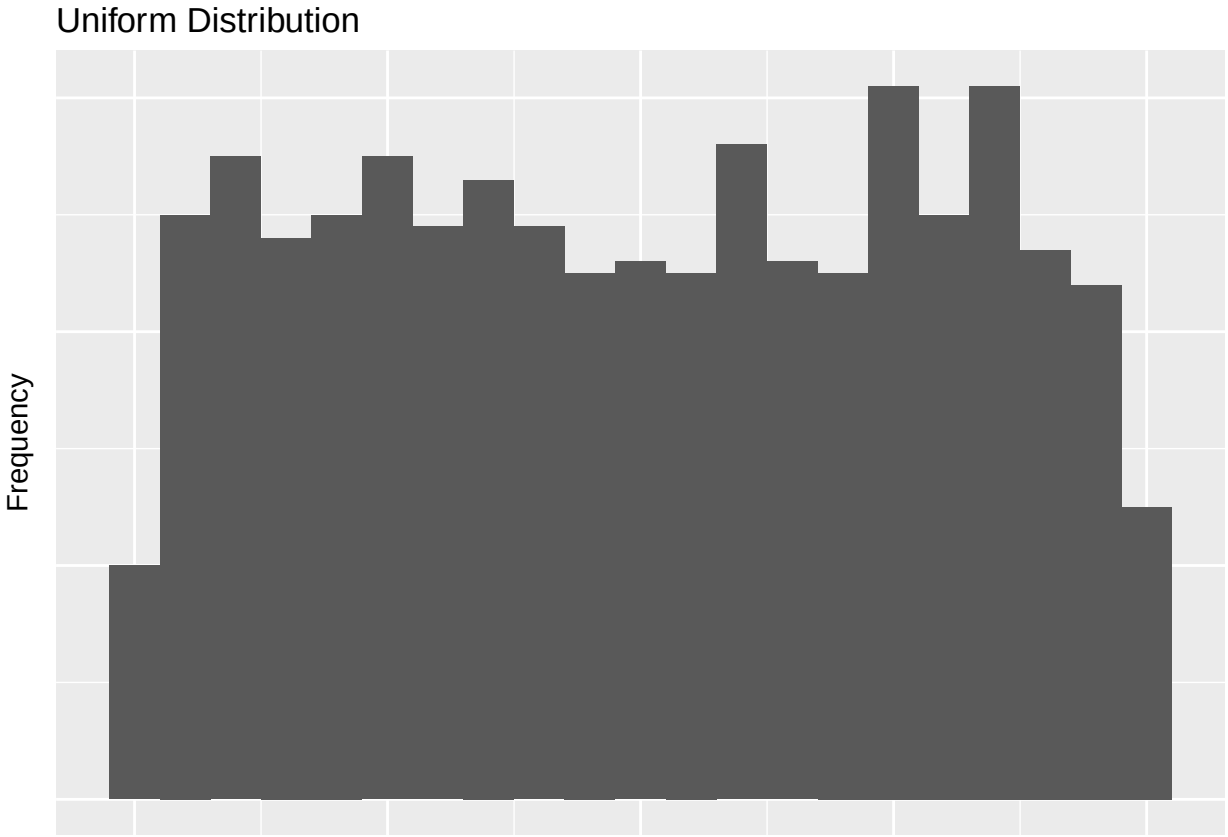
```
##   student SAT
## 1       1 480
## 2       2 530
## 3       3 560
## 4       4 650
## 5       5 720
## 6       6 760
```

5.1.2 Shape of Data

When visualized, data can take on a variety of shapes. Below are a few of the shapes you might come across when analyzing data.

The first, and probably least likely distribution you will find is the **uniform** or **rectangular distribution**. We can create this plot and the others by using the distribution functions from R's functionality. In each case we are going to take 1,000 samples from 0 to 1. We'll plot everything using `ggplot2` so we can also get used to using it for our packages.

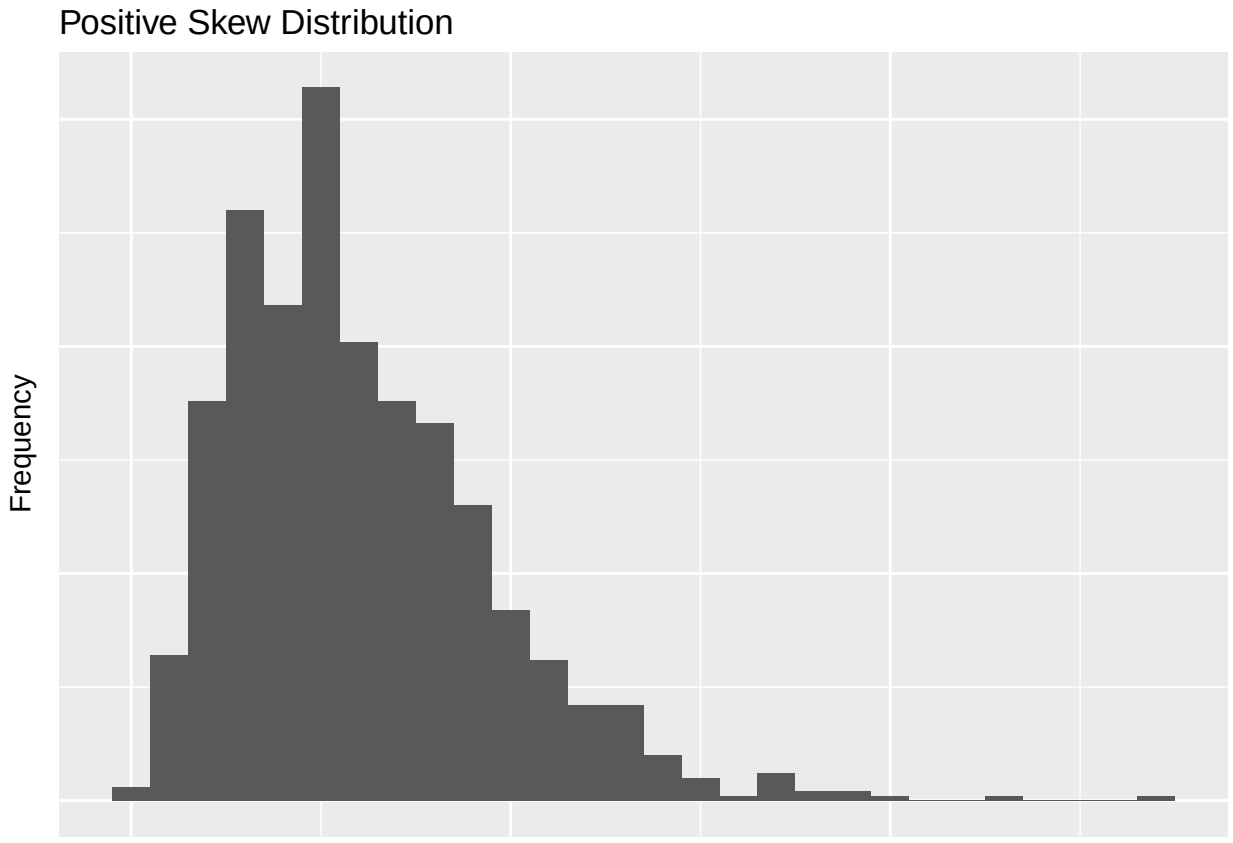
```
library(ggplot2)
set.seed(666)
uniformData <- runif(n=1000, min=0, max=100)
distributions <- data.frame(uniformData)

# Make variable to remove ggplot elements
cleanUpPlots <-  theme(axis.title.x=element_blank(),
                   axis.text.x=element_blank(),
                   axis.ticks.x=element_blank(),
                   axis.text.y=element_blank(),
                   axis.ticks.y=element_blank())

ggplot(distributions, aes(distributions$uniformData)) +
  geom_histogram(binwidth = 5) +
  labs(x = "Independent Variable", y = "Frequency", title = "Uniform Distribution") + cleanUpPlots
```



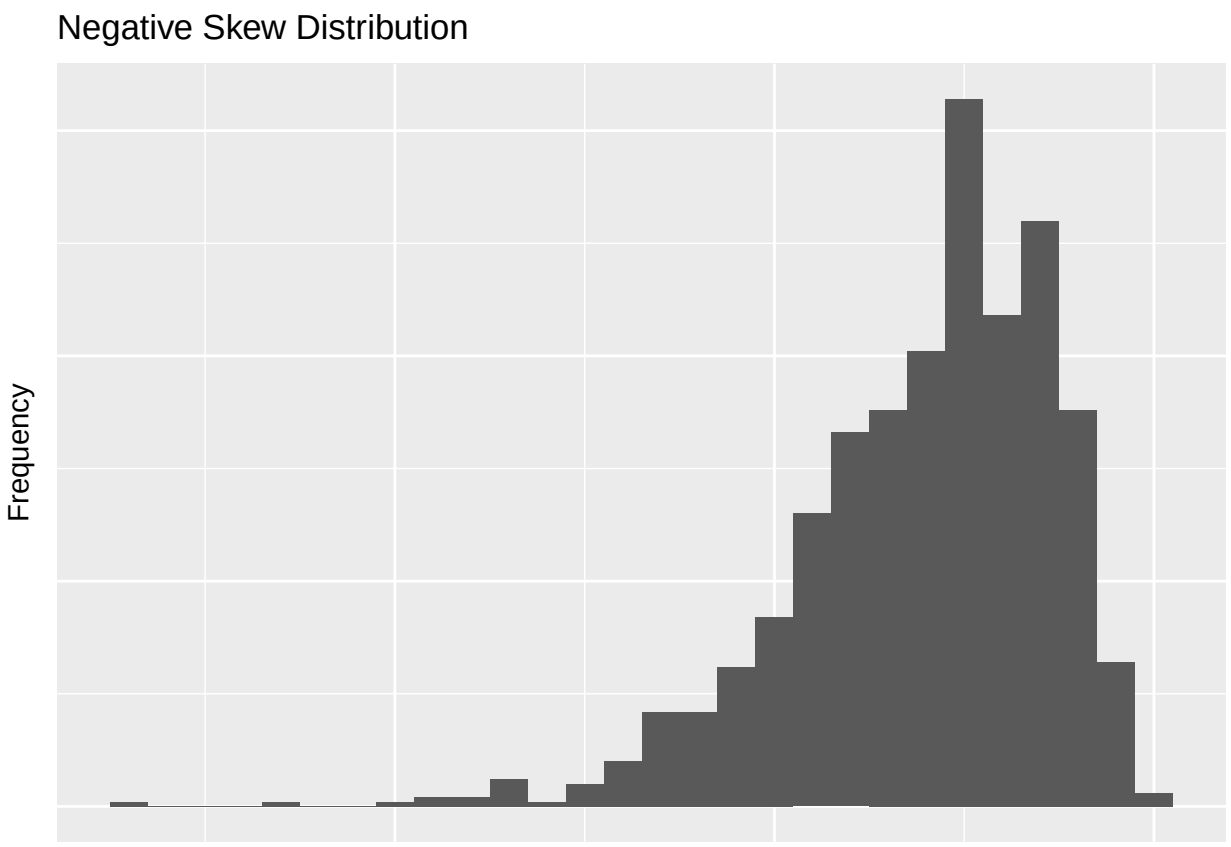
Try to run the above code with different arguments in the `binwidth` argument. You'll notice that the way you plot the data will actually represent it differntly.

We can also have **positively skewed** and **negatively skewed** distributions. If a distribution is skewed, it usually means that the **mode** does not equal the **mean**. We're going to approximate both of these with another one of R's probability functions.

```
positiveSkewData <- rchisq(1000,6) # Chi Square distributions are positively skewed
negativeSkewData <- - positiveSkewData # Flip it!
distributions <- data.frame(uniformData, positiveSkewData, negativeSkewData)
ggplot(distributions, aes(distributions$positiveSkewData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Positive Skew Distribution") +    cleanUpPl
```

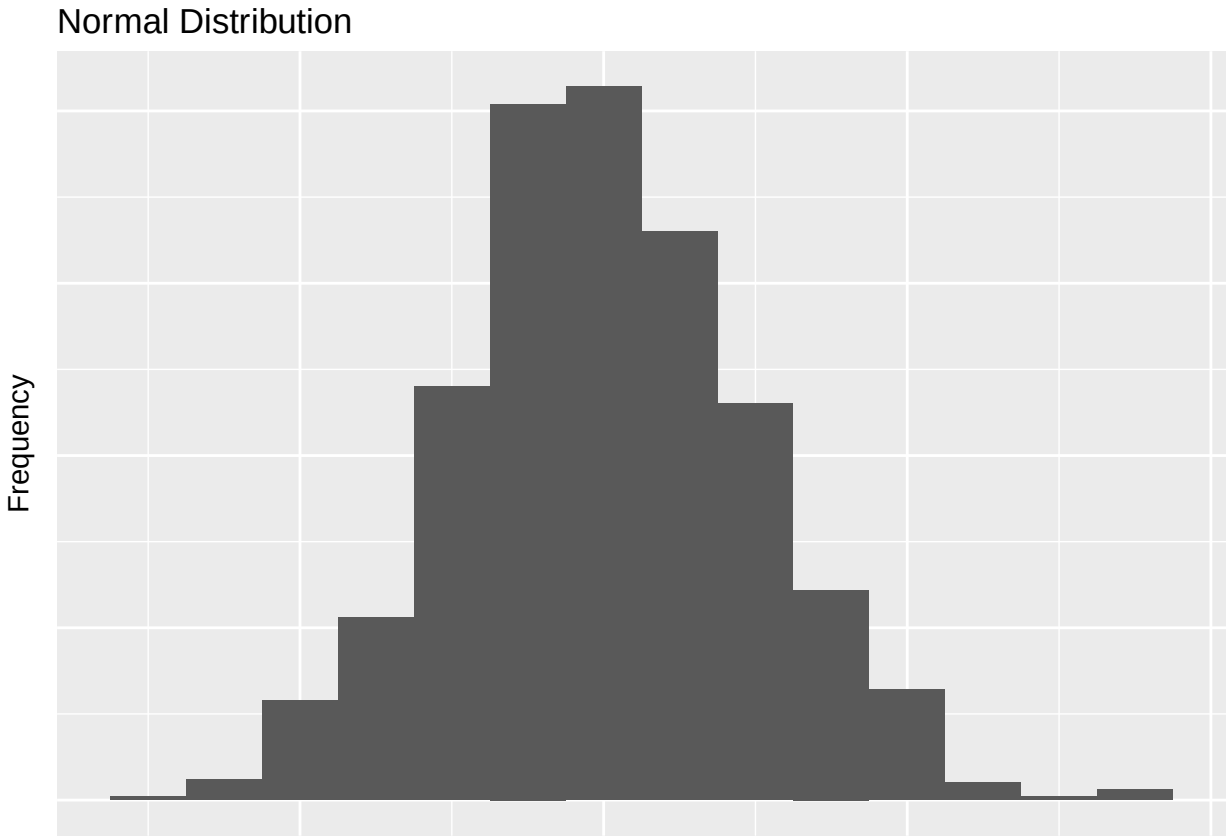



```
ggplot(distributions, aes(distributions$negativeSkewData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Negative Skew Distribution") +   cleanUpPl
```



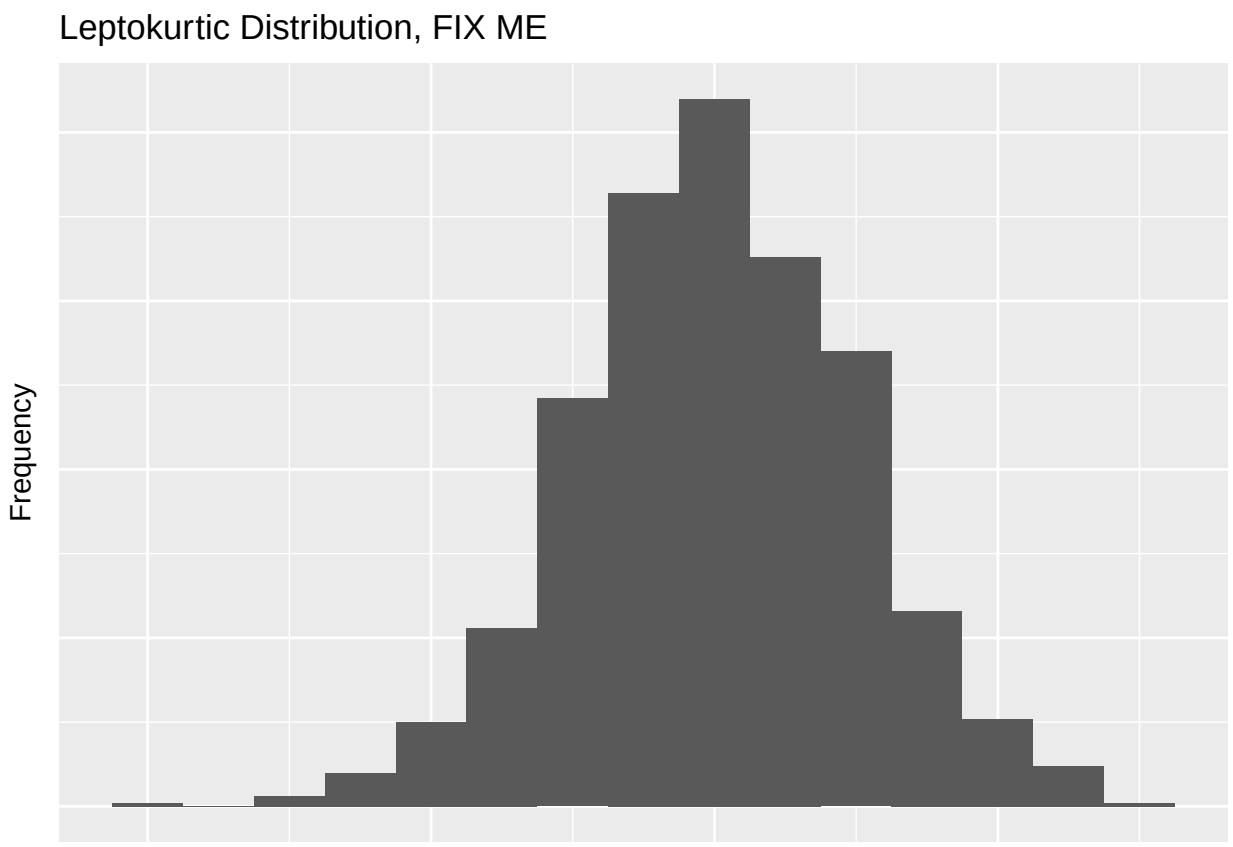
The most important distribution in the world of Frequentist statistics is a **normal distribution**. A normal distribution is defined by THIS HERE.

```
normalData <- rnorm(n = 1000,mean = 0,sd = 2) # Flip it!
distributions <- data.frame(uniformData, positiveSkewData, negativeSkewData, normalData)
ggplot(distributions, aes(distributions$normalData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Normal Distribution") +    cleanUpPlots
```

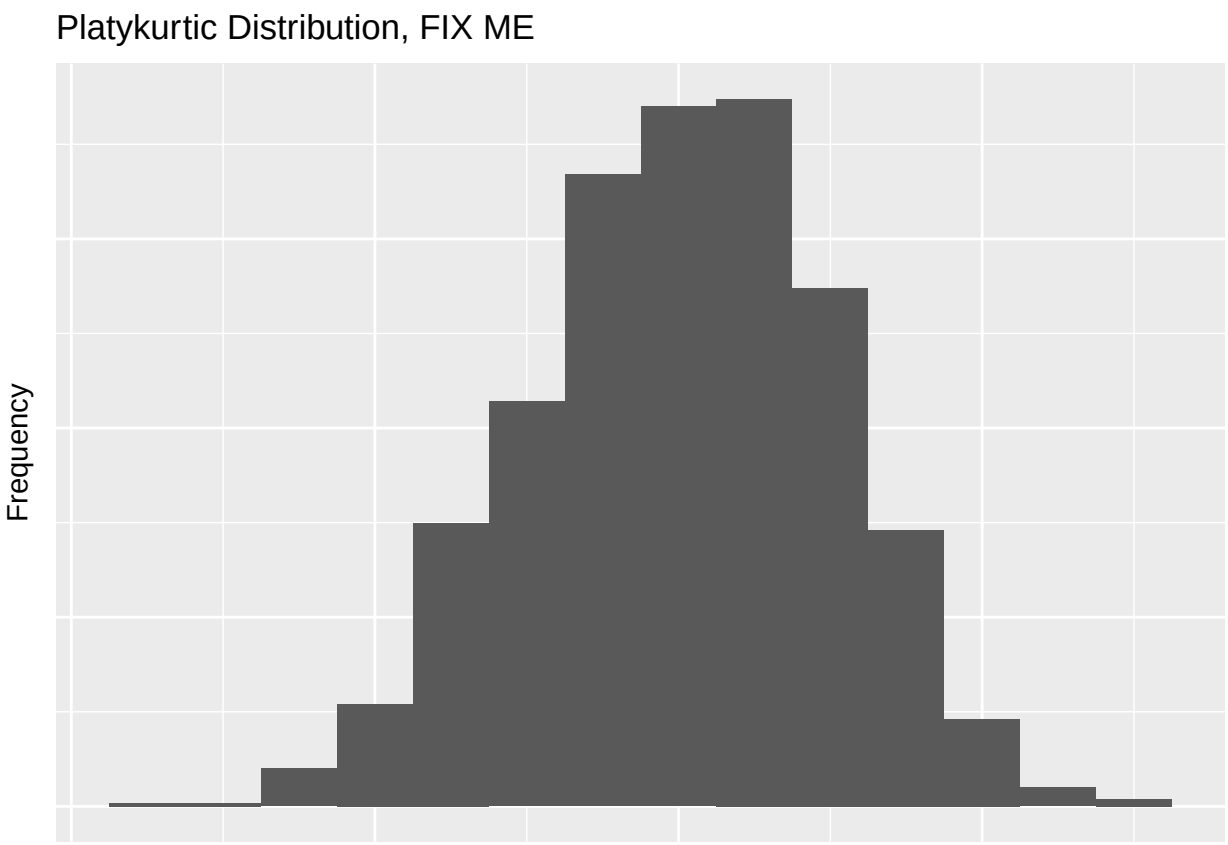


And the lastly we can have both **leptokurtic** and **platykurtic** distributions.

```
leptoData <- rnorm(n = 1000,mean = 0,sd = 2)
platyData <- rnorm(n = 1000,mean = 0,sd = 2)
distributions <- data.frame(uniformData, positiveSkewData, negativeSkewData, normalData,leptoData, platy
ggplot(distributions, aes(distributions$leptoData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Leptokurtic Distribution, FIX ME") +    cl
```




```
ggplot(distributions, aes(distributions$platyData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Platykurtic Distribution, FIX ME") +     cl
```



Generally measurses of **central tendency** are used to characterize the most typical score in a distribution.

For example we could calculate the **mean** or the **median** of our SAT data. The mean is calculated by adding up all our numbers, designated with the Greek letter Sigma Σ then dividing by the amount of numbers we added up, or n. As an equation it would look like this.

$$\bar{X} = \frac{(\Sigma\ x_i)}{n}$$

Take a second to talk yourself through that so later you will start to feel more comfortable with more complex equational notation. Some people find it helpful to just try to say the equation in plain English. In this case it would be the mean, or $\bar{x}$ is defined as or equal to what happens when you add up Σ every single value x that you have going up to i, then divide all those numbers by the amount of numbers you have, or n.

The median is defined by finding the middle number. If there is a tie because we have an even set of numbers, we take the mean of the middle two numbers.

We can do both of these in R as well. Below we can either type in the numbers as you would with a calculator, or use a function in R. Notice that each step when its typed out is saved into an object. By starting to think this way, it will pave the way for writing more elegant code later on.

```
# Typing it out
our.data <- c(480 + 530 + 560 + 650 + 720 + 760)
how.many <- length(our.data)
our.data/how.many
```

```
## [1] 3700
```

```
# Inbuilt functions
mean(satData$SAT)
```

```
## [1] 616.6667
```

```
median(satData$SAT)
```

```
## [1] 605
```

Notice here that for adding up the means by hand I could have done what programmers call hard coded the equation in. That would have looked like this.

```
our.answer <- c(480 + 530 + 560 + 650 + 720 + 760) / 6
our.answer
```

```
## [1] 616.6667
```

The problem with this, is that every time you get a new SAT score you have to both enter the score and update how many scores you are dividing by. Whenever you see a chance to take a shortcut like this, do it! It will save you tons of time in the future.

5.1.3 Important Considerations for Central Tendency

There are three big considerations to think about when choosing numbers to represent your data.

1. The mode is the most variable from sample to sample; the mean is the least variable
2. The mean is the most sensitive to extreme scores; e.g., skewed distributions
3. The mean is the most frequently used measure because *The sum of the deviations around the mean is 0* The sum of the squared deviations is the smallest around the mean, rather than the mode or median; this is known as the least squares principle

5.1.4 Measures of Variability

The **range** is simply the largest score minus the smallest score. It is the crudest measure of variability. In our dataset we would calculate it with the following code.

```
760 - 480
```

```
## [1] 280
```

```
# OR
range(satData$SAT)
```

```
## [1] 480 760
```

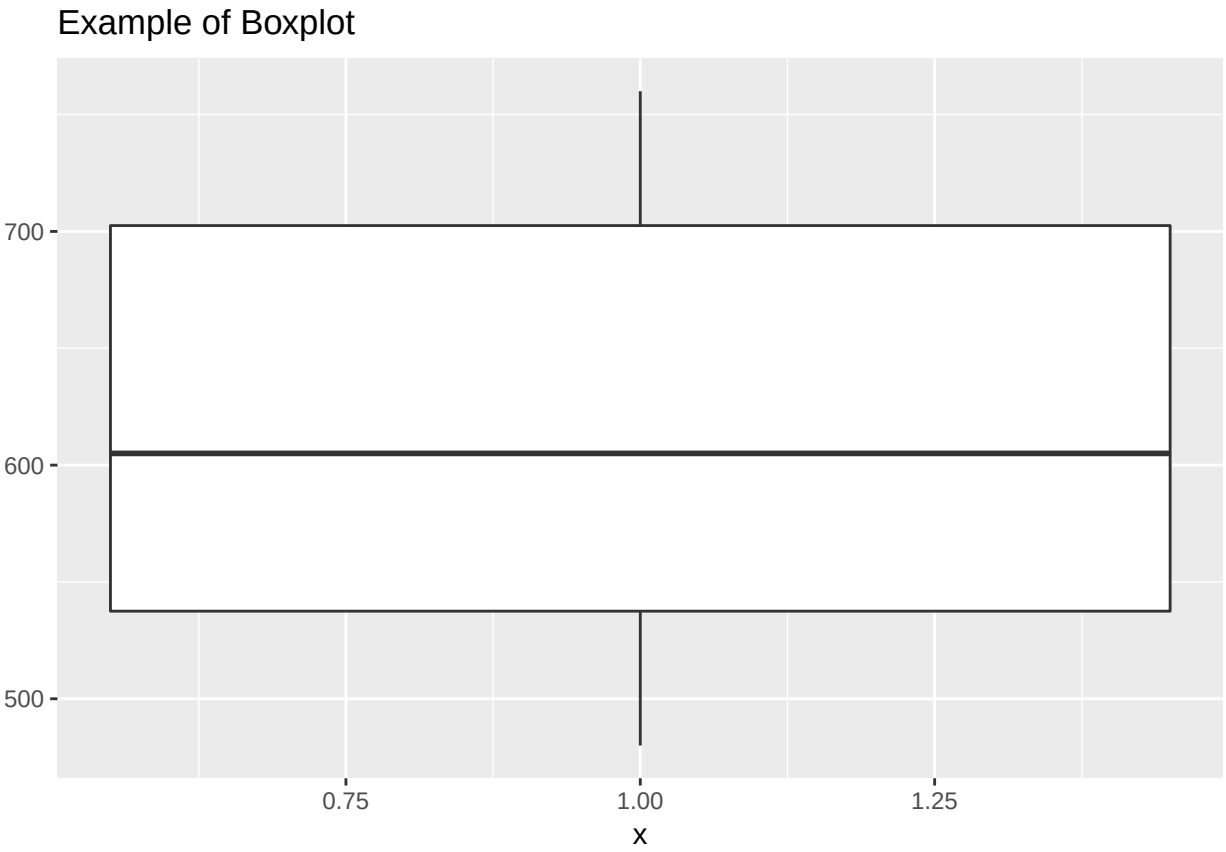
```
max(satData$SAT) - min(satData$SAT)
```

```
## [1] 280
```

The interquartile rangerepresents the spread between the score at the 75th and 25th percentiles. Boxplots are often used to graphically represent inner 50% of the scores.

```
y <- satData$SAT
boxplot.example <- data.frame(
          x = 1,
          y0 = min(y),
          y25 = quantile(y, 0.25),
          y50 = median(y),
          y75 = quantile(y, 0.75),
          y100 = max(y)
)
```

```
ggplot(boxplot.example, aes(x)) +
    labs(title = "Example of Boxplot") + geom_boxplot(aes(ymin = y0, lower = y25, middle = y50, upper =
    stat = "identity")
```



The **variance** is essentially the averaged squared deviation around the mean. Now there is a very important distinction that we will get to a bit later on, but that is the difference between a **population** value or σ^2 and a **sample** variance or s^2. In order to do frequentist statistics, we need to assume that there is some sort of True value that the group we are measuring has and it is a fundamental property of the group! For more on this see CHAPTER 3 and the work of Zoltan Dienes. Somewhere, maybe written on a golden plate in heaven is the actual value of the average weight of a labrador retriever. The problem is we will never have access to that information so we need to estimate it by using a sample. The logic is that if we can truly draw in a random way from our entire population, in this case labrador retrievers, the central limit theroum will give us a good approximation of what that True value will be. Since we want to be clear about when we are talking about the Platonic, True value and the actual sample we collected, we use different Greek notation. The σ^2 refers to the Platonic value and the σ^2 is the sample. They are defined as follows:

$$\sigma^2 = \frac{\Sigma(X_i - \mu)^2}{N}$$

$$s^2 = \frac{\Sigma(X_i - \mu)^2}{n - 1}$$

Note here that each of these formulas needs a mean. In the population equation that is defined as μ and in samples we used $\bar{X}$.

In our case with the SAT scores, we are wanting to know the True value of the SAT scores of whatever population our six students are theorized to come from. To do the calulations below we need to know the

mean which we calcualted above to be 616.67.

Now since these scores are to serve as a represntive **sample** in hopes of getting at the true population value we need to use the formula reflecting the **sample variance** or s^2.

$$s^2 = \frac{(480 - 616.7)^2 + ... + (760 - 616.676)^2}{6 - 1}$$

Doing this by hand we get an s^2 value of 12346.67. Or running it in R, we would use.

```
var(satData$SAT)
```

```
## [1] 12346.67
```

The **standard deviation** is the square root of the variance of the sample.

$$s = \sqrt{\frac{\Sigma(X_i - \mu)^2}{n - 1}}$$

And since we know s^2 from above, we can shorten this to

$$s = \sqrt{s^2} = \sqrt{12345.67} = 111.12$$

Or run it in R and get

```
sd(satData$SAT)
```

```
## [1] 111.1156
```

Standard scores represent the distance of raw scores form their mean in standard deviation units.

$$z = \frac{x_i - \bar{X}}{s}$$

So if we needed to find the z score or standardized score for someone who got a 560 on their SAT we could compute the following.

$$z_{560} = \frac{560 - 616.67}{111.12} = -0.51$$

Interpreted in-context, this would mean that if you scored a 560 on the SAT, based on our sample (which we think helps us get at the True popuation value), you would be scoring about less than 1 standard deviation (the unit of z) below the average.

5.1.5 Properties of z Scores

Z scores are defined by having having three separte properties:

1. The standardized distribution preserves the shape of the original raw score distribution
2. The mean of the standardized distribution is always 0
3. The variance & standard deviation are always 1

Many of the variables in behavioral sciences are distributed normally. In addition, the basis for parametric inferential statistics is based on the normal distribution.

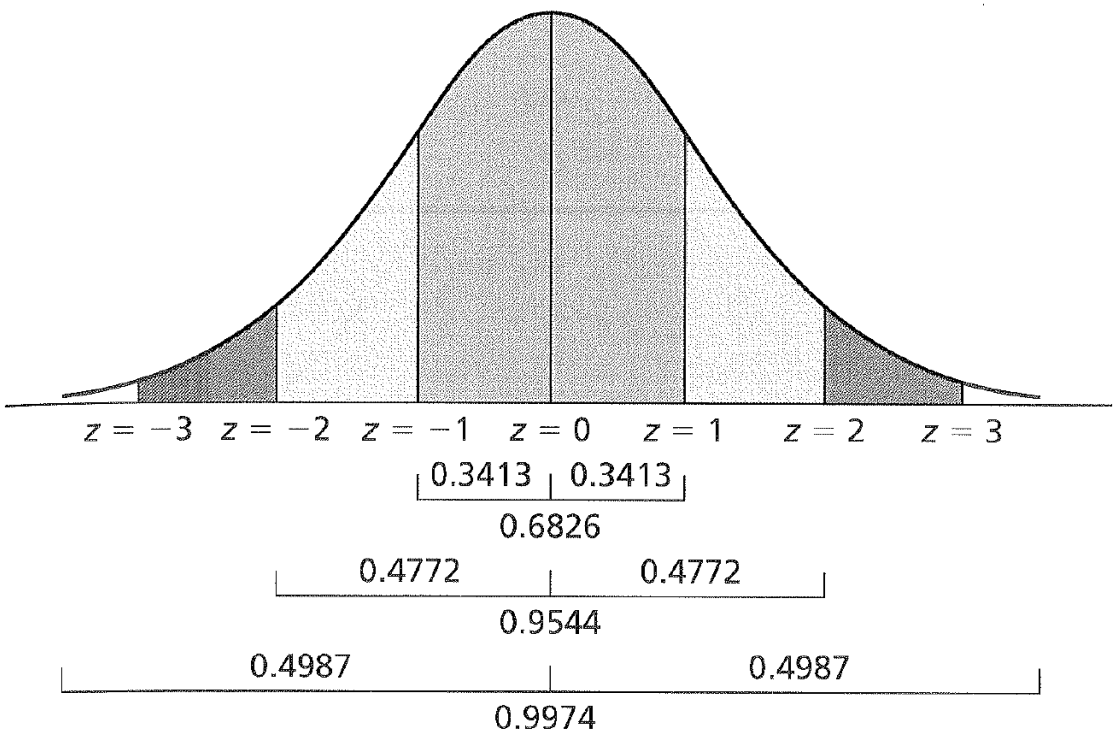


Figure 5.1: z Scores and Areas Under the Curve

The normal distribution is unimodal, symmetrical, bell shaped, with a maximum height at the mean. The normal distribution is continuous and additionallythe normal distribution is asymptotic to the X axis—it never actually touches the X axis and theoretically goes on to infinity.

The normal distribution is really a family of distributions defined by all possible combinations of means μ and standard deviations σ. We can use the standardized normal distribution to find areas of probability under the curve.

With a normal distribution, there is always a fixed area under the curve which we take the reflect the probability of getting a score when sampling from a population.

For example if we go 1 z unit (1 SD) away from the mean we find 34% of the total area of the curve there. If you then extend that out to the negative side, you then encapsulate 68% of the distribution. This would translate to a scenario where if you were to get a score at random from the distribution, 68% of the time you would get a score between 1 and -1 SD units from your mean.

This process can be extended as seen in the figure below.

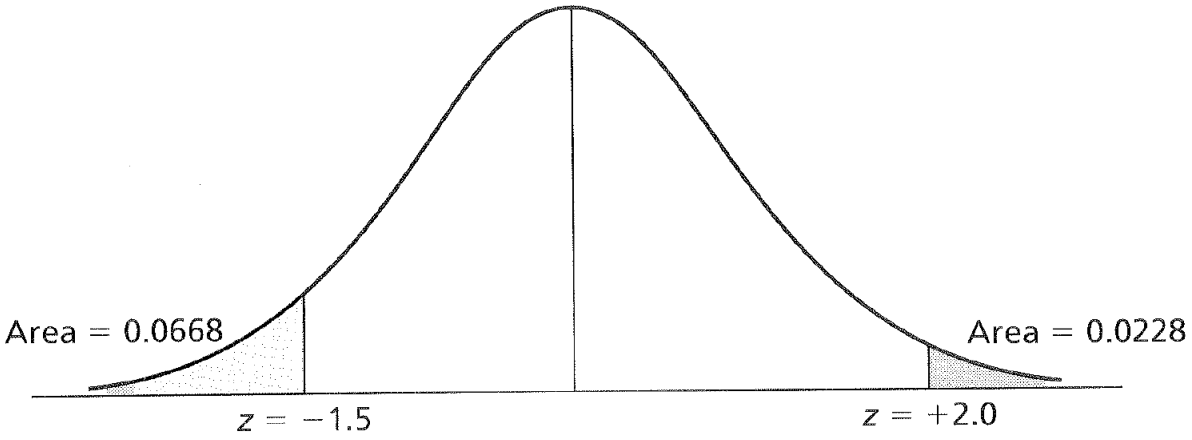
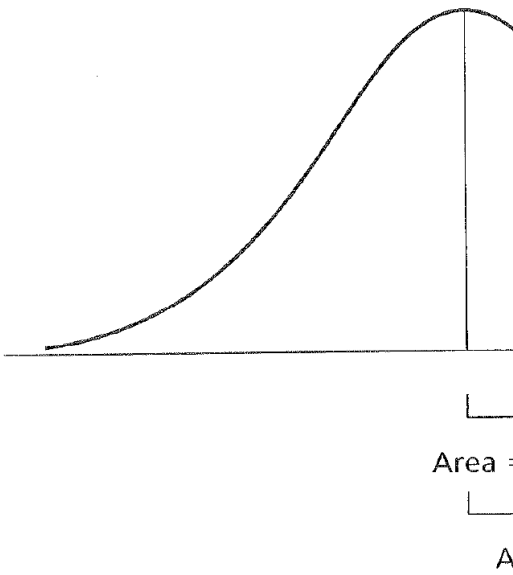


Figure 5.2: z Scores and Areas Under the Curve



We could also calculate the area between two z scores as shown here.

And we could also look at how much area under the distribution exists beyond two standard deviations beyond the mean.

Or we could pick any z score values and find the area under the mean!

5.2 Practice

We can now start to put this to use. Here are some past homework examples.

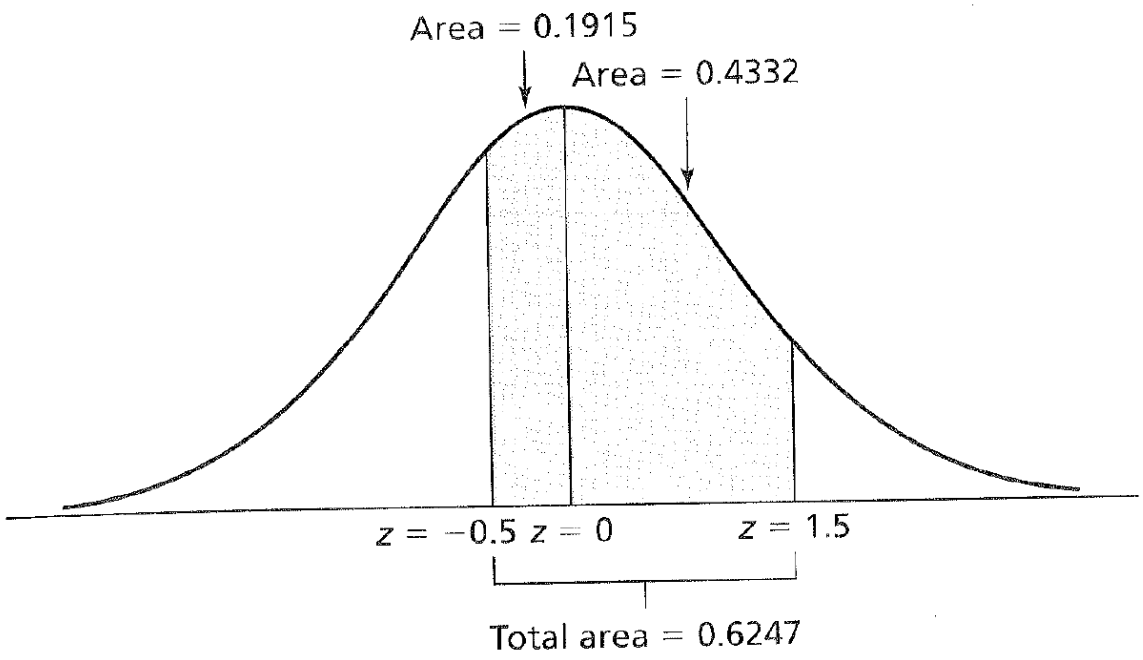


Figure 5.3: z Scores and Areas Under the Curve

During tryouts, a sample of ballet dancers were rated on their athletic ability andoverall knowledge of the art. Below are the ratings for each dancer (a score above 75 percent means that the dancer will join the troupe

83, 98, 45, 69, 52, 94, 82, 74, 71, 83, 62, 85, 90, 97, 61, 74, 74, 88

Let's put them into R so we can use answer a few questions about our data.

```
ballet <- c(83, 98, 45, 69, 52, 94, 82, 74, 71, 83, 62, 85, 90, 97, 61, 74, 74, 88)
```

What is the median percentage?

```
median(ballet)
```

```
## [1] 78
```

What is the mean percentage?

```
mean(ballet)
```

```
## [1] 76.77778
```

What is the standard deviation for the sample (assume we don't know any population characteristics)?

```
sd(ballet)
```

```
## [1] 15.02373
```

Demonstrate the least squares principle by showing that the sum of squares (SS) around the mean is smaller than the sum of squares around the median (remember to show your work for each).

```
ballet_mean <- mean(ballet)
ballet_median <- median(ballet)
ballet_sd <- sd(ballet)
```

Now we can use these values to do our math!

```
sum((ballet - ballet_mean)^2)
```

```
## [1] 3837.111
```

```
sum((ballet - ballet_median)^2)
```

```
## [1] 3864
```

```
# Then having R do the final work for us
sum((ballet - ballet_mean)^2) > sum((ballet - ballet_median)^2)
```

```
## [1] FALSE
```

What are the standardized (z) scores for the raw scores 73, 99, and 66? If you know the population mean and the sd, you can calulate a z score using the formula

$$z = \frac{x_i - \bar{X}}{s}$$

Or in our case

```
(73 - ballet_mean)/ ballet_sd
```

```
## [1] -0.2514541
```

```
(99 - ballet_mean)/ ballet_sd
```

```
## [1] 1.479142
```

```
(66 - ballet_mean)/ ballet_sd
```

```
## [1] -0.7173837
```

What proportion of scores exceeds a raw score of 73?

```
pnorm(q = 73, mean = ballet_mean, sd =  ballet_sd)
```

```
## [1] 0.4007315
```

To get the other side of the probability we can remember that we can treat the line above as an object!

```
1 - pnorm(q = 73, mean = ballet_mean, sd =  ballet_sd)
```

```
## [1] 0.5992685
```

What proportion of scores lies between the raw scores of 75 and 100? Let's be clever for this one and just put the two equations together for this one. Or if you want, you could save them into objects.

```
pnorm(q = 100, mean = ballet_mean, sd =  ballet_sd) - pnorm(q = 75, mean = ballet_mean, sd =  ballet_sd
```

```
## [1] 0.4860093
```

```
pnorm(q = 76, mean = ballet_mean, sd =  ballet_sd)
```

```
## [1] 0.479356
```

What raw score represents the 55thpercentile? To find out what raw score represents a percentile we can go back and use the formula from above, just rearranged a bit.

$$z = \frac{x_i - \bar{X}}{s}$$

or with a bit of basic algerbra

$$x_i = (z * s) + \bar{X}$$

```
(.05 * ballet_sd) + ballet_mean
```

```
## [1] 77.52896
```

Between what raw scores does the middle 60% of the distribution lie?

Lastly, we then need to find first what z scores map on 30% on either side of the disribution, then convert those z scores to raw scores on our data using the z score formula.

First we find the z score associated with what is 30% left and right of the mean (it will be the same number, only negative). In this case, it is +/-.84.

With that established, we then first solve for x

$$-0.84 = \frac{x - 76.78}{15.02}$$

Giving us a value of 64.1 And we do it again with the positive number.

$$0.84 = \frac{x - 76.78}{15.02}$$

Resulting in 89.547.

Chapter 6

Sampling Distributions

In this chapter, we'll cover three ideas/questions.

1. What are inferential statistics and the logic behind them
2. What the underlying distribution of all hypothetical sample estimates is known as the sampling distribution, and it constitutes the third of the three important distributions.
3. Several important implications follow from an understanding of the sampling distribution as a normal distribution and from the central limit theorem

Spoken about a bit before in the other chapter, we have both sample statistics like $\bar{X}$ and population parameters μ.

The idea of how frequentist inferential statistics is as follows. Samples must be selected *randomly* in order to make appropopriate inferences about the parent population. Sample estimates must be compared to an underlying distributionof estimates of all other hypothetical samples of that same sizefrom the parent population. Based on this comparison and the associated probability of obtaining certain outcomes, inferences can be made about population parameters.

It's important to note that there are three different distributions that we typically talk about. Two you should be familiar with – the populatation and the sample. The third is the **sampling distribution** which is a **distribution of sample means**. The sampling distribution of the mean is generated by considering all possible sample means of a given sample size.

As is demonstratd from the image below, in A we can see there is some sort of distribution, then with one sample (notice the $\bar{X}$), we now have one wide sample. As we increase that to $N = 16$, the sampling distribution becomes more narrow. This narrowing is reflective of the idea we are coming in on the true value of the popula-