

# R for Psych Handbook

*David John Baker*

*2018-05-04*



# Contents



# Chapter 1

## Preface

Welcome to the **R For Psychologists Handbook!**. This book serves as a collection of resources used in the Department of Psychology at Louisiana State University (LSU) for the statistics classes required of all Graduate Student. The book is **not** meant to serve as a comprehensive statistics textbook, for that please check out either Explaining Psychological Statistics or Andy Field's Discovering Statistics with R. The book **is** meant to be a reference manual that gives a brief overview of a concept, then the code needed to run that analysis in R.

I try to add in references in each of the chapters in order for anyone to explore the concepts at length, but the material in this handbook should be enough to get any psychologist through the 4111 Intermediate Statistics and 7111 Multivariate Statistics courses.

The majority of the content comes from having taken and then TA'ed for both of these courses and collecting a lot of material on my own. The content was generated by both Jason Hicks and Jason Harman, I just provided R as the glue to hold it all together. The idea behind the book is that anyone going through the LSU stats rotation would be able to put in the extra work and learn R while they are doing the assignments in SPSS.

As noted in next chapter, it's hard to learn R, but worth doing it. Personally found that R has opened up more opportunities than anything else I have invested in. You don't need any programming experience to start and if you feel like you can't do this, know that reason I started R is I knew so little about computers during my Masters that my adviser 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 something good 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.

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

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 re-inspired before R starts paying you back.

1. The R community is fantastic, check out `#rstats` on Twitter as well as everyone affiliated with the TidyVerse
2. R will always be free because the people behind it believe in open source principles.
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 transfer 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. It's also insane to think people would learn so much about statistics, the hardest part about becoming a data scientist, without learning the software to get you in the door.
6. When you make analyses and graphs in R they are very easy to reproduce. You just press 'Run' again.
7. If you do your data cleaning in R, then each step is documented. There is less chance for human error.
8. It makes gorgeous graphs.
9. There are a lot of ways that R integrates into other software. This book is written in bookdown, my website is written in blogdown, you can also make interactive data applications.
10. It's kind of fun.

## 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. If you use R without RStudio, you are basically a masochist.

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

## 2.2 A Note on 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 entire act of picking where you are working right away by saving your

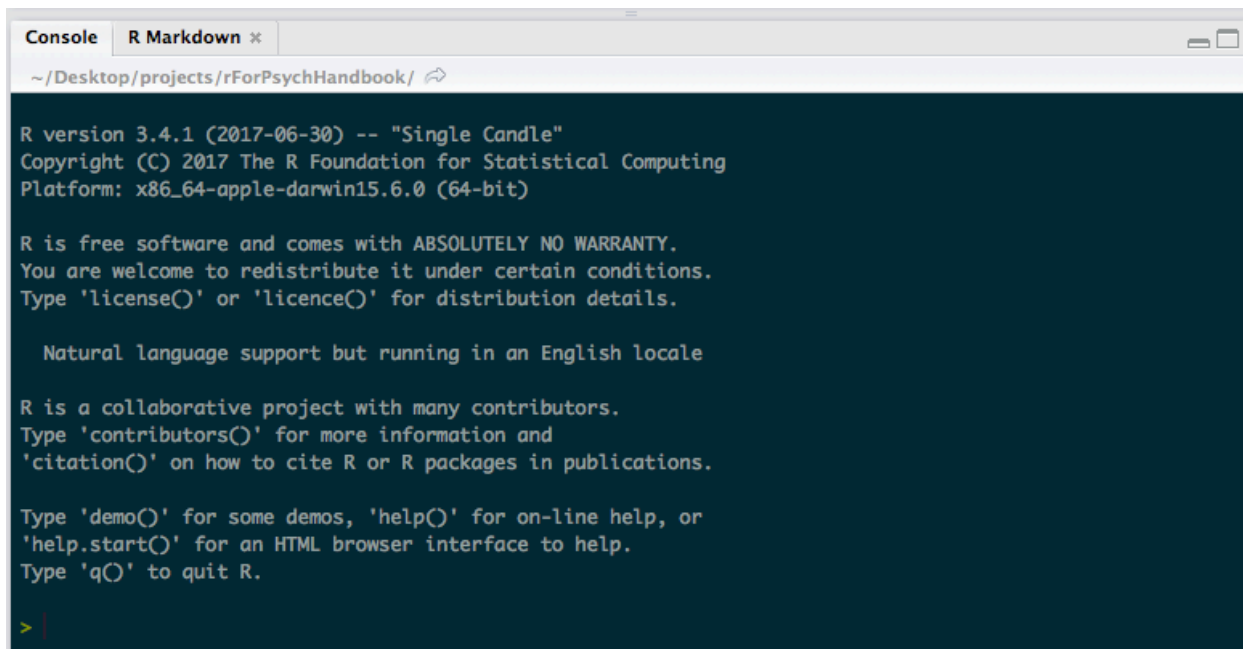


Figure 2.1: Current Working Directory Location

script (the what) and setting your working directory (the where). After running a couple of “teach yourself R” sessions, I have found that this concept is one of the most tedious parts of starting to learn R, but once you are over this hump, the others will not feel as bad. While these instructions below are meant to be comprehensive, it’s much easier to learn this part of the process with someone walking you through the steps.

When you first open R, it’s good to get in the habit of seeing where your working directory is set to. If RStudio is set on it’s default settings, there should be a little line right above the console just like the image below.

This whole book exists in a folder that lives in the projects folder on my Desktop. If you’re not familiar with how your computer is set up, it’s worth checking out an explanation or having someone walk you through the hierarchy your computer’s directories.

Once you have an basic understanding of this, you can find where you are in R by using the `getwd()` function. If you need to change your working directory at any point, the function is `setwd()`. While it might seem a bit tedious to have to think about things like this right away, by setting up analysis projects in a clean, and organized way you are going to be saving yourself hours in the future.

So with that, open up a new R script, and for your first line set your working directory to a folder where you are going to want to work out of.

## 2.3 The Basics

With this script set up, we can now start to play around with R! 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

## 2.4 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

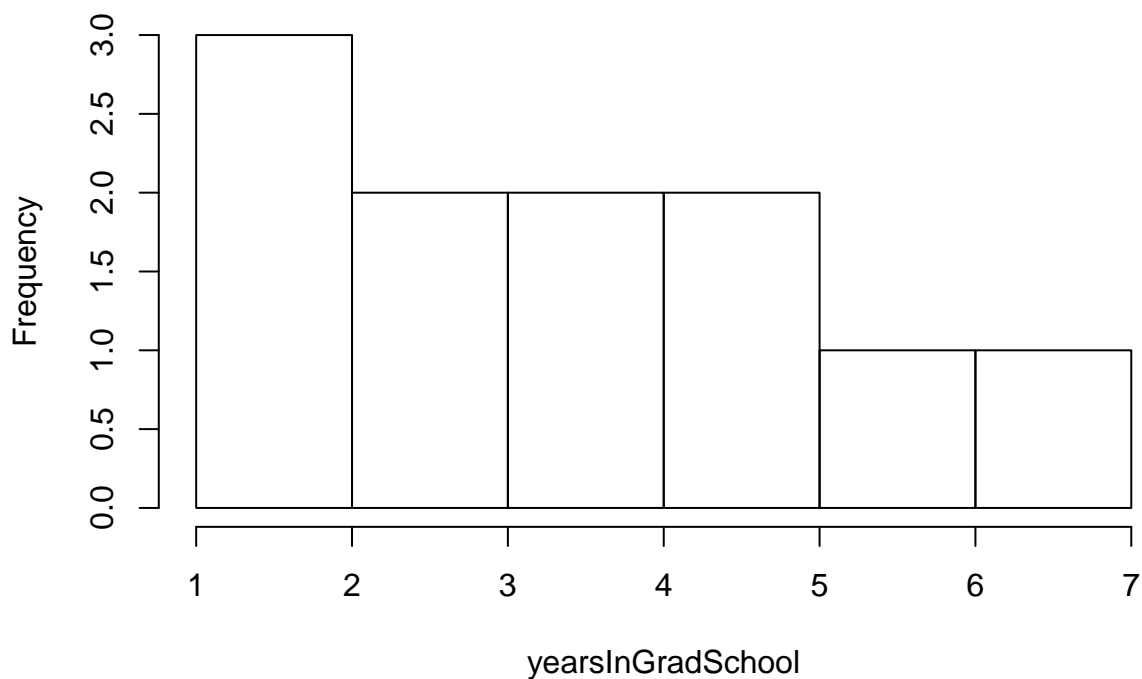
## [1] FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
mean(yearsInGradSchool)

## [1] 3.818182
sd(yearsInGradSchool)

## [1] 1.834022
hist(yearsInGradSchool)

```

**Histogram of 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
```

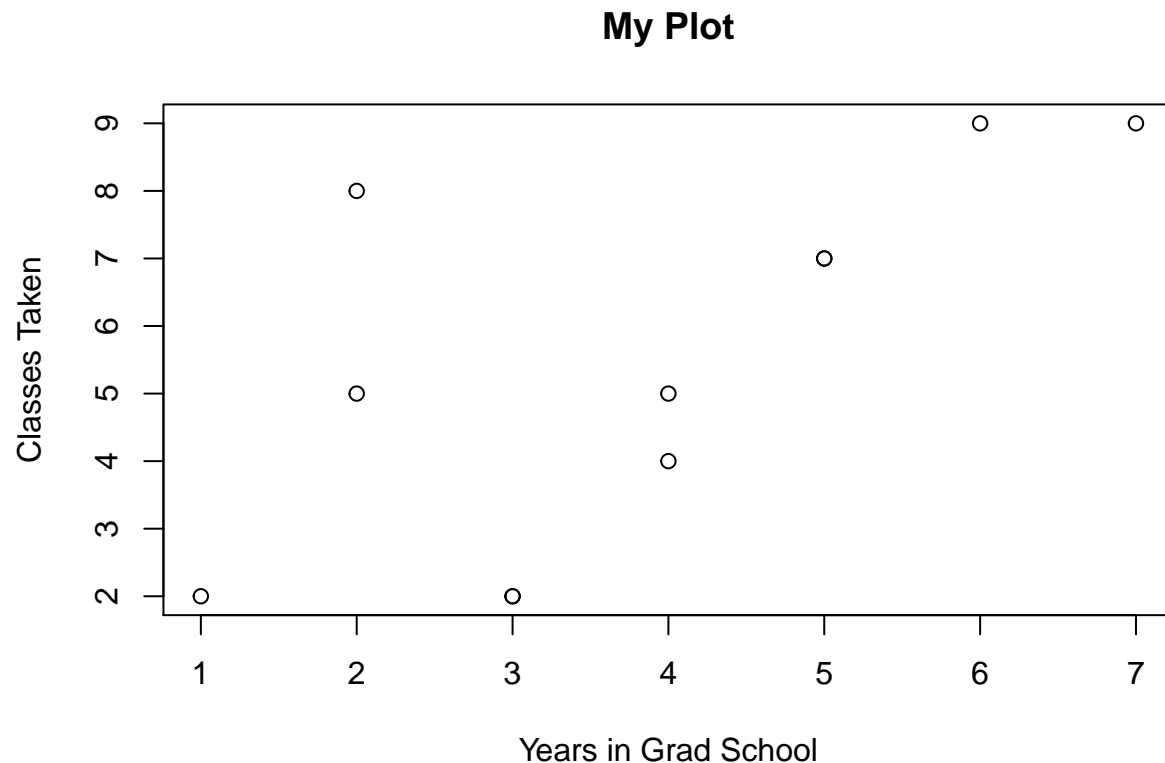
```
## Warning in plot.xy(xy, type, ...): "data" is not a graphical parameter
```

```
## 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

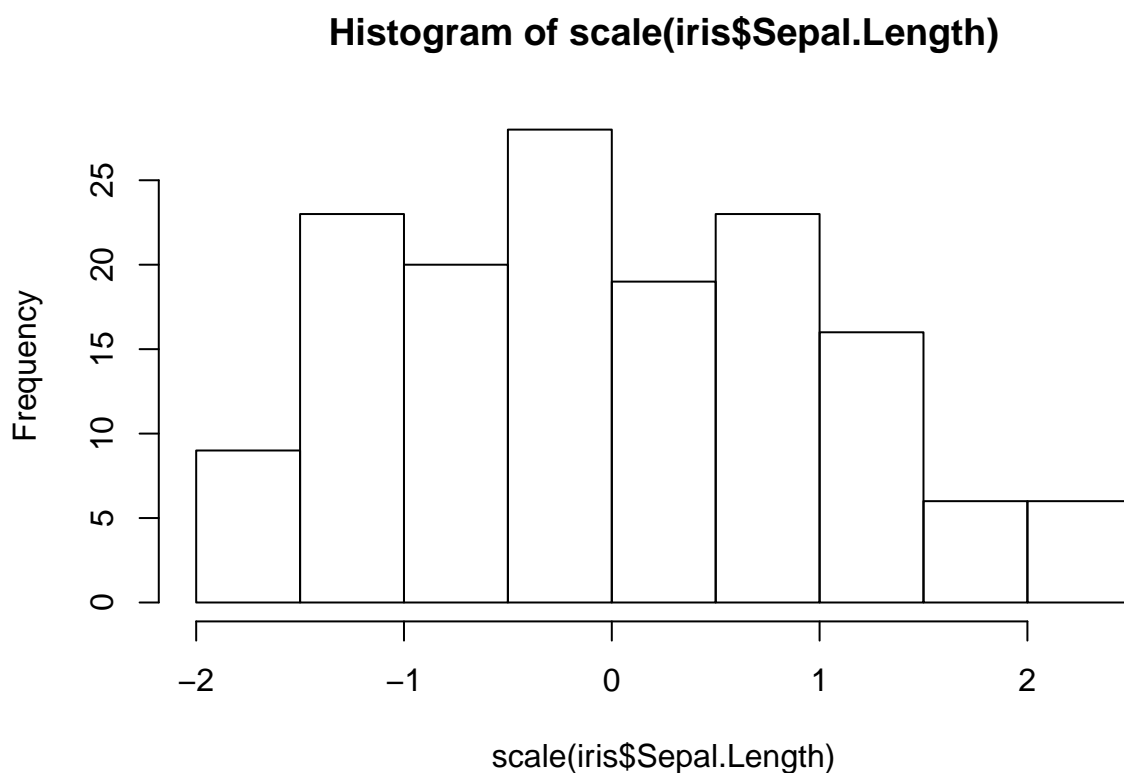
```
iris$Sepal.Length
```

```
## [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",]
```

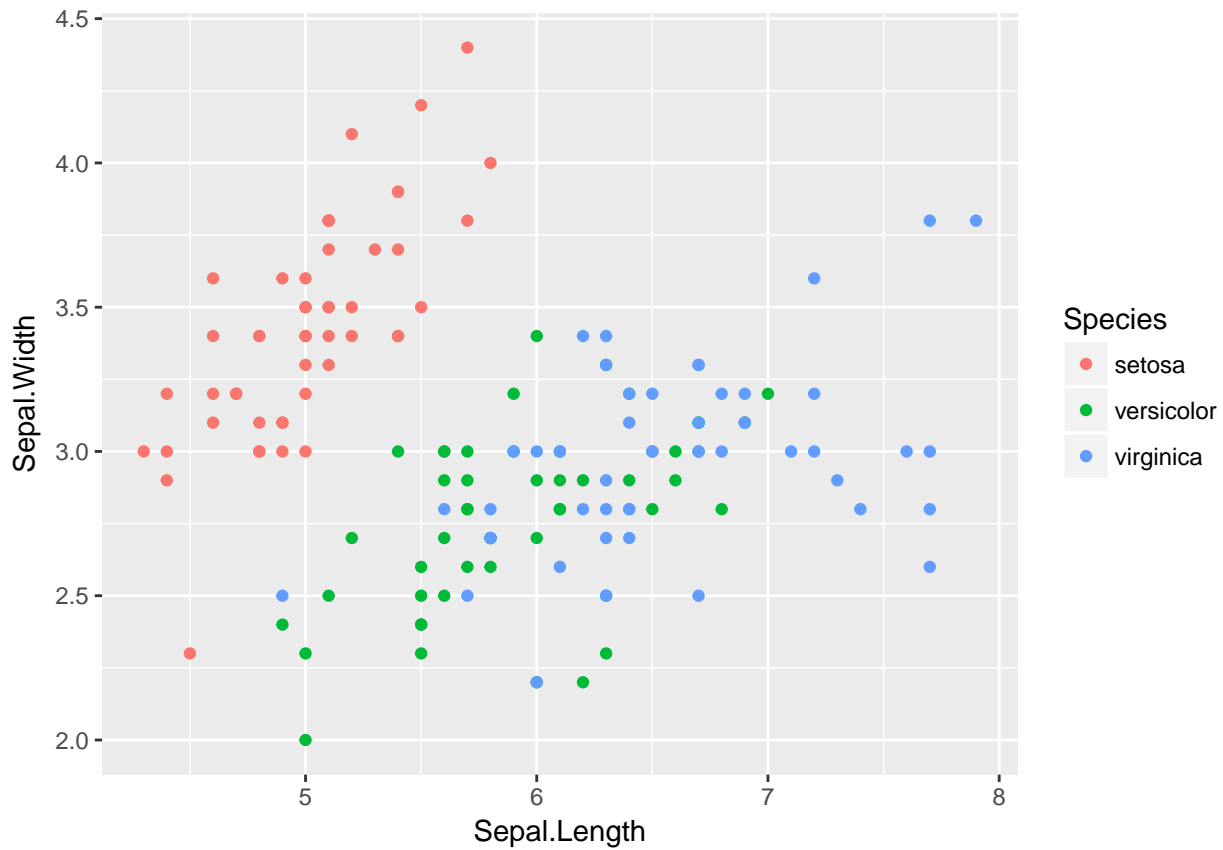
This could be an entire lecture by itself!!!

## 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)
```

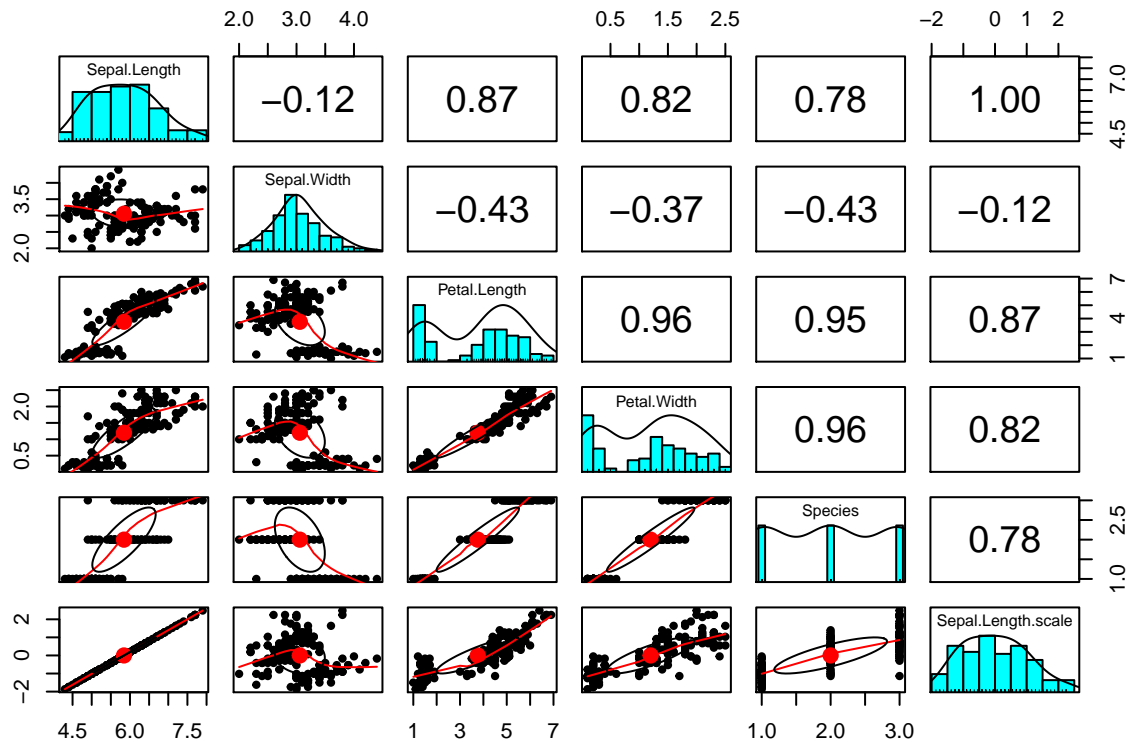
```
## Warning: package 'psych' was built under R version 3.4.4
```

```
##
## 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

- [swirl](#)
- [stackoverflow.com](#)
- [Twitter](#)
- Your peers
- R Community

### 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.

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

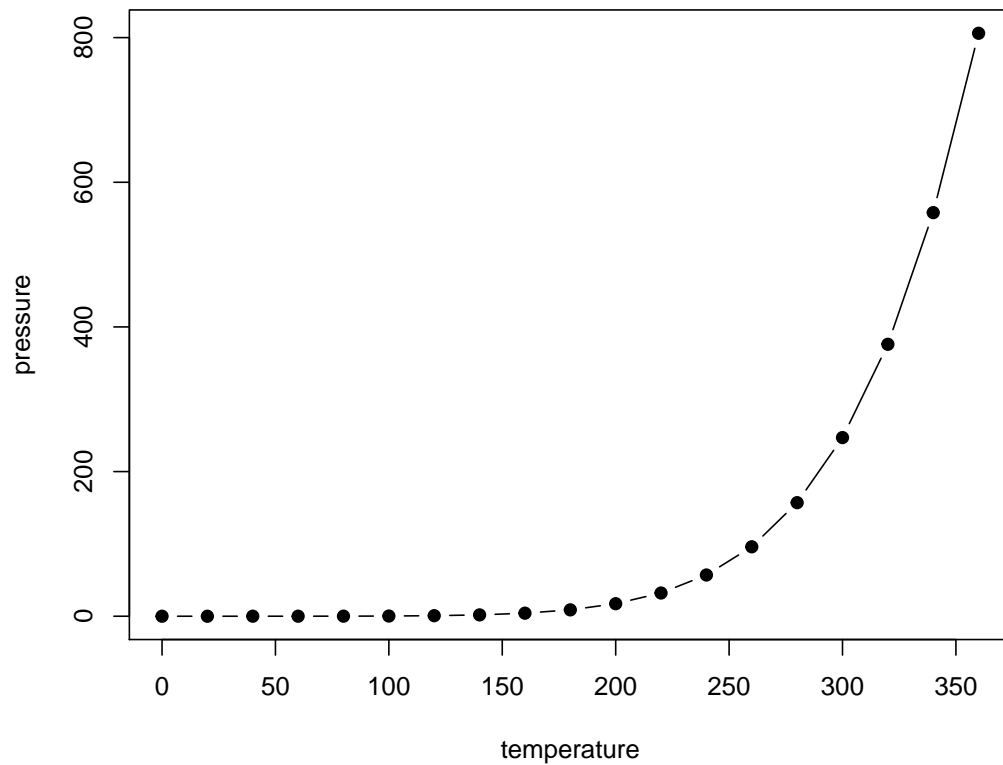


Figure 2.2: Here is a nice figure!

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.2. 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

This chapter will take you through a brief walk through of cleaning some data in R. When starting out with R, it's not important that you memorize every command and learn every line, but rather that you start to think about how a computer programmer would solve a question (you!) rather than get an undergrad RA to fix it by hand.

The first thing that you want to do to any script is to put a header on it. It should cover a few things such as

- A Title
- Who wrote the script?
- When did you edit it last?
- What does it do?
- How does it work?
- Does it take anything in specific?

The more information you can provide, the better. Remember that 6-Month-From-You now is going to be either your best or worst collaborator.

Each script in R out to follow a general pattern. First list out some information about it, then load in your libraries, and then set your working directory and load your data.

```
# Import Libraries  
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.4.3
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.2
```

```
library(psych)
```

Next thing you would do is set your working directory and load in your data. Note here that you need to tell R exactly where to look!

```
# 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")
```

You can also import data using RStudio's 'Import Data' function in the top right corner.



Figure 3.1: Import Data via RStudio's GUI

If you do do that, make sure to add in any of the libraries and code in your script that is printed out below in RStudio's console!

Next you should familiarize yourself with what your data looks like. Not only do you have to be cautious of the shape of your data (wide vs. long), but there are a few different types of data that R uses in its functionality.

Looking at the dataset from this experiment, we see lots of different types of data listed. R did its best to guess what 'kind' of data each one is, but sometimes you have to change data types.

```
str(experiment.data) # Check to see if R guessed correctly on data types
```

```
## 'data.frame':    251 obs. of  67 variables:
## $ Sub           : int  10 11 12 13 14 15 16 17 18 19 ...
## $ ACTIVE        : int  38 37 39 33 48 47 55 42 36 36 ...
## $ PERCEPTUAL  : int  47 47 49 42 57 55 60 61 43 36 ...
## $ MUSICAL       : int  28 16 21 26 7 8 46 39 19 24 ...
## $ SINGING       : int  33 22 40 19 24 24 45 25 29 28 ...
## $ EMOTIONS      : int  39 32 30 34 35 36 42 41 33 24 ...
## $ GENERAL       : int  75 67 82 68 60 65 118 86 69 68 ...
## $ inst          : Factor w/ 70 levels "", "air drums",...: 9 4 67 46 28 36 36 36 48 28 ...
## $ Age           : int  29 22 19 23 23 19 19 21 19 21 ...
## $ Gender        : Factor w/ 13 levels "", "COUNTRY", "f",...: 6 6 6 12 12 4 6 4 12 10 ...
## $ Major         : Factor w/ 147 levels "", "AGRIBUSINESS",...: 24 126 38 88 67 79 95 17 143 47 ...
## $ Minor         : Factor w/ 65 levels "", " Economics",...: 29 NA 50 NA 4 NA NA NA NA 39 ...
## $ SchoolUGorG   : int  1 1 1 1 1 1 1 1 1 1 ...
## $ MuscMajMinor  : int  1 1 1 1 1 1 2 1 1 1 ...
## $ BeginTrain    : Factor w/ 20 levels "", "10", "11", "12",...: 20 2 2 3 1 1 19 16 4 16 ...
## $ AbsPitch      : Factor w/ 7 levels "", "no", "No", "NO",...: 4 2 4 4 4 4 2 4 4 2 ...
## $ YearsPrac     : int  4 1 3 7 0 0 11 7 2 6 ...
## $ PeakPrac      : int  5 2 1 1 0 0 3 2 2 1 ...
## $ Live12Mo      : Factor w/ 23 levels "", "0", "1", "10",...: 9 15 3 2 20 9 17 12 14 18 ...
## $ TheoryTrain   : int  4 0 0 0 0 0 4 4 0 0 ...
## $ FormalYrs     : int  4 1 3 7 0 0 11 8 2 2 ...
## $ NoOfInst      : int  1 0 1 2 0 0 5 1 1 0 ...
## $ ActListenMin  : Factor w/ 34 levels "", "0", "1", "10",...: 30 30 9 24 5 30 9 24 24 22 ...
## $ SubjectNo     : int  10 11 12 13 14 15 16 17 18 19 ...
## $ SubRaven      : int  10 11 12 13 14 15 16 17 18 19 ...
## $ RavenB1       : int  10 7 3 10 7 6 8 8 4 3 ...
```



```
## $ RavenB2      : int  8 10 7 10 8 10 9 10 6 3 ...
## $ RavenB3      : int 10 6 7 11 5 6 8 8 5 3 ...
## $ RavenTotal   : int 28 23 17 31 20 22 25 26 15 9 ...
## $ OspanSub     : int 10 11 12 13 14 15 16 17 18 19 ...
## $ OspanAccError : int  8 7 7 1 2 12 4 8 5 37 ...
## $ OspanMathError : int 12 12 7 2 2 15 4 8 8 37 ...
## $ OspanAbsoluteScore : int 29 20 48 68 33 20 68 36 16 14 ...
## $ OspanPartialScore : int 47 41 65 73 50 44 70 64 30 23 ...
## $ OspanB1      : int 21 11 21 25 17 14 25 19 10 8 ...
## $ OspanB2      : int 14 18 23 23 16 17 20 24 11 6 ...
## $ OspanB3      : int 12 12 21 25 17 13 25 21 9 9 ...
## $ OspanSpeedErrTot : int 4 5 0 1 0 3 0 0 3 0 ...
## $ SymSub       : int 10 11 12 13 14 15 16 17 18 19 ...
## $ SymAccErrTot : int  1 4 2 1 0 4 2 3 2 23 ...
## $ SymSpeedErrTot : int  0 1 0 0 0 1 0 0 0 0 ...
## $ SspanAbsScore : int 10 15 21 27 15 9 30 24 12 18 ...
## $ SspanPartiaScore : int 19 29 34 34 26 25 38 30 18 24 ...
## $ SspanB1      : int  8 11 10 13 8 9 13 7 8 8 ...
## $ SspanB2      : int  4 13 13 10 8 10 14 14 4 7 ...
## $ SspanB3      : int  7 5 11 11 10 6 11 9 6 9 ...
## $ SymmErrTot   : int  1 5 2 1 0 5 2 3 2 23 ...
## $ ToneSub      : int 10 11 12 13 14 15 16 17 18 19 ...
## $ ToneAccErrTot : int  8 5 8 1 4 7 3 6 3 36 ...
## $ ToneMathErrTot : int  9 5 8 2 5 10 5 6 6 38 ...
## $ TspanAbsScore : int  9 21 27 37 10 4 23 14 3 0 ...
## $ TspanPartScore : int 36 47 55 63 31 34 58 48 37 26 ...
## $ TspanB1      : int 13 14 17 20 8 9 20 12 9 9 ...
## $ TspanB2      : int  6 19 21 22 9 12 19 18 12 5 ...
## $ TspanB3      : int 17 14 17 21 14 13 19 18 16 12 ...
## $ ToneSpeedErrTot : int  1 0 0 1 1 3 2 0 3 2 ...
## $ BeatSub      : int 10 11 12 13 14 15 16 17 18 19 ...
## $ BeatTotal    : int 11 10 14 13 14 10 14 11 7 12 ...
## $ BeatAcc      : num 0.61 0.56 0.78 0.72 0.78 0.56 0.78 0.61 0.39 0.67 ...
## $ MelodicSub   : int 10 11 12 13 14 15 16 17 18 19 ...
## $ MelodicTot   : int  7 7 6 10 5 5 6 7 7 7 ...
## $ MelodicAcc   : num 0.54 0.54 0.46 0.77 0.38 0.38 0.46 0.54 0.54 0.54 ...
## $ Native      : int  1 1 1 1 1 1 1 2 1 1 ...
## $ Paid        : int  1 1 1 1 1 1 1 1 1 1 ...
## $ jv_NoGMSI    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ jv_nonNative : int  0 0 0 0 0 0 0 1 0 0 ...
## $ jv_noOspan   : int  0 0 0 0 0 0 0 0 0 0 ...
```

Using `read.csv()` with its default setting, 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 easiest to manipulate a character or string. There are much more elegant ways to do this, but just to make a point, I've gone through and fixed each one manually.

Notice how the output of `str()` has changed.

```
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.
```

```
## 'data.frame':    251 obs. of  67 variables:
## $ Sub           : int  10 11 12 13 14 15 16 17 18 19 ...
## $ ACTIVE        : int  38 37 39 33 48 47 55 42 36 36 ...
## $ PERCEPTUAL  : int  47 47 49 42 57 55 60 61 43 36 ...
## $ MUSICAL       : int  28 16 21 26 7 8 46 39 19 24 ...
## $ SINGING       : int  33 22 40 19 24 24 45 25 29 28 ...
## $ EMOTIONS      : int  39 32 30 34 35 36 42 41 33 24 ...
## $ GENERAL       : int  75 67 82 68 60 65 118 86 69 68 ...
## $ inst          : chr  "clarinet" "bass" "Voice" "saxophone" ...
## $ Age           : int  29 22 19 23 23 19 19 21 19 21 ...
## $ Gender        : chr  "FEMALE" "FEMALE" "FEMALE" "MALE" ...
## $ Major         : chr  "BUSINESS MANAGEMENT" "psychology" "COMMUNICATION STUDIES" "Mechanical E
## $ Minor         : chr  "HUMAN RESOURCE MANAGEMENT" NA "SOCIAL WORK" NA ...
## $ SchoolUGorG   : int  1 1 1 1 1 1 1 1 1 1 ...
## $ MuscMajMinor   : int  1 1 1 1 1 1 2 1 1 1 ...
## $ BeginTrain     : chr  "9" "10" "10" "11" ...
## $ AbsPitch       : chr  "NO" "no" "NO" "NO" ...
## $ YearsPrac     : int  4 1 3 7 0 0 11 7 2 6 ...
## $ PeakPrac      : int  5 2 1 1 0 0 3 2 2 1 ...
## $ Live12Mo       : num  9 15 3 2 20 9 17 12 14 18 ...
## $ TheoryTrain    : int  4 0 0 0 0 0 4 4 0 0 ...
## $ FormalYrs     : int  4 1 3 7 0 0 11 8 2 2 ...
## $ NoOfInst       : int  1 0 1 2 0 0 5 1 1 0 ...
## $ ActListenMin   : chr  "60" "60" "180" "45" ...
## $ SubjectNo      : int  10 11 12 13 14 15 16 17 18 19 ...
## $ SubRaven       : int  10 11 12 13 14 15 16 17 18 19 ...
## $ RavenB1        : int  10 7 3 10 7 6 8 8 4 3 ...
## $ RavenB2        : int  8 10 7 10 8 10 9 10 6 3 ...
## $ RavenB3        : int  10 6 7 11 5 6 8 8 5 3 ...
## $ RavenTotal     : int  28 23 17 31 20 22 25 26 15 9 ...
## $ OspanSub       : int  10 11 12 13 14 15 16 17 18 19 ...
## $ OspanAccError   : int  8 7 7 1 2 12 4 8 5 37 ...
## $ OspanMathError  : int  12 12 7 2 2 15 4 8 8 37 ...
## $ OspanAbsoluteScore: int  29 20 48 68 33 20 68 36 16 14 ...
## $ OspanPartialScore: int  47 41 65 73 50 44 70 64 30 23 ...
## $ OspanB1        : int  21 11 21 25 17 14 25 19 10 8 ...
## $ OspanB2        : int  14 18 23 23 16 17 20 24 11 6 ...
## $ OspanB3        : int  12 12 21 25 17 13 25 21 9 9 ...
## $ OspanSpeedErrTot: int  4 5 0 1 0 3 0 0 3 0 ...
## $ SymSub         : int  10 11 12 13 14 15 16 17 18 19 ...
## $ SymAccErrTot   : int  1 4 2 1 0 4 2 3 2 23 ...
## $ SymSpeedErrTot : int  0 1 0 0 0 1 0 0 0 0 ...
## $ SspanAbsScore   : int  10 15 21 27 15 9 30 24 12 18 ...
## $ SspanPartiaScore: int  19 29 34 34 26 25 38 30 18 24 ...
## $ SspanB1        : int  8 11 10 13 8 9 13 7 8 8 ...
## $ SspanB2        : int  4 13 13 10 8 10 14 14 4 7 ...
## $ SspanB3        : int  7 5 11 11 10 6 11 9 6 9 ...
## $ SymmErrTot     : int  1 5 2 1 0 5 2 3 2 23 ...
```

```
## $ ToneSub          : int  10 11 12 13 14 15 16 17 18 19 ...
## $ ToneAccErrTot    : int   8 5 8 1 4 7 3 6 3 36 ...
## $ ToneMathErrTot   : int   9 5 8 2 5 10 5 6 6 38 ...
## $ TspanAbsScore    : int   9 21 27 37 10 4 23 14 3 0 ...
## $ TspanPartScore   : int  36 47 55 63 31 34 58 48 37 26 ...
## $ TspanB1          : int  13 14 17 20 8 9 20 12 9 9 ...
## $ TspanB2          : int   6 19 21 22 9 12 19 18 12 5 ...
## $ TspanB3          : int  17 14 17 21 14 13 19 18 16 12 ...
## $ ToneSpeedErrTot  : int   1 0 0 1 1 3 2 0 3 2 ...
## $ BeatSub          : int  10 11 12 13 14 15 16 17 18 19 ...
## $ BeatTotal        : int  11 10 14 13 14 10 14 11 7 12 ...
## $ BeatAcc          : num  0.61 0.56 0.78 0.72 0.78 0.56 0.78 0.61 0.39 0.67 ...
## $ MelodicSub       : int  10 11 12 13 14 15 16 17 18 19 ...
## $ MelodicTot       : int   7 7 6 10 5 5 6 7 7 7 ...
## $ MelodicAcc       : num  0.54 0.54 0.46 0.77 0.38 0.38 0.46 0.54 0.54 0.54 ...
## $ Native           : int   1 1 1 1 1 1 1 2 1 1 ...
## $ Paid             : int   1 1 1 1 1 1 1 1 1 1 ...
## $ jv_NoGMSI        : int   0 0 0 0 0 0 0 0 0 0 ...
## $ jv_nonNative     : int   0 0 0 0 0 0 0 1 0 0 ...
## $ jv_noOspan       : int   0 0 0 0 0 0 0 0 0 0 ...
```

The next thing we'd want to check in our data cleaning is for any sort of import errors. We can use combinations of the functions `names()`, `table()`, `is.na()`, and `complete.cases()` to get a brief summary of what is missing. You could also use R's plotting functionality to see if there are any participants with weird subject numbers or maybe negative values where there should not be. If you find any problems, code in the solution in R! Resist the urge to fix it in the Excel file!

```
# Check for Import Errors
```

```
table(complete.cases(experiment.data)) # Not all observations have everything!
```

```
##
```

```
## FALSE TRUE
```

```
##    74    177
```

```
complete.cases(experiment.data)
```

```
## [1] TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE TRUE FALSE
## [12] FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE TRUE TRUE FALSE
## [23] TRUE FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE
## [34] FALSE TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE TRUE
## [45] TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE TRUE FALSE FALSE
## [56] FALSE TRUE FALSE TRUE FALSE FALSE TRUE FALSE TRUE FALSE FALSE
## [67] FALSE TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE TRUE
## [78] FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE
## [89] FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE TRUE TRUE
## [100] TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE
## [111] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [122] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [133] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [144] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [155] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [166] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [177] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
## [188] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [199] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [210] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
## [221] TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE FALSE TRUE TRUE
## [232] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [243] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
table(is.na(experiment.data))
```

```
##
## FALSE TRUE
## 16723 94
```

```
# Gotta decide what to do about it!!
```

Later we'll cover how to change individual cases, but as I will note, you will want to avoid hard coding in any fixes if you can. Let's first look at how we might fix a messy gender problem.

### 3.1 Cleaning 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 efficient way to fix the gender column? What format does the variable 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 in the right order!!! As we are not dealing with huge amounts of data, the `table()` function will help out.

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
```

```
## [1] "9"      "10"     "10"     "11"     ""
## [6] ""       "8"      "6"      "12"     "6"
## [11] ""       ""       "10"     ""       "7"
## [16] "16"     "10"     "18"     "15"     "7"
## [21] "8"      "10"     "11"     "8"      "12"
## [26] "6"      ""       "17"     ""       ""
## [31] "10"     "15"     "15"     "8"      "6"
## [36] "13"     "16"     ""       "7"      "7"
## [41] "10"     ""       "12"     "5"      "11"
## [46] "9"      ""       "21"     ""       "16"
## [51] "17"     "11"     ""       "13"     ""
## [56] "15"     "6"      "15"     "9"      "16"
## [61] "10"     "18"     ""       "11"     "9"
## [66] "12"     "10"     "4"      ""       "9"
## [71] "9"      ""       "14"     "11"     "10"
## [76] "7"      ""       "17"     "9"      "16"
## [81] "7"      "7"      "5"      ""       ""
## [86] "10"     "10"     "9"      "14"     "17"
## [91] "11"     ""       ""       "12"     "6"
## [96] "14"     "8"      "11"     ""       ""
## [101] "12"     "6"      "10"     "14"     "8"
## [106] "13"     "12"     "8"      "11"     "8"
## [111] "14"     "11"     "8"      ""       ""
## [116] "10"     "13"     "13"     "5"      "15"
## [121] ""       "11"     "8"      "12"     "10"
## [126] "9"      "6"      "7"      "10"     "6"
```

```
## [131] "8"      ""      "16"     ""      ""
## [136] ""      "11"     "12"     "7"     ""
## [141] "4"      "7"      "7"      ""      "10"
## [146] "10"     "12"     "10"     ""      "12"
## [151] "12"     "6"      "14"     "7"     ""
## [156] "6"      "18"     "8"      "12"     "15"
## [161] "13"     ""      "10"     "11"     "10"
## [166] "13"     "8"      ""      "6"      "8"
## [171] "8"      ""      "9"      "13"     ""
## [176] "16"     "10"     "9"      "10"     "5"
## [181] "11"     ""      "15"     "11"     "13"
## [186] "14"     "11"     "2"      "12"     "11"
## [191] "8"      "13"     "8"      "9"      "12"
## [196] "9"      "7"      "12"     "9"      "8"
## [201] "9"      "6"      "11"     "11"     "9"
## [206] "10"     "12"     "12"     "10"     "6"
## [211] "9"      "14"     "5"      "11"     "12"
## [216] "8"      "9"      "11"     "5"      "13"
## [221] "10"     "11"     "9"      "9"      "10"
## [226] "5"      "8"      "10"     "7"      "10"
## [231] "12"     "10"     "7"      "13"     "11"
## [236] "11"     "10"     "9"      "8"      "12"
## [241] "11"     "7"      "10"     "12"     "29"
## [246] "5"      "13"     "9"      "9"      "7th grade"
## [251] "8"
```

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]
```

```
## [1] "7th grade"
```

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
```

```
## [1] "9" "10" "10" "11" "" "" "8" "6" "12" "6" "" "" "10" ""
## [15] "7" "16" "10" "18" "15" "7" "8" "10" "11" "8" "12" "6" "" "17"
## [29] "" "" "10" "15" "15" "8" "6" "13" "16" "" "7" "7" "10" ""
## [43] "12" "5" "11" "9" "" "21" "" "16" "17" "11" "" "13" "" "15"
## [57] "6" "15" "9" "16" "10" "18" "" "11" "9" "12" "10" "4" "" "9"
## [71] "9" "" "14" "11" "10" "7" "" "17" "9" "16" "7" "7" "5" ""
## [85] "" "10" "10" "9" "14" "17" "11" "" "" "12" "6" "14" "8" "11"
## [99] "" "" "12" "6" "10" "14" "8" "13" "12" "8" "11" "8" "14" "11"
## [113] "8" "" "" "10" "13" "13" "5" "15" "" "11" "8" "12" "10" "9"
## [127] "6" "7" "10" "6" "8" "" "16" "" "" "" "11" "12" "7" ""
## [141] "4" "7" "7" "" "10" "10" "12" "10" "" "12" "12" "6" "14" "7"
## [155] "" "6" "18" "8" "12" "15" "13" "" "10" "11" "10" "13" "8" ""
## [169] "6" "8" "8" "" "9" "13" "" "16" "10" "9" "10" "5" "11" ""
```

```
## [183] "15" "11" "13" "14" "11" "2" "12" "11" "8" "13" "8" "9" "12" "9"
## [197] "7" "12" "9" "8" "9" "6" "11" "11" "9" "10" "12" "12" "10" "6"
## [211] "9" "14" "5" "11" "12" "8" "9" "11" "5" "13" "10" "11" "9" "9"
## [225] "10" "5" "8" "10" "7" "10" "12" "10" "7" "13" "11" "11" "10" "9"
## [239] "8" "12" "11" "7" "10" "12" "29" "5" "13" "9" "9" "12" "8"
```

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)
```

```
## [1] "9" "10" "10" "11" "0" "0" "8" "6" "12" "6" "0" "0" "10" "0"
## [15] "7" "16" "10" "18" "15" "7" "8" "10" "11" "8" "12" "6" "0" "17"
## [29] "0" "0" "10" "15" "15" "8" "6" "13" "16" "0" "7" "7" "10" "0"
## [43] "12" "5" "11" "9" "0" "21" "0" "16" "17" "11" "0" "13" "0" "15"
## [57] "6" "15" "9" "16" "10" "18" "0" "11" "9" "12" "10" "4" "0" "9"
## [71] "9" "0" "14" "11" "10" "7" "0" "17" "9" "16" "7" "7" "5" "0"
## [85] "0" "10" "10" "9" "14" "17" "11" "0" "0" "12" "6" "14" "8" "11"
## [99] "0" "0" "12" "6" "10" "14" "8" "13" "12" "8" "11" "8" "14" "11"
## [113] "8" "0" "0" "10" "13" "13" "5" "15" "0" "11" "8" "12" "10" "9"
## [127] "6" "7" "10" "6" "8" "0" "16" "0" "0" "0" "11" "12" "7" "0"
## [141] "4" "7" "7" "0" "10" "10" "12" "10" "0" "12" "12" "6" "14" "7"
## [155] "0" "6" "18" "8" "12" "15" "13" "0" "10" "11" "10" "13" "8" "0"
## [169] "6" "8" "8" "0" "9" "13" "0" "16" "10" "9" "10" "5" "11" "0"
## [183] "15" "11" "13" "14" "11" "2" "12" "11" "8" "13" "8" "9" "12" "9"
## [197] "7" "12" "9" "8" "9" "6" "11" "11" "9" "10" "12" "12" "10" "6"
## [211] "9" "14" "5" "11" "12" "8" "9" "11" "5" "13" "10" "11" "9" "9"
## [225] "10" "5" "8" "10" "7" "10" "12" "10" "7" "13" "11" "11" "10" "9"
## [239] "8" "12" "11" "7" "10" "12" "29" "5" "13" "9" "9" "12" "8"
```

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$BeginTrain))
```

And Alas!

```
experiment.data$BeginTrain
```

```
## [1] 9 10 10 11 0 0 8 6 12 6 0 0 10 0 7 16 10 18 15 7 8 10 11
## [24] 8 12 6 0 17 0 0 10 15 15 8 6 13 16 0 7 7 10 0 12 5 11 9
## [47] 0 21 0 16 17 11 0 13 0 15 6 15 9 16 10 18 0 11 9 12 10 4 0
## [70] 9 9 0 14 11 10 7 0 17 9 16 7 7 5 0 0 10 10 9 14 17 11 0
## [93] 0 12 6 14 8 11 0 0 12 6 10 14 8 13 12 8 11 8 14 11 8 0 0
## [116] 10 13 13 5 15 0 11 8 12 10 9 6 7 10 6 8 0 16 0 0 0 11 12
## [139] 7 0 4 7 7 0 10 10 12 10 0 12 12 6 14 7 0 6 18 8 12 15 13
## [162] 0 10 11 10 13 8 0 6 8 8 0 9 13 0 16 10 9 10 5 11 0 15 11
## [185] 13 14 11 2 12 11 8 13 8 9 12 9 7 12 9 8 9 6 11 11 9 10 12
## [208] 12 10 6 9 14 5 11 12 8 9 11 5 13 10 11 9 9 10 5 8 10 7 10
## [231] 12 10 7 13 11 11 10 9 8 12 11 7 10 12 29 5 13 9 9 12 8
```

Let's now clean up the Gender column, first let's look at it

```
experiment.data$Gender # Are there any common trends?
```

```
## [1] "FEMALE" "FEMALE" "FEMALE" "MALE" "MALE" "female" "FEMALE"
## [8] "female" "MALE" "male" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
## [15] "FEMALE" "FEMALE" "FEMALE" "male" "FEMALE" "FEMALE" "FEMALE"
```

```
## [22] "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE" "MALE" "FEMALE"
## [29] "FEMALE" "" "MALE" "FEMALE" "female" "female" "FEMALE"
## [36] "female" "female" "Female" "FEMALE" "FEMALE" "MALE" "MALE"
## [43] "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE" "FEMALE" "female"
## [50] "MALE" "MALE" "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE"
## [57] "MALE" "FEMALE" "FEMALE" "FEMALE" "female" "FEMALE" NA
## [64] "Male" "Female" "MALE" "MALE" "FEMALE" "FEMALE" "FEMALE"
## [71] "MALE" "FEMALE" "FEMALE" "MALE" "MALE" "female" "FEMALE"
## [78] "female" "FEMALE" "female" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
## [85] "FEMALE" "female" "MALE" "FEMALE" "FEMALE" "MALE" "FEMALE"
## [92] "m" "female" "male" "FEMALE" "FEMALE" "Female" "female"
## [99] "FEMALE" "FEMALE" "MALE" "female" "FEMALE" "MALE" "MALE"
## [106] "male" "FEMALE" "MALE" "female" "female" "female" "Male"
## [113] "FEMALE" "FEMALE" "FEMALE" "FEMALE" "Male" "FEMALE" "FEMALE"
## [120] "MALE" "FEMALE" "male" "FEMALE" "FEMALE" "MALE" "FEMALE"
## [127] "FEMALE" "FEMALE" "Female" "FEMALE" "FEMALE" "female" "female"
## [134] "FEMALE" "FEMALE" "FEMALE" "MALE" "MALE" "FEMALE" "FEMALE"
## [141] "MALE" "Male" "FEMALE" "female" "FEMALE" "MALE" "FEMALE"
## [148] "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE" "male" "FEMALE"
## [155] "FEMALE" "FEMALE" "male" "FEMALE" "FEMALE" "FEMALE" "MALE"
## [162] "FEMALE" "female" "FEMALE" "FEMALE" "FEMALE" "f" "FEMALE"
## [169] "MALE" "FEMALE" "male" "Male" "FEMALE" "MALE" "female"
## [176] "female" "MALE" "FEMALE" "FEMALE" "female" "FEMALE" "COUNTRY"
## [183] "MALE" "MALE" "female" "male" "MALE" "FEMALE" "MALE"
## [190] "FEMALE" "FEMALE" "Female" "FEMALE" "FEMALE" "MALE" "MALE"
## [197] "FEMALE" "MALE" "FEMALE" "FEMALE" "MALE" "Male" "MALE"
## [204] "MALE" "MALE" "MALE" "MALE" "Male" "FEMALE" "FEMALE"
## [211] "FEMALE" "MALE" "female" "FEMALE" "MALE" "FEMALE" "MALE"
## [218] "FEMALE" "MALE" "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE"
## [225] "MALE" "FEMALE" "FEMALE" "MALE" "MALE" "FEMALE" "FEMALE"
## [232] "FEMALE" "FEMALE" "FEMALE" "MALE" "FEMALE" "FEMALE" "FEMALE"
## [239] "FEMALE" "MALE" "FEMALE" "FEMALE" "FEMALE" "MALE" "MALE"
## [246] "FEMALE" "M" "FEMALE" "FEMALE" "FEMALE" "FEMALE"
```

```
table(experiment.data$Gender)
```

```
##
##      COUNTRY      f female Female FEMALE female      m      M
##      1      1      1     24      5     137      3      1      1
##   male   Male   MALE   MALE
##      9      7     59      1
```

### 3.1.1 Cleaning Up 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.

```
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$Gender[30] <- "No Response" #Something Might Be Up w this datapoint?
experiment.data$Gender[63] <- "No Response" #Something Might Be Up w this datapoint?
```

```
experiment.data$Gender <- as.factor(experiment.data$Gender)
table(experiment.data$Gender)
```

```
##
##      Female      Male No Response
##      170        78          3
```

```
#-----
```

```
# Can we do same thing for AP?
```

```
experiment.data$AbsPitch
```

```
## [1] "NO" "no" "NO" "NO" "NO" "NO" "no" "NO" "NO" "no" "NO"
## [12] "NO" "NO" "NO" "NO" "NO" "NO" "no" "No" "NO" "no" "NO"
## [23] "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "" "No" "NO" "no"
## [34] "no" "NO" "no" "no" "no" "NO" "NO" "NO" "NO" "no" "NO" "NO"
## [45] "NO" "no" "NO" "NO" "no" "NO" "NO" "NO" "NO" "NO" "NO" "NO"
## [56] "NO" "NO" "No" "no" "NO" "No" "NO" "no" "no" "no" "No" "NO"
## [67] "NO" "no" "no" "NO" "NO" "NO" "no" "NO" "NO" "NO" "no" "NO"
## [78] "no" "no" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "no" "NO" "NO"
## [89] "NO" "NO" "NO" "NO" "no" "no" "NO" "NO" "NO" "no" "no" "NO"
## [100] "no" "NO" "no" "NO" "NO" "NO" "no" "NO" "NO" "NO" "no" "no"
## [111] "no" "no" "NO" "NO" "NO" "NO" "no" "NO" "NO" "no" "NO" "NO"
## [122] "NO" "NO" "No" "NO" "NO" "no" "NO" "no" "NO" "NO" "NO" "no"
## [133] "no" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "No" "No" "NO"
## [144] "no" "NO" "NO" "NO" "NO" "NO" "no" "NO" "NO" "NO" "no" "NO"
## [155] "Non" "YES" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "no" "NO" "NO"
## [166] "NO" "no" "NO" "NO" "no" "no" "NO" "NO" "NO" "no" "No" "no"
## [177] "NO" "NO" "no" "non" "NO" "NO" "NO" "NO" "NO" "No" "no" "NO"
## [188] "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO"
## [199] "no" "NO" "NO" "No" "NO" "No" "NO" "NO" "NO" "NO" "no" "NO"
## [210] "NO" "NO" "NO" "NO" "No" "No" "NO" "NO" "NO" "NO" "NO" "NO"
## [221] "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "NO"
## [232] "NO" "NO" "NO" "NO" "NO" "NO" "No" "No" "NO" "NO" "NO" "NO"
## [243] "no" "NO" "NO" "NO" "NO" "NO" "NO" "NO" "no" "NO"
```

```
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)
```

```
##
## no yes
## 250 1
```

## 3.2 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)
```

```
## [1] "Sub"           "ACTIVE"        "PERCEPTUAL"
## [4] "MUSICAL"       "SINGING"       "EMOTIONS"
## [7] "GENERAL"       "inst"          "Age"
## [10] "Gender"        "Major"         "Minor"
## [13] "SchoolUGorG"   "MuscMajMinor"  "BeginTrain"
## [16] "AbsPitch"      "YearsPrac"     "PeakPrac"
## [19] "Live12Mo"      "TheoryTrain"   "FormalYrs"
## [22] "NoOfInst"      "ActListenMin"  "SubjectNo"
## [25] "SubRaven"      "RavenB1"       "RavenB2"
## [28] "RavenB3"       "RavenTotal"    "OspanSub"
## [31] "OspanAccError" "OspanMathError" "OspanAbsoluteScore"
## [34] "OspanPartialScore" "OspanB1"       "OspanB2"
## [37] "OspanB3"       "OspanSpeedErrTot" "SymSub"
## [40] "SymAccErrTot"  "SymSpeedErrTot" "SspanAbsScore"
## [43] "SspanPartiaScore" "SspanB1"       "SspanB2"
## [46] "SspanB3"       "SymmErrTot"    "ToneSub"
## [49] "ToneAccErrTot" "ToneMathErrTot" "TspanAbsScore"
## [52] "TspanPartScore" "TspanB1"       "TspanB2"
## [55] "TspanB3"       "ToneSpeedErrTot" "BeatSub"
## [58] "BeatTotal"     "BeatAcc"       "MelodicSub"
## [61] "MelodicTot"    "MelodicAcc"    "Native"
## [64] "Paid"          "jv_NoGMSI"     "jv_nonNative"
## [67] "jv_noOspan"
```

```
names(item.level.data)
```

```
## [1] "tmp.dat.subject.1." "X1"           "X2"
## [4] "X3"                 "X4"           "X5"
## [7] "X6"                 "X7"           "X8"
## [10] "X9"                 "X10"          "X11"
## [13] "X12"                "X13"          "X14"
## [16] "X15"                "X16"          "X17"
## [19] "X18"                "X19"          "X20"
## [22] "X21"                "X22"          "X23"
## [25] "X24"                "X25"          "X26"
## [28] "X27"                "X28"          "X29"
## [31] "X30"                "X31"          "X32"
## [34] "X33"                "X34"          "X35"
## [37] "X36"                "X37"          "X38"
```

First off our subject ID columns are not the same. Let's swap that.

```
names(item.level.data)
```

```
## [1] "tmp.dat.subject.1." "X1"           "X2"
## [4] "X3"                 "X4"           "X5"
## [7] "X6"                 "X7"           "X8"
## [10] "X9"                 "X10"          "X11"
## [13] "X12"                "X13"          "X14"
## [16] "X15"                "X16"          "X17"
## [19] "X18"                "X19"          "X20"
## [22] "X21"                "X22"          "X23"
## [25] "X24"                "X25"          "X26"
```

```
## [28] "X27"          "X28"          "X29"
## [31] "X30"          "X31"          "X32"
## [34] "X33"          "X34"          "X35"
## [37] "X36"          "X37"          "X38"
```

```
names(experiment.data)
```

```
## [1] "Sub"          "ACTIVE"       "PERCEPTUAL"
## [4] "MUSICAL"      "SINGING"      "EMOTIONS"
## [7] "GENERAL"      "inst"         "Age"
## [10] "Gender"       "Major"        "Minor"
## [13] "SchoolUGorG" "MuscMajMinor" "BeginTrain"
## [16] "AbsPitch"     "YearsPrac"    "PeakPrac"
## [19] "Live12Mo"     "TheoryTrain"  "FormalYrs"
## [22] "NoOfInst"     "ActListenMin" "SubjectNo"
## [25] "SubRaven"     "RavenB1"      "RavenB2"
## [28] "RavenB3"     "RavenTotal"   "OspanSub"
## [31] "OspanAccError" "OspanMathError" "OspanAbsoluteScore"
## [34] "OspanPartialScore" "OspanB1"      "OspanB2"
## [37] "OspanB3"     "OspanSpeedErrTot" "SymSub"
## [40] "SymAccErrTot" "SymSpeedErrTot" "SspanAbsScore"
## [43] "SspanPartiaScore" "SspanB1"      "SspanB2"
## [46] "SspanB3"     "SymmErrTot"   "ToneSub"
## [49] "ToneAccErrTot" "ToneMathErrTot" "TspanAbsScore"
## [52] "TspanPartScore" "TspanB1"      "TspanB2"
## [55] "TspanB3"     "ToneSpeedErrTot" "BeatSub"
## [58] "BeatTotal"   "BeatAcc"      "MelodicSub"
## [61] "MelodicTot"  "MelodicAcc"   "Native"
## [64] "Paid"        "jv_NoGMSI"    "jv_nonNative"
## [67] "jv_noOspan"
```

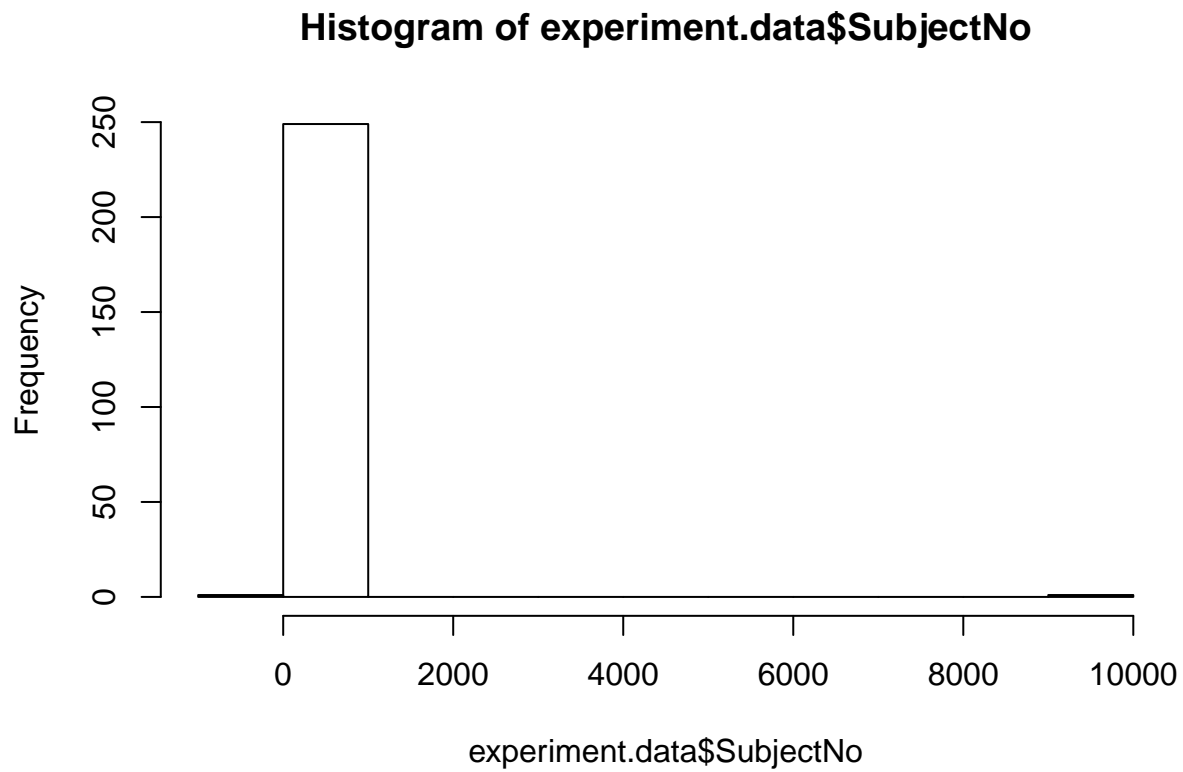
```
setnames(item.level.data,"tmp.dat.subject.1.", "SubjectNo")
setnames(experiment.data,"Sub", "SubjectNo") # Make this clearer!!!
```

If you need to do more than 1, use the `c()` operator! Now if we look at this column, it's all messed 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", replac
item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "C", replac
item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "M", replac
item.level.data$SubjectNo <- str_replace_all(string = item.level.data$SubjectNo, pattern = "CM", replac
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!
```

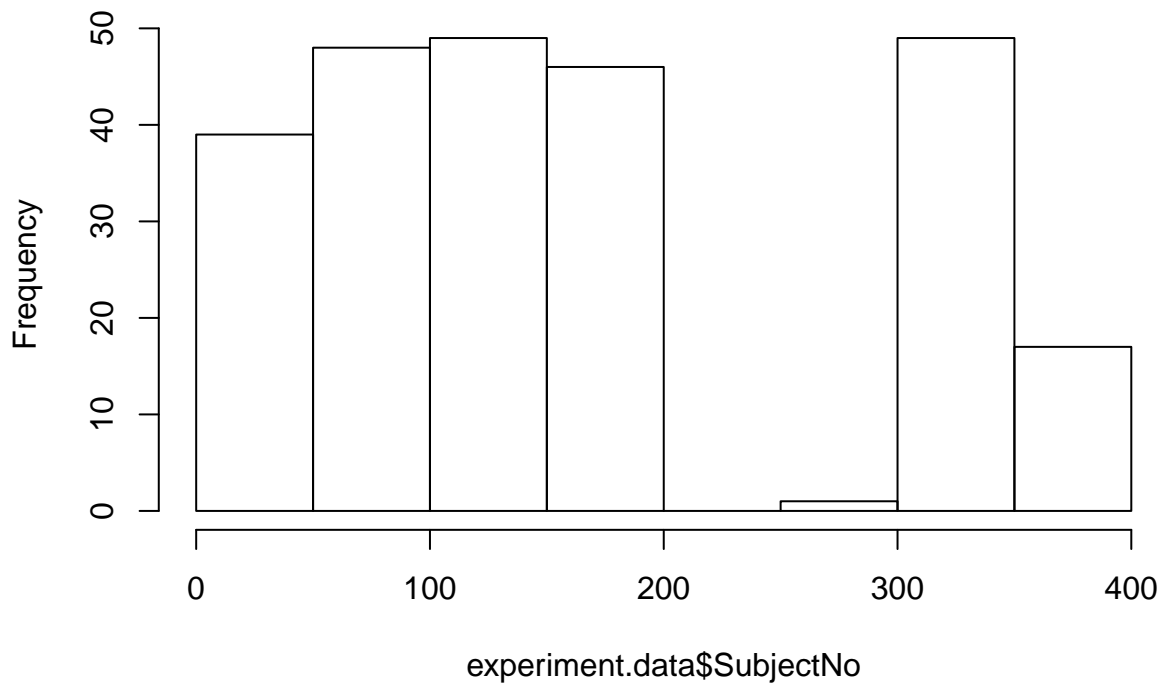


Notice that the variable of `SubjectNo` has negative values and placeholders! We can use R's indexing function to find these entries.

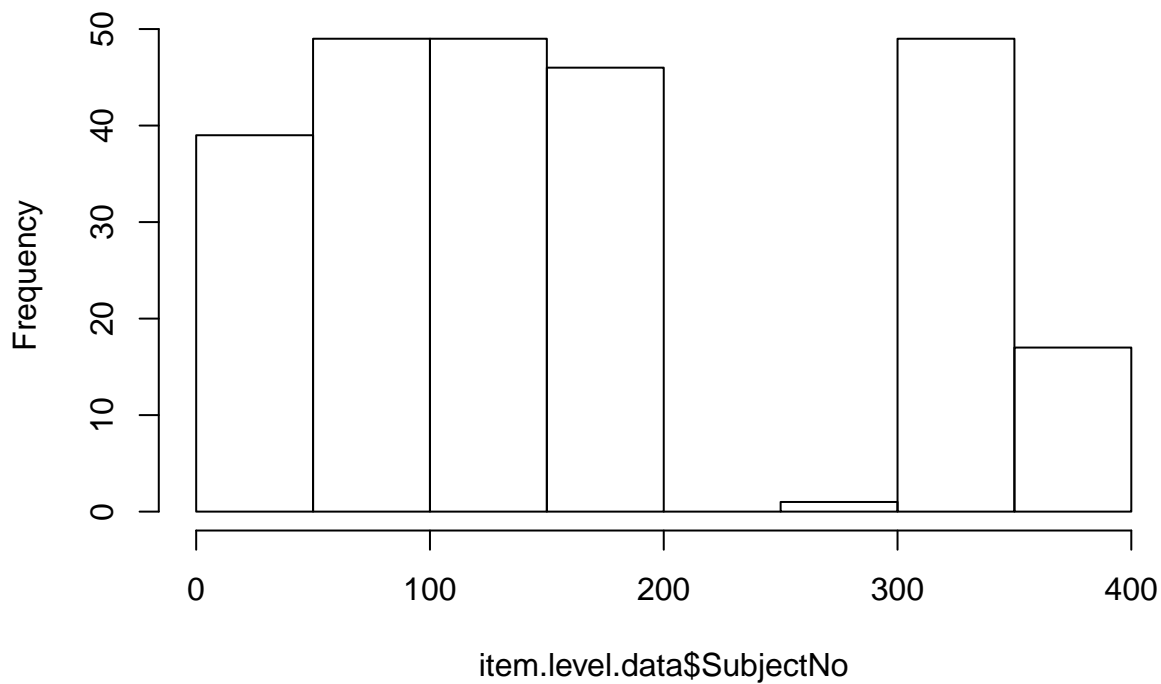
```
# Drop those  
experiment.data <- experiment.data[experiment.data$SubjectNo > 0 & experiment.data$SubjectNo < 1000,]
```

Note this works because the `SubjectNo` variable is numeric

```
hist(experiment.data$SubjectNo)
```

**Histogram of experiment.data\$SubjectNo**

```
hist(item.level.data$SubjectNo)
```

**Histogram of item.level.data\$SubjectNo**

Ok, finally we merge our datasets. What we are doing here is called an “inner join” Here we will keep 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
```

```
##      SubjectNo X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17
##  1:         100  2  1  5  4  1  5  5  1  4  5  7  1  5  4  3  5  6
##  2:         101  7  5  7  7  3  7  7  4  7  2  6  4  5  5  5  4  6
##  3:         102  4  2  5  4  3  7  4  5  3  4  6  3  2  4  4  5  2
##  4:         103  5  7  7  5  5  7  7  5  7  6  7  2  5  6  5  4  4
##  5:         104  2  1  1  2  1  4  1  5  3  5  6  2  5  6  6  6  1
##  ---
## 246:          95  3  2  6  5  3  2  3  3  4  2  6  6  5  6  6  6  3
## 247:          96  5  1  3  5  5  5  6  5  3  5  6  4  2  5  5  5  1
## 248:          97  5  2  4  4  2  3  5  4  2  5  6  4  3  5  5  3  6
## 249:          98  6  2  7  7  6  7  5  3  4  4  7  6  6  6  6  6  7
## 250:          99  5  4  5  5  6  7  7  4  7  2  4  5  2  5  5  4  1
##      X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34
##  1:    2   2   2   3   3   3   4   2   2   7   2   5   5   2   1   6   7
##  2:    6   2   1   3   5   1   1   2   3   4   2   4   3   3   2   7   7
##  3:    4   2   1   2   6   4   3   2   2   2   1   2   3   2   2   5   6
##  4:    4   2   1   1   1   3   3   1   1   2   1   5   1   5   4   7   3
##  5:    5   1   1   1   1   1   1   1   3   2   1   1   5   7   5   1   6
##  ---
## 246:    6   2   2   1   1   1   1   1   6   6   3   6   6   2   2   2   6
## 247:    6   1   1   1   1   1   1   1   2   6   1   1   6   6   5   2   6
## 248:    5   6   7   3   5   1   5   2   2   5   2   2   4   5   3   5   6
## 249:    7   6   3   7   5   4   4   2   6   6   5   7   7   7   3   6   7
## 250:    5   1   1   2   3   1   1   1   6   5   2   2   3   7   6   6   7
##      X35 X36 X37 X38
##  1:    7   2   4   7
##  2:    7   4   6   7
##  3:    7   5   5   7
##  4:    7   7   7   7
##  5:    6   3   5   7
##  ---
## 246:    6   5   5   5
## 247:    7   2   5   6
## 248:    5   5   4   5
## 249:    7   5   6   7
## 250:    7   2   6   7
```

```
exp.data <- item.level.data[experiment.data, on="SubjectNo"]
```

```
exp.data
```

```
##      SubjectNo X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17
##  1:          10  2  2  5  6  5  4  7  3  4  6  7  4  6  6  5  4  4
##  2:          11  4  2  4  5  2  5  6  5  4  6  6  3  6  5  5  5  5
##  3:          12  4  2  6  5  2  6  5  2  7  6  6  5  3  6  6  6  6
##  4:          13  5  2  3  7  2  5  5  1  3  4  6  2  4  6  6  5  3
##  5:          14  6  1  6  6  5  6  6  6  6  7  7  6  6  6  6  7  6
##  ---
## 245:         363  6  1  5  2  1  7  2  2  1  7  6  3  5  6  6  6  7
## 246:         364  5  3  6  4  3  6  4  4  6  7  5  6  6  6  6  7  7
```

```

## 247:      365  3  1  5  2  1  5  5  2  1  6  6  6  5  6  5  6  5
## 248:      366  5  3  5  4  2  5  5  3  3  5  6  5  6  5  6  6  5
## 249:      367  7  5  7  7  7  6  6  7  7  7  7  6  7  6  7  7  5
##      X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34
## 1:    5   2   1   5   7   6   5   2   4   6   4   3   4   6   5   7   7
## 2:    6   2   2   2   5   1   3   1   4   5   2   2   2   5   2   5   5
## 3:    5   4   2   4   3   1   5   2   7   6   5   6   6   6   4   5   6
## 4:    6   3   4   6   3   1   6   3   5   5   2   2   2   2   1   6   6
## 5:    6   1   1   1   1   1   1   1   3   1   5   1   2   6   6   7   7
## ---
## 245:    6   7   2   6   5   1   6   2   6   7   7   6   5   2   1   5   6
## 246:    4   7   6   7   5   4   6   3   5   7   6   1   7   6   5   5   6
## 247:    6   6   3   6   3   1   6   2   3   4   2   4   4   2   2   6   6
## 248:    6   5   4   5   3   1   5   3   5   6   4   5   3   4   3   5   6
## 249:    6   7   7   7   7   6   6   3   4   7   6   6   2   6   3   7   7
##      X35 X36 X37 X38 ACTIVE PERCEPTUAL MUSICAL SINGING EMOTIONS GENERAL
## 1:    6   6   6   7      38      47      28      33      39      75
## 2:    6   5   5   6      37      47      16      22      32      67
## 3:    7   2   4   6      39      49      21      40      30      82
## 4:    7   5   5   5      33      42      26      19      34      68
## 5:    6   6   7   2      48      57       7      24      35      60
## ---
## 245:    6   5   7   6      27      52      29      34      35      88
## 246:    6   5   6   7      41      54      38      37      35      96
## 247:    5   3   6   6      25      51      27      21      32      61
## 248:    6   5   7   5      35      50      26      30      34      82
## 249:    7   6   7   7      59      58      43      34      41     103
##      inst Age Gender                      Major
## 1:  clarinet 29 Female                      BUSINESS MANAGEMENT
## 2:    bass  22 Female                      psychology
## 3:   Voice  19 Female                      COMMUNICATION STUDIES
## 4: saxophone 23   Male                      Mechanical Engineering
## 5:    none  23   Male                      Interdisciplinary Studies
## ---
## 245:   voice 21   Male                      psychology
## 246:   voice 22 Female Psychology and English Literature
## 247:   flute 22 Female                      PSYCHOLOGY
## 248: saxophone 21 Female                      psychology
## 249:  Bassoon 23 Female                      Bassoon Performance
##      Minor SchoolUGorG MuscMajMinor BeginTrain
## 1: HUMAN RESOURCE MANAGEMENT              1              1              9
## 2:                                NA              1              10
## 3:                                SOCIAL WORK  1              10
## 4:                                NA              1              11
## 5:                                Anthropology 1              0
## ---
## 245:                                1              1              13
## 246:                                Sociology  1              1              9
## 247:                                CHEMISTRY  1              1              9
## 248:                                theatre    1              1              12
## 249:                                2              2              8
##      AbsPitch YearsPrac PeakPrac Live12Mo TheoryTrain FormalYrs NoOfInst
## 1:    no              4          5          9          4          4          1
## 2:    no              1          2         15          0          1          0

```

```

## 3:      no      3      1      3      0      3      1
## 4:      no      7      1      2      0      7      2
## 5:      no      0      0     20      0      0      0
## ---
## 245:     no      9      2      3      0      9      1
## 246:     no     14      2     12      2      8      2
## 247:     no      8      1      3      0      9      1
## 248:     no      4      1      9      0      3      2
## 249:     no     12      5     16      6      9      2
##      ActListenMin i.SubjectNo SubRaven RavenB1 RavenB2 RavenB3 RavenTotal
## 1:           60           10      10      10      8      10      28
## 2:           60           11      11      7      10      6      23
## 3:          180           12      12      3      7      7      17
## 4:           45           13      13     10     10     11     31
## 5:          120           14      14      7      8      5     20
## ---
## 245:           6          363      363      6      7      6     19
## 246:          160          364      364      7      8      8     23
## 247:           10          365      365     10     10      9     29
## 248:           35          366      366     10      7      9     26
## 249:          240          367      367      7      6      6     19
##      OspanSub OspanAccError OspanMathError OspanAbsoluteScore
## 1:           10              8              12              29
## 2:           11              7              12              20
## 3:           12              7              7              48
## 4:           13              1              2              68
## 5:           14              2              2              33
## ---
## 245:          363              2              4              18
## 246:          364              1              1              68
## 247:          365              2              8              4
## 248:          366              4              7              38
## 249:          367              1              2              54
##      OspanPartialScore OspanB1 OspanB2 OspanB3 OspanSpeedErrTot SymSub
## 1:              47      21      14      12              4      10
## 2:              41      11      18      12              5      11
## 3:              65      21      23      21              0      12
## 4:              73      25      23      25              1      13
## 5:              50      17      16      17              0      14
## ---
## 245:              28      11      8      9              2     363
## 246:              72      22      25      25              0     364
## 247:              21      10      5      6              6     365
## 248:              58      22      18      18              3     366
## 249:              66      25      22      19              1     367
##      SymAccErrTot SymSpeedErrTot SspanAbsScore SspanPartiaScore SspanB1
## 1:              1              0              10              19      8
## 2:              4              1              15              29     11
## 3:              2              0              21              34     10
## 4:              1              0              27              34     13
## 5:              0              0              15              26      8
## ---
## 245:              0              0              7              21      8
## 246:              0              0             32              38     14

```

```

## 247:      0      0      18      28      9
## 248:      2      1      33      40     14
## 249:      2      0      17      28     10
##      SspanB2 SspanB3 SymmErrTot ToneSub ToneAccErrTot ToneMathErrTot
## 1:      4      7      1     10      8      9
## 2:     13      5      5     11      5      5
## 3:     13     11      2     12      8      8
## 4:     10     11      1     13      1      2
## 5:      8     10      0     14      4      5
## ---
## 245:      7      6      0    363      7      8
## 246:     13     11      0    364      3      5
## 247:      8     11      0    365      3      4
## 248:     13     13      3    366      6      9
## 249:     10      8      2    367      2      2
##      TspanAbsScore TspanPartScore TspanB1 TspanB2 TspanB3 ToneSpeedErrTot
## 1:      9      36     13      6     17      1
## 2:     21     47     14     19     14      0
## 3:     27     55     17     21     17      0
## 4:     37     63     20     22     21      1
## 5:     10     31      8      9     14      1
## ---
## 245:      6     37     15     13      9      1
## 246:     55     67     22     24     21      2
## 247:     21     38      9     13     16      1
## 248:     32     56     17     16     23      3
## 249:     58     67     23     23     21      0
##      BeatSub BeatTotal BeatAcc MelodicSub MelodicTot MelodicAcc Native
## 1:     10      11    0.61      10      7     0.54      1
## 2:     11      10    0.56      11      7     0.54      1
## 3:     12      14    0.78      12      6     0.46      1
## 4:     13      13    0.72      13     10     0.77      1
## 5:     14      14    0.78      14      5     0.38      1
## ---
## 245:    363      16    0.89     363      7     0.54      1
## 246:    364      11    0.61     364      7     0.54      1
## 247:    365      10    0.56     365     10     0.77      1
## 248:    366       9    0.50     366      9     0.69      1
## 249:    367      12    0.67     367      9     0.69      1
##      Paid jv_NoGMSI jv_nonNative jv_noOspan
## 1:      1      0      0      0
## 2:      1      0      0      0
## 3:      1      0      0      0
## 4:      1      0      0      0
## 5:      1      0      0      0
## ---
## 245:      2      0      0      0
## 246:      2      0      0      0
## 247:      2      0      0      0
## 248:      2      0      0      0
## 249:      2      0      0      0

```

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



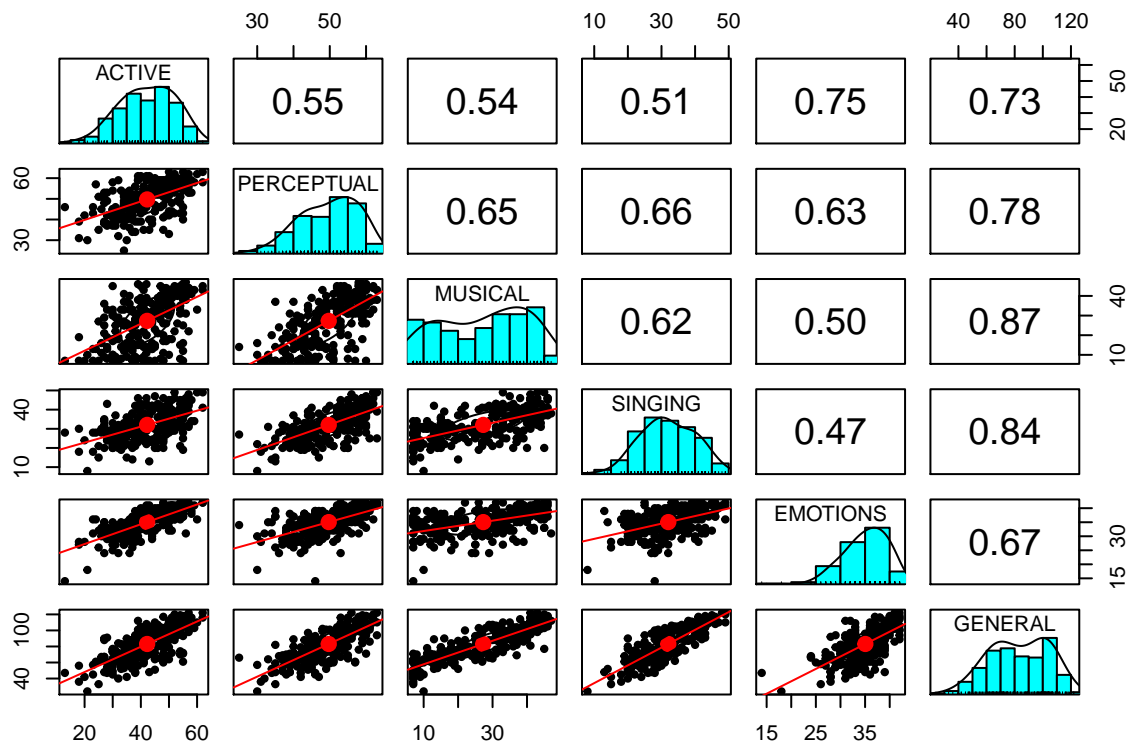
big tour du R right now and learning to think

```
exp.data <- data.frame(exp.data)
exp.data <- exp.data[,c(1,40:100,2:39)]
```

### 3.3 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` package 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 arguments. The first is what you want to manipulate, the second is if it's rows 1 or columns 2 (remember 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 threshold we want!*

```
table(gmsi.z.scores > 2)
```

```
##
```

```
## FALSE TRUE
```

```
## 1487      6
gmsi.z.indexer <- gmsi.z.scores > 2
gmsi.z.scores[gmsi.z.indexer] # See what they are, find them , decide to get rid of

## [1]      NA 2.053441 2.053441 2.142711 2.016431 2.142711 2.142711
```

### 3.3.1 Checking for Multivariate Outliers

A bit trickier, I learned 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 Measures

cutoff <- qchisq(.999, ncol(gmsi.responses))                         ## Create cutoff object .00
summary(mahal < cutoff)                                             ## 11 Subjects greater than

##      Mode  FALSE    TRUE
## logical    11     238

# 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]
```

##	SubjectNo	ACTIVE	PERCEPTUAL	MUSICAL	SINGING	EMOTIONS	GENERAL
##	1:	10	38	47	28	33	75
##	2:	11	37	47	16	22	67
##	3:	12	39	49	21	40	82
##	4:	13	33	42	26	19	68
##	5:	15	47	55	8	24	65
##	---						
##	234:	363	27	52	29	34	88
##	235:	364	41	54	38	37	96
##	236:	365	25	51	27	21	61
##	237:	366	35	50	26	30	82
##	238:	367	59	58	43	34	103
##		inst	Age	Gender		Major	
##	1:	clarinet	29	Female		BUSINESS MANAGEMENT	
##	2:	bass	22	Female		psychology	
##	3:	Voice	19	Female		COMMUNICATION STUDIES	
##	4:	saxophone	23	Male		Mechanical Engineering	
##	5:	piano	19	Female		marketing	
##	---						
##	234:	voice	21	Male		psychology	
##	235:	voice	22	Female	Psychology and English	Literature	
##	236:	flute	22	Female		PSYCHOLOGY	
##	237:	saxophone	21	Female		psychology	
##	238:	Bassoon	23	Female		Bassoon Performance	
##					Minor SchoolUGorG	MuscMajMinor	BeginTrain
##	1:	HUMAN RESOURCE MANAGEMENT			1	1	9

```

## 2:          NA          1          1          10
## 3:        SOCIAL WORK    1          1          10
## 4:          NA          1          1          11
## 5:          NA          1          1           0
## ---
## 234:          1          1          13
## 235:        Sociology    1          1           9
## 236:        CHEMISTRY    1          1           9
## 237:         theatre    1          1          12
## 238:          2          2           8
##      AbsPitch YearsPrac PeakPrac Live12Mo TheoryTrain FormalYrs NoOfInst
## 1:         no         4         5         9         4         4         1
## 2:         no         1         2        15         0         1         0
## 3:         no         3         1         3         0         3         1
## 4:         no         7         1         2         0         7         2
## 5:         no         0         0         9         0         0         0
## ---
## 234:         no         9         2         3         0         9         1
## 235:         no        14         2        12         2         8         2
## 236:         no         8         1         3         0         9         1
## 237:         no         4         1         9         0         3         2
## 238:         no        12         5        16         6         9         2
##      ActListenMin i.SubjectNo SubRaven RavenB1 RavenB2 RavenB3 RavenTotal
## 1:             60             10         10         10         8         10         28
## 2:             60             11         11         7         10         6         23
## 3:            180             12         12         3         7         7         17
## 4:             45             13         13        10         10        11         31
## 5:             60             15         15         6         10         6         22
## ---
## 234:             6          363          363         6         7         6         19
## 235:            160          364          364         7         8         8         23
## 236:             10          365          365        10        10         9         29
## 237:             35          366          366        10         7         9         26
## 238:            240          367          367         7         6         6         19
##      OspanSub OspanAccError OspanMathError OspanAbsoluteScore
## 1:          10             8             12             29
## 2:          11             7             12             20
## 3:          12             7              7             48
## 4:          13             1              2             68
## 5:          15            12             15             20
## ---
## 234:          363             2              4             18
## 235:          364             1              1             68
## 236:          365             2              8              4
## 237:          366             4              7             38
## 238:          367             1              2             54
##      OspanPartialScore OspanB1 OspanB2 OspanB3 OspanSpeedErrTot SymSub
## 1:                   47        21        14        12             4        10
## 2:                   41        11        18        12             5        11
## 3:                   65        21        23        21             0        12
## 4:                   73        25        23        25             1        13
## 5:                   44        14        17        13             3        15
## ---
## 234:                   28        11         8         9             2       363

```

```

## 235:      72      22      25      25      0      364
## 236:      21      10       5       6      6      365
## 237:      58      22      18      18      3      366
## 238:      66      25      22      19      1      367
##      SymAccErrTot SymSpeedErrTot SspanAbsScore SspanPartiaScore SspanB1
## 1:      1      0      10      19      8
## 2:      4      1      15      29     11
## 3:      2      0      21      34     10
## 4:      1      0      27      34     13
## 5:      4      1       9      25      9
## ---
## 234:      0      0       7      21      8
## 235:      0      0      32      38     14
## 236:      0      0      18      28      9
## 237:      2      1      33      40     14
## 238:      2      0      17      28     10
##      SspanB2 SspanB3 SymmErrTot ToneSub ToneAccErrTot ToneMathErrTot
## 1:      4      7      1      10      8      9
## 2:     13      5      5      11      5      5
## 3:     13     11      2      12      8      8
## 4:     10     11      1      13      1      2
## 5:     10      6      5      15      7     10
## ---
## 234:      7      6      0     363      7      8
## 235:     13     11      0     364      3      5
## 236:      8     11      0     365      3      4
## 237:     13     13      3     366      6      9
## 238:     10      8      2     367      2      2
##      TspanAbsScore TspanPartScore TspanB1 TspanB2 TspanB3 ToneSpeedErrTot
## 1:      9      36     13      6     17      1
## 2:     21     47     14     19     14      0
## 3:     27     55     17     21     17      0
## 4:     37     63     20     22     21      1
## 5:      4     34      9     12     13      3
## ---
## 234:      6      37     15     13      9      1
## 235:     55     67     22     24     21      2
## 236:     21     38      9     13     16      1
## 237:     32     56     17     16     23      3
## 238:     58     67     23     23     21      0
##      BeatSub BeatTotal BeatAcc MelodicSub MelodicTot MelodicAcc X1 X2 X3
## 1:     10      11    0.61      10      7    0.54 2 2 5
## 2:     11      10    0.56      11      7    0.54 4 2 4
## 3:     12      14    0.78      12      6    0.46 4 2 6
## 4:     13      13    0.72      13     10    0.77 5 2 3
## 5:     15      10    0.56      15      5    0.38 7 2 5
## ---
## 234:     363      16    0.89     363      7    0.54 6 1 5
## 235:     364      11    0.61     364      7    0.54 5 3 6
## 236:     365      10    0.56     365     10    0.77 3 1 5
## 237:     366       9    0.50     366      9    0.69 5 3 5
## 238:     367      12    0.67     367      9    0.69 7 5 7
##      X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 X21 X22
## 1:  6  5  4  7  3  4  6  7  4  6  6  5  4  4  5  2  1  5  7

```

```

##      2:  5  2  5  6  5  4  6  6  3  6  5  5  5  5  6  2  2  2  5
##      3:  5  2  6  5  2  7  6  6  5  3  6  6  6  6  5  4  2  4  3
##      4:  7  2  5  5  1  3  4  6  2  4  6  6  5  3  6  3  4  6  3
##      5:  7  5  7  7  3  4  7  7  3  5  6  7  7  7  6  2  1  1  1
##      ---
## 234:  2  1  7  2  2  1  7  6  3  5  6  6  6  7  6  7  2  6  5
## 235:  4  3  6  4  4  6  7  5  6  6  6  6  7  7  4  7  6  7  5
## 236:  2  1  5  5  2  1  6  6  6  5  6  5  6  5  6  6  3  6  3
## 237:  4  2  5  5  3  3  5  6  5  6  5  6  6  5  6  5  4  5  3
## 238:  7  7  6  6  7  7  7  7  6  7  6  7  7  5  6  7  7  7  7
##      X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34 X35 X36 X37 X38
##      1:   6   5   2   4   6   4   3   4   6   5   7   7   6   6   6   7
##      2:   1   3   1   4   5   2   2   2   5   2   5   5   6   5   5   6
##      3:   1   5   2   7   6   5   6   6   6   4   5   6   7   2   4   6
##      4:   1   6   3   5   5   2   2   2   2   1   6   6   7   5   5   5
##      5:   1   1   1   1   7   3   1   1   6   5   5   7   7   5   5   7
##      ---
## 234:   1   6   2   6   7   7   6   5   2   1   5   6   6   5   7   6
## 235:   4   6   3   5   7   6   1   7   6   5   5   6   6   5   6   7
## 236:   1   6   2   3   4   2   4   4   2   2   6   6   5   3   6   6
## 237:   1   5   3   5   6   4   5   3   4   3   5   6   6   5   7   5
## 238:   6   6   3   4   7   6   6   2   6   3   7   7   7   6   7   7
##      mahal
##      1: 51.21448
##      2: 25.99414
##      3: 29.78012
##      4: 49.15429
##      5: 43.12685
##      ---
## 234: 58.63224
## 235: 35.67417
## 236: 34.53619
## 237: 17.72908
## 238: 20.60059

```

### 3.3.2 Checking for Skew and Kurtosis

```
apply(gmsi.responses, 2, skew)
```

```

##      X1      X2      X3      X4      X5
## -0.751959493  0.403676499 -1.078653410 -0.813983811  0.067614732
##      X6      X7      X8      X9     X10
## -1.133338356 -1.056888584  0.003114209  0.174997119 -1.251450954
##      X11     X12     X13     X14     X15
## -1.209870193 -0.643154118 -1.046476716 -1.023134965 -1.512836098
##      X16     X17     X18     X19     X20
## -0.918246164 -0.549121749 -0.931025766 -0.607086647 -0.009797227
##      X21     X22     X23     X24     X25
## -0.324554196 -0.342355019  0.502388714 -0.404073175  1.103530636
##      X26     X27     X28     X29     X30
## -0.419299836 -1.204955610 -0.411695100 -0.409111747 -0.082358311
##      X31     X32     X33     X34     X35
## -1.037773224  0.068284056 -1.083784471 -1.825310783 -1.693280548

```

```
##           X36           X37           X38
## -0.651012271 -1.218176414 -2.364635137
```

```
apply(gmsi.responses, 2 , kurtosi)
```

```
##           X1           X2           X3           X4           X5           X6
## -0.27214952 -0.83327603  1.14629947  0.02931285 -1.30675305  0.83305054
##           X7           X8           X9           X10          X11          X12
##  1.36849018 -0.92937707 -0.69656989  1.58298149  2.00805464 -0.58320443
##           X13          X14          X15          X16          X17          X18
##  0.49964877  1.30350514  2.47139233  0.56622834 -0.89787441  0.80425156
##           X19          X20          X21          X22          X23          X24
## -1.18236897 -1.67596746 -1.49414255 -1.24558633 -1.41680710 -1.30953622
##           X25          X26          X27          X28          X29          X30
##  0.42141756 -0.69749482  0.73800105 -1.08494701 -0.90943305 -1.31818613
##           X31          X32          X33          X34          X35          X36
##  0.86327735 -1.00536155  0.46640336  4.08373503  4.90078751 -0.09764595
##           X37           X38
##  2.04104393  9.60389299
```

### 3.3.3 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, "datasets/My_Experiment_Data.csv")
```

## Chapter 4

# Epistemology of Statistics

Provide summary here of Dienes book. Just want to note a bit on the history of p values (there is split in world) and note example of probability coming from the fundamental nature of the object itself, or as a product of observation.

Could just just jump straight into things, but I think having a bit of familiarity with the philosophy of it all actually really helps.

### 4.1 Karl Popper

### 4.2 R.A. Fisher

### 4.3 Neyman-Pearson

### 4.4 Bayes





## 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.

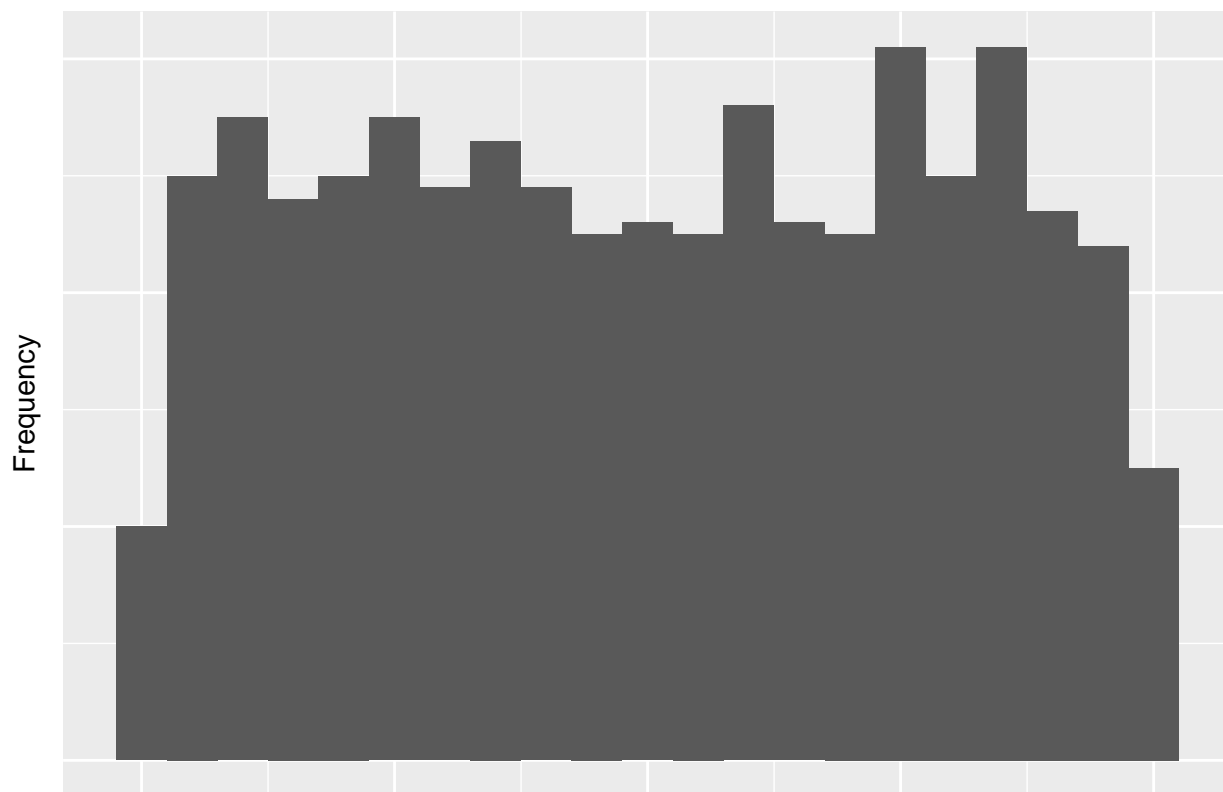
The first, and probably least likely distribution you will find is the **uniform** or **rectangular distribution**. We can create this plot and the others by using the distribution functions from R's functionality. In each case we are going to take 1,000 samples from 0 to 1. We'll plot everything using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

Uniform Distribution



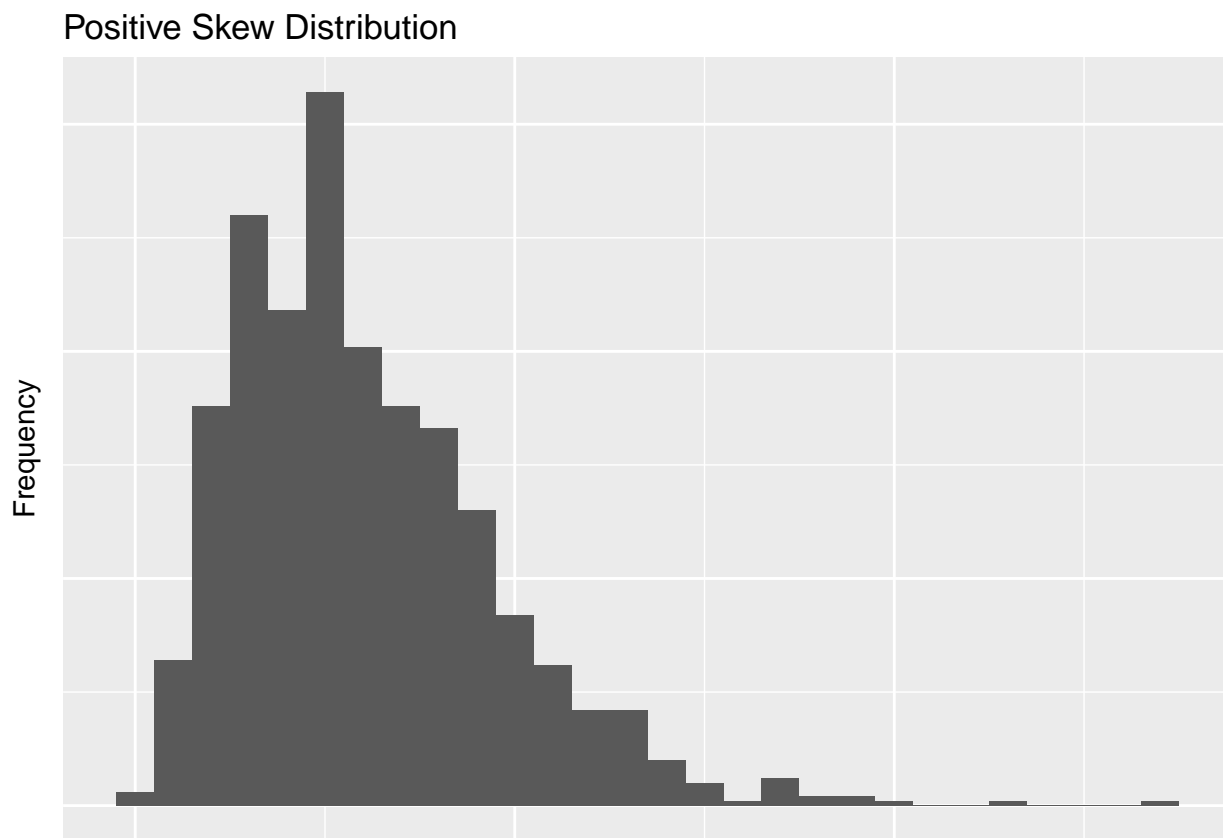
Try to run the above code with different arguments in the **binwidth** argument. You'll notice that the way you plot the data will actually represent it differently.

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") +
  cleanUpP

```

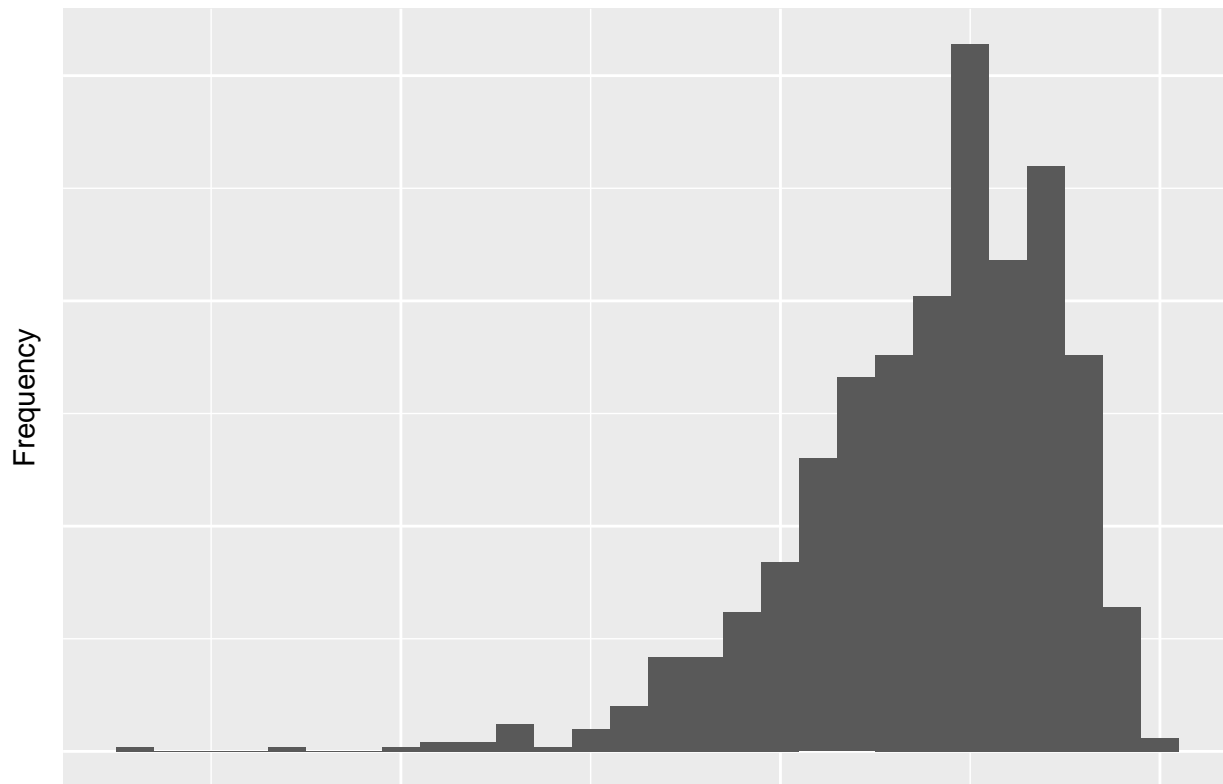


```

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

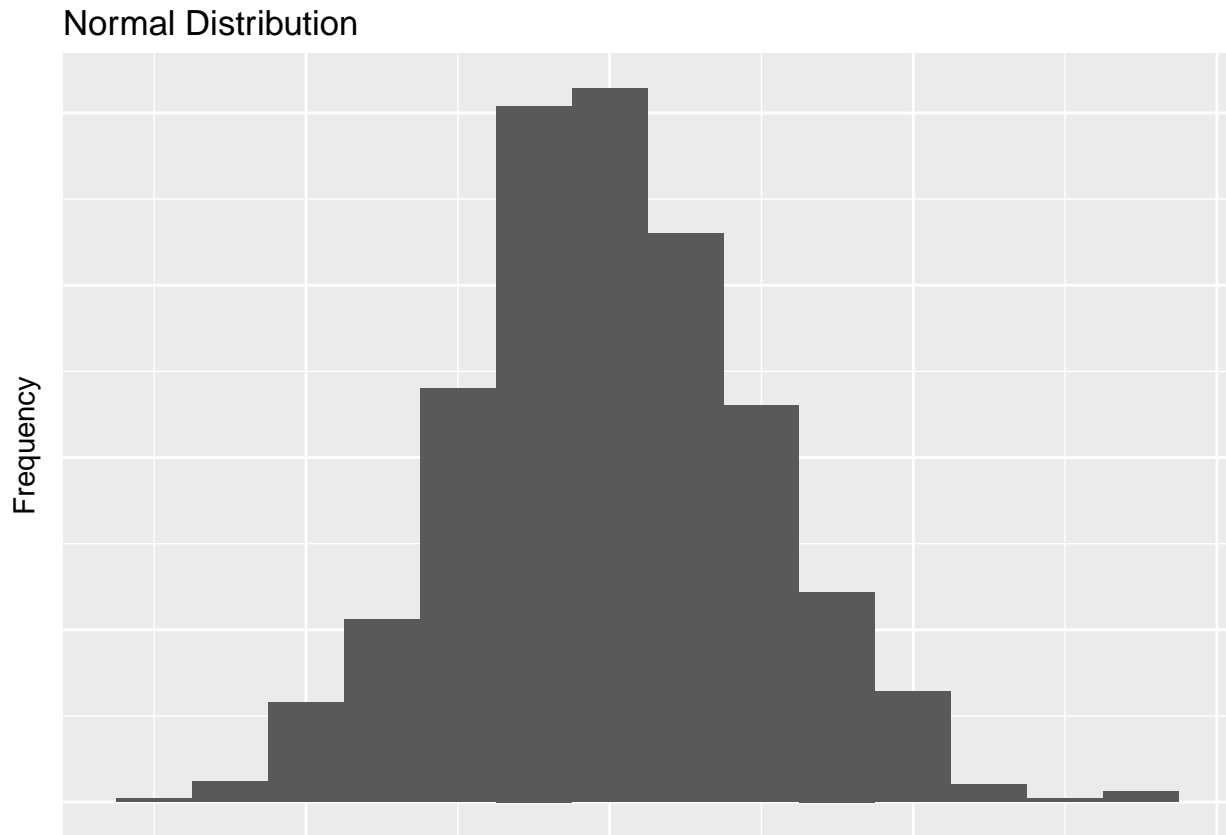
```

## Negative Skew Distribution



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

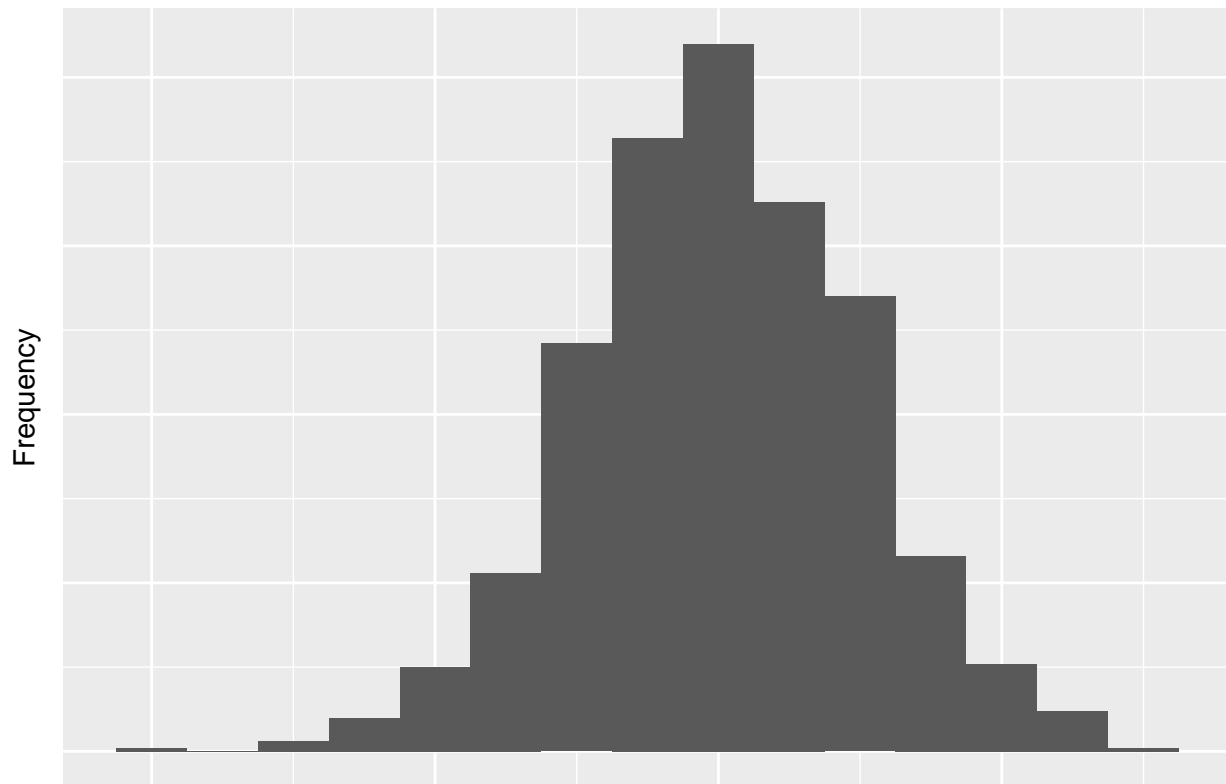
```
normalData <- rnorm(n = 1000, mean = 0, sd = 2) # Flip it!
distributions <- data.frame(uniformData, positiveSkewData, negativeSkewData, normalData)
ggplot(distributions, aes(distributions$normalData)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Independent Variable", y = "Frequency", title = "Normal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



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

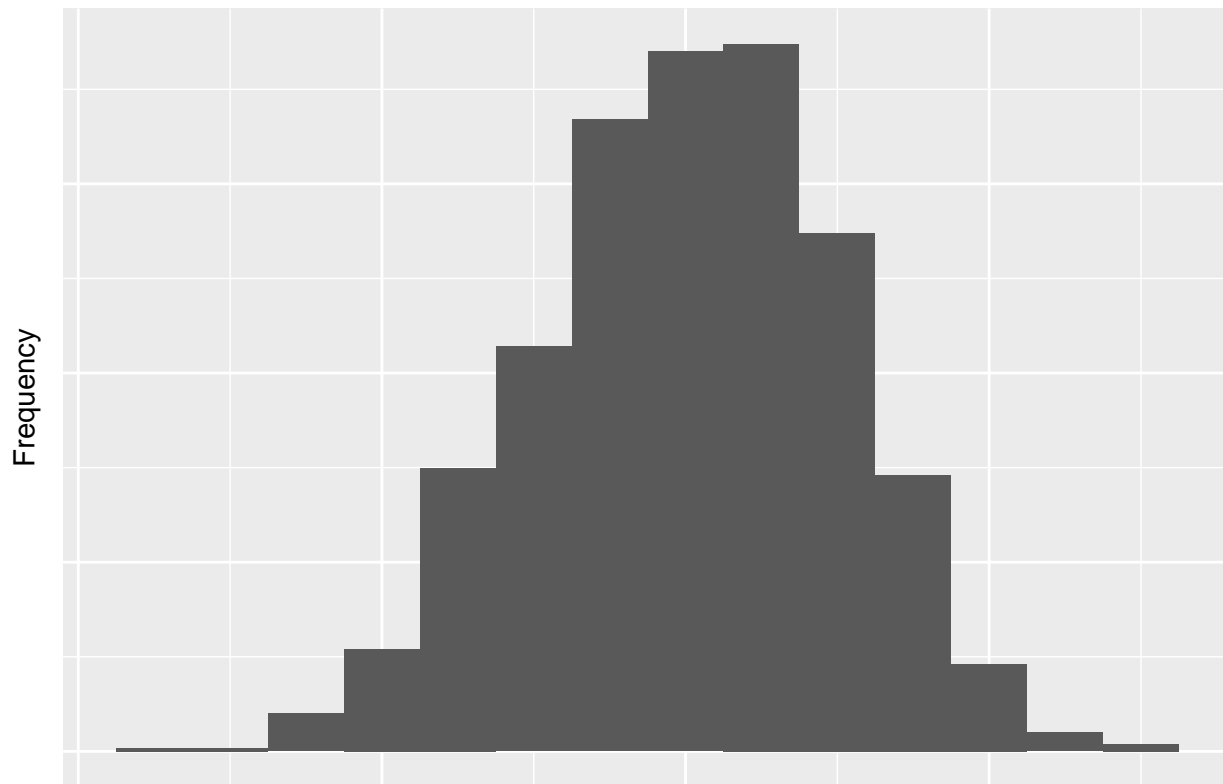
```
leptoData <- rnorm(n = 1000, mean = 0, sd = 2)
platyData <- rnorm(n = 1000, mean = 0, sd = 2)
distributions <- data.frame(uniformData, positiveSkewData, negativeSkewData, normalData, leptoData, platyData)
ggplot(distributions, aes(distributions$leptoData)) +
  geom_histogram(binwidth = 1) +
  labs(x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

## Leptokurtic Distribution, FIX ME



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

## Platykurtic Distribution, FIX ME



Generally measures 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  $\Sigma$  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  $\Sigma$  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
```

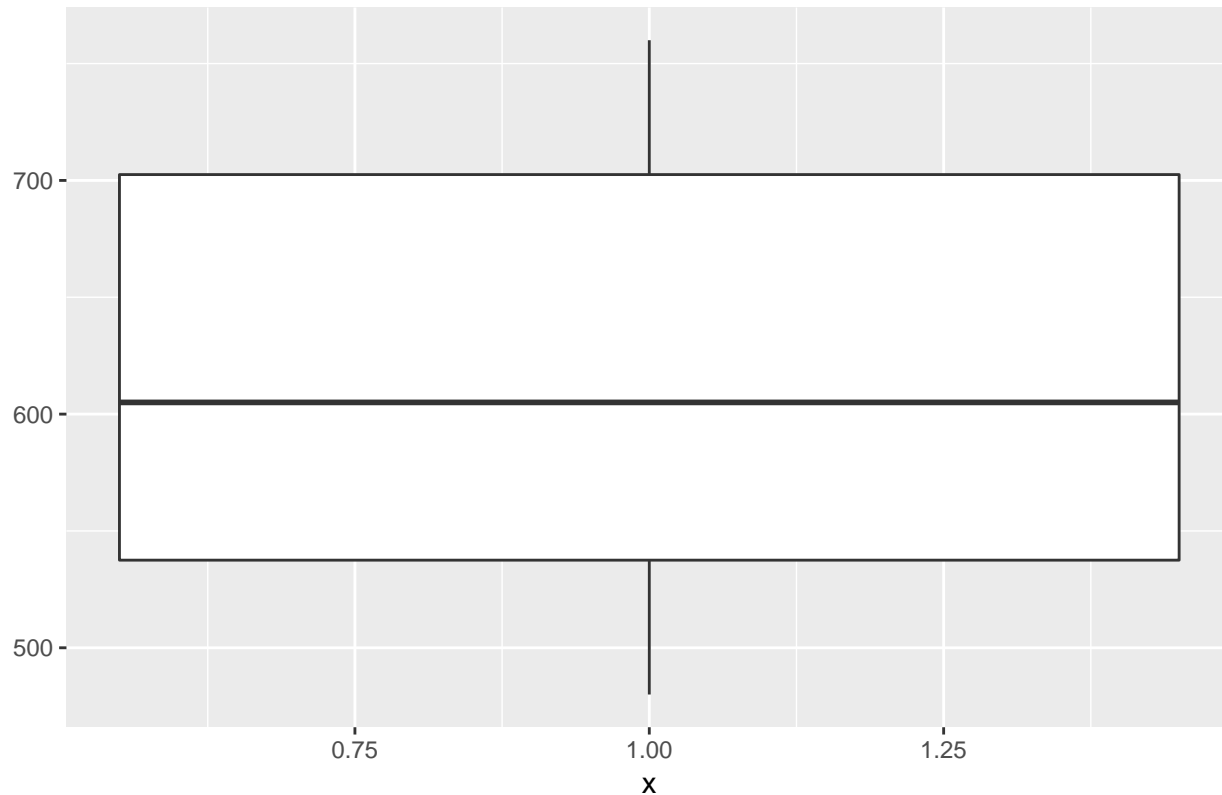
The interquartile range represents the spread between the score at the 75th and 25th percentiles. Boxplots are often used to graphically represent inner 50% of the scores.

```
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"))
```

Example of Boxplot



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  $\sigma^2$  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 theorem 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  $\sigma^2$  refers to the Platonic value and the  $s^2$  is the sample. They are defined as follows:

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

$$s^2 = \frac{\sum (X_i - \bar{X})^2}{n - 1}$$

Note here that each of these formulas needs a mean. In the population equation that is defined as  $\mu$  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 calculations below we need to know the

mean which we calculated above to be 616.67.

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

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

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.

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

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

$$s = \sqrt{s^2} = \sqrt{12346.67} = 111.12$$

Or run it in R and get

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

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

**Standard scores** represent the distance of raw scores from 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 population 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 separate 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.

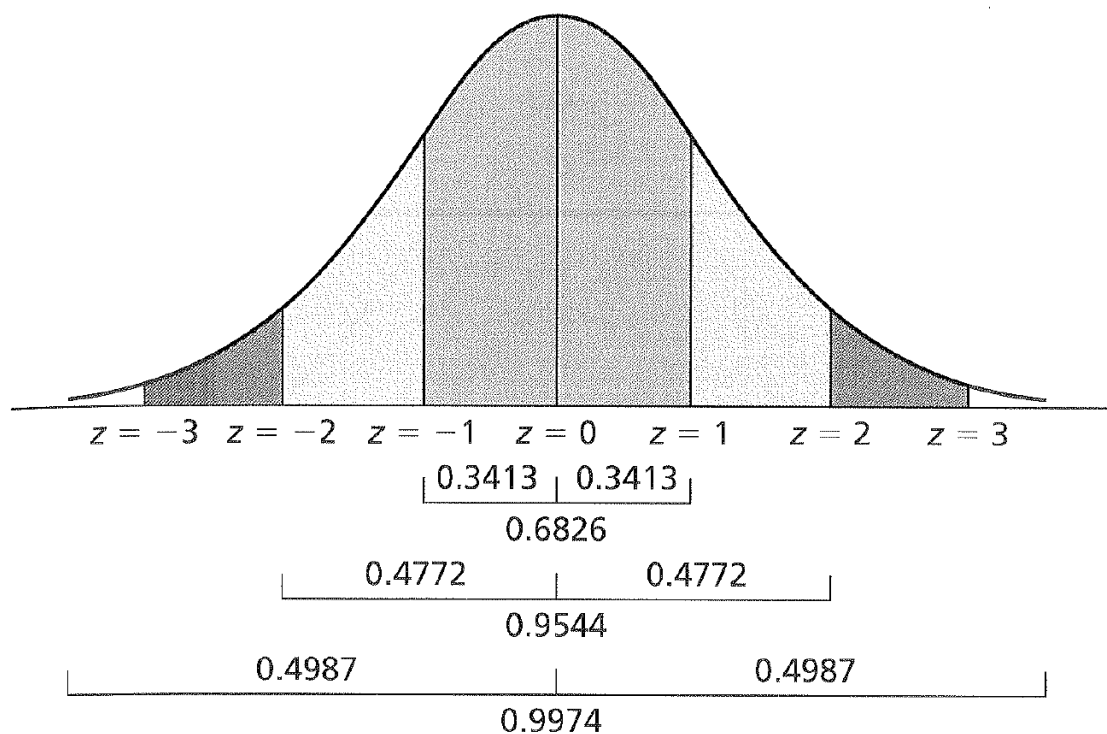


Figure 5.1: z Scores and Areas Under the Curve

The normal distribution is unimodal, symmetrical, bell shaped, with a maximum height at the mean. The normal distribution is continuous and additionally the 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  $\mu$  and standard deviations  $\sigma$ . 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.

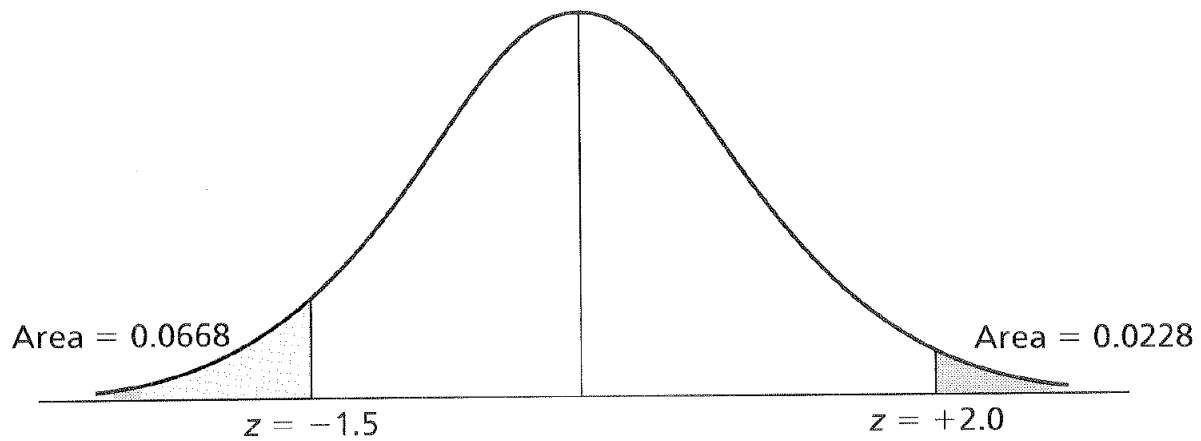
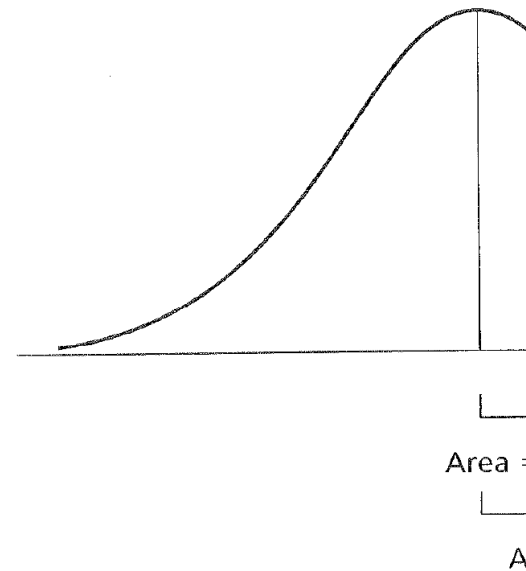


Figure 5.2: z Scores and Areas Under the Curve



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.

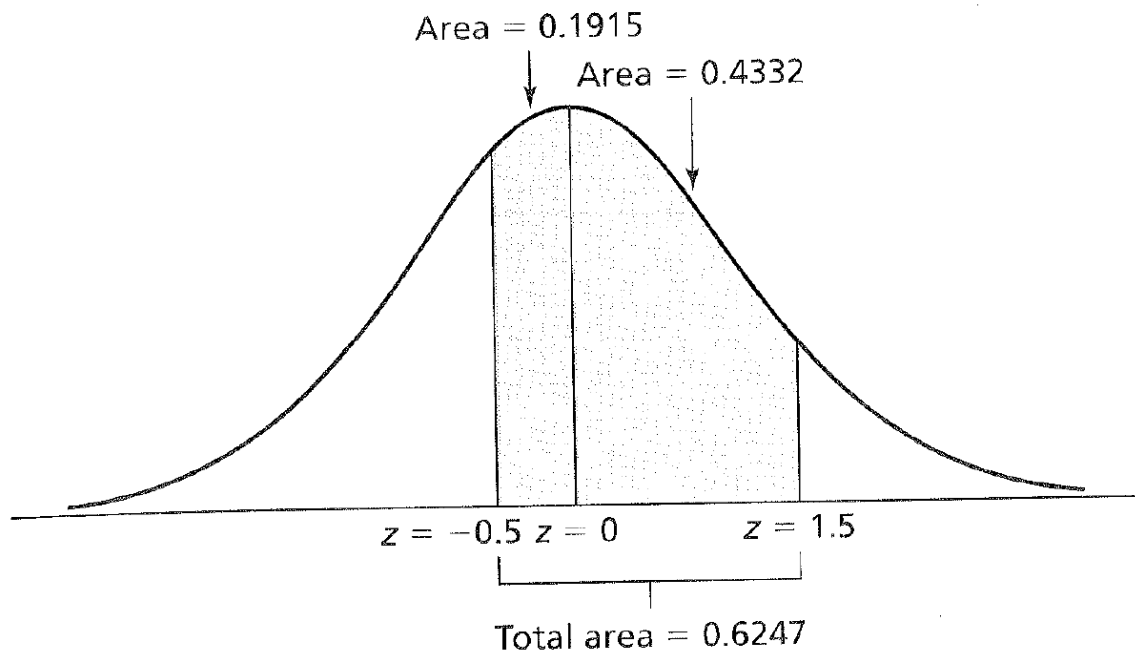


Figure 5.3: z Scores and Areas Under the Curve

During tryouts, a sample of ballet dancers were rated on their athletic ability and overall 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 calculate 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 55th percentile? 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 algebra

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

```
(.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 distribution, 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  $\mu$ .

The idea of how frequentist inferential statistics is as follows. Samples must be selected *randomly* in order to make appropriate inferences about the parent population. Sample estimates must be compared to an underlying distribution of estimates of all other hypothetical samples of that same size from 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 population 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 demonstrated 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-