Or At Least The Really New

Shep Sheppard

FORWARD ONE

Why are you here? This book is written for someone who has an interest in Data Science (DS), but no clue what it really is, or how to get started. It is written for someone who has never taken a serious "DS" class, but has an interest. Your next steps after this book is to continue to apply the things you have learned at home or at work. This may compel you to start a career in data science, though i will be the first to tell you that will require significantly more education than any one book can provide. I think i have somewhere around 50 books i have purchased in the last three years. Sometimes i will buy them realize i am in over my head, put them down and come back to them in six months and come back to them later, don't be afraid to do the same thing.

Why did i bother with a book? Well, i need it as a companion to a one week class i teach for Intro to Data Science class for professionals. These are folks that already have jobs and their company is looking to expand their skills a bit, but now pay for or wait for a year of training, and see paragraph one.

Whats next? After you have been through this book, read more books and take some classes, online, in person or whatever. If you take a class and find out you are in over your head withdraw as quickly as possible and take a lower level class. There is no shame in it, no one will care, you will be better in the long run, just don't give up. When you take a class, don't just learn to get through it, learn it as if you had to teach it.

Forward Two

In case you do not read any further, know this, i will not use math notation, i will not use PhD speak. This is an introduction to data science for the common folk. Those of us who work too many hours, and need to have a clue as fast as we can without spending the next year in school. This book is just to break down some barriers and give you a un-intimidating peak into the field.

The first item we have to come to terms with is what is Data Science? There is a certain audacity in a book title actually being called Data Science for the Absolute Beginner, only because Data Science is an umbrella term or as Wikipedia states “is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured, similar to data mining.”[1].

Well, that was super helpful, having spent the last couple years trying to wrap my brain around it, let me say this; Data Science is an interdisciplinary field that is made up of statistics, probability, linear algebra, matrix algebra and programming. The expectation is to attempt to detect patterns in data, be it the iris data set or edge device such as a sensor in an electric generator or maybe even your electric meter on your house.

The Wikipedia Data Science page goes on to states that “Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science.” [1].

I do like this definition and in my experience it is accurate, though one key piece of information that is frequently overlooked in the sweeping generalizations of Data Science and Machine learning are what i have come to call the pillars of Data Science. In short, the pillars are Visualization, Mathe-

matics, Database, Software Engineering, Business Analyst. A data Scientist encompasses many if not all of these traits in some capacity. A Data Scientist will typically be an expert in the Mathematics pillar with significant knowledge of all of the other pillars. At some level i will dive in and out off all of these pillars in this book. I will go back and forth between Python and R, as i think you should learn them both and have some level of knowledge in both.

I am not going to make you a Data Scientist in this book, but i will give you a clue and i will break down some of the barriers in communication. What i will not be doing is using PhD speak and i will not be using math notation. There are thousands of books that have LaTek and lots of PhD speak, you can buy those and smash your own brain with them, i'm not going to do that. If you prefer math notation, this is not the book for you.

Lets get to it.

Chapter 1

TOOLS

As crazy as it sounds the one thing i do not want to do is spend an entire chapter on tools. They are going to change with time, and even if i provide you with uber detailed instructions on where and how to get everything installed by the time i use this book in my first class the instructions will be obsolete. So what to do?

Well, get **R** from here https://www.r-project.org/ and **R Studio** from here https://www.rstudio.com/products/RStudio/, down load the latest version of each. For Python i use **Spyder** which can be found at https://www.spyder-ide.org/ and **jupyter notebook** which can be found here http://jupyter.org/.

R Studio is the IDE for R, and Spyder is the IDE for python, and jupyter is the notebook environment I will use for Python and R.

If you can get all of those installed and working you are ready, the packages we will be using will change month to month and i cannot guarantee anything i demonstrate will work by the time you get to the chapter. That is the worst thing about these tools and packages and the absolute worst thing about open source, one update to R or R Studio can actually break the very code that has been working for weeks, months or years, so keep that in mind as you work and update.

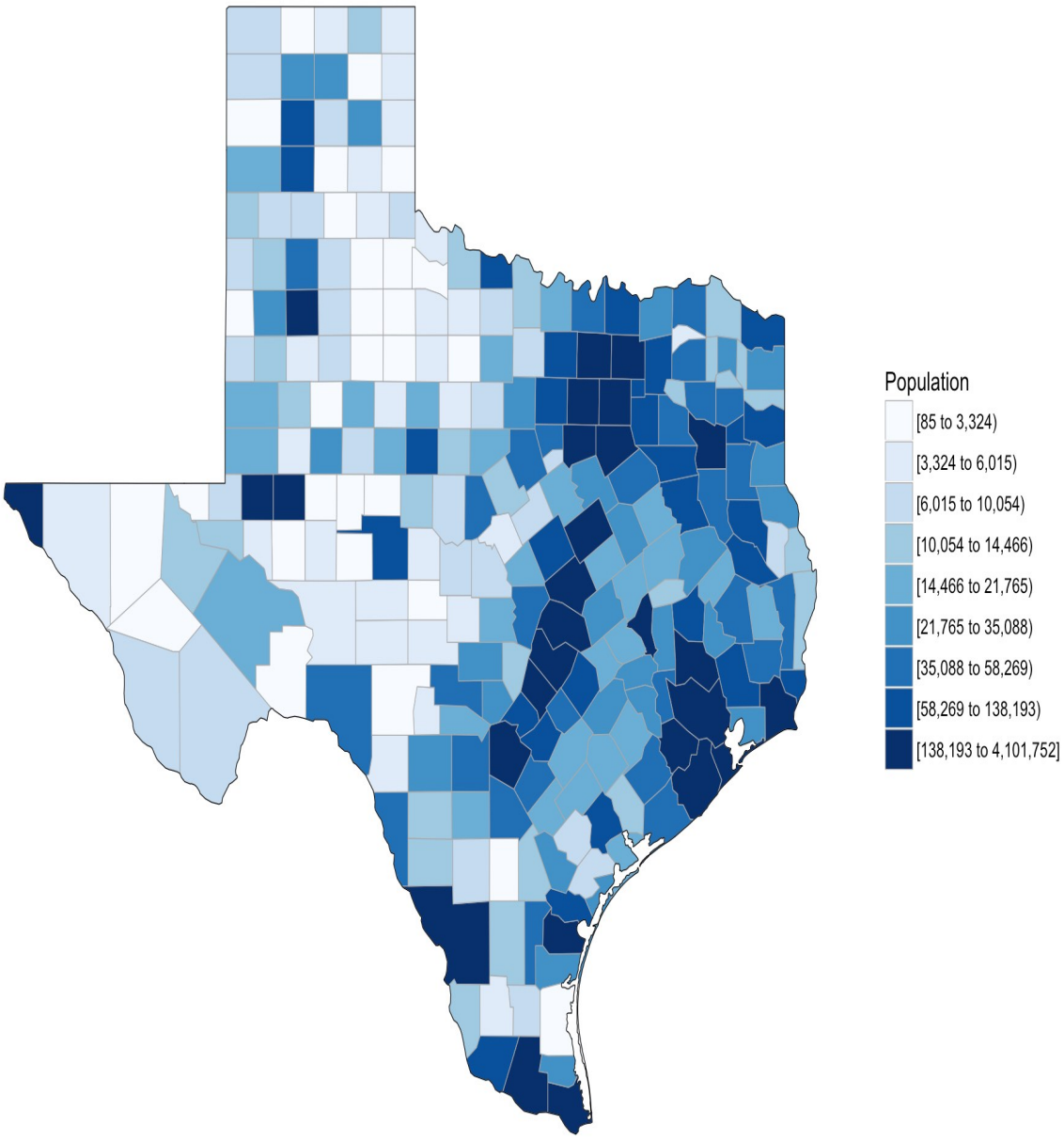
You can find the github site for this book at https://github.com/sqlshep/DS4New

Chapter 2

THE GATEWAY DRUG

Visualization Is The Gateway Drug To Statistics

Population Density of Texas



What if i told you that you can create this map with one command?

Section 1

CHOROPLETH

The main point of this chapter is to show you how much you can do with very little. We will get into the hard core (introductory) data science stuff soon, but first lets play. R need not be intimidating you just have to find something fun to do with it and off you will go.

I'm actually not sure who said visualization is the gateway drug to statistics but i like it, and its true. If i find out i will add an online errata to the book. But the best way to break down the barriers of R and the scary statistics is to visualize something, so lets get started.

I expect that after you see how easy some visuals are in R you will be off and running with your own data explorations. Data visualization is one of the Data Science pillars, so it is critical that you have a working knowledge of as many visualizations as you can, and be able to produce as many as you can. Even more important is the ability to identify a bad visualization, if for no other reason to make certain you do not create one and release it into the wild, there is a site for those people, don't be those people! Edward Tufte has done some great work in the field of visualizations, one phrase i want to introduce to you is chart junk. Just knowing that phrase exists will make you're visualizations better, you certainly do not want to have one of your visualizations end up on http://viz.wtf.

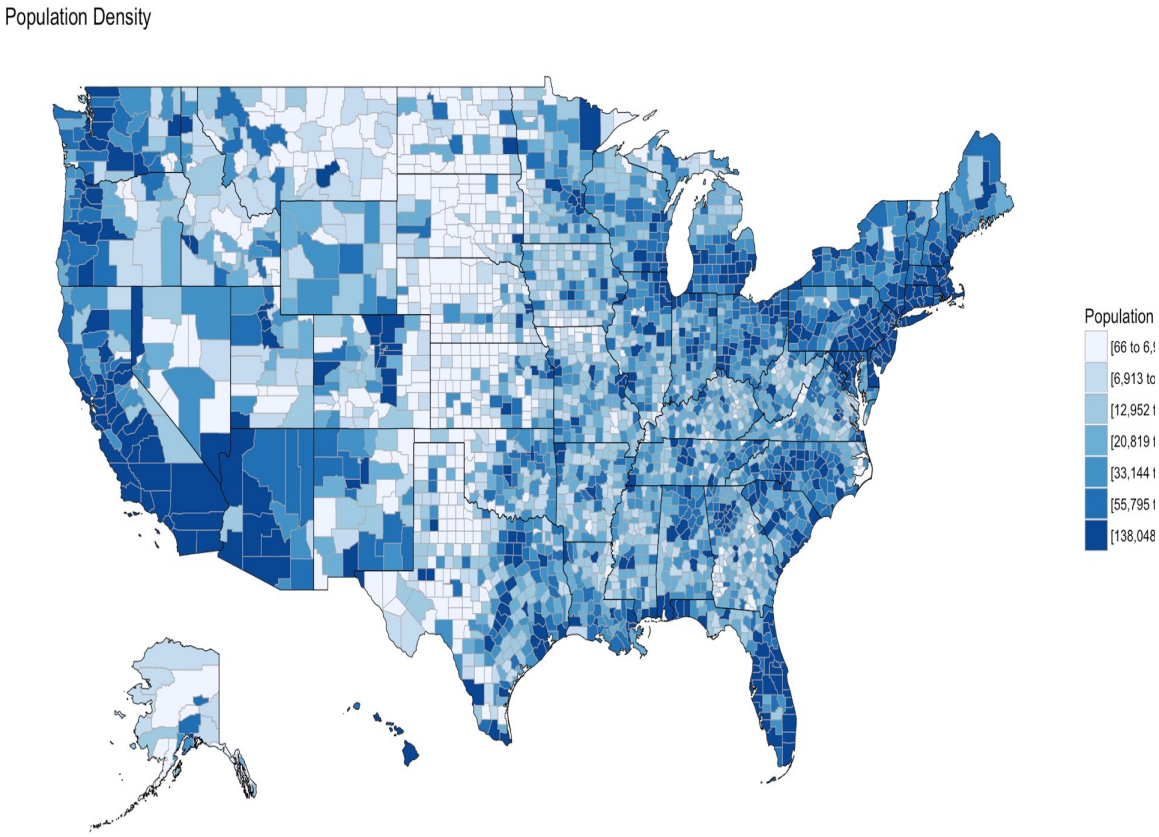
We are going to start easy, you have installed R Studio, if you have not this would be a great time to do it. Your first visualization is what is typically considered advanced, but I will let you be the judge of that after we are done.

Some lingo to learn:

Packages – Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sometimes sample data.

Choropleth – is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map,

such as population density or per-capita income or just about anything you can imagine to stuff into a map.



Below is the code for a choropleth, using the package choroplethr and the data set df_pop_county, which is the population of every county in the US, though very likely out of date.

Install package called choroplethr, the quotes are required, you will get a meaningless error without them

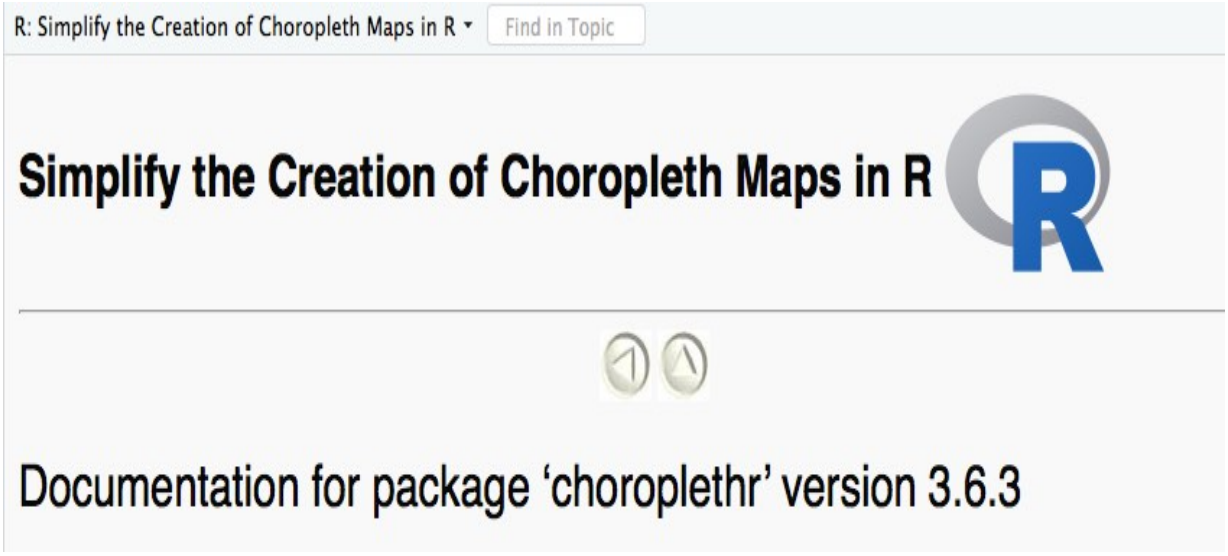
```
#Only needs to be installed once per machine

install.packages("choroplethr")
```

The library function will load the installed package to make any functions available for use.

```
library("choroplethr")
```

To find out what functions are in a package use help(package=""). This will bring up the help doc in the RStudio Console.

```
help(package="choroplethr")
```



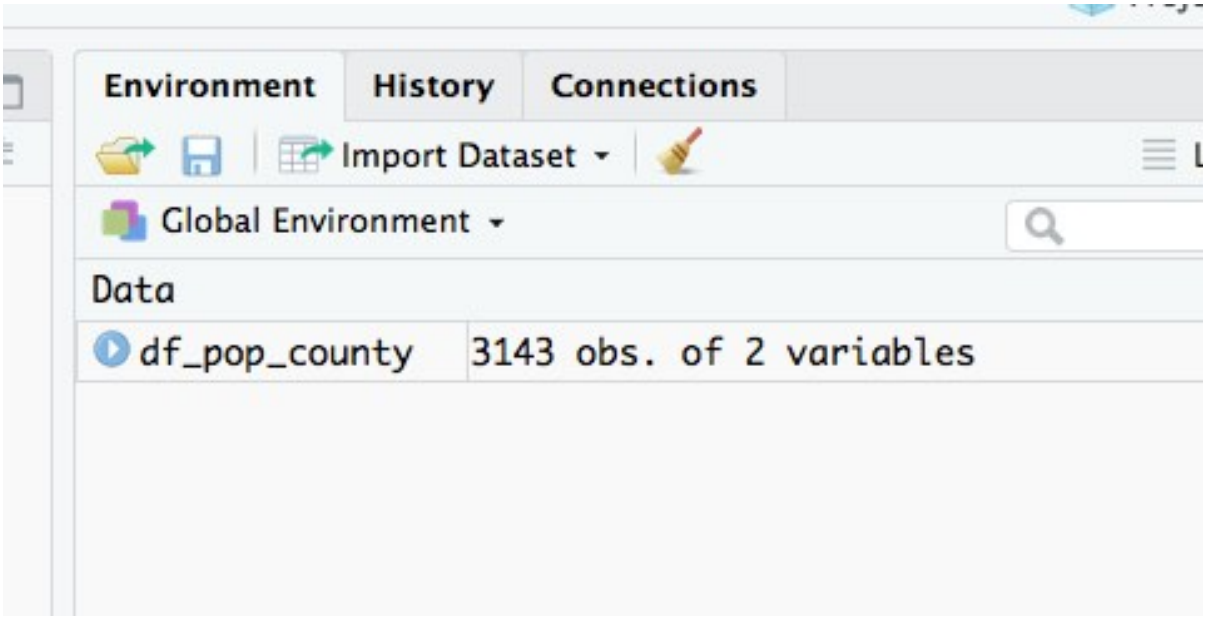
Many packages come with test or playground datasets, you will use many in classes and many for practice, data(package="") will list the datasets that ship with a package.

```
data(package="choroplethr")
```

```
Data sets in package 'choroplethr':

continental_us_states          A vector of the names of US Continental US States.
df_county_demographics         A data.frame containing demographic statistics for ea
df_japan_census                A data.frame containing basic demographic information
df_pop_country                 A data.frame containing population estimates for Coun
df_pop_county                  A data.frame containing population estimates for US C
df_pop_state                   A data.frame containing population estimates for US S
df_president                   A data.frame containing election results from the 201
df_president_ts                A data.frame containing all US presdiential election
df_state_age_2010              A data.frame containing median age estimates for US s
df_state_age_2015              A data.frame containing median age estimates for US s
df_state_demographics          A data.frame containing demographic statistics for ea
                               Columbia.
```

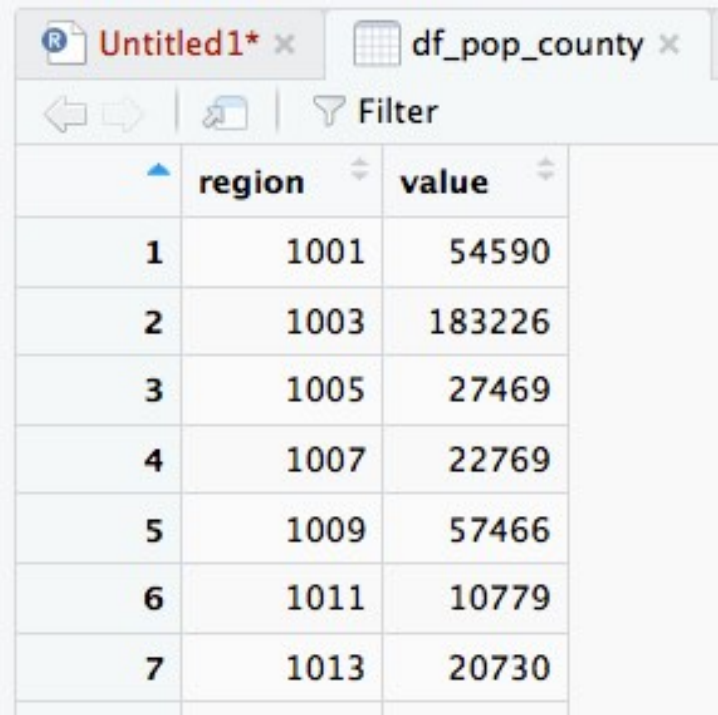
For this example we will be using the df_pop_county dataset, this command will load it from the package and you will be able to verify it is available by checking out the Environment Pane in R Studio.

```
data("df_pop_county")
```



View("") will open a view pane so you can explore the dataset. Similar to clicking on the dataset name in the Environment Pane.

```
View(df_pop_county)
```



	region	value
1	1001	54590
2	1003	183226
3	1005	27469
4	1007	22769
5	1009	57466
6	1011	10779
7	1013	20730

Part of learning R is learning the features and commands for data exploration, str will provide you with details on the structure of the object it is passed.

```
str(df_pop_county)
```

```
> str(df_pop_county)
'data.frame':   3143 obs. of  2 variables:
 $ region: num  1001 1003 1005 1007 1009 ...
 $ value : num  54590 183226 27469 22769 57466 ...
```

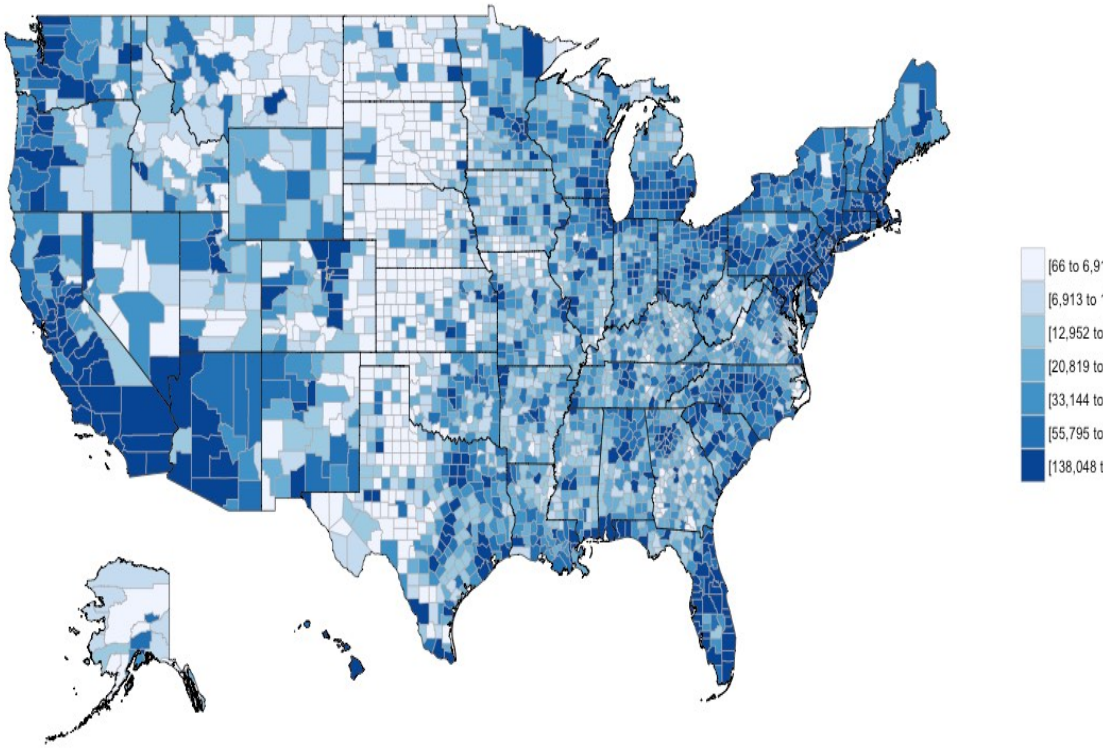
Summary will provide basic statistics about each column/variable in the object that it is passed. While it does not make much sense to perform stats on region, population will show some interesting statistics about the dataset.

```
summary(df_pop_county)
```

```
> summary(df_pop_county)
     region          value
 Min.   : 1001   Min.   :     66
 1st Qu.:18178   1st Qu.:  11061
 Median :29177   Median :  25772
 Mean   :30390   Mean   :  98358
 3rd Qu.:45082   3rd Qu.:  66914
 Max.   :56045   Max.   :9840024
```

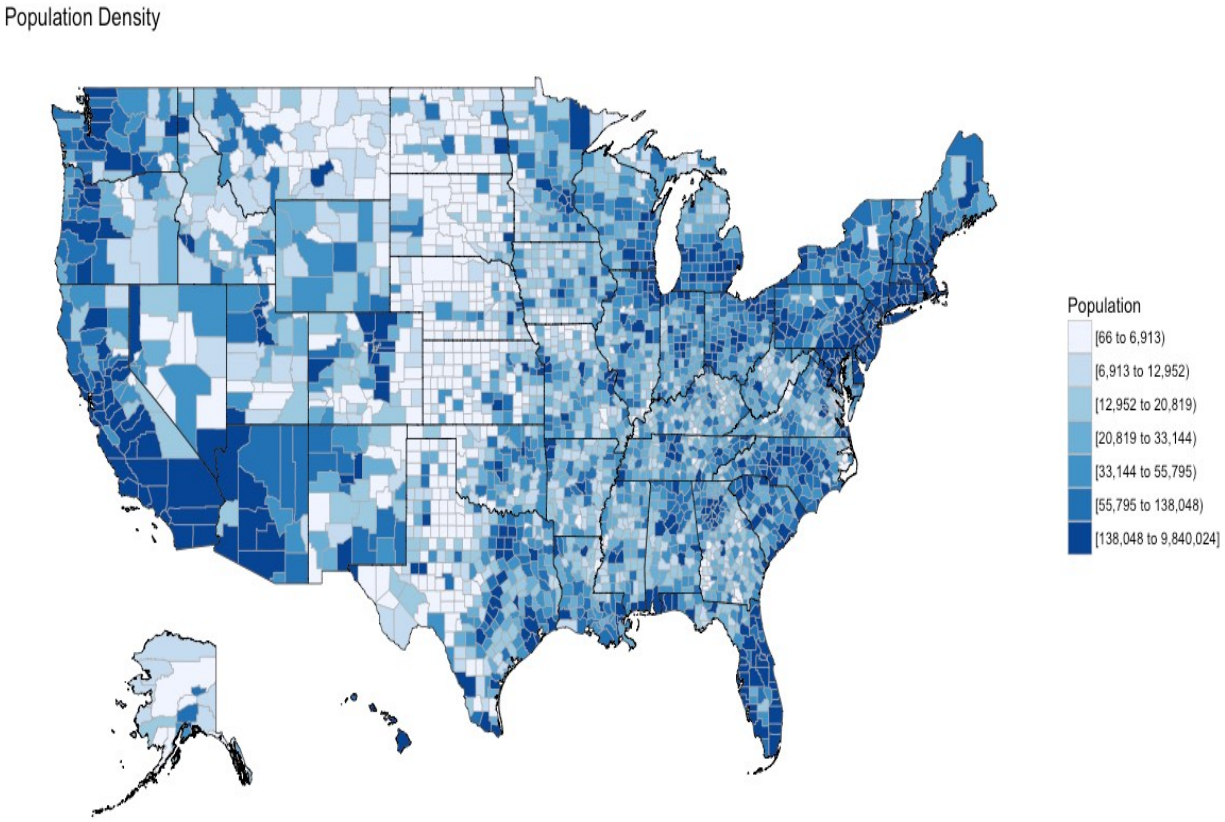
If your heart is true, you should get something very similar to the image above after running the following code. county_choropleth is a function that resides in the choroplethr package, it is used to generate a county level US map. The data passed in must be in the format of county number and value, the value will populate the map. WHen the map renders it will be in the plot pane of the RStudio IDE, be sure to select zoom and check out your work.

```
county_choropleth(df_pop_county)
```



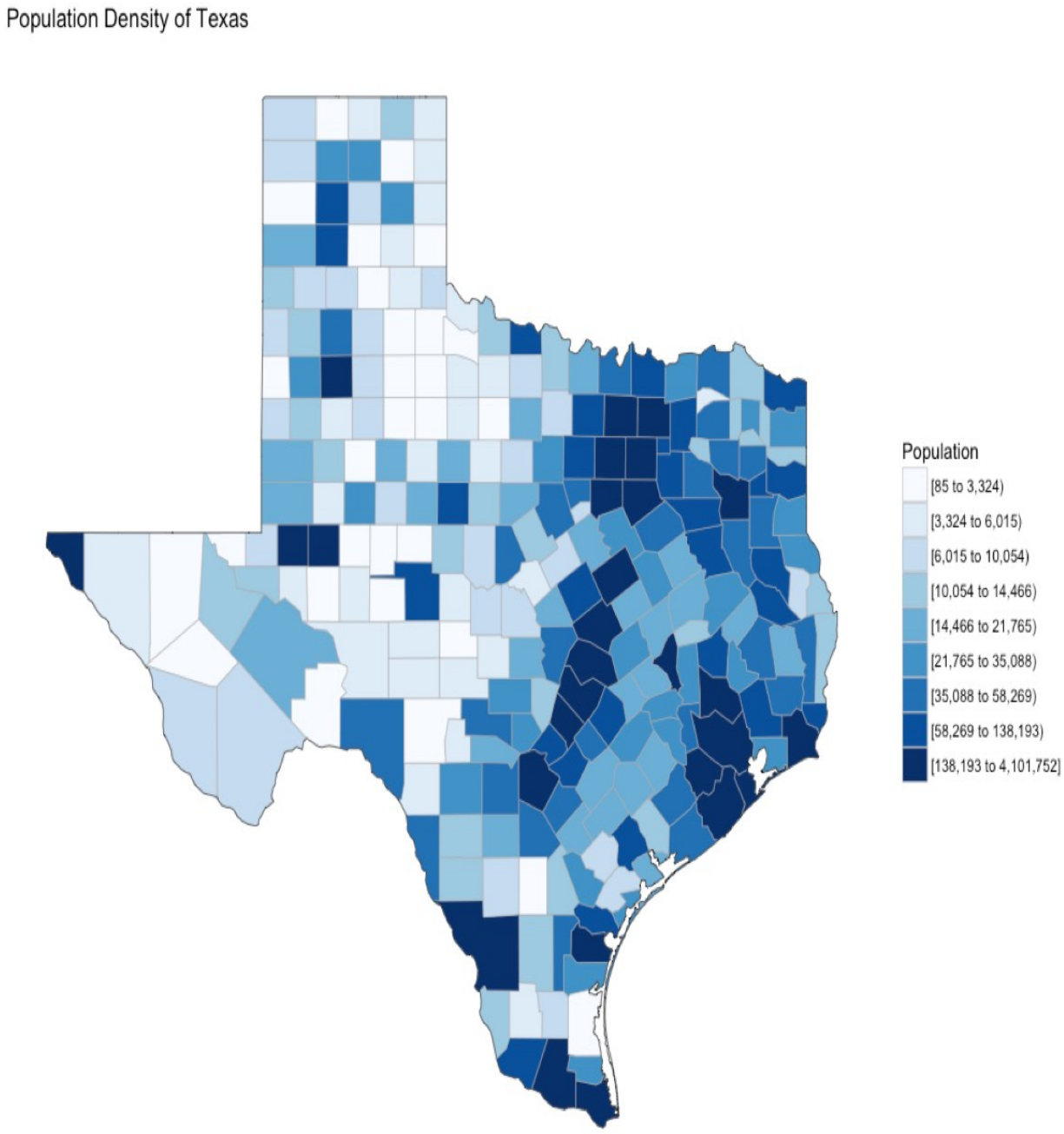
There are some additional parameters we can pass to the function, use help to find more. The following code will add a title to the image and a title to the legend. You can lean more about the parameters by running "**?county_choropleth**". Note the question mark in front of the command.

```
county_choropleth(df_pop_county,
                  title = "Population Density",
                  legend="Population")
```



Try changing the number of colors and change the state zoom. If your state is not working read the help to see if you can find out why.

```
county_choropleth(df_pop_county,
                  title = "Population Density of
Texas",
                  legend="Population",
                  num_colors=9,
                  state_zoom="texas")
```



state

There is an additional option for county_choropleth, reference_map. If it does not work for you do not fret, as of this writing it is not working for me either, an R upgrade a few versions back whacked it, be ready for this to happen and make sure you have backs and versions, especially before you get up on stage in front 200 people to present. I will show this in python as well, as it is fun to play with maps.

There you have it! Explore the commands used, look at the other datasets that ship with choroplethr and look at the other functions that ship with choroplethr, it can be tricky to figure out which ones work, be sure to check the help for each function you want to run, no help may mean no longer supported. Remember that these packages are community driven and written, which is good, but sometimes they can be a slightly imperfect.

Section 2

CHOROPLETH II USING YOUR OWN DATA

In the last section you were able to get a dataset with county and population data to display on a US map and zoom in on a state, and maybe even a county if you went exploring. In this demo we will be using the same choroplethr package but this time we will be using external data. Specifically, we will focus on one state, and check out the education level per county for one state.

The data is hosted by the USDA Economic Research Division, under Data Products / County-level Data Sets, though depending on website updates you are better off searching for "usda datasets college graduates". What will be demonstrated is the proportion of the population who have completed college, the datasets "completed some college", "completed high school", and "did not complete high school" are also available on the USDA site.

To make life easier on all of us the data is on the book github site under data at https://github.com/sqlshep/DS4New For this exercise you can use Edu_CollegeDegree-FL.csv.

In any data project it is important to understand the data, below is the basic schema

Column	Description
FIPS	Five-digit Federal Information Processing Standards code which uniquely identified counties and county equivalents in the United States.
region	County and State
2013RuralUrban-Code	Classification scheme that distinguishes metropolitan counties by the population size of their metro area, and nonmetropolitan counties by degree of urbanization and adjacency to a metro area.
CollegeDegree1970	Percentage os respondents that claim a college degree for 1970
CollegeDegree1980	Percentage os respondents that claim a college degree for 1980
CollegeDegree1990	Percentage os respondents that claim a college degree for 1990
CollegeDegree2000	Percentage os respondents that claim a college degree for 2000
CollegeDegree2010-2014	Percentage os respondents that claim a college degree for 2010-2014

Generally speaking when you start working with GIS data of any sort you enter a whole new world of acronyms and in many cases mathematics to deal with the craziness. The package we are using eliminates almost all of this for quick and dirty graphics via the choroplethr package. The county choropleth takes two values, the first is the region which must be the FIPS code for that county. If you happen to be working with states, then the FIPS state code must be used for region. To make it somewhat easier, the first two digits of the county FIPS code is the state code, the remainder is the county code for the data we will be working with.

So let's get to it;

Install and load the choroplethr package

```
install.packages("choroplethr")
library(choroplethr)
```

Use the setwd() to set the local working directory, getwd() will display what the current R working directory.

```
setwd("/Users/Data")
getwd()
```

Read.csv will read in a comma delimited file. "<-" is the assignment operator, much like using the "=". The "=" can be used as well. Which to assignment operator to use is a bit if a religious argument in the R community, i will stay out of it.

```
# read a csv file from my working directory
edu.CollegeDegree <- read.csv("Edu_CollegeDegree-FL.csv")
```

View() will open a new tab and display the contents of the data frame.

```
View(edu.CollegeDegree)
```

Untitled1* × edu.CollegeDegree ×

Filter

	FIPS	region	X2013RuralUrbanCode	CollegeDegree1970	CollegeDegree1980	CollegeDegree199
1	12001	Alachua, FL	2	0.231	0.294	
2	12003	Baker, FL	1	0.036	0.057	
3	12005	Bay, FL	3	0.092	0.132	
4	12007	Bradford, FL	6	0.045	0.076	
5	12009	Brevard, FL	2	0.151	0.171	
6	12011	Broward, FL	1	0.097	0.151	
7	12013	Calhoun, FL	6	0.060	0.069	
8	12015	Charlotte, FL	3	0.088	0.128	
9	12017	Citrus, FL	3	0.060	0.071	0.134

str() will display the structure of the data frame, essentially what are the data types of the data frame

```
str(edu.CollegeDegree)
```

```
> str(edu.CollegeDegree)
'data.frame':   68 obs. of  8 variables:
 $ FIPS                 : int  12001 12003 12005 12007 12009 12011
 $ region               : Factor w/ 68 levels "Alachua, FL",..: 1 2
 $ X2013RuralUrbanCode  : int  2 1 3 6 2 1 6 3 3 1 ...
 $ CollegeDegree1970    : num  0.231 0.036 0.092 0.045 0.151 0.097
 $ CollegeDegree1980    : num  0.294 0.057 0.132 0.076 0.171 0.151
 $ CollegeDegree1990    : num  0.346 0.057 0.157 0.081 0.204 0.188
 $ CollegeDegree2000    : num  0.387 0.082 0.177 0.084 0.236 0.245
 $ CollegeDegree2010.2014: num  0.408 0.109 0.216 0.104 0.267 0.302
```

Looking at the structure of the dataframe we can see that the counties imported as Factors, for this task it will not matter as i will not need the county names, but in the future it may become a problem. To nip this we will reimport using stringsAsFactors option of read.csv we will get into factors later, but for now we don't need them.

```
edu.CollegeDegree <- read.csv("Edu_CollegeDegree-
FL.csv",stringsAsFactors=FALSE)
```

```
#Recheck our structure
str(edu.CollegeDegree)
```

```
> str(edu.CollegeDegree)
'data.frame':   68 obs. of  8 variables:
 $ FIPS                 : int  12001 12003 12005 12007 12009
 $ region               : chr  "Alachua, FL" "Baker, FL" "Bay
 $ X2013RuralUrbanCode  : int  2 1 3 6 2 1 6 3 3 1 ...
 $ CollegeDegree1970    : num  0.231 0.036 0.092 0.045 0.151
 $ CollegeDegree1980    : num  0.294 0.057 0.132 0.076 0.171
 $ CollegeDegree1990    : num  0.346 0.057 0.157 0.081 0.204
 $ CollegeDegree2000    : num  0.387 0.082 0.177 0.084 0.236
```

Now the region/county name is a character however, the there is actually more data in the file than we need. While we only have 68 counties, we have more columns/variables than we need. The only year i am interested in is the CollegeDegree2010.2014 so there are several ways to remove the unwanted columns.

The following is actually using index to include only columns 1,2,3,8 much like using column numbers in SQL vs the actual column name, this can bite you in the butt if the order or number of columns change though not required for this import, header=True never hurts. You only need to run one of the following commands below, but you can see two ways to reference columns.

This can also be done after the data is imported, we will cover that later.

```
    edu.CollegeDegree <- read.csv("Edu_CollegeDegree-FL.csv", head-
er=TRUE,stringsAsFactors=FALSE)[c(1,2,3,8)]


    # or Use the column names

    edu.CollegeDegree <- read.csv("Edu_CollegeDegree-FL.csv", head-
er=TRUE,stringsAsFactors=FALSE)
[c("FIPS","region","X2013RuralUrbanCode","CollegeDegree2010.2014")]


    #Lets check str again

    str(edu.CollegeDegree)
```

```
> str(edu.CollegeDegree)
'data.frame':   68 obs. of  4 variables:
 $ FIPS                  : int  12001 12003 12005 12007 12009 12011 1
 $ region                : chr  "Alachua, FL" "Baker, FL" "Bay, FL" "
 $ X2013RuralUrbanCode   : int  2 1 3 6 2 1 6 3 3 1 ...
 $ CollegeDegree2010.2014: num  0.408 0.109 0.216 0.104 0.267 0.302 0
>
```

Using summary() we can start reviewing the data from A statistical perspective. The CollegeDegree2010.2014 variable, we can see the county with the lowest proportion of college graduates is .075, or 7.5% of the population of that county the max value is 44.3%. The average across all counties is 20.32% that have completed college.

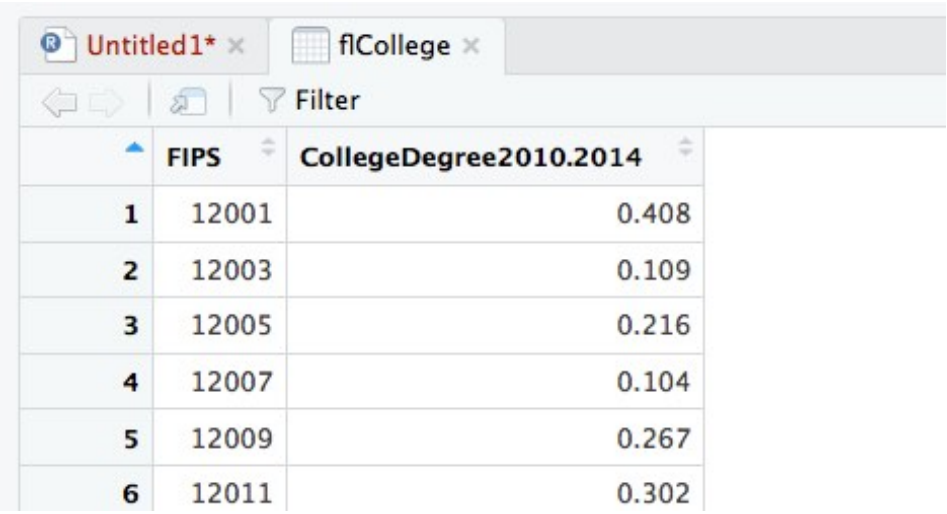
```
summary(edu.CollegeDegree)
```

```
> summary(edu.CollegeDegree)
      FIPS          region          X2013RuralUrbanCode CollegeDegree2010.2014
 Min.   :12000   Length:68          Min.   :0.000       Min.   :0.0750
 1st Qu.:12034   Class :character   1st Qu.:1.750       1st Qu.:0.1155
 Median :12068   Mode  :character   Median :2.000       Median :0.1880
 Mean   :12067                      Mean   :3.147       Mean   :0.2032
 3rd Qu.:12100                      3rd Qu.:6.000       3rd Qu.:0.2670
 Max.   :12133                      Max.   :9.000       Max.   :0.4430
```

Looking at the data we can see that we have a FIPS code, and the only other column we are interested in for mapping is CollegeDegree2010.2014, so lets create a dataframe with just what we need.

```
# the follwoing will create a datafram with just the FIPS and percentage of college grads

flCollege <- edu.CollegeDegree[c(1,4)]

# Alternatively, you can use the column names vs. the positions. Probably smarter ;-)

flCollege <- edu.CollegeDegree[c("FIPS","CollegeDegree2010.2014")]
```

```
View(flCollege)
```



Untitled1* flCollege

Filter

	FIPS	CollegeDegree2010.2014
1	12001	0.408
2	12003	0.109
3	12005	0.216
4	12007	0.104
5	12009	0.267
6	12011	0.302

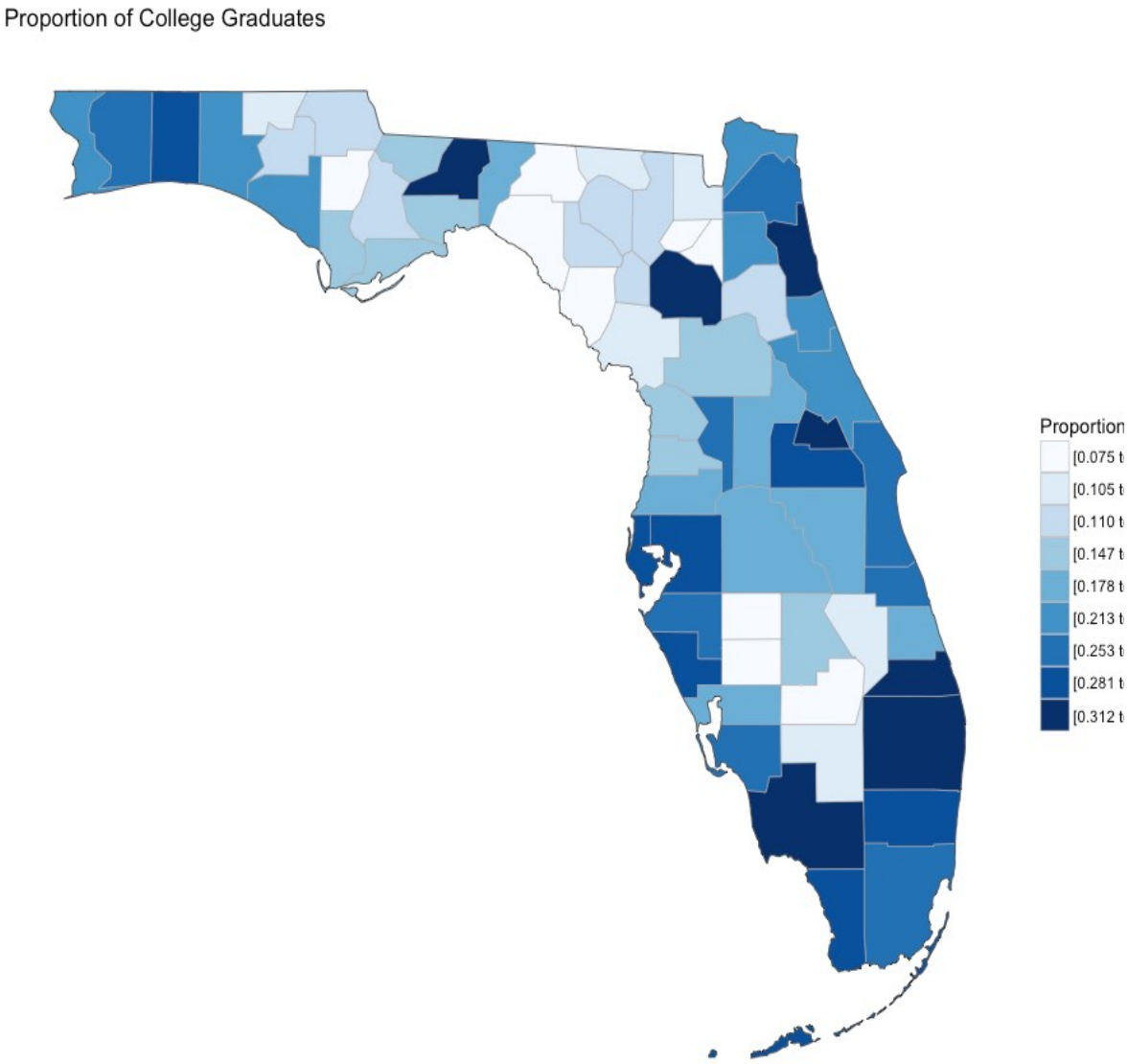
Reading the help file on county_choropleth, it requires that only two variables(columns) be passed in, region, and value. Region must be a FIPS code so, we need to rename the columns using colnames().

```
    colnames(flCollege)[which(colnames(flCollege) == 'FIPS')] <-
'region'
    colnames(flCollege)[which(colnames(flCollege) == 'CollegeDe-
gree2010.2014')] <- 'value'
```

So, lets map it!

Since we are only using Florida, set the state_zoom, it will work without the zoom but you will get many warnings. You will also notice a warning that 12000 is not mappable. Looking at the data you will see that 12000 is the entire state of Florida.

```
county_choropleth(flCollege,
                  title = "Proportion of College Graduates ",
                  legend="Proportion",
                  num_colors=9,
                  state_zoom="florida")
```



For your next task, go find a different state and a different data set from the USDA or anywhere else for that matter and create your own map. Be aware of the "value", that must be an integer, sometimes these get imported as character if there is a comma in the number.

Chapter 3

VISUALIZATIONS

Lorem Ipsum

It has been my opinion and others that R was written by an angry teenager to get even with boomer parents, while not entirely true R has many frustrations. The nice thing is, you can write your own package to handle many of these more complex visualizations, i stick to Base R for much of this and it does get the point across, but ggplot provides much better graphics and legends. We will visit ggplot in a later chapter once we have grown bored of base R graphics. Ggplot can be dense and complex, but once you learn it in R, you have it in python too.

Section 1

THE HISTOGRAM

You probably have noticed that R behaves very much like a scripting language which for me, having been a T-SQL guy seemed familiar. You may have noticed that it behaves like a programing language too in that you can install a package, invoke function or data set stored in that package, very much like a dll, though no compiling is required. It's clear that it is very flexible as a language, which you will learn is its strength and its downfall. If you decide to start designing your own R packages, you can write them as terribly as you want, though i would rather you didn't.

If you want to find out what datasets are available in a package run "data()", and as we covered in a prior chapter, data(package=) will give you the datasets for a specific package. This will provide you nice list of datasets to doodle with, as you learn something new explore the datasets to see how you can apply your new-found knowledge.

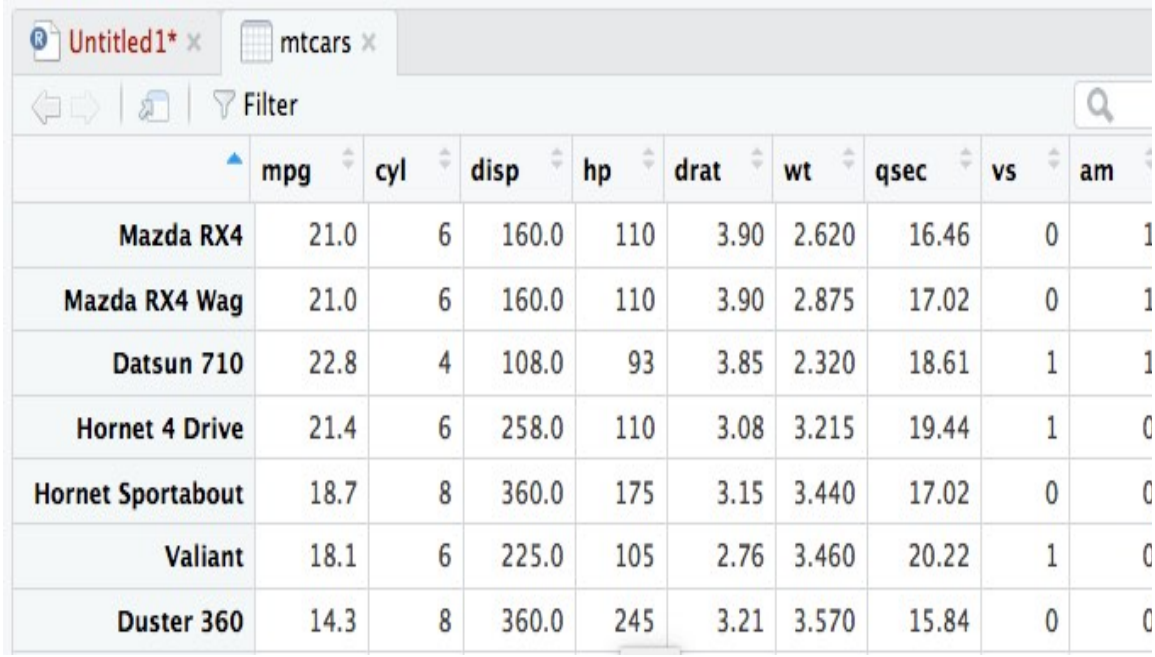
First lets check out the histogram. If you have worked with SQL you know what a histogram is, and it is marginally similar to a statistics visual histogram. We are going to look at a real one. The basic definition is that it is a graphical representation of the distribution of numerical data.

When to use it? When you want to know the distribution of a single column or variable.

"Hist" ships with base R, which means no package required. We will go through a few Histograms from different packages.

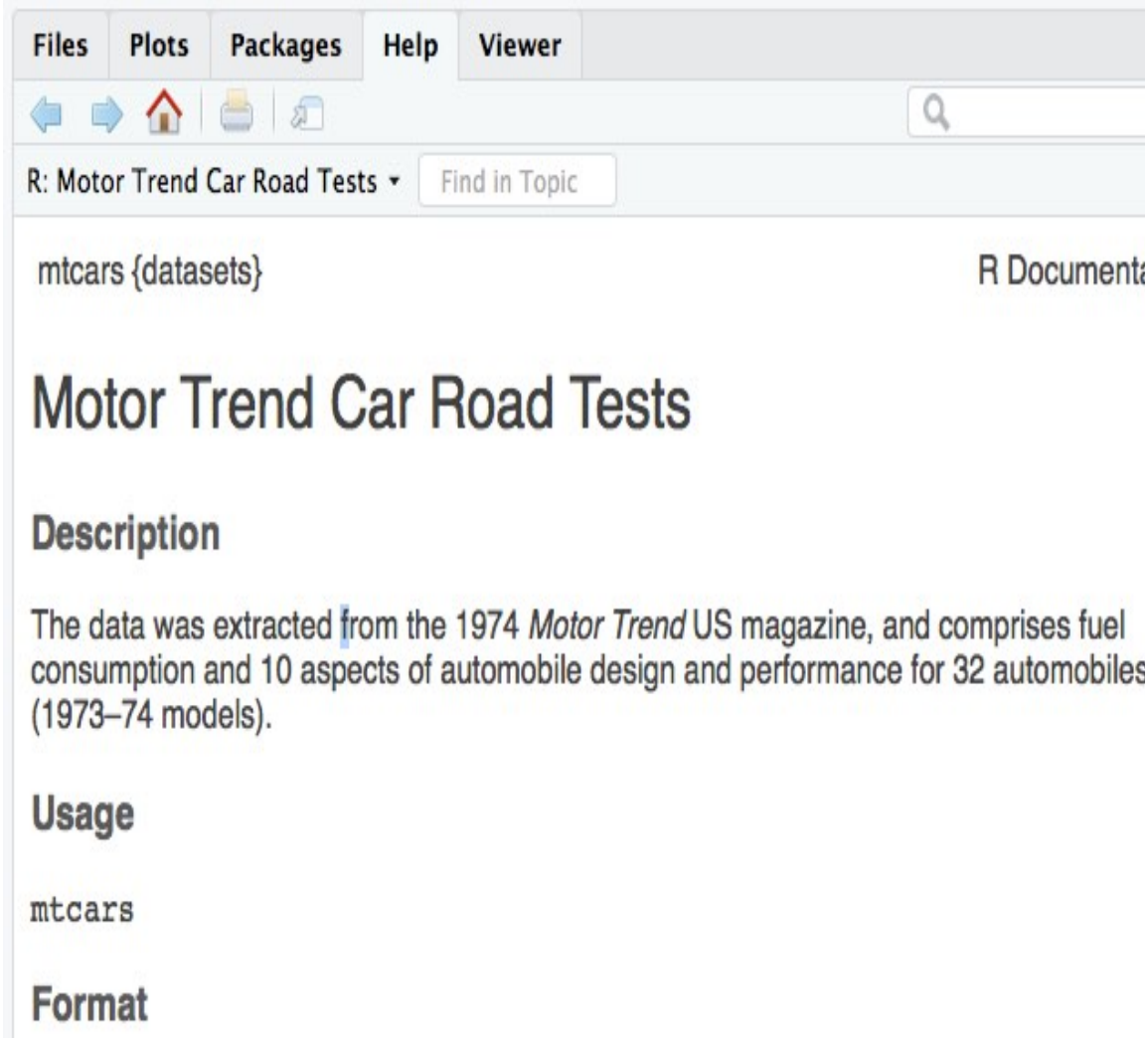
You have become familiar with this by now, "data()" will load the mtcars dataset for use, "View" will open a new pane so you can review it, and help() will provide some information on the columns/variables in the dataset. Fun fact, if you run "view" (lowercase v) it will display the contents to the console, not a new pane. "?hist" will open the help for the hist command.

```
data(mtcars)

View(mtcars)
```

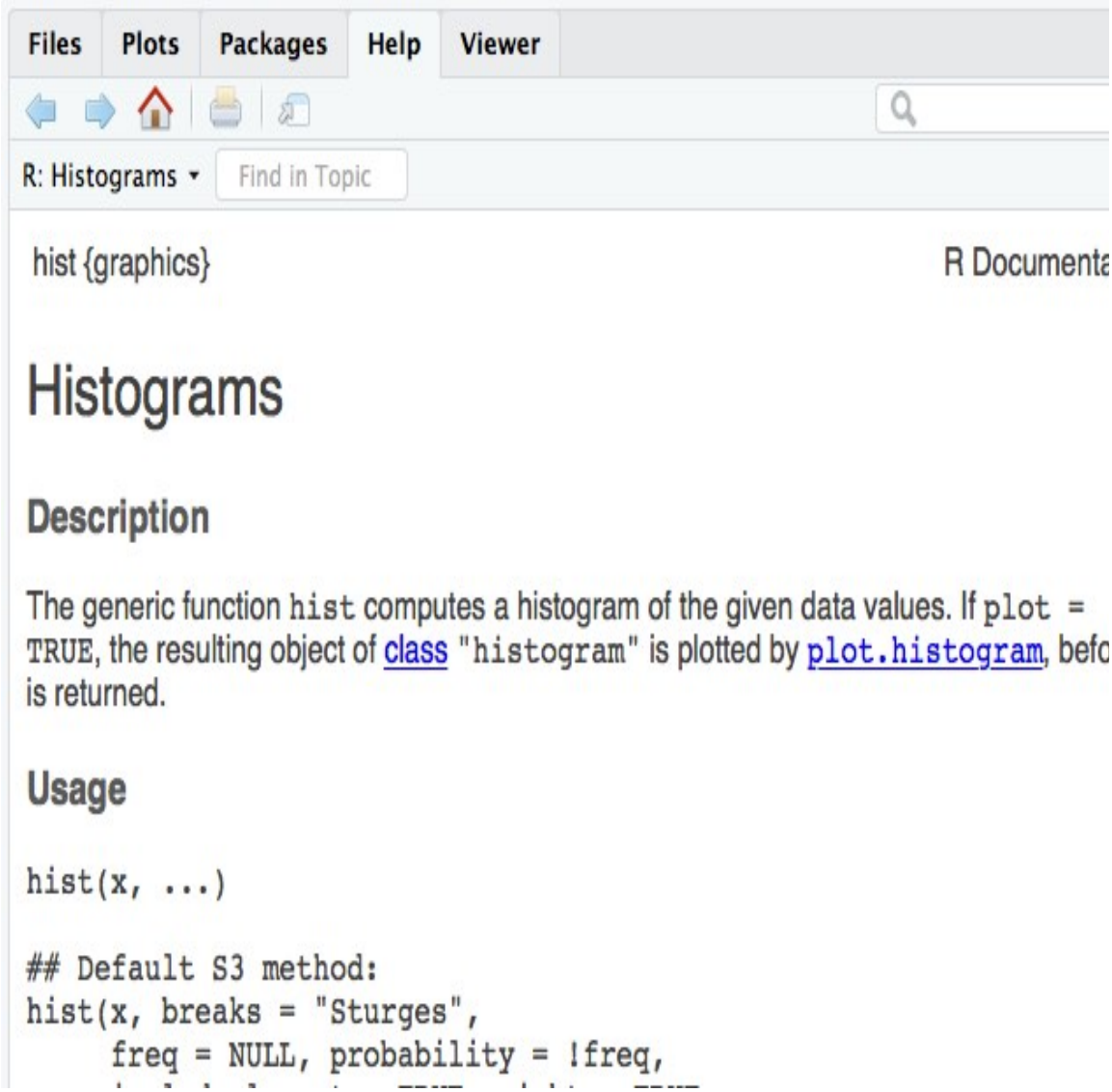


	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0

```
help(mtcars)
```


Files
Plots
Packages
Help
Viewer
R: Motor Trend Car Road Tests
Find in Topic
mtcars {datasets}
R Documenta
Motor Trend Car Road Tests
Description
The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).
Usage
mtcars
Format

```
?hist
```



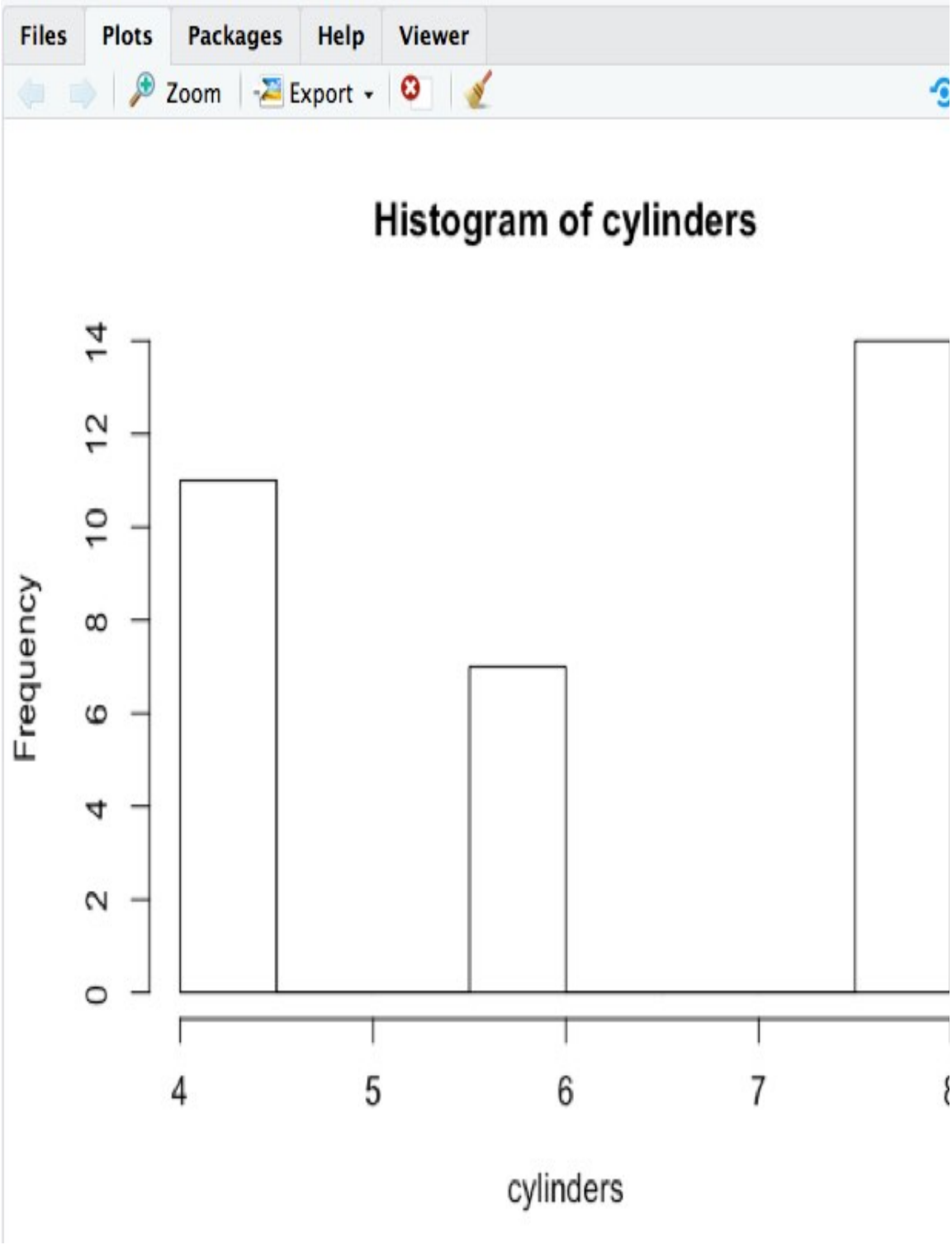
Did you notice we did not load a package? mtcars ships with base R, run “data()” to see all the base datasets.

Hist takes at a single quantitative variable, this can be passed by creating a vector or referencing just the dataframe variable you are interested in.

Try each of these out one at a time.

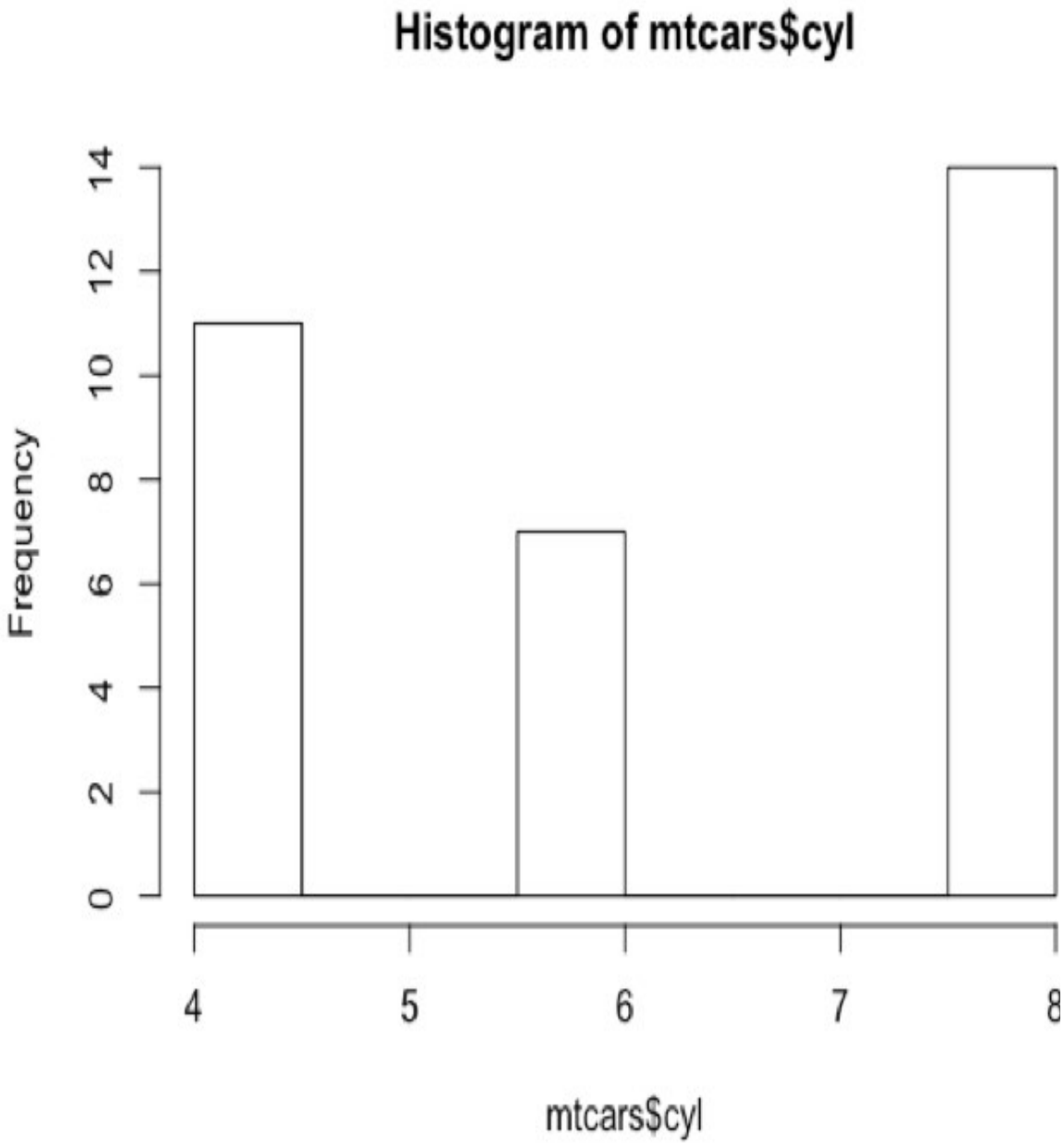
```
    #Have you figure out that a # is a comment line?

    #When you see the following code, this is copying
the contents

    #of one variable into a vector, it is not neces-
sary for what
```

```
    #we are doing but it is an option. Once copied
just pass it in to the function.


    cylinders <- mtcars$cyl

    hist(cylinders)
```




```
    #Otherwise, invoke the function and pass just the
variable you are interested in.
```

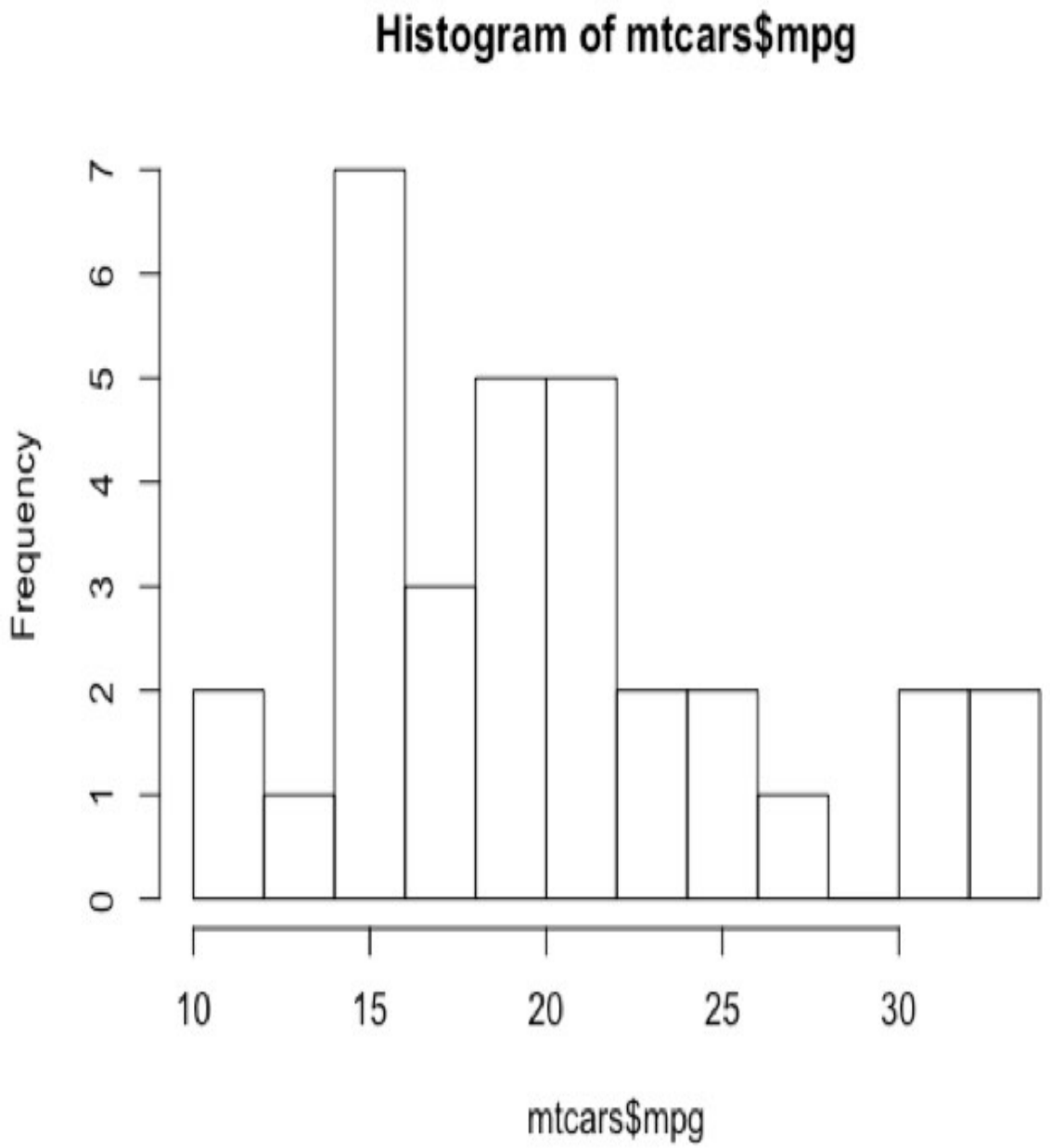
```
hist(mtcars$cyl)
```



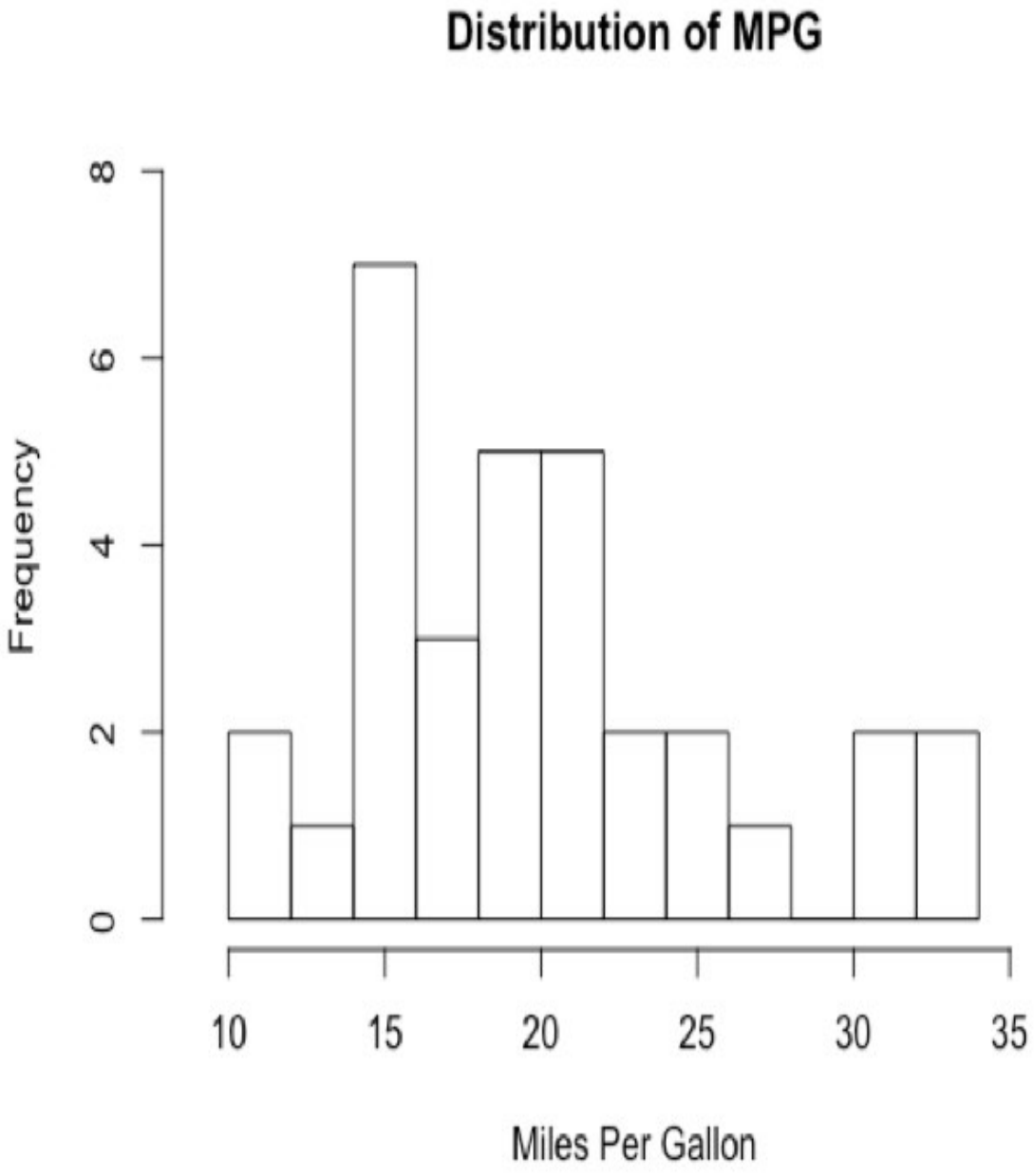

```
hist(mtcars$cyl, breaks=3)
hist(mtcars$disp)
hist(mtcars$wt)
hist(mtcars$carb)
```

As you get wiser in the ways of R you can start adding more options to clean up the histogram so its starts to look a little more appealing, inherently histograms are not visually appealing but they are a good for data discovery and exploration. Without much thought you can see that more vehicles get 15mpg.

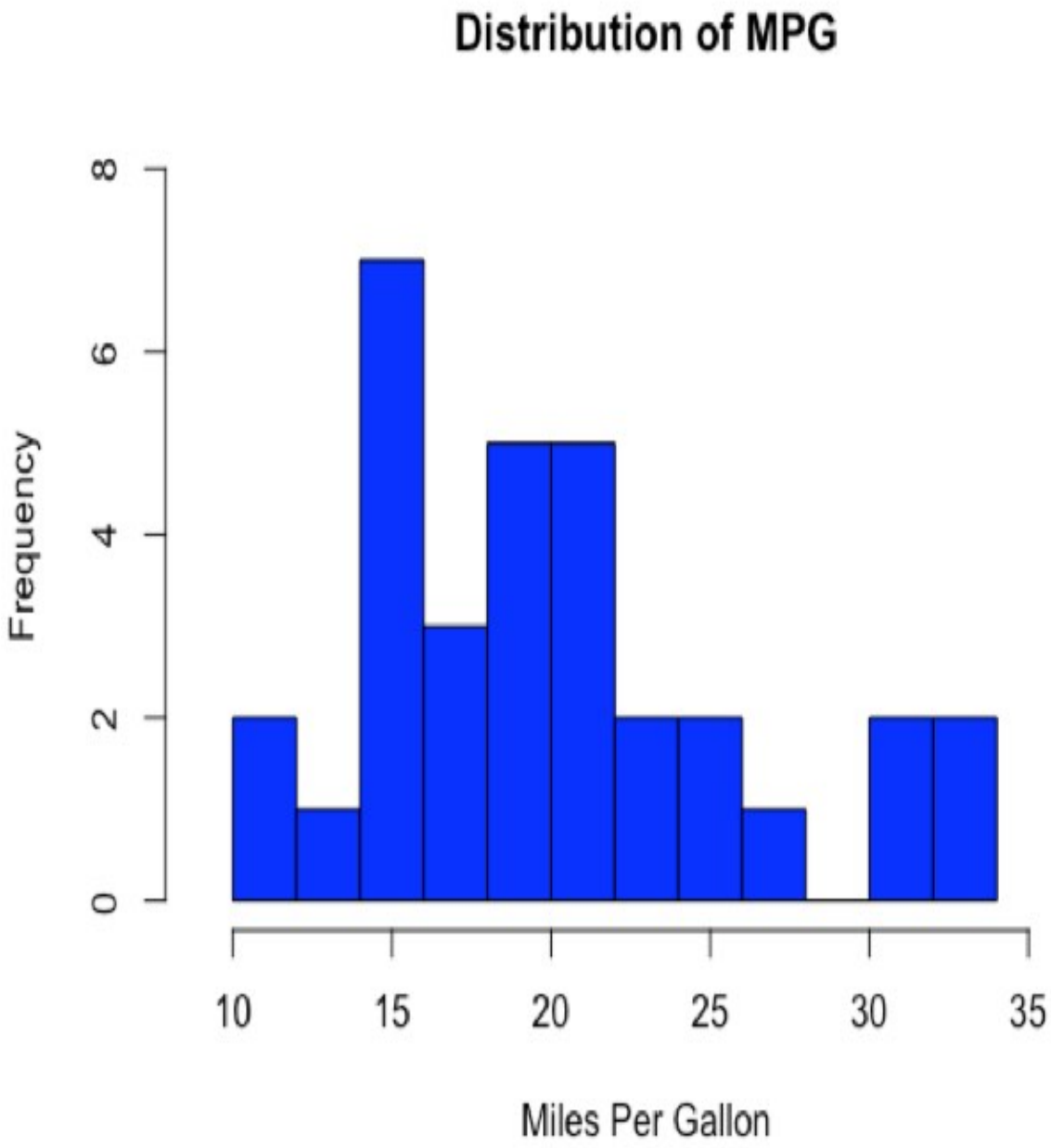
```
hist(mtcars$mpg, breaks = 15)
```



```
hist(mtcars$mpg, breaks = 15,
        xlim = c(9,37),
        ylim = c(0,8),
        main = "Distribution of MPG",
        xlab = "Miles Per Gallon")
```

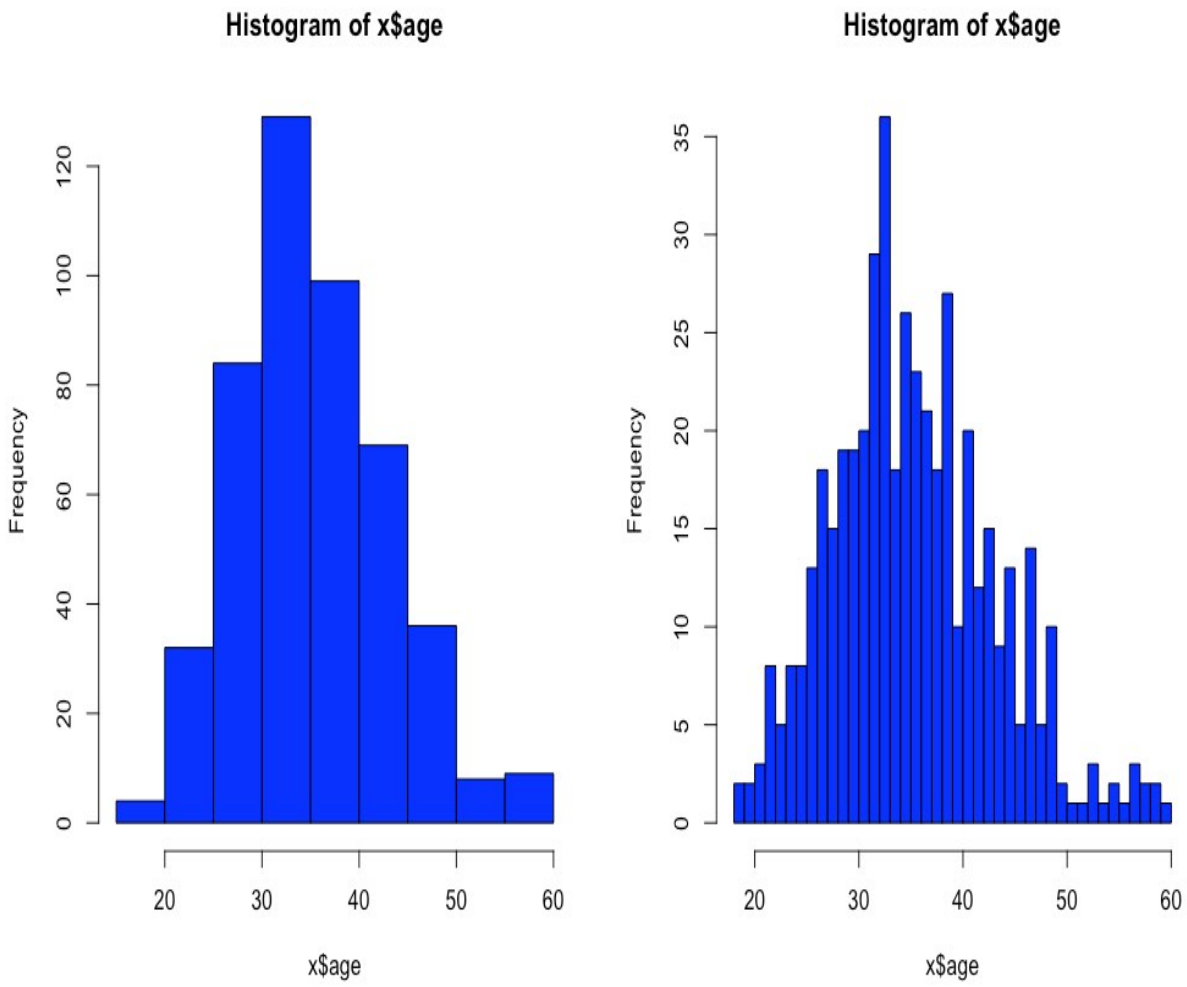


```
hist(mtcars$mpg, breaks = 15,
     xlim = c(9,37),
     ylim = c(0,8),
     col='blue',
     main = "Distribution of MPG",
     xlab = "Miles Per Gallon")
```



Using the code below you can see that the number of bins does change the graph dramatically though some averaging clearly is taking place, but the x and y axis both make sense. We are basically looking at a bar chart, and based on shape we can derive some level of data distribution. Also notice the R optoin par() that will allow two visuals side by side when (mfrow=c(1,2)) is passed. (mfrow=c(2,2)) will allow 4 visuals in one pane.

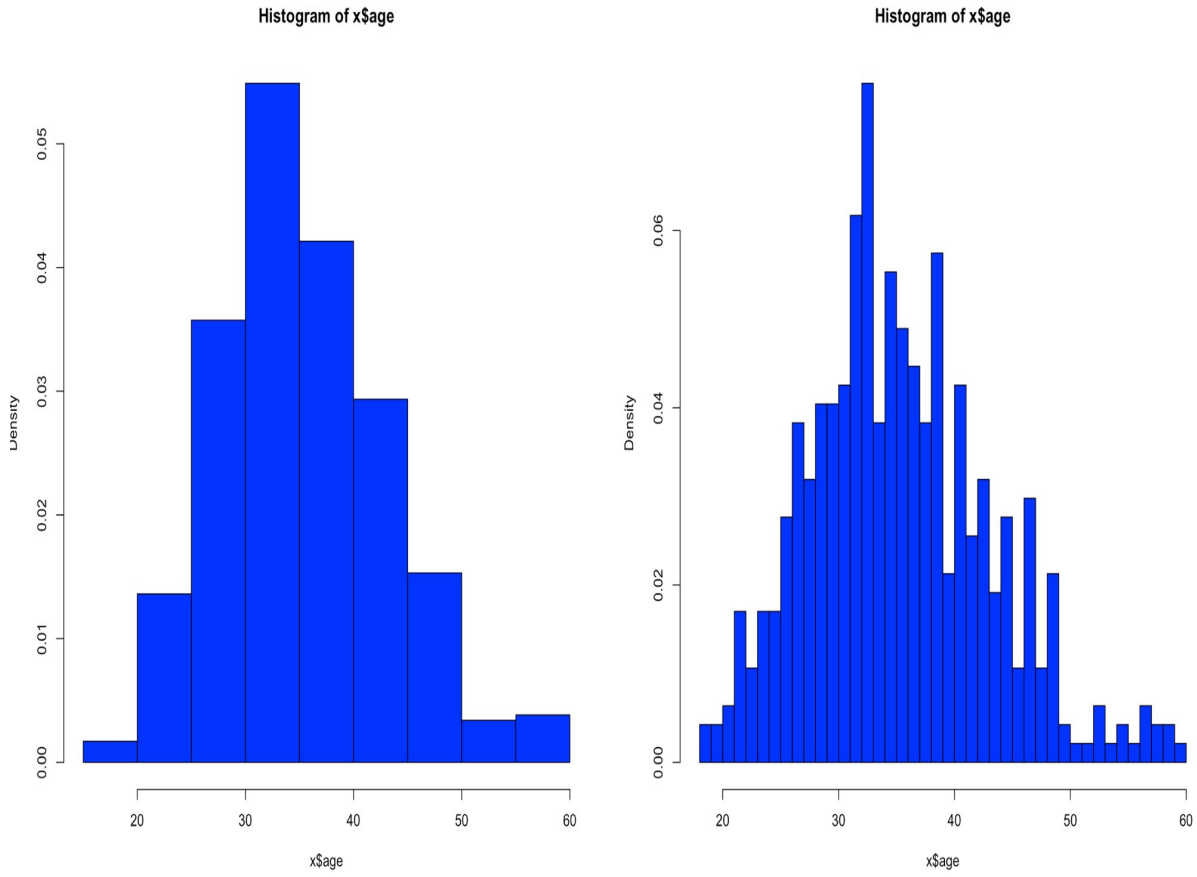
```
install.packages("mosaicData")

library("mosaicData")



data(HELPmiss)

View(HELPmiss)


str(HELPmiss)


x <-
HELPmiss[c("age","anysub","cesd","drugrisk","sex","g1b","homeless","race
grp","substance")]


#the par command will allow you to show more than one plot side by
side

#par(mfrow=c(2,2)); or par(mfrow=c(3,3)); etc...


par(mfrow=c(1,2))


hist(x$age,col="blue",breaks=10)

hist(x$age,col="blue",breaks=50)
```



So, what happens when we add the line, the density curve, the PDF call it what you will? I have mentioned before it is a probability density function, but what does that really mean? And why do the numbers on the y axis charge so dramatically?

Well, first lets look at the code to create it. Same dataset as above, but lets add the "prob=TRUE" option.

```
par(mfrow=c(1,2))
hist(x$age,prob=TRUE,col="blue",breaks=10)
hist(x$age,prob=TRUE,col="blue",breaks=50)
```



Notice the y axis scale is no longer frequency, but density. Okay, whats that? Density is a probability of that number occurring relative to all other numbers. Being a probability function and being relative to all numbers it should probably add up to 100% then, or in the case of proportions which is what we have all should add up to 1. The problem is a histogram is a miserable way to figure that out. And clearly changing the bins from 20 to 50 not only changed the number of bins as we would expect it also changed the y-axis. Whaaaaaaa? As if to tell me that even though the data is the same the probability has changed. So how can i ever trust this?

Well, there is a way to figure out what is going on! I'm not going to lie, the first time i saw this my mind was a little blown. You can get all the data that is generated and used to create the histogram.

Pipe the results into a variable then simply display the variable.

```
y<-hist(x$age,prob=TRUE,col="blue",breaks=10)
y
```

```
> y<-hist(x$age,prob=TRUE,col="blue",breaks=10)
> y
$breaks
 [1] 15 20 25 30 35 40 45 50 55 60

$counts
[1]   4  32  84 129  99  69  36   8   9

$density
[1] 0.001702128 0.013617021 0.035744681 0.054893617 0.042127660 0.029361702 0.015319149 0.003404255
[9] 0.003829787

$mids
[1] 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5

$xname
[1] "x$age"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

So, based on any probability we should get "1" indicating 100% of the data if we sum $density... Lets find out.

```
sum(y$density)
```

```
> sum(y$density)
[1] 0.2
```

I got .2...

Lets increase the bins and see what happens, we know the scale changed, maybe my density distribution will to.

```
z<-hist(x$age,prob=TRUE,col="blue",breaks=50)

#z

sum(y$density)
```

```
> z<-hist(x$age,prob=TRUE,col="blue",breaks=50)
> sum(z$density)
[1] 1
```

Hmm, this time i get a "1". You will also notice that when the bins are increased to 50 the number of values in $mids and $density increased, it just so happens that for this dataset 50 was a better number of bins, though you will see that $density and $mids only went to a count of 42 values, which is enough.

With the below code you can get an estimation of the probability of specific numbers occurring. For the dataset i am using, 32.5 occurs 7.66% of the time, so that is the likely probability of that age occurring on future random variables. With less bins, you are basically getting an estimated average but not necessarily the full picture, so a density function in a histogram can be tricky and misleading if you attempt to make a decision on this.

```
z<-hist(x$age,prob=TRUE,col="blue",breaks=50)

z


distz<- as.data.frame(list(z$density,z$mids))

names(distz) <- c("density","mids")

distz
```

```
> distz<- as.data.frame(list(z$density,z$mids))
> names(distz) <- c("density","mids")
> distz
      density mids
1  0.004255319 18.5
2  0.004255319 19.5
3  0.006382979 20.5
4  0.017021277 21.5
5  0.010638298 22.5
6  0.017021277 23.5
7  0.017021277 24.5
8  0.027659574 25.5
```

So what is all of this? well, when you switch the hist() function to prob=true, you pretty much leave descriptive statistics and are now in probability. The thing that is used to figure out the distribution is called a kernel density estimation and in some cases called the Parzen-Rosenblatt-Window Density estimation, or kernel estimation. The least painful definition " to estimate a probability density function p(x) for a specific point p(x) from a sample p(xn) that doesn't require any knowledge or assumption about the underlying distribution." If you are up for some modern day geekery, the Parzen mentioned above is the father of the Michael Parzen who teaches at Harvard. So, no all stats and probability is centuries old. Well, stats is centuries old, probability is a bit newer obviously. He is a great instructor and does happen to teach an intro to stats course.

The definitions of the hist fields are int eh CRAN Help under hist() in the values section. Try this out, try different data and check out how the distribution changes with more or less bins.

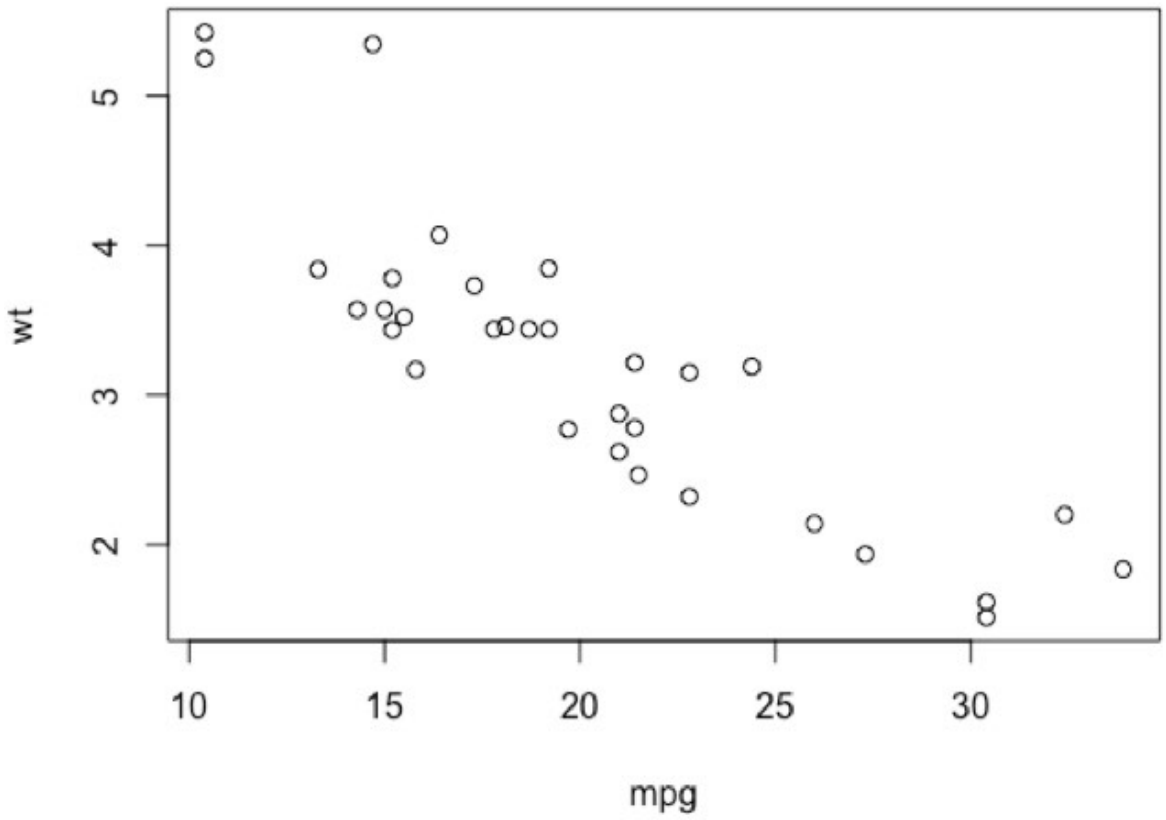
Section 2

THE SCATTERPLOT

In the ongoing visualization show and tell scatterplots have come up next on my list. As I write this I try very hard to check and double check my knowledge and methods, I usually have a dataset or two in mind long before I get to the point I want to write about it. I want to use the mtcars dataset again to play around with and run a line through the scatter plot to show a trend.

Below we have loaded the mtcars dataset, and run an attach(). Attach() gives the ability to access the variables/columns of the dataset without having to reference the dataset name. So, instead of mtcars$cyl we can just reference cyl in functions after the attach. It has down sides, so be careful, sort of like a global anything in programming.

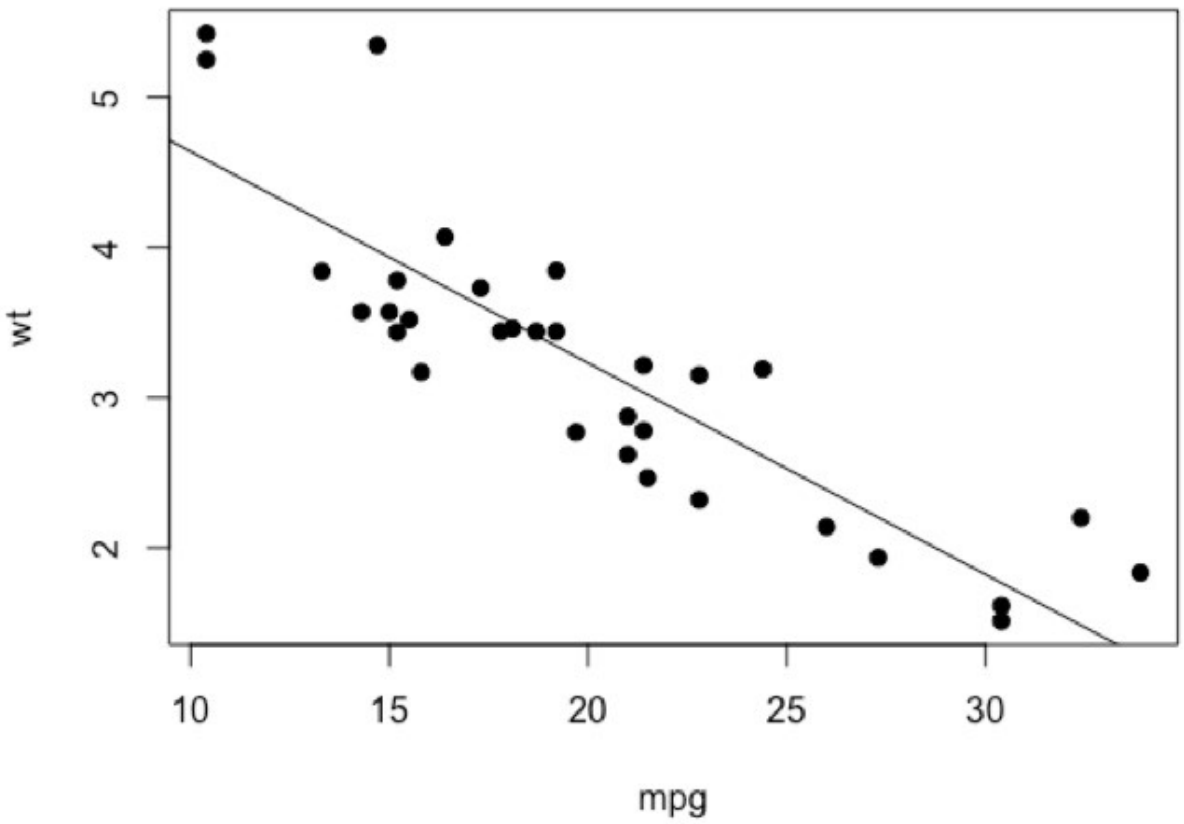
```
data(mtcars)

#Notice when you execute just the dataframe it displays to the console

mtcars

attach(mtcars)

plot(mpg,wt)
```



With the scatterplot we have two dimensions of data, on the left is the y axis, the weight (wt) of the vehicle in thousands of pounds, and on the bottom is the x axis, mpg or miles per gallon.

That was cool, no? Lets add "pch=19" to the next plot to make the dots a bit more visible, and add a line through the data. Abline() draws a straight line through the plot using intercept and slope, we can get the intercept and slope by passing the wt and mpg into a linear model function called lm(). Run lm(wt ~ mpg) by itself and see what you get. Make sure mpg and wt are in the order specified below, "mpg, wt" for the plot and wt~mpg for the lm function. if you dig into the lm() function you will see that we are passing in just a formula of "wt ~ mpg", from the R documentation for lm() y must be first which is the response variable. Much, Much more on lm later, just know for now, y must be first when using lm(), not x.

```
plot(mpg,wt,pch=19)
abline(lm(wt ~ mpg))
```

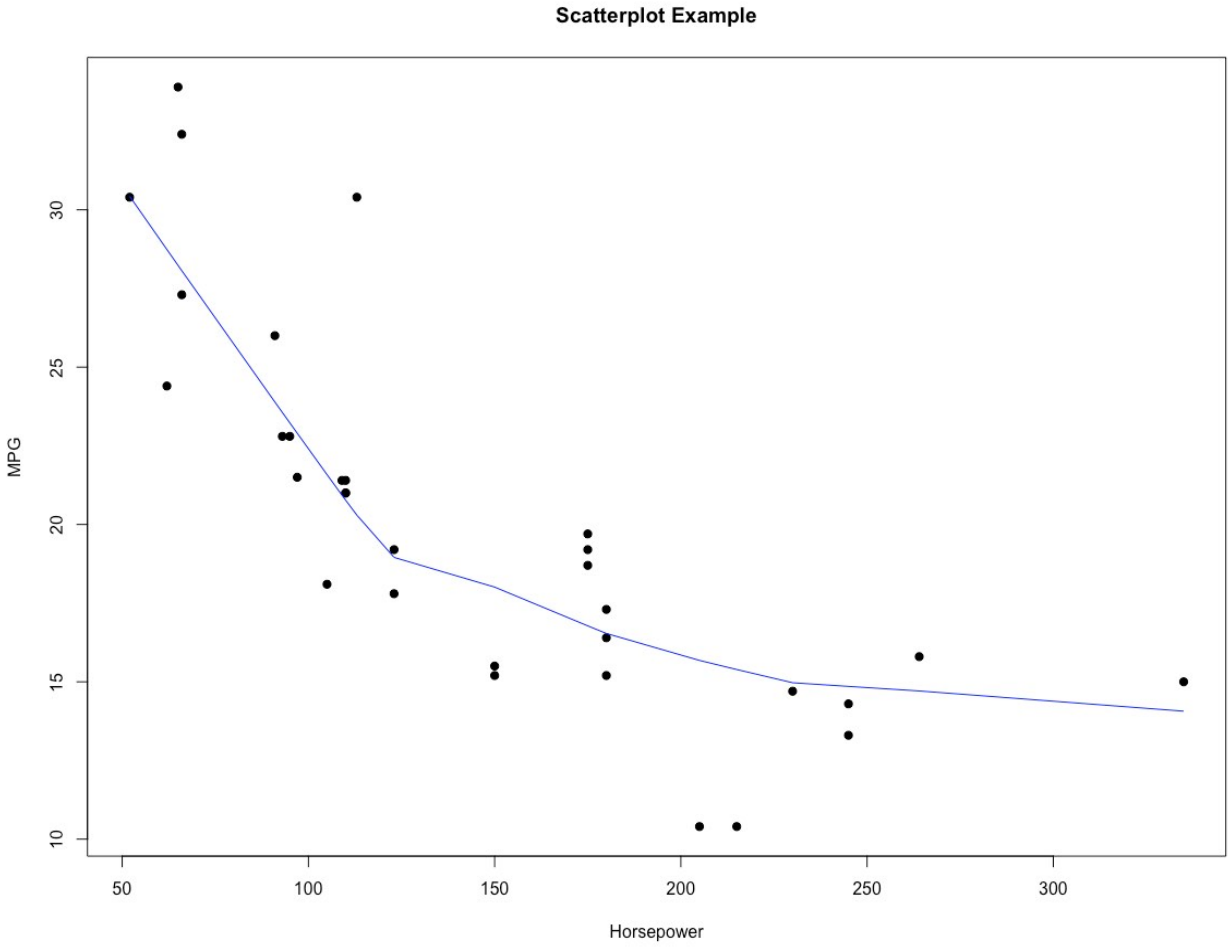


So, using our scatterplot and the lm function it would appear that as weight increases mpg decreases.

Well thats all pretty cool, but a straight line through my data gives me an idea of the trend but can be misleading if the data is wiggly in the scatter plot, or appears not to be trending.

Using the lines() and "lowess" option you can see that the line is a little more in tune with the trend of the data. LOWESS is locally weighted scatterplot smoothing.

```
plot(hp, mpg, main="Scatterplot Example",
     xlab="Horsepower", ylab="MPG", pch=19)
lines(lowess(hp,mpg), col="blue")
```



Depending on the package you are using, there is more than one way to get a line to fit the data. In short, lowess is a smoothing function that uses polynomial regression as opposed to just linear regression. If that made your brain skip, consider linear (straight line) regression versus polynomial (curvy line) regression. We will touch on both soon enough, using this in the scatter plot is just for visualization purposes. But, you can see that as horsepower increases mpg does drop bu starts to level out a bit. The point of the curvy line is to demonstrate that not everything can be addressed by a straight line and there is a function for that.

Lets have a little fun. Hopefully you have played with the mtcars dataset a little bit and maybe even tried out some of the other base R datasets, or loaded your own. The best way to engage in topics like this is to use a dataset you have some passion or curiosity about.

I have a dataset for you on the book github site, FL-Median-Population.

This dataset contains the following;

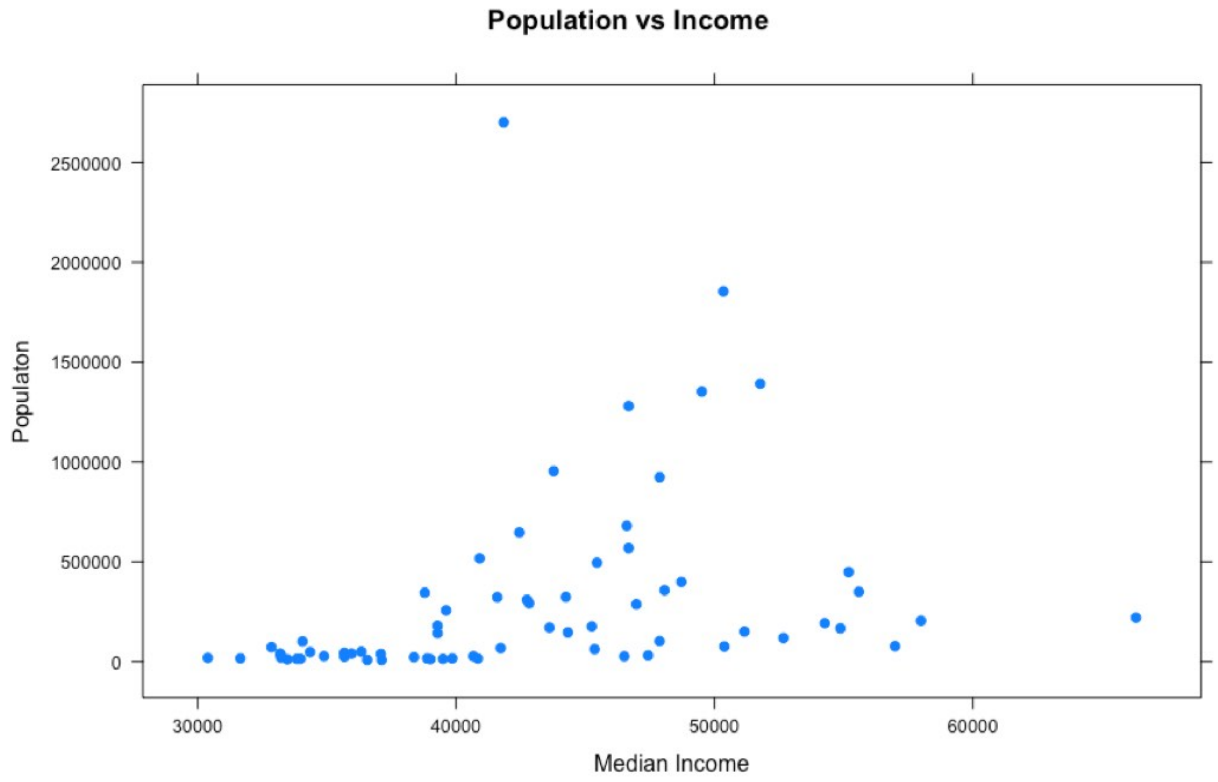
Column	Description
region	County name
CountyFipsCode	Five-digit Federal Information Processing Standards code which uniquely identified counties and county equivalents

	in the United States.
population	
CollegeDegree	
College	

We will be using the xyplot from the lattice package, and the dataset listed above, be sure to change the file location two where every you put it, or use setwd to set your working directory. Or you can leave it alone and load the file directly from the book github location. For this we will start with just the population and median income.

```
    install.packages("lattice")

    library(lattice)


    florida <-
read.csv("https://raw.githubusercontent.com/sqlshep/DS4New/master/Chapter%203/data/FL-Median-Population.csv")


    xyplot(population ~ MedianIncome,
           data=florida,
           pch=19,
           main="Population vs Income",
           xlab="Median Income",
           ylab = "Populaton")
```

Hopefully your plot looks a little bit, or exactly like this one. I think this is a good example because it is an imperfect sample. Hopefully on a good day you will get something more like this, and not complete randomness. One point to notice right away is there is one dot way out of range of the rest of the dots at the top of population. Not hard to figure out that is probably the county that Miami resides in, Miami-Dade, and then four other counties coming in at over a population of 1 million. You will also notice there is one county way off to the right in median income, that is St Johns county, which is where the city of Jacksonville is located. If this were your dataset, this is where you will need to do some research on your own to figure out what is going on. Now the median income of Jacksonville the city is lower than the median income of the county, so what could be going on there?

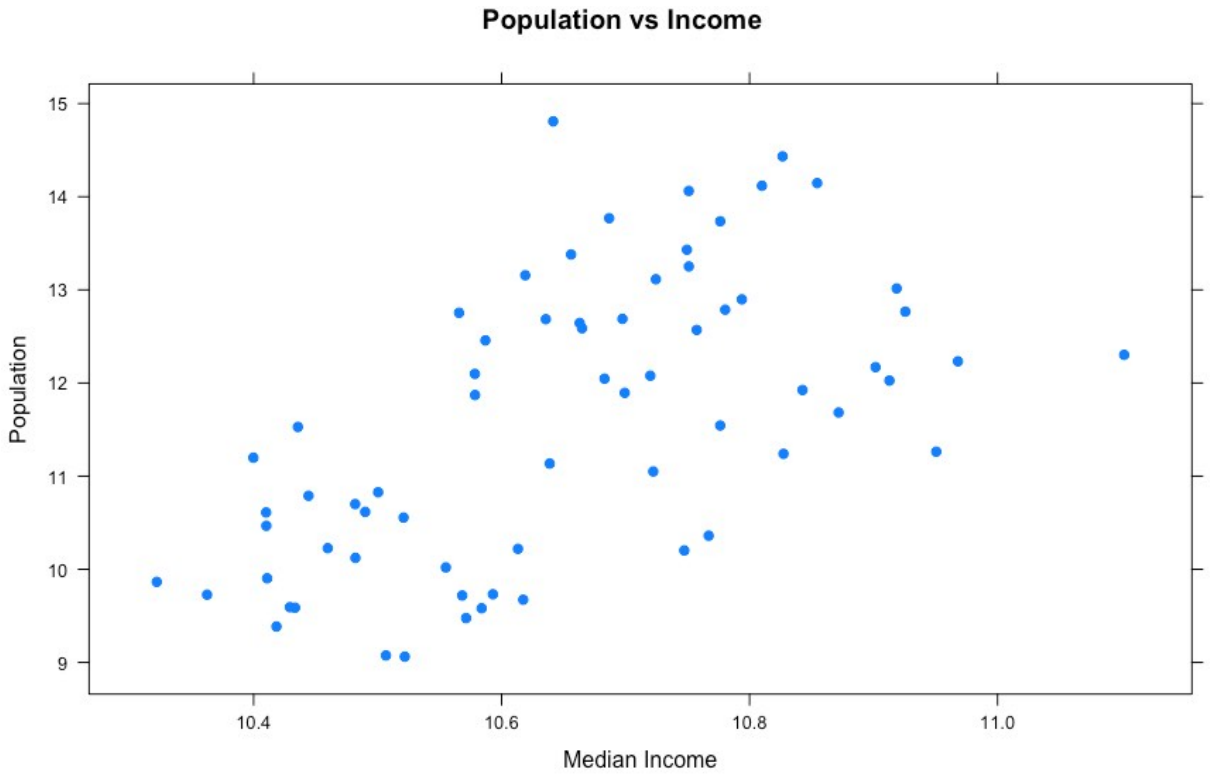


Hmmm, this just raises more questions, it just so happens, with a population of about 27,000 Ponte Vedra Beach has a median income of $116,000 according to wikipedia, so this one city is dragging the average for the entire county of 220,257 up pretty significantly compared to other counties. So, the with the population of Miami-Dade and the median income of St. Johns being far away from the rest of the data, these are what we call outliers. For now we are going to leave them in, i will discuss outliers throught the book, they are complicated to deal with at best. Clearly one like the county of St. Johns will need to be handled eventually.

So from looking at the scatter plot, we can kind of make out a general direction of the relationship of income to population, but it is sort of vague. In cases like this there are a few things to do.

1. When your data looks like it has gone to crazy town, try applying a transformation function for a better graphical representation. This will change the x/y scale, but will still represent the trend of the data. More on log transforms here.

```
xyplot(log(population) ~ log(MedianIncome),
       data=florida,
       main="Population vs Income",
       xlab="Median Income",
       ylab = "Population",
       pch=19)
```

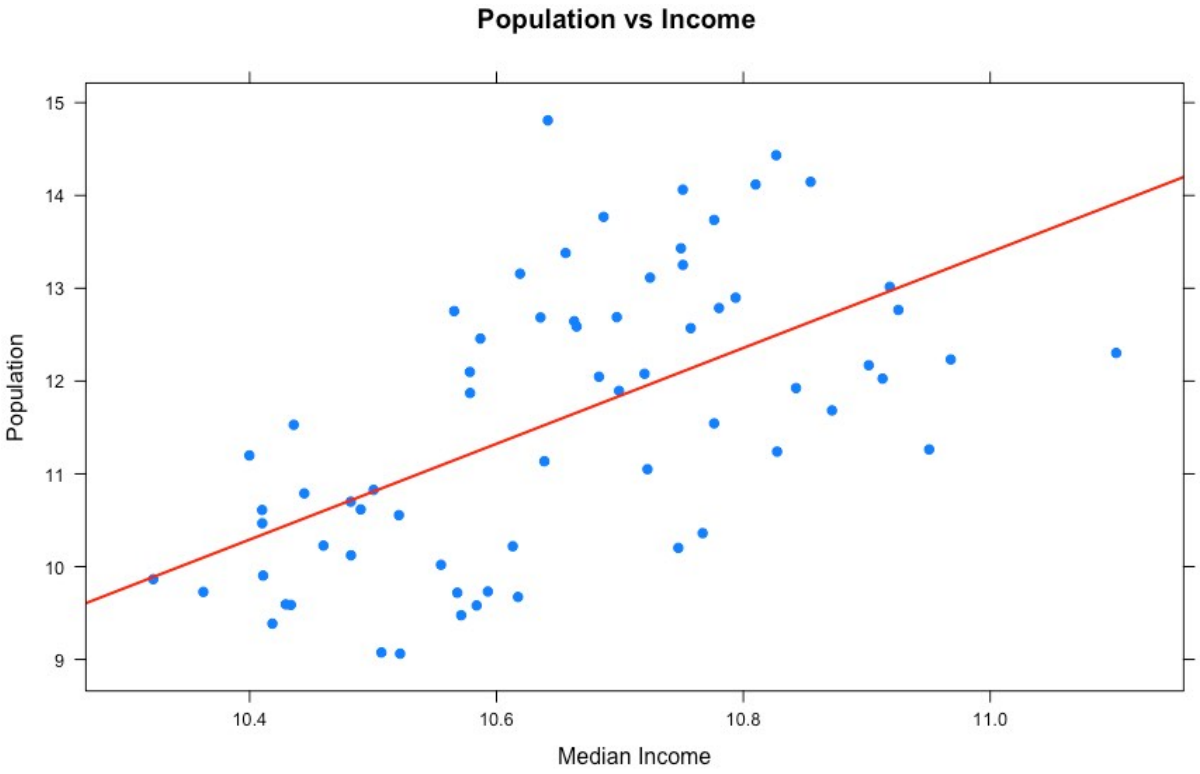


The log transform did clean up the scatterplot quite a bit, there does appear to be a trend.

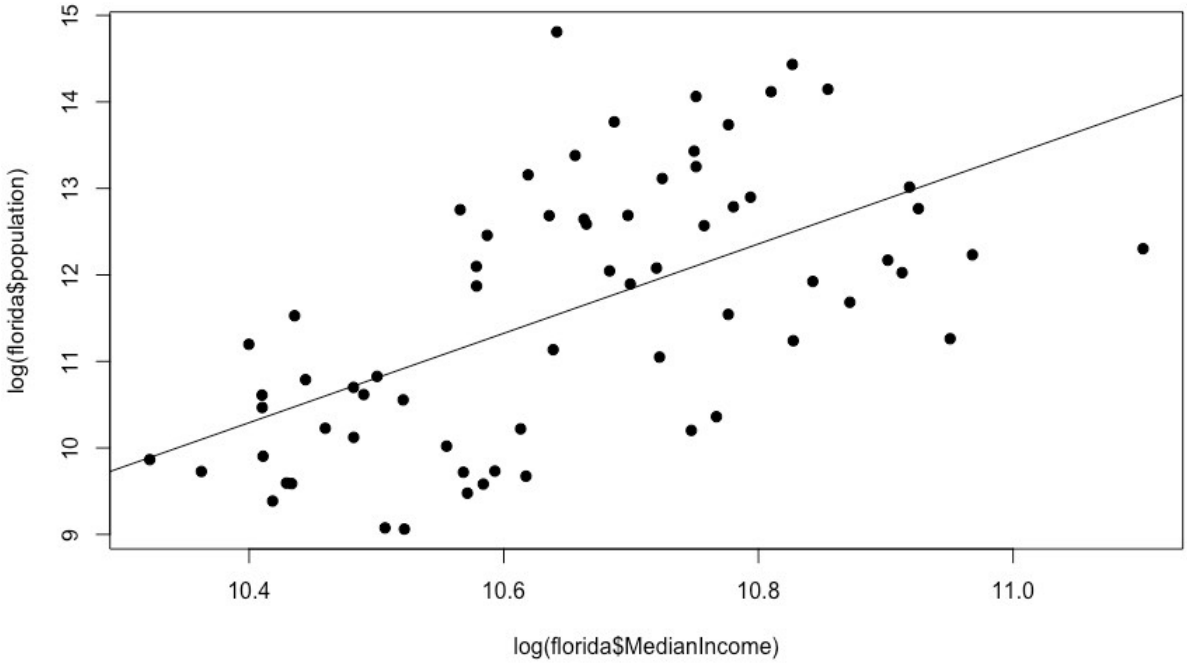
2. Draw a linear regression line through it so see if there is a trend. We will get into lm() later, but it takes the (y ~ x) as model input and returns an intercept and slope, remember algebra?

There is more than one way to do this, xyplot uses the panel function, the much lengthier syntax.

```
xyplot(log(population) ~ log(MedianIncome),
       data=florida,
       main="Population vs Income",
       xlab="Median Income",
       ylab = "Population",
       panel = function(x, y, ...) {
       panel.xyplot(x, y, ...)
       panel.abline(lm(y~x), col='red',lwd=2)},
       pch=19)
```



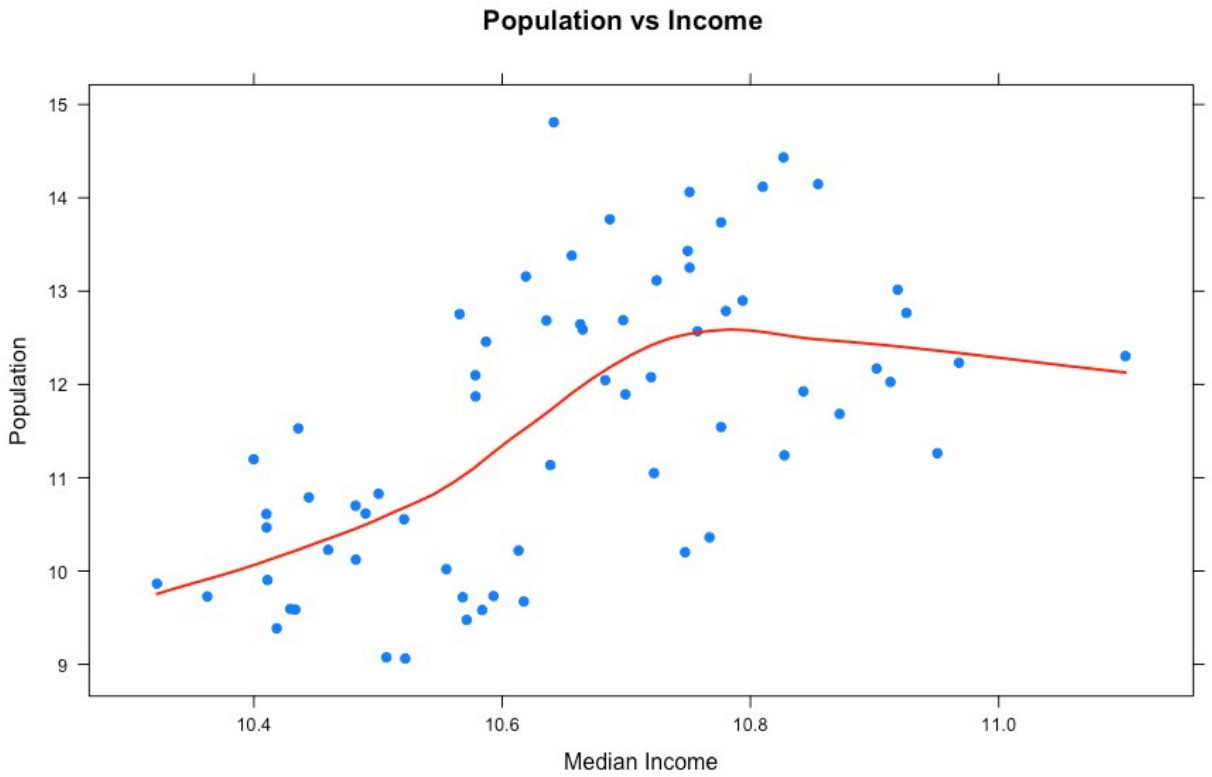

```
##
## OR
##

plot(log(florida$MedianIncome),log(florida$population),pch=19)
abline(lm(log(florida$population) ~ log(florida$MedianIncome)))
```



3. In the previous sample we just used a lm, straight line, slope and intercept to run a line through the data, that alone does show a trend. Even if you remove the log function you still seen upward trend of greater population seems to indicate greater income.

So lets try the LOWESS again, this time with xyplot. The type parameter below is using a "p" parameter, this is the LOWESS (locally weighted scatterplot smoothing), it will take our bumpy data and smooth the line to the data.

```
xyplot(log(population) ~ log(MedianIncome),
       data=florida,
       main="Population vs Income",
       xlab="Median Income",
       ylab = "Population",
       type = c("p", "smooth"), col.line = "red", lwd = 2,
       pch=19)
```



One thing appears to be somewhat clear from this, as the population of the county increases, the income does increase to a point hen it seems to stabilize.

There are a couple more columns in the Florida data provided that you can try on your own, see if you can visually show a relationship between college and income, or even college and population. Do more rural counties have more or less college educated population?

Section 3

STEM AND LEAF

To be truthful these are not visualization i use everyday. Stem and Leaf plot, Boxplot, Frequency Polygon.

First lets get some data loaded, we will switch to a mosaic data package for this one. We will be using a data set called HELPmiss, Health Evaluation And Linkage To Primary Care – The HELP study was a clinical trial for adult inpatients recruited from a detoxification unit. Patients with no primary care physician were randomized to receive a multidisciplinary assessment and a brief motivational intervention or usual care, with the goal of linking them to primary medical care. There are three HELP datasets in mosaicData, feel free to go exploring.

```
install.packages("mosaicData")

library("mosaicData")


data(HELPmiss)

View(HELPmiss)


str(HELPmiss)


x <-
HELPmiss[c("age","anysub","cesd","drugrisk","sex","g1b","homeless","race
grp","substance")]


#stem is the first one

stem(x$cesd)
```

Magical right? In the days of PowerBi, Tableau, ggplot and Edward Tufte this does not look like anything i want to see on a report. But, lets figure it out.

For this output, the stem is the first digit on the left, and the leaf would be each digit on the right of the "|". the first row, "0 | 1334444" is equal to the values 1,3,3,4,4,4,4, and continued on the second row of the stem and leaf plot. Looking at the last row you will see "6 | 0" which means that one 60 occurred in the vector. I have not used this nor seen this used in modern day modern visualizations.

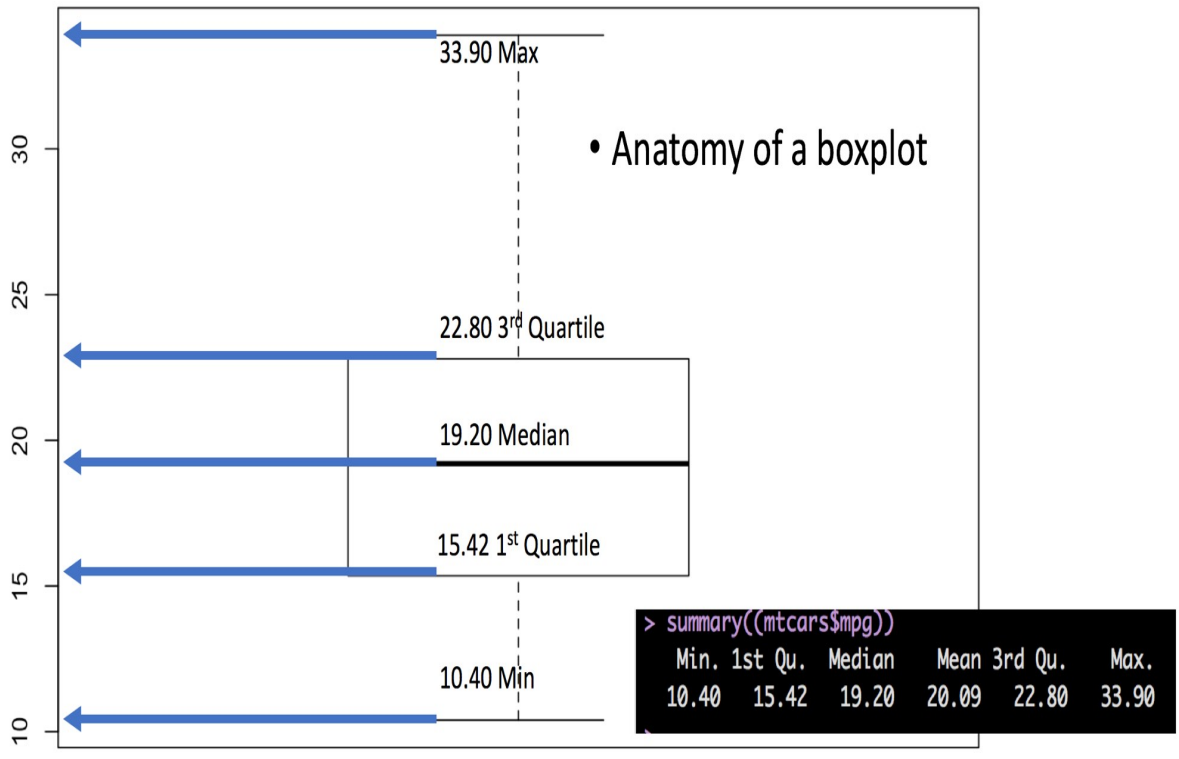
```
The decimal point is 1 digit(s) to the right of the |

0 | 1334444
0 | 56666778889999
1 | 00001111122223444
1 | 55555555566666667777777788888899999999
2 | 000011111111222222333333333444444444
2 | 555555566666666667777777777888888888888999999999999999
3 | 00000000000000111111111111111222222223333333333333344444444444444444
3 | 555555555555666666666666666666667777777777777777788888888888889999+3
4 | 0000000000000000000111111111111222222222333333333444444444
4 | 55555555556666666666777777788888899999999
5 | 00111111111111222223333344444
5 | 555666777888
6 | 0
```

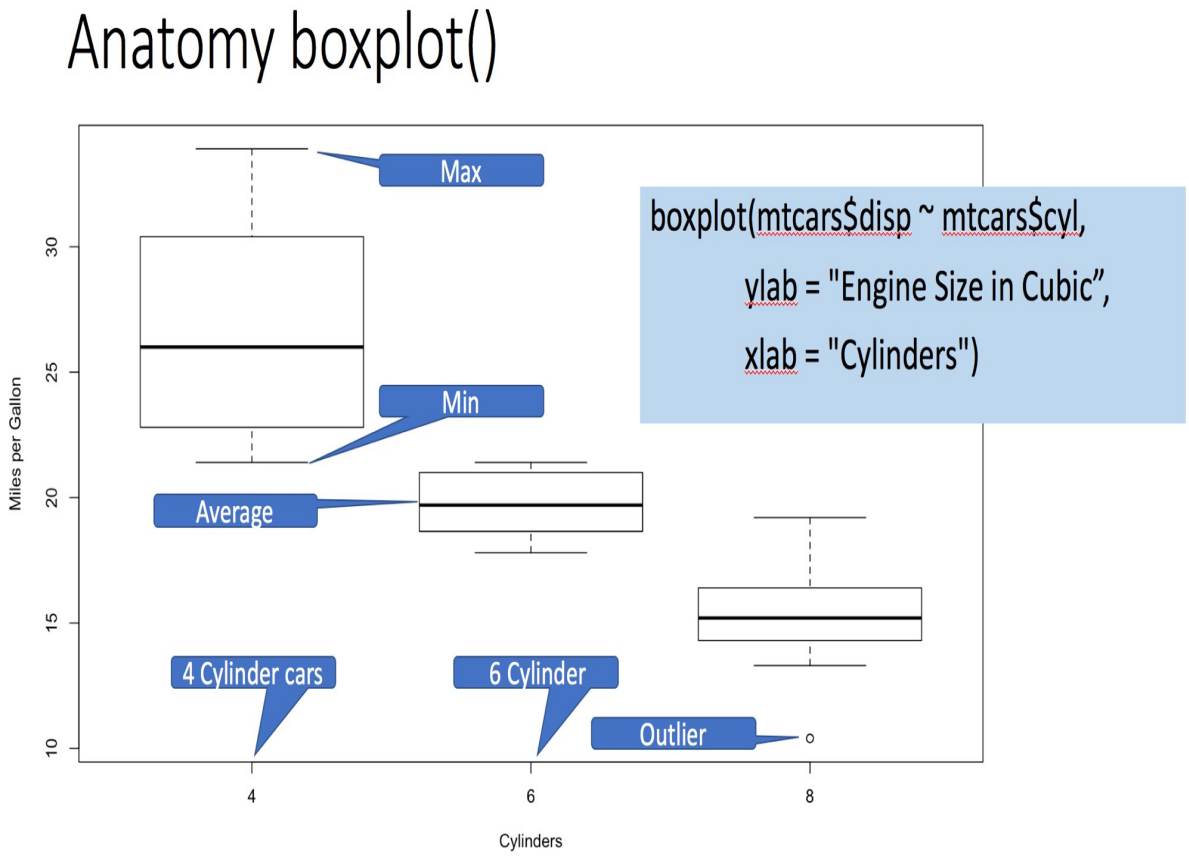
Section 4

BOXPLOT

Boxplot or, whisker plot! You will see this one, and its always a little fuzzy to recall what is what until you have been doing it for a while.

It is said a picture is worth a thousand words, so will leave these here. THer following two samples are from the mtcars dataset that ships with R, so you can easily recreate them. Quartiles is coming up in the next chapter, so if you need to can jump to the Range and IQR section if you need to then come back to this.



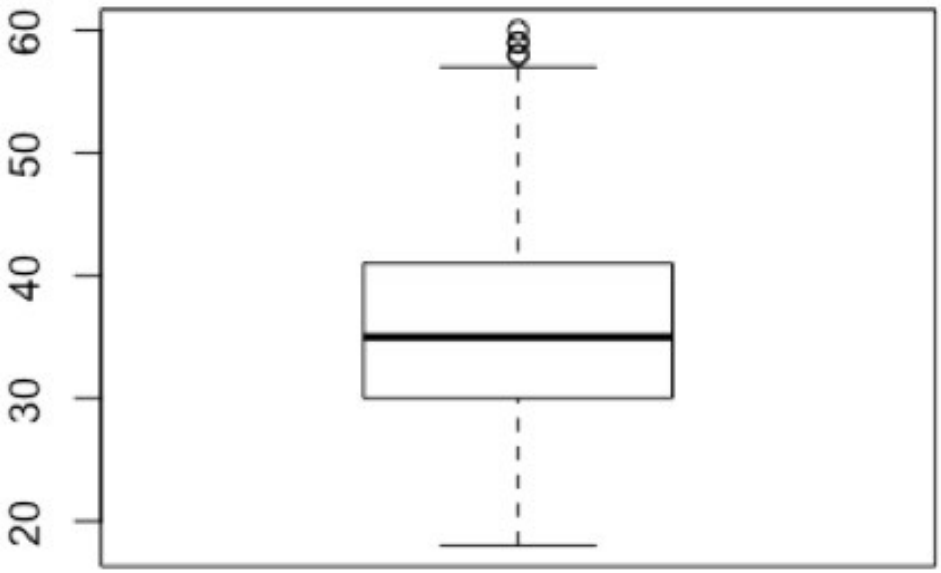


You can also go a little crazy of you like by using one plot, or combining one quantitative variable and many qualitative, like the last one.

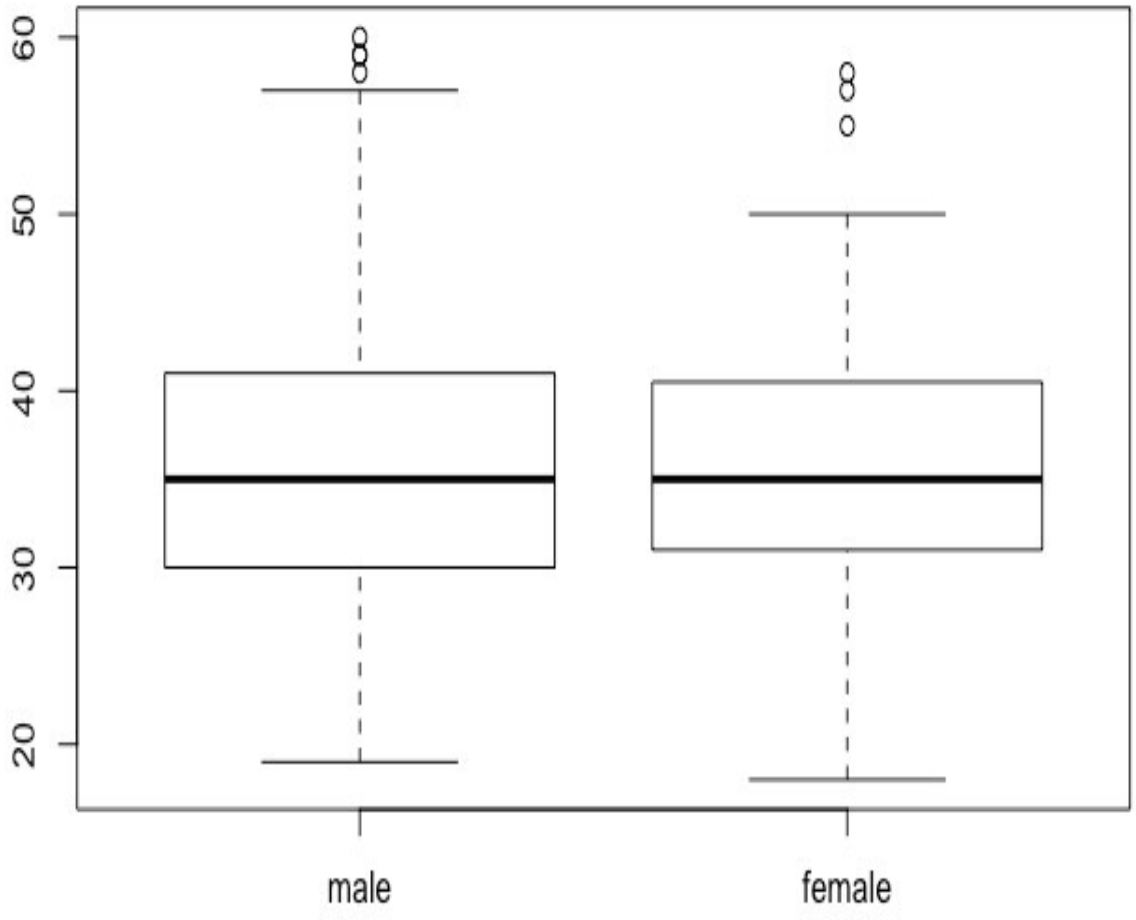
```
install.packages("mosaicData")

library("mosaicData")


data(HELPmiss)

View(HELPmiss)


str(HELPmiss)


x <-
HELPmiss[c("age","anysub","cesd","drugrisk","sex","g1b","homeless",
"racegrp","substance")]
```

```
?boxplot
```

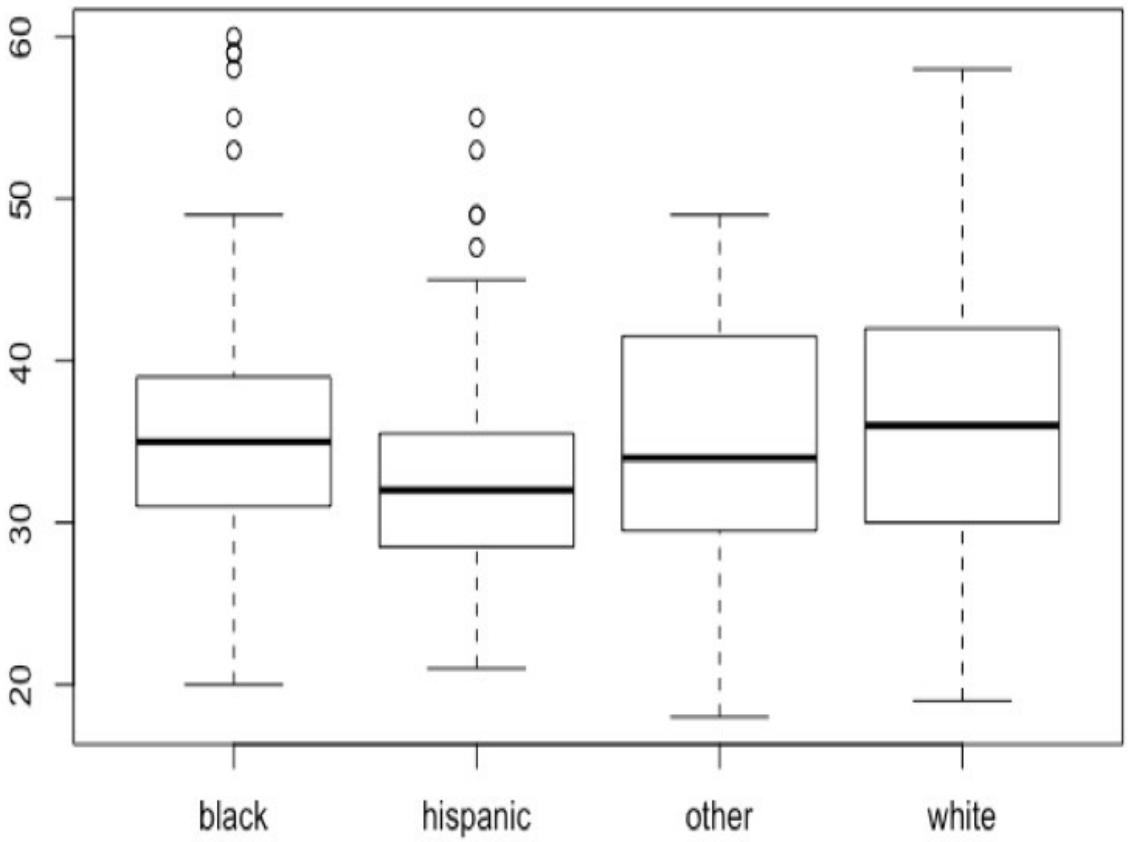
```
boxplot(x$age)
```




```
boxplot(age ~ sex, data=x)
```




```
boxplot(age ~ racegrp, data=x)
```



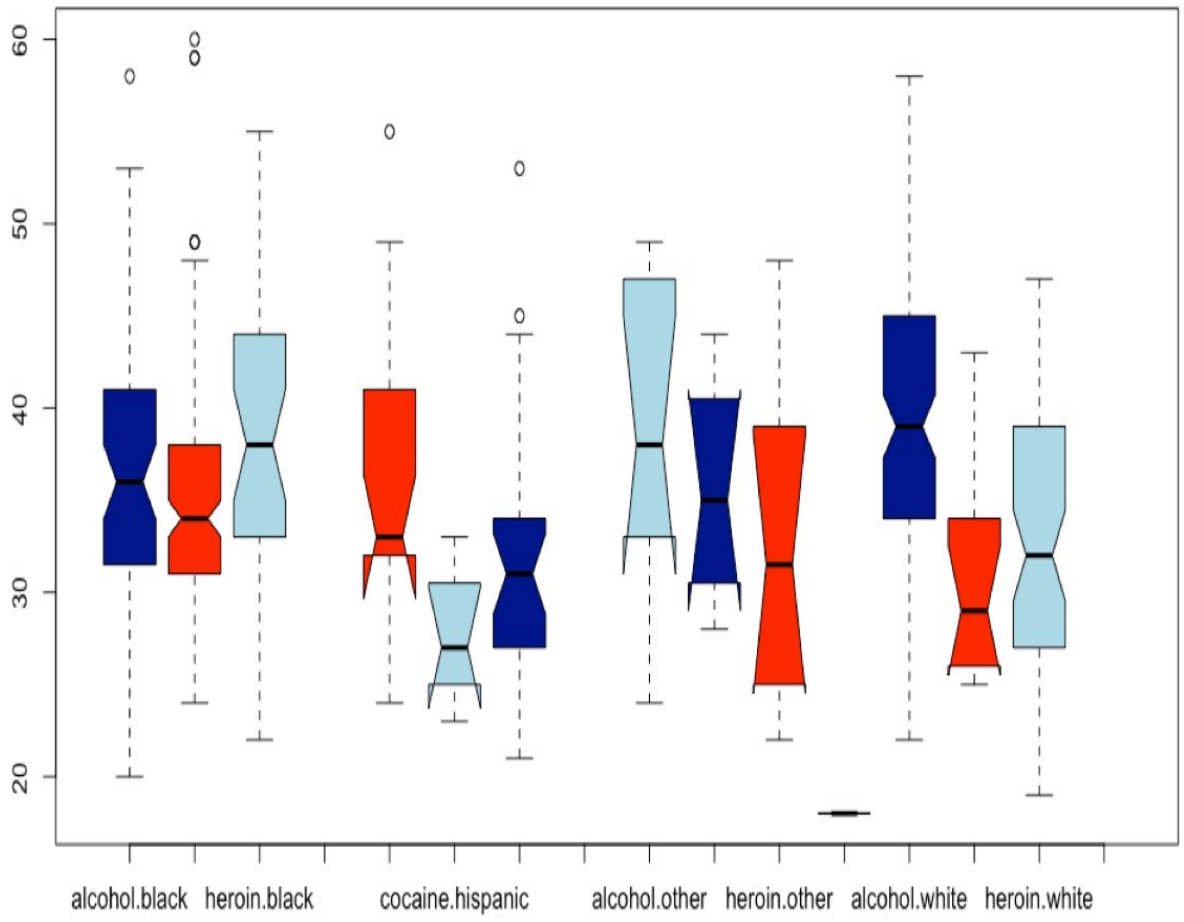

```
> summary(subset(x$age, x$racegrp=="black" ))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   20.0    31.0    35.0    35.8    39.0    60.0
> sd(subset(x$age, x$racegrp=="black" ))
[1] 7.12124
```

Lets look at age and racegrp black, the median of age is 35, the standard deviation is 7.1 (rounded). Using 2 standard deviations this would give us a range of 20.8 to 49.2, if you look at the boxplot for racegrp black, you will see that this does cover age, about 20 to 50. So what are the dots? Remember a few blogs back when using IQR has a tool to detect outliers? This is it,

anything beyond two standard deviations for this boxplot is considered an outlier, the hollow circle.

```
boxplot(age ~ substance*racegrp, data=x, notch=TRUE, col=
(c("darkblue","red","lightblue")))
```

This one is mostly eye candy to show how much you can do, this one is called a notched boxplot that demonstrates statistical distributions of data.



Chapter 4

GGPLOT

Lorem Ipsum

I debated on whether to make this a section or a chapter, i clearly decided on a chapter so it has some level of organization and division. This is meant to be brief, and i will keep it as brief as possible.

GGplot has for the most part become the defacto standard in visualizations for R. If you are in a hurry use base R graphics, otherwise use ggplot. Ggplot is very wordy, meaning the syntax can get pretty lengthy. Luckily there are dozens of books available to you and hundreds of websites to help you along your way.

Ggplot2 is a package that can be downloaded by itself, or you can downlaod it thorugh a package called tidyverse, of which i will not get into. If you plan on a career in R, you will want to learn the tidyverse at some point. I personally have a cursory knowledge of the tidyverse packages as i moved into Python pretty quickly.

Ggplot was based on a philosphy set forth in a book called the Grammar of Graphics, i have it it, its not worth readin gunless you feel like you need a PhD in visualization and graphic design, i do not.

For a rapid start ggplot cheat sheet will get you running. Most of this section will be short description with extended examples.

Section 1

SCATTERPLOT

Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet, sagittis felis sodales, dolor sociis mauris, vel eu est libero cras. Interdum at. Eget habitasse elementum est, ipsum purus pede porttitor class, ut lorem adipiscing, aliquet sed auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec pellentesque leo, temporibus scelerisque nec.

Ac dolor ac adipiscing amet bibendum nullam, massa lacus molestie ut libero nec, diam et, pharetra sodales eget, feugiat ullamcorper id tempor eget id vitae. Mauris pretium eget aliquet, lectus tincidunt. Porttitor mollis imperdiet libero senectus pulvinar. Etiam molestie mauris ligula eget laoreet, vehicula eleifend. Repellat orci eget erat et, sem cum, ultricies sollicitudin amet eleifend dolor nullam erat, malesuada est leo ac. Varius natoque turpis elementum est. Duis montes, tellus lobortis lacus amet arcu et. In vitae vel, wisi at, id praesent bibendum libero faucibus porta egestas, quisque praesent ipsum fermentum placerat tempor.

Curabitur auctor, erat mollis sed fusce, turpis vivamus a dictumst congue magnis. Aliquam amet ullamcorper dignissim molestie, sed mollis. Tortor vitae tortor eros wisi facilisis.Consectetuer arcu ipsum ornare pellentesque vehicula, in vehicula diam, ornare magna erat felis wisi a risus. Justo fermentum id. Malesuada eleifend, tortor eros.

Section 2

HISTOGRAM

Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet, sagittis felis sodales, dolor sociis mauris, vel eu est libero cras. Interdum at. Eget habitasse elementum est, ipsum purus pede porttitor class, ut lorem adipiscing, aliquet sed auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec pellentesque leo, temporibus scelerisque nec.

Ac dolor ac adipiscing amet bibendum nullam, massa lacus molestie ut libero nec, diam et, pharetra sodales eget, feugiat ullamcorper id tempor eget id vitae. Mauris pretium eget aliquet, lectus tincidunt. Porttitor mollis imperdiet libero senectus pulvinar. Etiam molestie mauris ligula eget laoreet, vehicula eleifend. Repellat orci eget erat et, sem cum, ultricies sollicitudin amet eleifend dolor nullam erat, malesuada est leo ac. Varius natoque turpis elementum est. Duis montes, tellus lobortis lacus amet arcu et. In vitae vel, wisi at, id praesent bibendum libero faucibus porta egestas, quisque praesent ipsum fermentum placerat tempor.

Curabitur auctor, erat mollis sed fusce, turpis vivamus a dictumst congue magnis. Aliquam amet ullamcorper dignissim molestie, sed mollis. Tortor vitae tortor eros wisi facilisis.Consectetuer arcu ipsum ornare pellentesque vehicula, in vehicula diam, ornare magna erat felis wisi a risus. Justo fermentum id. Malesuada eleifend, tortor eros.

Chapter 5

JUST THE STATS MAM

Basic Intro To Stats

In this chapter we will cover statistics terms and basic visualizations.

Let me take this opportunity to apologize. Statistics is necessary, i will try to make it as painless as possible and i am going to give you the absolute essentials. There is no escaping it, you are going to have to know some stats stuff. Through out the book i will cover some of this, its good for me to brain dump to paper every now and then, and its good for you to try and assimilate it. Nothing is going to make sense without it. It only hurts a little bit and I am going to try and give as many samples as possible and break down each concept as much as I possibly can. In the end you will be grateful for software, but stats came from long hand math calculations and have formulas that look like they could single handedly put a rocket in space.

Stick with it, it gets better, except for probability, probability does not get better.

Section 1

COMMON TERMS

If you are a RDBMS gal or guy some of the definitions will seem wonky, and if you are talking to an academic this will be the first communication breakdown, hopefully this will help.

Lets get some definitions out of the way, these are critical, it does not mean you need to question the lingo you use on a daily basis, but if you hear it from you statisticians or DS folks, make sure you are on the same page.

Population – A population is ALL THE DATA. As a data guy I lived in a world where if we were trying to make a decision we used all the data, this is why we had SQL performance problems, our customers sucked at statistics and did not know there was a way to sample a data set. That being said, i also would not like to check my bank account balance based on a sample. So, when you see a population referenced that means the entire data, all of it. If I am using a population of the United States, that means I am using all 330 million or so people in the US for my research. In Statistics, it is denoted as an uppercase N.

Census – A census is a study of EVERYTHING in a given population. Most countries have a census. One of the more popular results of the census is the American Community Survey, it frequently provides great statistical training and research material.

Sample – A sample is as it sounds, it's a sample of a population, hence not all the data. There are sub-classes of samples we will get into later. But for now know that a sample is a portion of a population. In Statistics, it is denoted as an lowercase n.

Parameter – A parameter is a numerical quantity that tells us something about the population, such as quantity of a specific ethnicity, number high school graduates, proportion of singles. Do not confuse this with a numeric quantity of sample, that is called a statistic. Ah ha!

Variable – A variable in statistics is what most SQL Folks, me included call a column, there is a very long definition, they contain anything that describe a characterization, qualitative and quantitative.

Case – A case in statistics is a single row of data. You can imagine that if you have a patient, all of the columns(variables) will make up the data for that patient, hence it is called a case. So, if you hear this outside of academia see if they are discussing a row, or something else entirely. Usually, this terminology references something sciencey, less so outside of medical research or scientific fields. I have never heard a row of banking data referred to as a case.

Data – Plural for of data. Some get bent out of shape over the use of this, I find them more annoying than the usage, I mean its not like I am using there their they're incorrectly.

Datum – Singular form of data.

Qualitative Data – In many respects this is an easy one, if its not Quantitative its probably Qualitative, another more familiar name for it is categorical or, a category. Categorical data is defined as which? Which color, which model of, which dog breed, which grade are you in.

If you recall, looking at the dataset we used for the Florida education choropleth you will recall that there was a variable called ruralurbancontinuum , though it was a number it was used as a categorical value. The values of this field are 1-9 and related to a category of population density used US wide. In this case no math could be done gainst the value even though it is a number, the Census could just have easily used A-I instead of 1-9.

Quantitative Data – This one is pretty easy if you remember that you can do math on a quantitative variable. Its is always a number. It can be my height, my weight, my pulse rate, the money in my checking account, my shoe size. If I have population or sample of these items I can average them, get a standard deviation etc. The word quantitative has a root of quantity, that should help you remember it.

To go a little bit deeper into the rabbit whole there are two types of Quantitative data, I know, I'm sorry… Hopefully looking at the root word of each will help.

Quantitative Discreet – Counting data. They ask How Many?

How many people on a bus?
-There are 20 people on the bus, not 19.5 or 20.5.

How many cars in my driveway?
-There are 2 cars in my driveway and one in the yard, though most of that on is missing it is still one car.

How many books do you own?
-I own 100 books, not 99.5, though an argument could be made for owning half a book, it is still one registered ISBN even if you do not have all of the book.

How many emails did you get to day?
-I received 50 emails today.

Quantitative Continuous – Measuring data, this asks How Much?

What is your height?
What is your weight?
What is the weight of a vehicle?
What iS the MPG of a vehicle?

Section 2

RANGE AND IQR

The next topics (Range, IQR, Variance and Standard Deviation) took up a combined 120 power point slides in my stats class, which means that describing all in a single post will not happen, and maybe two posts minimum, but I will try to keep it under 120 slides or pages.

So, range, IQR (Interquartile Range), variance and standard deviation fall under summary measures as ways to describe numerical data.

Range – is the measure of dispersion or spread. Using central tendency methods we can see where most of the data is piled up, but what do we know about the variability of the data? The range of the data is basically the maximum value – the minimum value.

What to know about range? It is sensitive to outliers. It is unconcerned about the distribution of the data in the set.

For instance, if I had a hybrid car in my mtcars dataset that achieved 120 mpg by the petrol standards set forth by the EPA, my range for mpg would be 10.40mpg to 120mpg. If I told you the cars in my sample had a mpg range of 10.40mpg to 120mpg what would you think of the cars? What range fails to disclose is that the next highest mpg car is 33.9, that's pretty far away and not all representative of the true dataset.

Run the following, try it out on your own data sets.

```
data(mtcars)

View(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1

```
range(mtcars$mpg)

range(mtcars$wt)

range(mtcars$hp)
```

```
> range(mtcars$mpg)
[1] 10.4 33.9
> range(mtcars$wt)
[1] 1.513 5.424
> range(mtcars$hp)
[1]  52 335
>
```

```
# if you are old school hard core,

# "c" is to concatenate the results.


c(min(mtcars$hp),max(mtcars$hp))
```

```
> c(min(mtcars$hp),max(mtcars$hp))
[1]  52 335
```

Interquartile Range – since we have already discussed quartiles this one is easy, the inter-quartile-range is simply the middle 50%, the values that reside between the 1st quartile(25%) and the first 3rd(75%) quartile. Summa-

ry() and favstats will give us the min(0%), Q1, Q2, Q3, max (100%)as will quantile().

```
quantile(mtcars$mpg)
```

```
> quantile(mtcars$mpg)
    0%    25%    50%    75%   100%
10.400 15.425 19.200 22.800 33.900
```

```
summary(mtcars$mpg)
```

```
> summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

IQRs help us find Outlier which is an observation point that is distant from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set.

One of the techniques for removing outliers is to use the IQR to isolate the center 50% of the data. Lets use the Florida dataset from the scatterplot blog and see how the plot changes.

I am going to demonstrate the way i know to do this. Understand this is a method to perform this task, two years from now i will probably think this is amateuresque, but until then here we go.

We will need the first quintile and the third quantile and then subtract one from the other. to do this we are going to use summary().

```
florida <- read.csv("https://raw.githubusercontent.com/sqlshep/DS4New/master/Chapter%203/data/FL-Median-Population.csv")
```

```
# Checkout everything summary tells us
summary(florida)
```

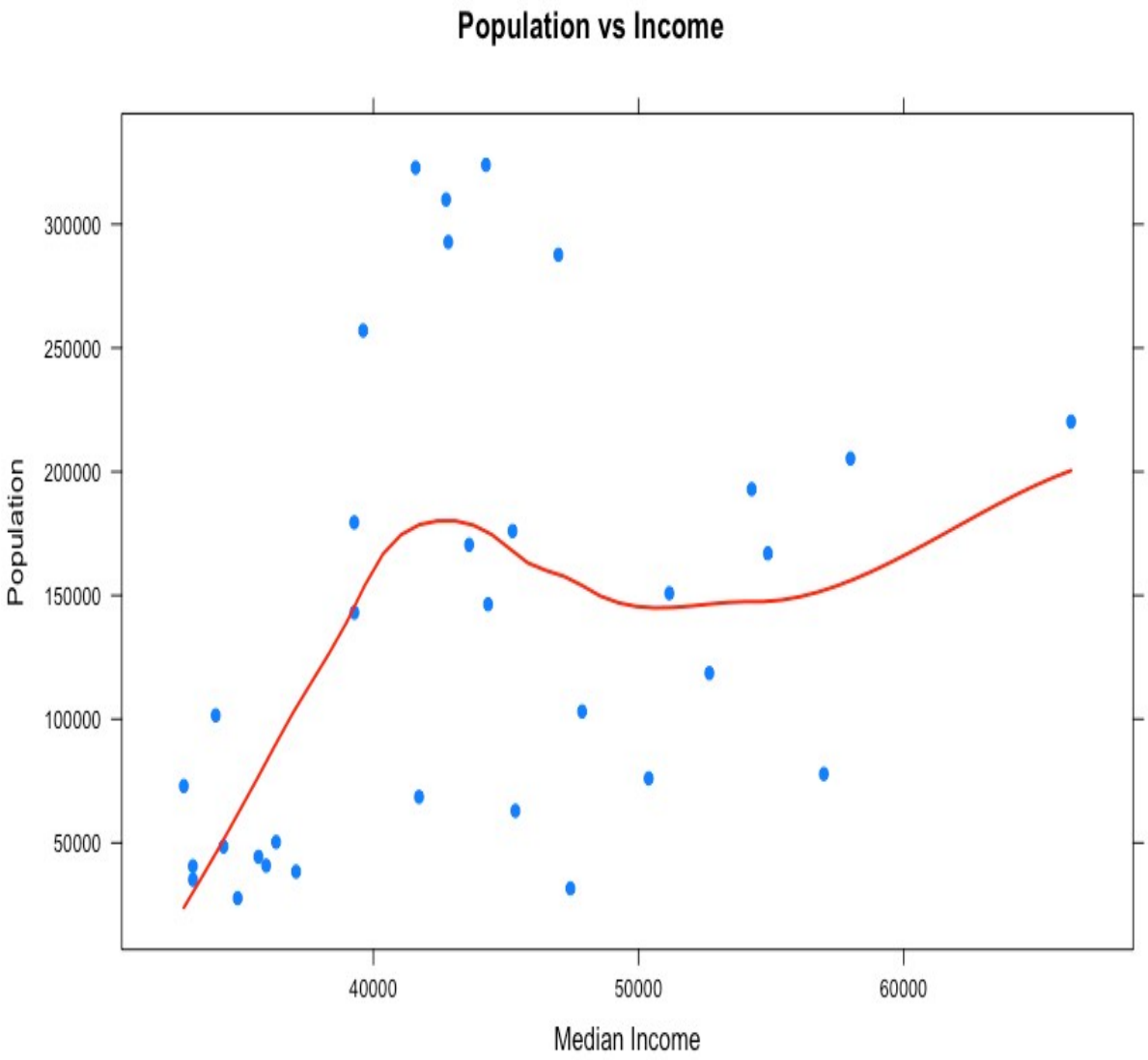
```
> summary(florida)
     region   CountyFipsCode   MedianIncome     population      CollegeDegree      College
 Alachua : 1   Min.   :12001   Min.   :30391   Min.   :  8621   Min.   :0.0750   Min.   :28.10
 Baker   : 1   1st Qu.:12036   1st Qu.:36823   1st Qu.: 27538   1st Qu.:0.1150   1st Qu.:40.25
 Bay     : 1   Median :12069   Median :41840   Median : 118577   Median :0.1860   Median :49.90
 Bradford: 1   Mean   :12068   Mean   :42896   Mean   : 300726   Mean   :0.2023   Mean   :49.86
 Brevard : 1   3rd Qu.:12100   3rd Qu.:47651   3rd Qu.: 334867   3rd Qu.:0.2660   3rd Qu.:59.50
 Broward : 1   Max.   :12133   Max.   :66312   Max.   :2700794   Max.   :0.4430   Max.   :72.60
 (Other) :61
```

```
# Now isolate the column we are interested in
summary(florida$population)
```

```
> summary(florida$population)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   8621   27540  118600  300700  334900 2701000
```

```
# Now a little R indexing,
# the values we are interested in are the 2nd and 5th position
# of the output so we just reference those

summary(florida$population)[2]
summary(florida$population)[5]
```

```
    # load the values into a variable
    q1 <- summary(florida$population)[2]
    q3 <- summary(florida$population)[5]

    #Now that we have the variables run subset to
grab the middle 50%
    x<-subset(florida,population >=q1 & population
<= q3)

    #And lets run the scatterplot again
    xyplot((population) ~ (MedianIncome),
           data=x,
           main="Population vs Income",
           xlab="Median Income",
           ylab = "Population",
           type = c("p", "smooth"), col.line =
"red", lwd = 2,
           pch=19)
```



Notice what happened? By removing everything outside of the IQR our observations (Rows) went from 67 counties to 33 counties, that is quite literally half the data that got identified as an outlier because of the IQR outlier methodology. On the bright side our scatter plot looks a little more readable and realistic and the regression looks similar but bit more wiggly than before.

So what to do? When you wipe out half your data as an outlier this is when you need to consult the powers that be. In real life you will be solving a problem and there will be some guidance and boundaries provided. Since this is just visualization, the stakes are pretty low. If you are in exploration and discovery phase, guess what, you just discovered something. If you are looking at this getting ready to make a predictive model, is throwing out

50% of the data as outlier data the right decision? Its time to make a decision. The decision i am going to make is to try out a different outlier formula. How about we chop 5% of both ends and see what happens? If the dataset were every single count in the US, this may be different.

To do this we are going to need use quantile().

```
    # Using quantile will give us some control of
the proportions

    # Run quantile first to see the results.
```

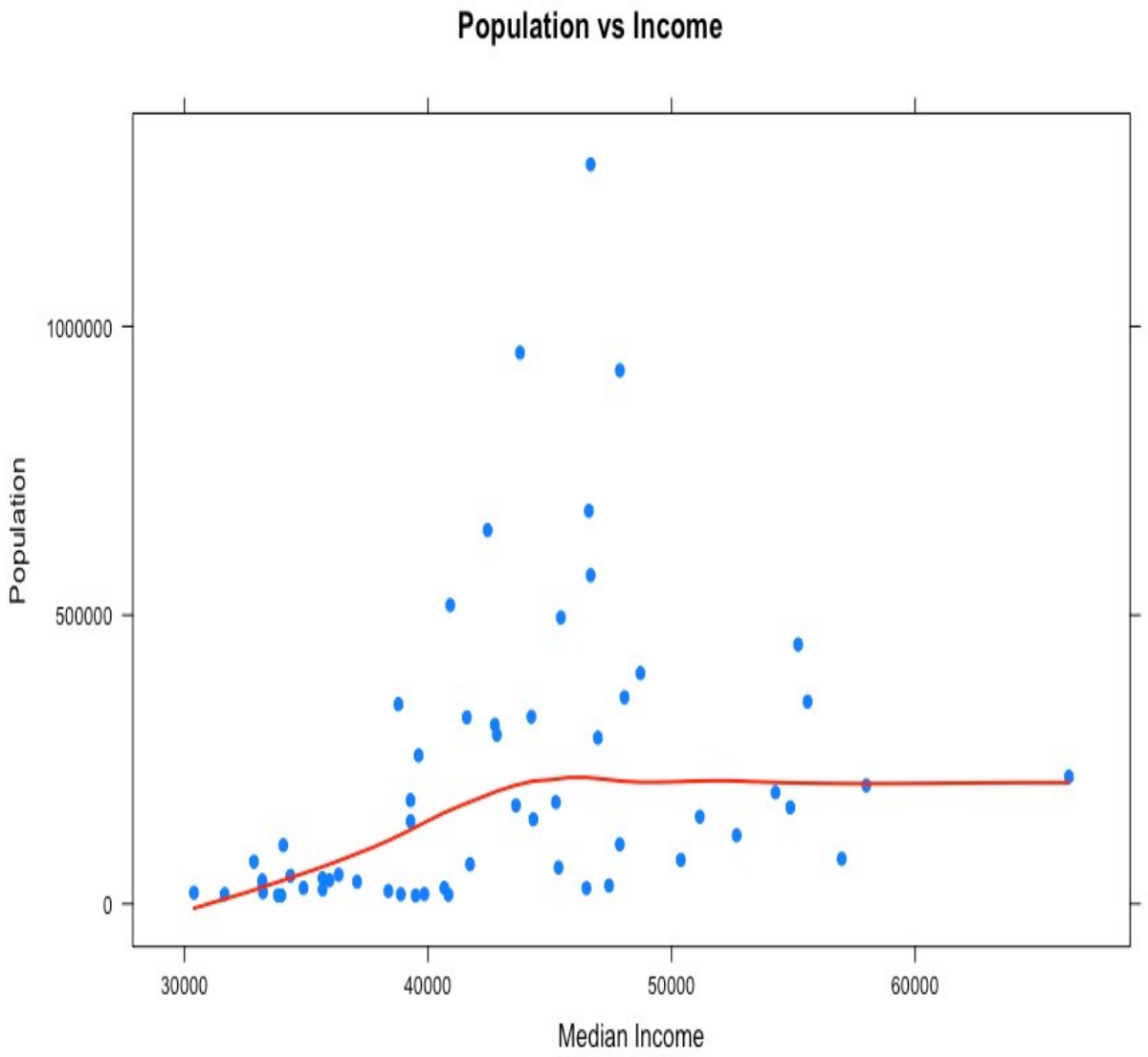
```
    quantile(florida$population,probs = seq(0, 1,
0.05))
```

```
> quantile(florida$population,probs = seq(0, 1, 0.05))
      0%       5%      10%      15%      20%      25%      30%      35%
  8621.0  13482.3  15398.2  16840.5  22960.0  27538.5  37724.2  44762.7
     40%      45%      50%      55%      60%      65%      70%      75%
 65192.2  77302.8 118577.0 155711.7 178108.2 218763.4 296258.0 334867.0
     80%      85%      90%      95%     100%
391148.6 522561.8 777782.2 1331074.0 2700794.0
```

```
    #Load q05 with the results of quantile at the
5% percentile

    #Load q95 with the results of quantile at the
95% percentile
```

```
    q05 <- quantile(florida$population,probs =
seq(0, 1, 0.05))[2]

    q95 <- quantile(florida$population,probs =
seq(0, 1, 0.05))[20]
```

```
    #Create the dataframe with the subset
    x<-subset(florida,population >=q05 & population
<= q95)


    #try the xyplot again
    xyplot((population) ~ (MedianIncome),
           data=x,
           main="Population vs Income",
           xlab="Median Income",
           ylab = "Population",
           type = c("p", "smooth"), col.line =
"red", lwd = 2,
           pch=19)
```



Did we make it better? We made it different. We also only dropped 8 counties from the dataset, so it was less impactful to the dataset. You can see that some of these are not going to be as perfect or as easy as mtcars, and that's the point. Using the entire population of the US with the interquartile range may be a reasonable method for detecting outliers, but its never just that easy. More often than not my real world data is never in a perfect mound with all the data within 2 standard deviations of the mean, also called the normal distribution. If this had been county election data, 5 of those 8 counties voted for Clinton in the last presidential election, if you consider that we tossed out 5 of the 9 counties she won what is the impact of dropping the outliers? Keep in mind that 67 observations(rows) is a very small dataset too. The point is, always ask questions!

Take these techniques and go exploring with your own data sets.

Section 3

VARIANCE AND STANDARD DEVIATION

It is said by some that standard deviation and variance are tedious to calculate by hand, I would agree with that but it is likely you will never ever do any of this by hand. That was more likely the stats of years gone by. But, in R you only have to know two commands to achieve standard deviation and variance, sd() and var(). Bam we are done! Okay one more, in SQL Server the commands are stdev and var! Bam, we are done!

Fine! Here is my frustration with all of this, exactly what is it? My intro to stats class took 50 slides for this part and only two of them made sense, the two with words, the other 48 were an x,y grid, not helpful at all.

Variance is the expected value of the squared deviation of a random variable for its mean. Looking at the R formula is easier. Variance = sum((x – mean(x)) ^2) / (length(x)-1). Work your way from the inside of the formula out of you need to.

Standard deviation is a measure of the variation of dispersion of the data. This has a nice easy formula as well, and it is based on variance. Standard Deviation = sqrt(sum((x – mean(x)) ^2) / (length(x)-1)). It's the square root of the variance, how cool is that, you only need to know one formula!

Just to get started, here is a short and simple demo of both formulas and the R function;

```
#Load up a vector with some numbers

x<- c(1,2,3,5,8)


#This is the long hand of variance

#hopefully, the formula will produce the same results as var()
```

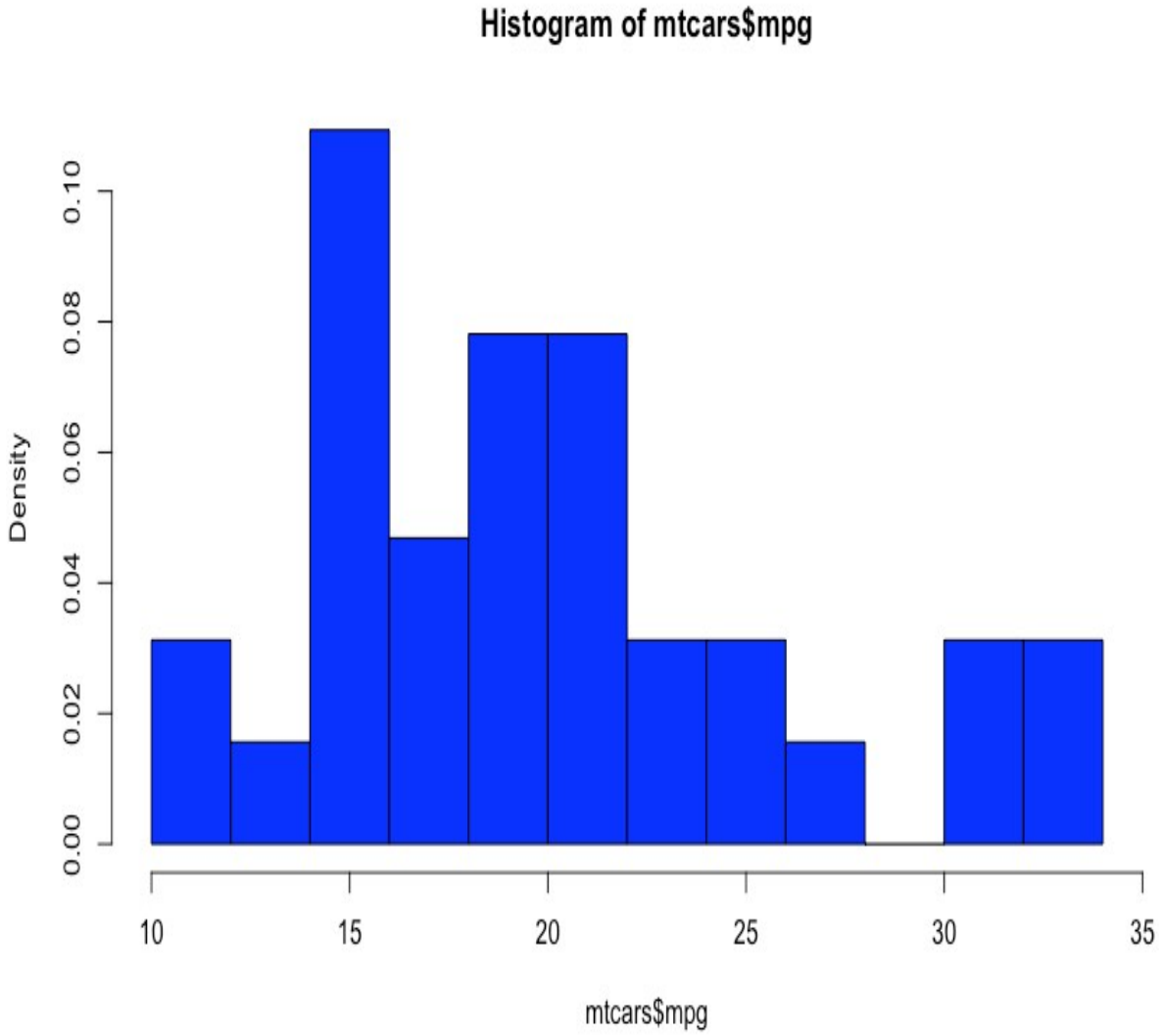
```
    sum((x - mean(x)) ^2 ) / (length(x)-1)


    var(x)


    #This is the long hand of standard deviation
    #Notica that it is the square root of variance formula
    #hopefully, the formula will produce the same results as var()
    sqrt(sum((x - mean(x)) ^2 ) / (length(x)-1))


    #you could also just use the square root of the output of var()
    sqrt(var(x))


    #or just run sd()
    sd(x)
```

In the long hand formula, did you notice we are taking the length of x and subtracting 1 for both sd and var? Do you know why? Blame Fred Bessel, he died in 1846 so i don't think there is much chance of getting around this. Its Bessels' correction, it is unique to a sample except that all statistical software will use the correction by default even if you are using an entire population. It is stated that this is used when the population mean is un-known. Though you will notice regardless if you or i know the population it is computed as if we do not, get used to it, n-1 is built into every software, the de facto standard.

I am going to cover normal distribution in the upcoming section, but lets make a hot mess of the mtcars$mpg data first.

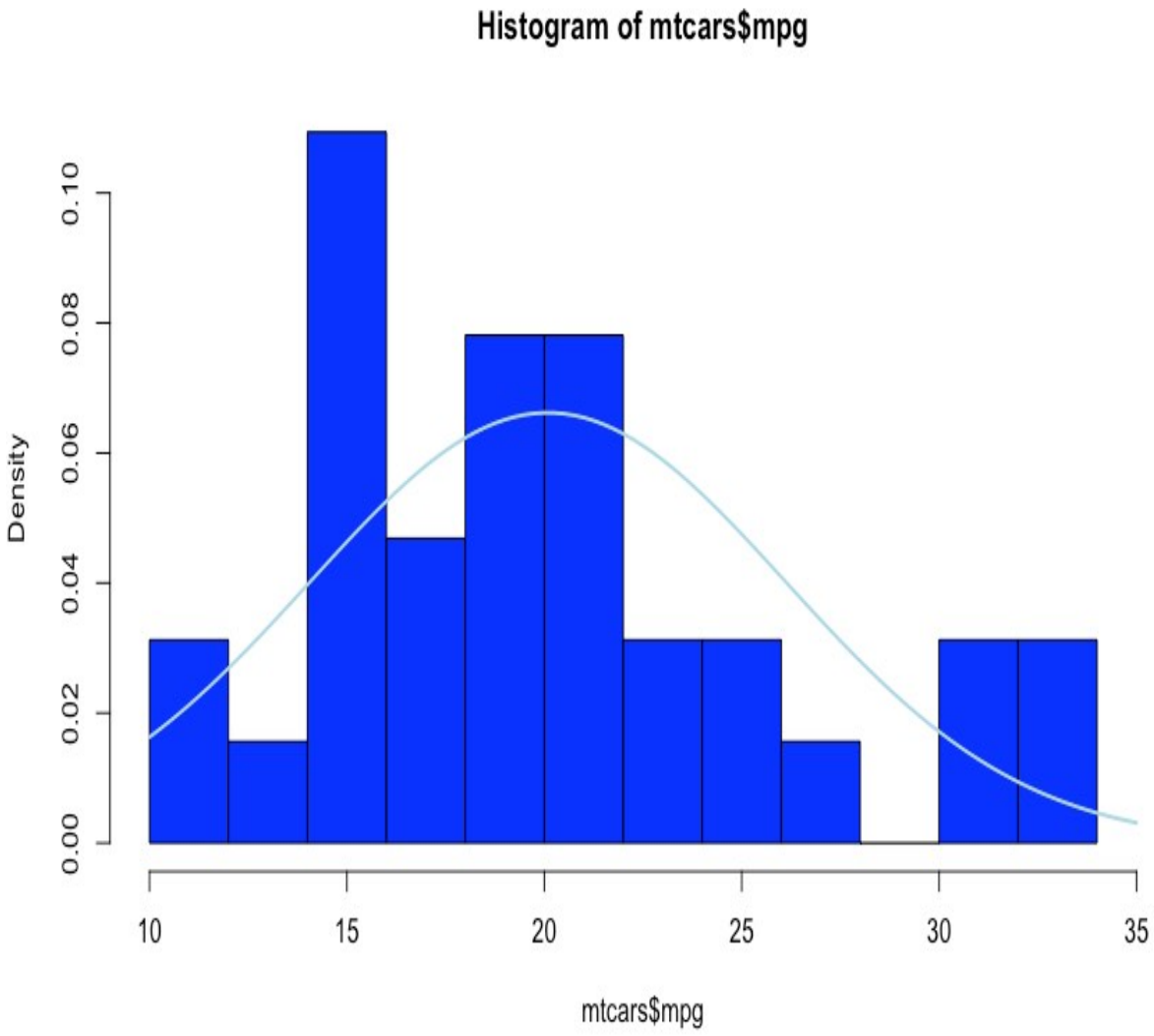
Lets get a histogram first;

```
    hist(mtcars$mpg,col="blue",breaks=15,freq=FALSE
,xlim=c(10,35))
```



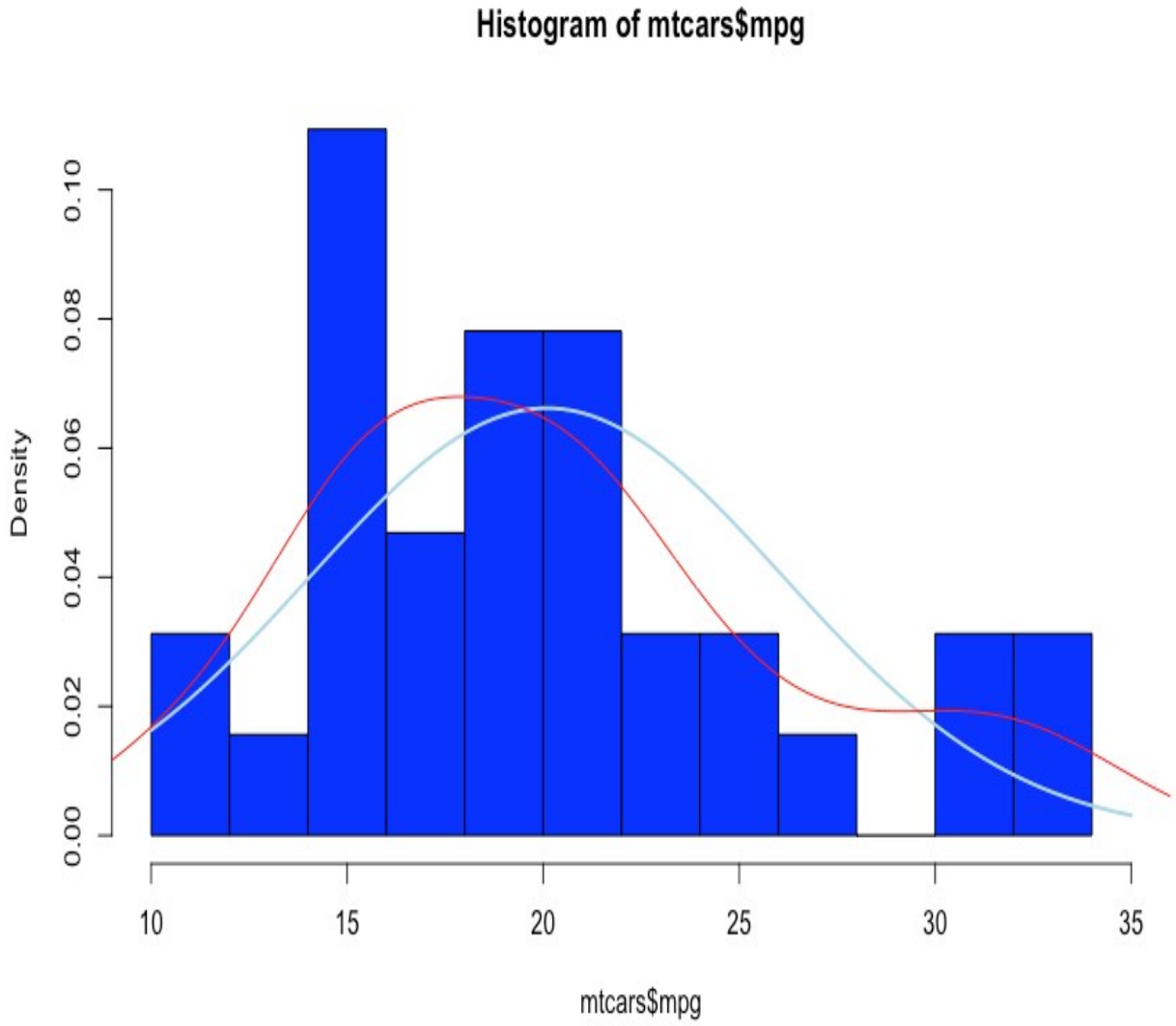
Brace yourself, i am going to use as few effective words as possible to describe what curve() and dnorm() are. Looking at at the light blue line below; what we have now is a PDF, probability density function line. What this means is, since the norm took in the standard deviation and the mean it creates a density line to try and predict where a new piece of incoming data is likely to fall.

```
    curve(dnorm(x, mean=mean(mtcars$mpg), sd=sd(mt-
cars$mpg)), col="lightblue",add=TRUE, lwd=2)
```



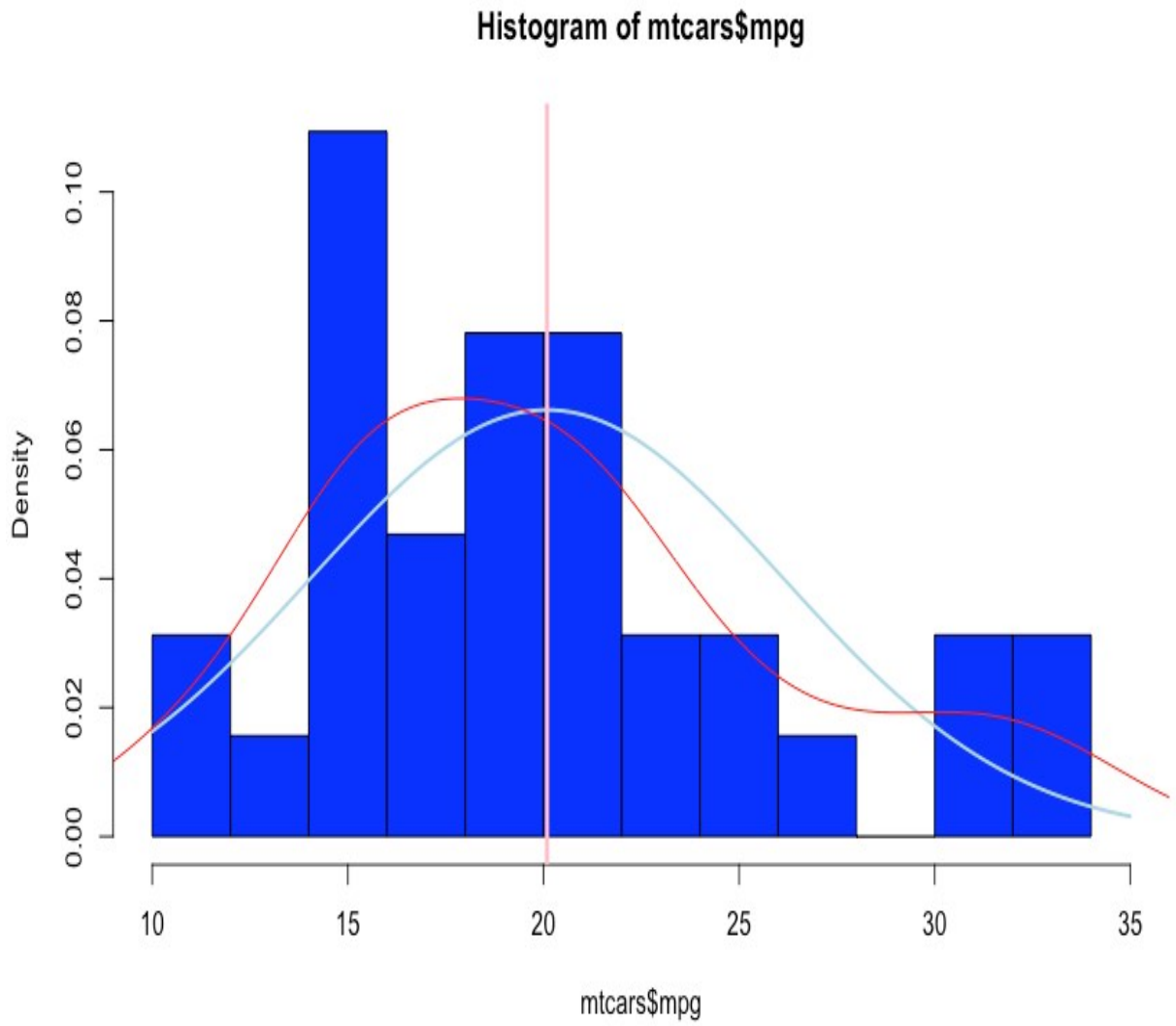

```
lines(density(mtcars$mpg),col="red")
```

Looking at the red line, this is a kernel density estimate plot layer over the histogram, this basically smoothes the data based on the sample provided. We may get deeper into this way later, as this is pretty advanced. But notice how it models the underlying data.



Lets add a mean line in pink.

```
        abline(v = mean(mtcars$mpg),col = "pink",lwd
= 3)
```



The pink line is the mean and you will recall is the average of the values in mtcars$mpg. Notice that as screwy as the data actual is the mean is still relatively centered.

Section 4

CENTRAL TENDENCY

We have determined that quantitative data is essentially numeric data, or "measuring data". Quantitative data asks how much. Knowing that, we can start to look at statistical techniques to analyze this data. To do this we will need to get some definitions out of the way, and some demonstrations to help figure out what these things are. None of them are derived from magic, though sometimes they certainly appear that way.

There is a thing called central tendency, or the measure of central tendency. It is not very hard to derive the definition from the words, it is the tendency of the things to be centered, or the center or location of the distribution of data. For the most part, as the dataset we are using increases in size, it tends to have much of the data centered in a specific location. We measure this by using mean, median, and mode. To be clear, Mean, Mode, Median are all measurers of central tendency.

Central Tendency will be one of the very important foundations of prediction, it's a principle that assumes all of the data will be within a certain distribution vs data all over the place. If you were to use a histogram with many bins, the data would be mound shaped. That mound means that we can probably predict other new data based on certain factors, those factors will come later.

Mean – the mean is the average, sum the data and divide by the number of values.

```
(1+2+3+4+5+6+6+7) / 8
#or
```

```
mean(c(1,2,3,4,5,6,6,7))
```

```
> (1+2+3+4+5+6+6+7) / 8
[1] 4.25
>
> mean(c(1,2,3,4,5,6,6,7))
[1] 4.25
```

Median is the middle number in the data set, using (1,2,3,4,5,6,6,7) we have eight numbers in the dataset, an even number, so we take the two middle numbers add them and divide by two, in this case the Median is 4.5. In R you can run median(c(1,2,3,4,5,6,6,7)). If we had an odd number of values in the dataset as in (1,1,1,2,9,9,9), "2" is the median. It is just the middle number.

```
median(c(1,2,3,4,5,6,6,7))

median(c(1,2,3,4,5,6,7))

median(c(1,1,1,2,9,9,9))
```

```
> median(c(1,2,3,4,5,6,6,7))
[1] 4.5
> median(c(1,2,3,4,5,6,7))
[1] 4
> median(c(1,1,1,2,9,9,9))
[1] 2
```

Mode is the number that shows up most often in the dataset. Mode is a little tricky, there is no built in function for Mode in base R. Stack overflow has a nice mode reference here, which demonstrates the following

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
Mode(c(1,1,1,2,9,9))
```

```
> Mode <- function(x) {
+   ux <- unique(x)
+   ux[which.max(tabulate(match(x, ux)))]
+ }
> Mode(c(1,1,1,2,9,9))
[1] 1
```

If you are use to T-SQL that last thing may have tripped you up a bit, a function being stored in a variable. Its called a “Call by Reference”, this demonstrates some of the power of R.

Its really not very hard, and we will be using central tendency a lot. try it out with some of the datasets. Try this out on a few datasets that ship with base R, or your favorite dataset, graph it using hist() see if you can spot some of the tendencies and do they match what you would expect using just mean, median, and mode. Otherwise run the code below line by line and make sure you understand what is going on.

```
?cars
data(cars)
View(cars)

mean(cars$dist)
```

```
median(cars$dist)
Mode(cars$dist) #using the above Mode function


data(mtcars)
View(mtcars)


mean(mtcars$mpg)
median(mtcars$mpg)
Mode(mtcars$mpg)
```

Somewhere between this topic and the next topic there lives a thing called percentiles and quartiles.

Percentile - In statitistics a percentile is the measure indicating the value below which observations in a group fall. Yeah right, lets use an example. When occupy Wallstreet was all the rage they frequently help signs of 99%, that meant that you and i are the 99 percentile of income. Hence they were attempting to annoy anyone in the US who fell into the 1 percentile. According to this article from investopedia, you would need to make more than $456,626 AGI to be in the one percent. From that we can determine that anyone who makes less than $456,626 AGI is in the 99 percentile. To be in the 95 percentile you need to make less than $214,462 AGI per year. So, percentile is a measure of location. Where am i? Where are you?

Quartiles - In descriptive statistics, the quartiles of a ranked set of data values are the three points that divide the data set into four equal groups, each group comprising a quarter of the data. Clear as mud, well four equal groups makes sense. They are, 25%, 50%, 75%, these divide the data into four groups.

Lets look at some data. R has a base function summary(), summary is your friend. Mtcars is should be loaded already, if it is not you are a wizard at get-

ting in memory by now. But, since it is part of base R, you may not need to load it at all, its secret R voodoo i am not going to get into.

```
summary(mtcars$mpg)
```

```
> summary(mtcars$mpg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

Min is the minimum value of the dataset,
1st quartile or 25 percentile is 15.42
2nd quartile or 50 percentile or **median** or **average** is 19.20
3rd quartile or 75 percentile is 22.80
Mean or average is 20.09
Maximum value for the dataset is 33.90

Lets use a few more words to describe whats going on above. IF the mpg of your car is 12, like my Jeep, you are in the 25th percentile. If you get 22mpg, you are in the 75th percentile and above the median (19.20) and above the mean(20.09). Why this matters will come up later, but be cognizant of where a datum falls.

You can run summary against an entire dataset as well vs just a column/variable summary(mtcars) for instance. This will provide statistics for all columns/variables in the dataset.

Section 5

NORMAL DISTRIBUTION AND IMPERICAL RULE

So, we have covered standard deviation and mean, discussed central tendency, and we have demonstrated some histograms. You are familiar with what a histogram looks like and that depending on the data, it can take many shapes. This section will discuss distribution that specifically applies to mound shaped data. We happen to have a couple of datasets that meet this criteria perfectly, or at least it does in shape. Lets load them and see what we find.

We will use datasets from US Educational attainment to demonstrate this. They appear to be mound shaped, mound being the key word. If it is mound shaped, we should be able to make some predictions about the data using the **Empirical Rule**, and if not mound shape, the perhaps Chebyshevs rule, more on that later.

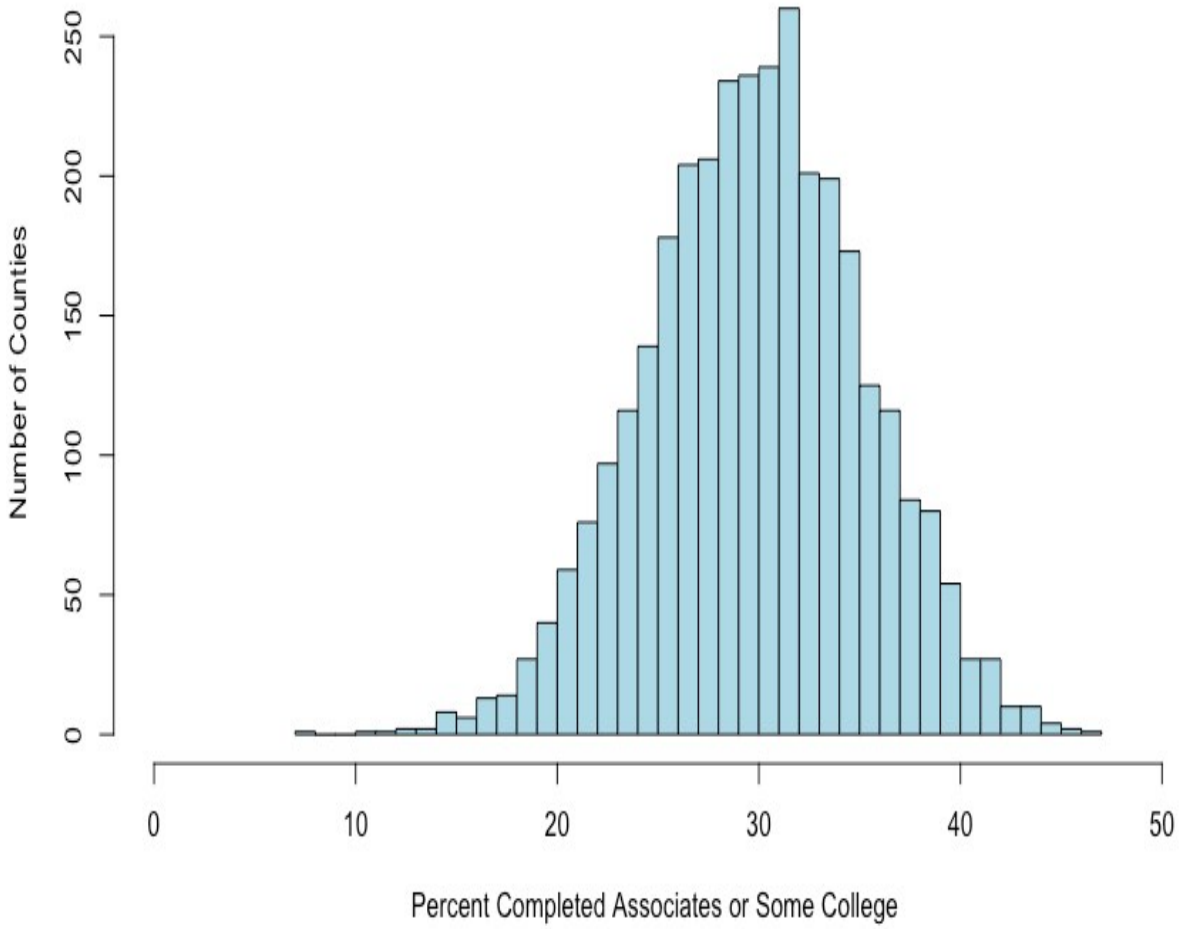
The point of this is to link visualization of distributions to numerical measures of center and location. This will only apply to mound shaped data, like the following;

The following four lines of code will;

- Load the csv file into a dataframe called “usa”.
- create a new dataframe with only the two columns we are interested in.
- Rename the columns to make just a tiny bit friendlier to write and deal with.
- Display a histogram with the new dataframe.

```
usa <- read.csv("https://raw.githubusercontent.com/sqlshep/DS4New/master/Chapter%204/data/US-Education.csv")
```

```
    someCollege <- subset(usa[c("FIPS.Code","Per-
cent.of.adults.completing.some.college.or.associ-
ate.s.degree..2010.2014")],FIPS.Code >0)


    names(someCollege) <-c("FIPS","SomeCollege")


    hist(someCollege$SomeCollege,xlim=c(0,50),break
s=30, xlab = "Percent Completed Associates or Some
College ", ylab="Number of
Counties",main="",col="lightblue")
```



When someone says mound shaped data, this is the text book example of mound shaped.

Just so you fully understand wha this data is, every person in the US reports their level of educational attainment to the Census every ten years, every few years this data is updated and projected to estimate reasonably current values. This we will be using is for the 2010-2014 years which is the five year average compiled by the American Community Survey. I highly encourage use of this website for test data, all of it has to be manipulated a little bit, but it typically takes minutes to get it into a format R can use.

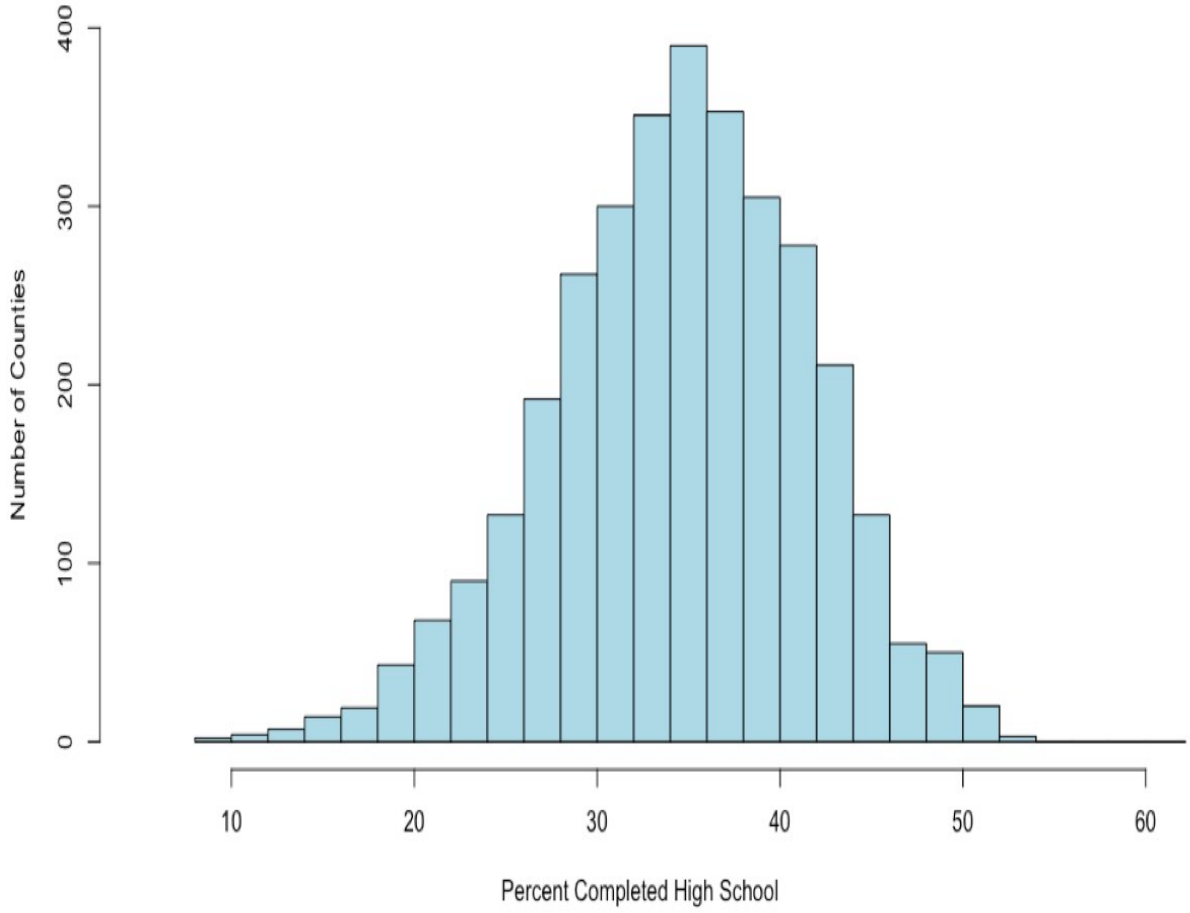
The **Empirical rule** states that;

> 68% of the data will fall with in 1 standard deviation of the mean,
> 95% of the data will fall within 2 standard deviations of the mean, and
> 99.7% of the data will fall within 3 standard deviations of them mean.

Lets find out!

```
    #While not required, i want to isolate the data
we will be working with

    highSchool <- subset(usa[c("FIPS.Code","Per-
cent.of.adults.with.a.high.school.diploma.only..20
10.2014")],FIPS.Code >0)


    #reanme the second column to something less
annoying

    colnames(highSchool)[which(colnames(highSchool)
== 'Percent.of.adults.with.a.high.school.diplo-
ma.only..2010.2014')] <- 'percent'


    #Display a histogram

    hist(highSchool$percent
```

```
,xlim=c(5,60)
,breaks=20
,xlab = "Percent Completed High School "
,ylab = "Number of Counties"
,main = ""
,col = "lightblue")
```




```
#create a variable with the mean and the standard devaiation
hsMean <- mean(highSchool$percent,na.rm=TRUE)
```

```
hsSD <- sd(highSchool$percent,na.rm=TRUE)


#one standard deviation from the mean will "mean" one SD

#to the left (-) of the mean and one SD to the right(+) of the
mean.

#lets calculate and store them


oneSDleftRange <- (hsMean - hsSD)

oneSDrightRange <- (hsMean + hsSD)


oneSDleftRange;oneSDrightRange


##[1] 27.51472 is one sd to the left of the mean

##[1] 41.60826 is one sd to the right of the mean


#lets calculate the number of rows that fall

#between 27.51472(oneSDleftRange) and
41.60826(oneSDrightRange)

oneSDrows <- nrow(subset(highSchool,percent > oneSDleftRange &
percent < oneSDrightRange))


# whats the percentage?

oneSDrows / nrow(highSchool)
```

If everything worked properly, you should have seen that the percentage of counties within one standard deviation of the mean is "0.6803778" or 68.04%. Wel that was kinda creepy wasn't it? The empirical rule states that 68% of the data will be within one standard deviation.

Lets keep going.

```
#two standard deviations from the mean will "mean" two SDs
```

```
     #to the left (-) of the mean and two SDs to the right(+) of
the mean.

     twoSDleftRange <- (hsMean - hsSD*2)

     twoSDrightRange <- (hsMean + hsSD*2)



     twoSDleftRange;twoSDrightRange


     ##[1] 20.46795 is two sds to the left of the mean

     ##[1] 48.65503 is two sds to the right of the mean


     twoSDrows <- nrow(subset(highSchool,percent > twoSDleftRange &
percent < twoSDrightRange))


     twoSDrows / nrow(highSchool)
```

If your math is the same as my math, you should have gotten 95.09%, so far the empirical rule is holding...

What about three standard deviations?

```
     threeSDleftRange <- (hsMean - hsSD*3)

     threeSDrightRange <- (hsMean + hsSD*3)

     threeSDleftRange;threeSDrightRange


     threeSDrows <- nrow(subset(highSchool,percent > threeSDleft-
Range & percent < threeSDrightRange))


     threeSDrows / nrow(highSchool)
```

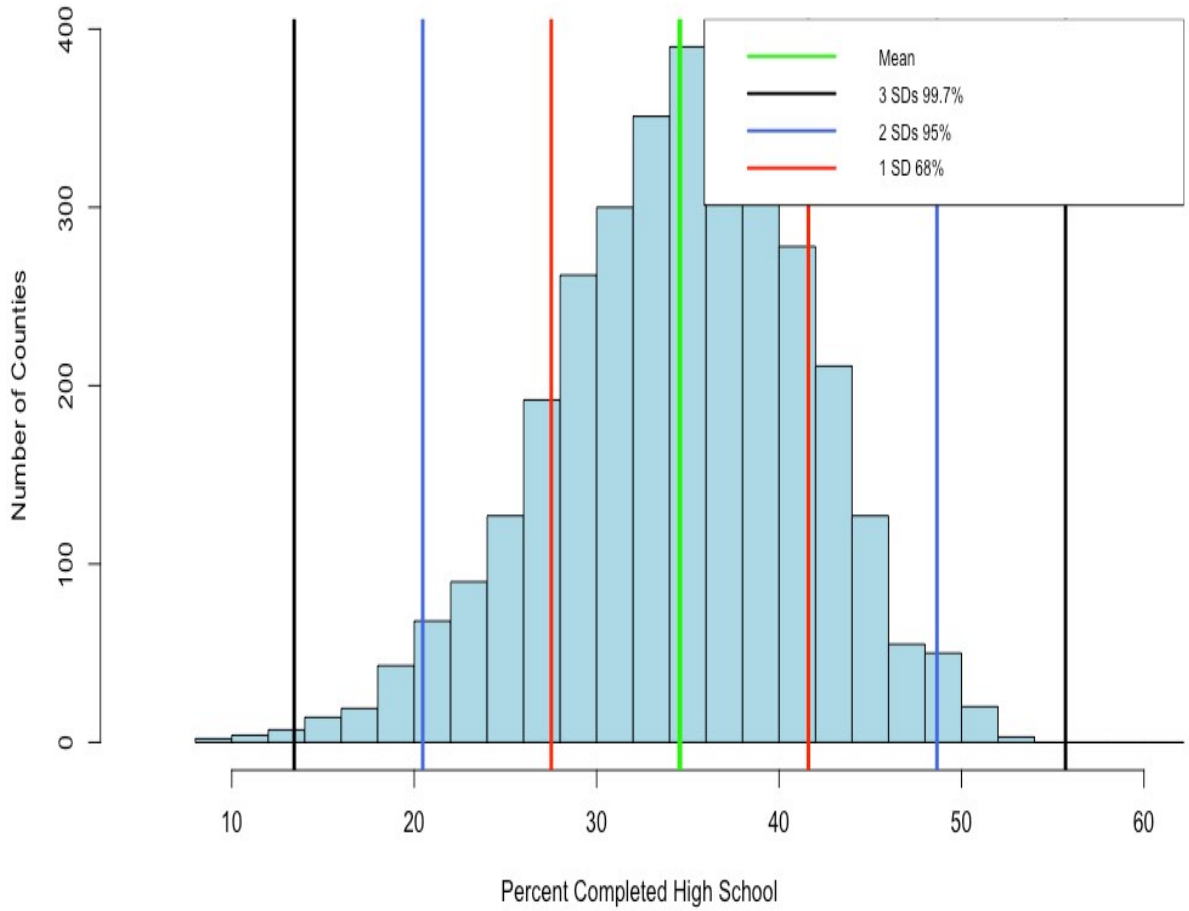
99.32% at three standard deviations, its like the empirical rule knows our data! Before we move on, lets add some lines...

For the following code to work you will have had to run the prior code in this section as it uses that variables created.

```
hist(highSchool$percent
     ,xlim=c(5,60)
     ,breaks=20
     ,xlab = "Percent Completed High School "
     ,ylab = "Number of Counties"
     ,main = ""
     ,col = "lightblue")

abline(v = threeSDleftRange,col = "black",lwd = 3)
abline(v = threeSDrightRange,col = "black",lwd = 3)

abline(v = twoSDleftRange,col = "royalblue",lwd = 3)
abline(v = twoSDrightRange,col = "royalblue",lwd = 3)

abline(v = oneSDleftRange,col = "red",lwd = 3)
abline(v = oneSDrightRange,col = "red",lwd = 3)

abline(v = hsMean,col = "green",lwd = 4)

legend(x="topright",
       c("Mean","3 SDs 99.7%", "2 SDs 95%", "1 SD 68%"),
       col = c("Green","black", "royalblue", "red"),
       lwd = c(2, 2, 2),
```

```
cex=0.75
)
```

You can see the distribution of the data below, it really does seem to fall into pretty predictable standard deviations.



Section 6

CHEBYSHEVS RULE

Picking up from the last section we will now look at **Chebyshevs rule**. For this one we will be using ggplot2 histogram with annotations just to shake things up a bit.

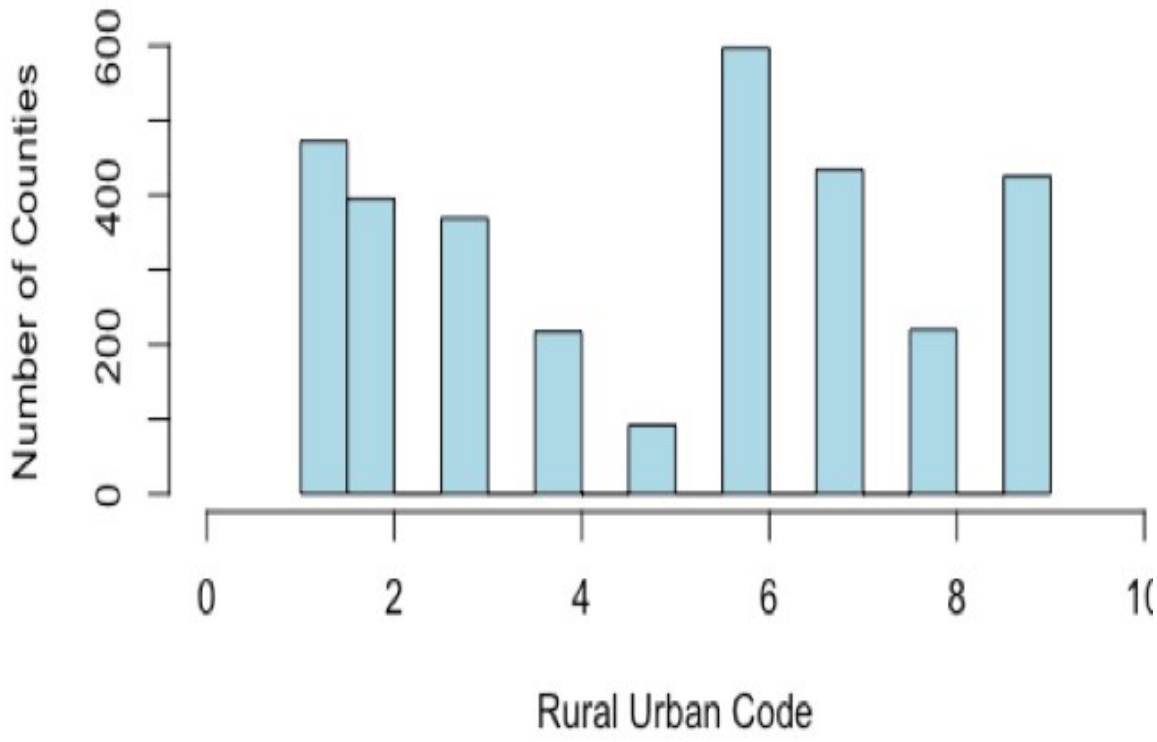
Chebyshevs rule, theorem, inequality, whatever you want to call it states that all possible datasets regardless of shape will have 75% of the data within 2 standard deviations, and 88.89% within 3 standard deviations. This should apply to mound shaped datasets as well as bimodal (two mounds) and multimodal (many mounds).

The **Chebyshevs rule** states that;

75% of the data will fall within 2 standard deviations of the mean, and 88.89% of the data will fall within 3 standard deviations of them mean.

```
usa <-
read.csv("https://raw.githubusercontent.com/sqlshep/DS4New/master/Chapter%204/data/US-Education.csv")


hist(usa$X2013.Rural.urban.Continuum.Code
     ,xlim=c(0,10)
     ,xlab = "Rural Urban Code"
     ,ylab = "Number of Counties"
     ,main = ""
     ,col = "lightblue")
```

Comparatively speaking, this one looks a little funky, this is certainly bimodal, if not nearly trimodal. This should be a good test for Chebyshev.



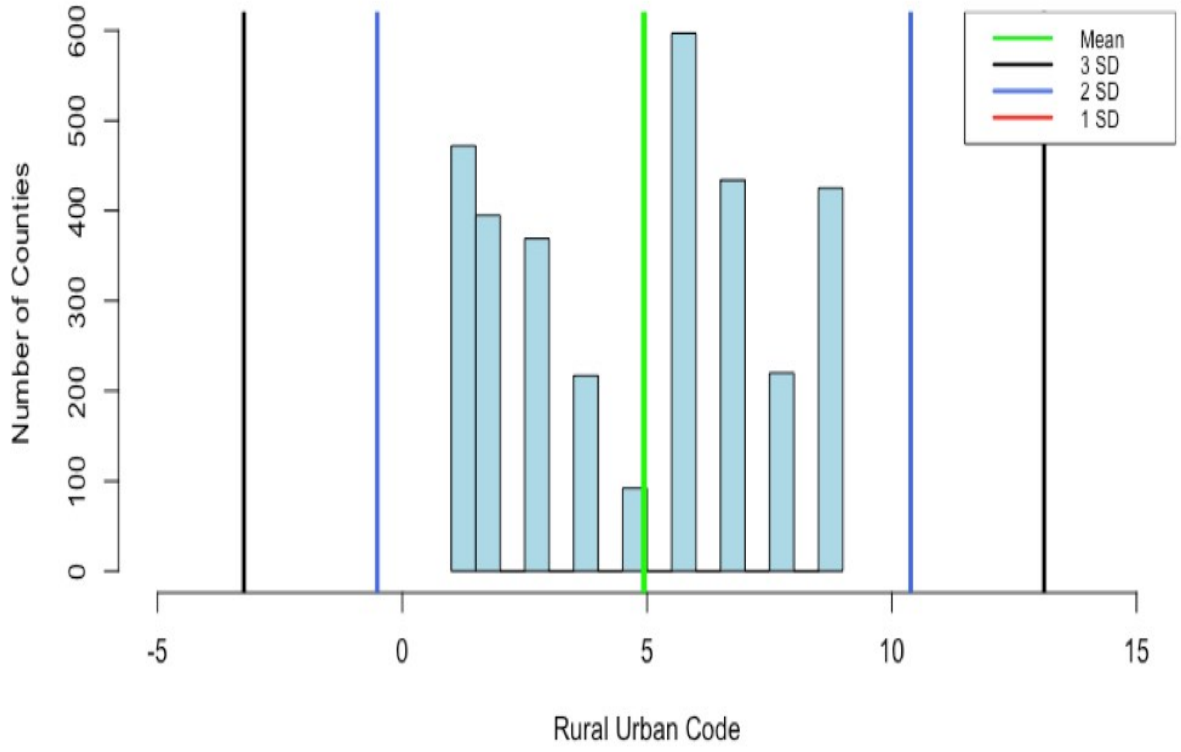
So, lets reuse some of the code above, drop the first standard deviation since Chebyshev does not need it and see if we can get this to work with "X2013.Rural.urban.Continuum.Code"

```
urbanMean <- mean(usa$X2013.Rural.urban.Continuum.Code,na.rm=TRUE)

urbanSD <- sd(usa$X2013.Rural.urban.Continuum.Code,na.rm=TRUE)



#two standard deviations from the mean will "mean" two SDs

#to the left (-) of the mean and two SDs to the right(+) of the
mean.

twoSDleftRange <- (urbanMean - urbanSD*2)

twoSDrightRange <- (urbanMean + urbanSD*2)
```

```
     twoSDleftRange;twoSDrightRange


     twoSDrows <- nrow(subset(usa,X2013.Rural.urban.Continuum.Code >
twoSDleftRange & usa$X2013.Rural.urban.Continuum.Code <
twoSDrightRange))

     twoSDrows / nrow(usa)


     #two standard deviations from the mean will "mean" two SDs

     #to the left (-) of the mean and two SDs to the right(+) of the
mean.

     threeSDleftRange <- (urbanMean - urbanSD*3)

     threeSDrightRange <- (urbanMean + urbanSD*3)

     threeSDleftRange;threeSDrightRange


     threeSDrows <- nrow(subset(usa,X2013.Rural.urban.Continuum.Code >
threeSDleftRange & X2013.Rural.urban.Continuum.Code <
threeSDrightRange))

     threeSDrows / nrow(usa)


hist(usa$X2013.Rural.urban.Continuum.Code

          ,xlim=c(-5,15)

          ,xlab = "Rural Urban Code"

          ,ylab = "Number of Counties"

          ,main = ""

          ,col = "lightblue")


     abline(v = threeSDleftRange,col = "black",lwd = 3)

     abline(v = threeSDrightRange,col = "black",lwd = 3)


     abline(v = twoSDleftRange,col = "royalblue",lwd = 3)

     abline(v = twoSDrightRange,col = "royalblue",lwd = 3)
```

```
abline(v = urbanMean,col = "green",lwd = 4)



legend(x="topright",
       c("Mean","3 SDs 99.7%", "2 SDs 95%", "1 SD 68%"),
       col = c("Green","black", "royalblue", "red"),
       lwd = c(2, 2, 2),
       cex=0.75
)
```

If you look at the range of two standard deviations above, you should know we have a problem; 98% of the data fell within 2 standard deviations. While yes, 68% of the data is also in the range it turns out this is a terrible example. The reason i include it is because it is just as important to see a test result that fails your expectation as it is for you to see on one that fits! You will notice that the 3rd standard deviations is far outside the data range. Notice i had to extend the x axis so the 2nd and 3rd standard deviation would even show up.



SO, what do we do? fake data to the rescue!

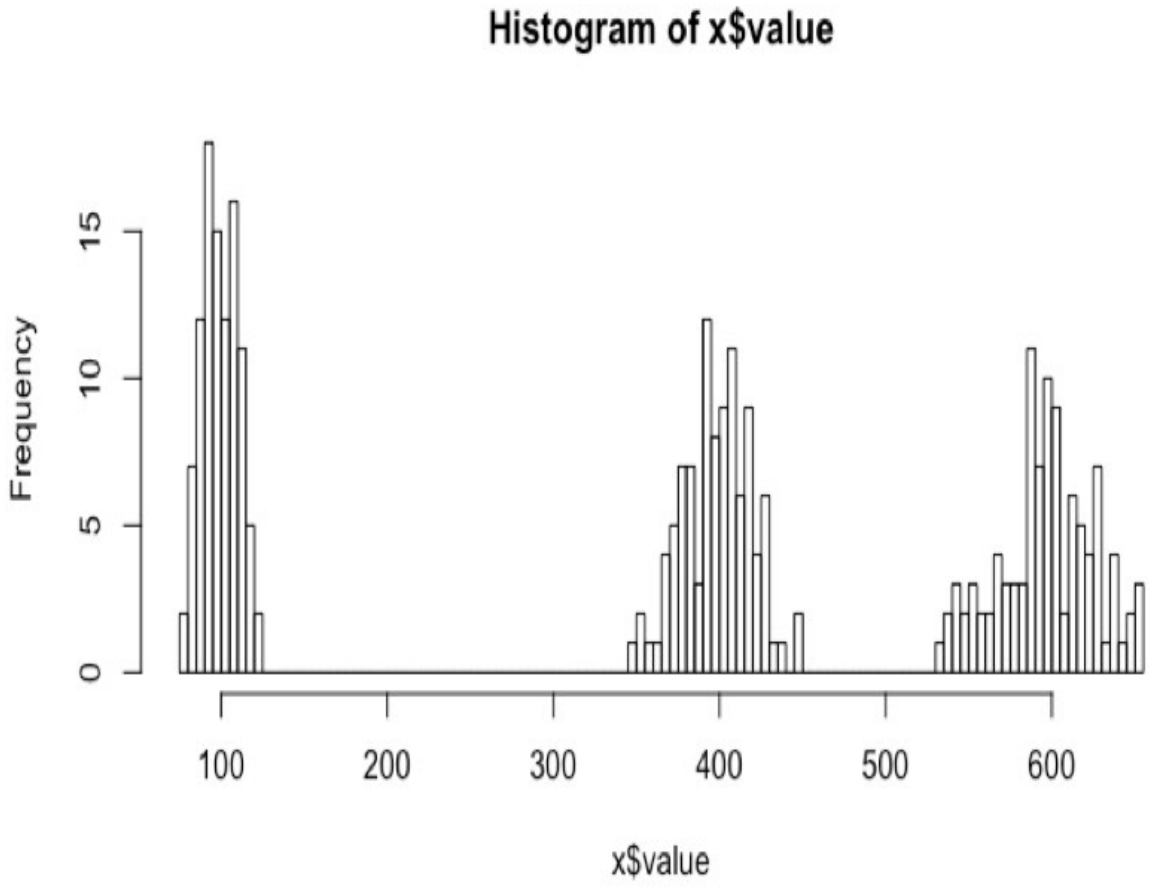
I try really hard to avoid using made up data because to me it makes no sense, where as car data, education data, population data, that all makes sense. But, there is no getting around it! Here is what you need to know, rnorm() generates random data based on a normal distribution using the variables standard deviation and a mean! But wait, we are trying to get multi-modal distribution. Then concatenate more than one normal distribution, eh? Lets try three.

We are going to test for one standard deviation just to see what it is, even though Chebyshevs rule has no interest in it, remember the rule states that 75% the data will fall within 2 standard deviations.

```
#set.seed() will make sure the random number generation is not random
every time

      set.seed(500)

      x <- as.data.frame(c(rnorm(100,100,10)
```

```
,(rnorm(100,400,20))
,(rnorm(100,600,30))))

colnames(x) <- c("value")
hist(x$value,nclass=100)
```

To be fair, if your data in the real word looks like this you may have an opportunity to classify it by what ever is causing it to be this far apart. But, this is for demonstration purposes only.




```
sd(x$value)

    mean(x$value)
```

```
> sd(x$value)
[1] 205.8439
>
> mean(x$value)
[1] 364.6718
```

```
#if you are interested in looking at just the first few values

head(x)


xMean <- mean(x$value)

xSD <- sd(x$value)


#one standard deviation from the mean will "mean" 1 * SD

#to the left (-) of the mean and one SD to the right(+) of the
mean.

oneSDleftRange <- (xMean - xSD)

oneSDrightRange <- (xMean + xSD)

oneSDleftRange;oneSDrightRange

oneSDrows <- nrow(subset(x,value > oneSDleftRange & x <
oneSDrightRange))

print("Data within One standard deviations");oneSDrows / nrow(x)
```

```
> oneSDleftRange <- (xMean - xSD)
> oneSDrightRange <- (xMean + xSD)
> oneSDleftRange;oneSDrightRange
[1] 158.828
[1] 570.5157
> oneSDrows <- nrow(subset(x,value > oneSDleftRange & x < oneSDrightRange))
> print("Data within One standard deviations");oneSDrows / nrow(x)
[1] "Data within One standard deviations"
[1] 0.3966667
```

```
#two standard deviations from the mean will "mean" 2 * SD
```

```
     #to the left (-) of the mean and two SDs to the right(+) of the
mean.

     twoSDleftRange <- (xMean - xSD*2)

     twoSDrightRange <- (xMean + xSD*2)

     twoSDleftRange;twoSDrightRange

     twoSDrows <- nrow(subset(x,value > twoSDleftRange & x$value <
twoSDrightRange))

     print("Data within Two standard deviations");twoSDrows / nrow(x)
```

```
> twoSDleftRange <- (xMean - xSD*2)
> twoSDrightRange <- (xMean + xSD*2)
> twoSDleftRange;twoSDrightRange
[1] -47.01587
[1] 776.3596
> twoSDrows <- nrow(subset(x,value > twoSDleftRange & x$value < twoSDrightRange))
> print("Data within Two standard deviations");twoSDrows / nrow(x)
[1] "Data within Two standard deviations"
[1] 1
```

```
     #three standard deviations from the mean will "mean" 3 * SD

     #to the left (-) of the mean and two SDs to the right(+) of the
mean.

     threeSDleftRange <- (xMean - xSD*3)

     threeSDrightRange <- (xMean + xSD*3)

     threeSDleftRange;threeSDrightRange

     threeSDrows <- nrow(subset(x,value > threeSDleftRange & x$value <
threeSDrightRange))

     print("Data within Three standard deviations");threeSDrows /
nrow(x)
```

```
> threeSDleftRange <- (xMean - xSD*3)
> threeSDrightRange <- (xMean + xSD*3)
> threeSDleftRange;threeSDrightRange
[1] -252.8597
[1] 982.2034
> threeSDrows <- nrow(subset(x,value > threeSDleftRange & x$value < threeSDrightRange
> print("Data within Three standard deviations");threeSDrows / nrow(x)
[1] "Data within Three standard deviations"
[1] 1
```

WOOHOO, Multimodal! Chebyshev said it works on anything, lets find out. The histogram below is a hot mess based on how the data was created, but it is clear that the empirical rule will not apply here, as the data is not mound shaped and is multimodal or trimodal.

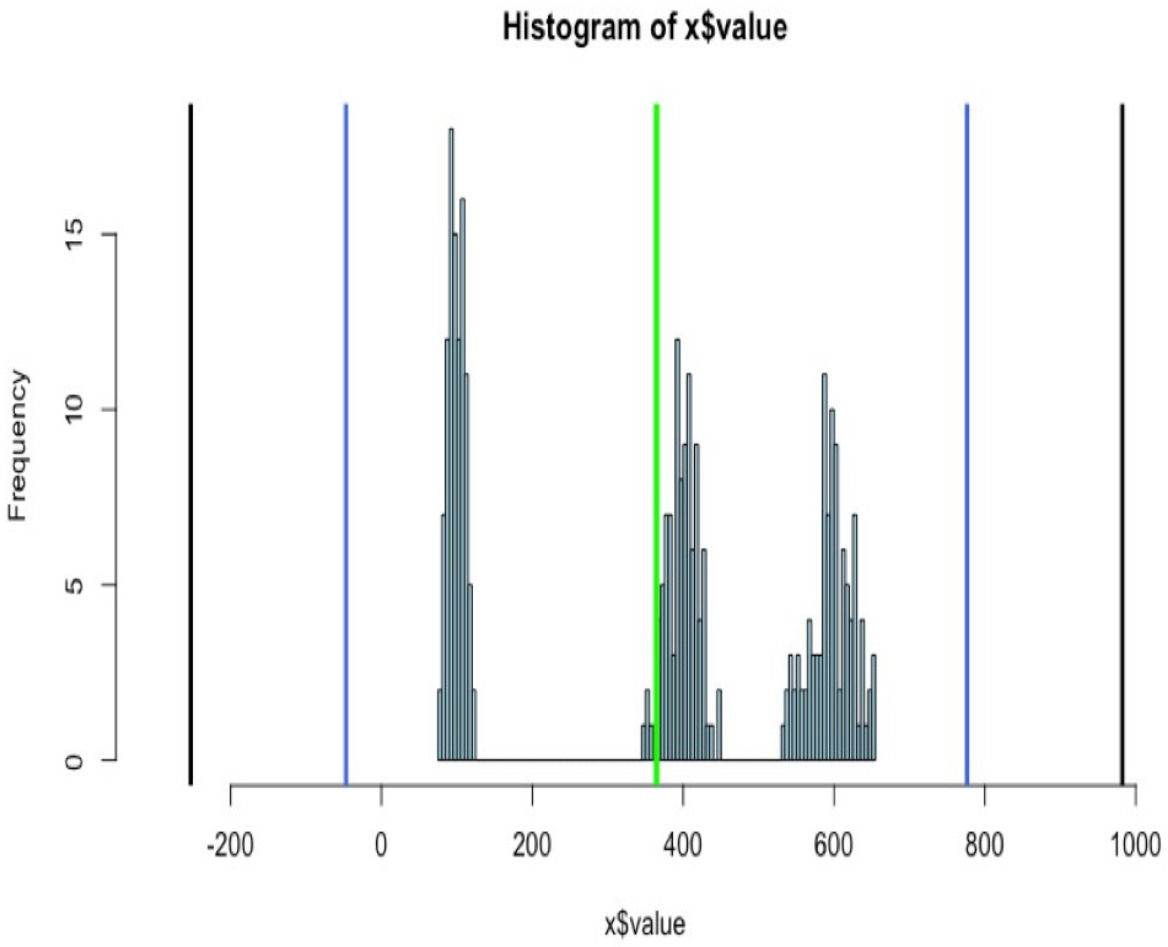
Though Chebyshevs rule has no interest in 1 standard deviation i wanted to show it just so you could see what the 1 SD looks like. I challenge you to take the rnorm and see if you can modify the mean and SD parameters passed in to make it fall outside o the 75% of two standard deviations. If you ran or even read the output above you already know that all of the data fell within 2 standard deviations.

[1] "Data within One standard deviations" = 0.3966667 # or 39.66667%
[2] "Data within Two standard deviations" = 1 # or 100%
[3] "Data within Three standard deviations" = 1 or 100%

```
hist(x$value,nclass=100,
     xlim=c(-300,1000),
     col="lightblue")



abline(v = threeSDleftRange,col = "black",lwd = 3)
abline(v = threeSDrightRange,col = "black",lwd = 3)


abline(v = twoSDleftRange,col = "royalblue",lwd = 3)
abline(v = twoSDrightRange,col = "royalblue",lwd = 3)
```

```
abline(v = xMean,col = "green",lwd = 4)
```

Notice how far to the left and right the x axis had to extend to accommodate the 2nd and 3rd standard deviation. THe point here, the wider the data the larger the sd so it will always be very hard to find data where the empirical rule or Cheby-shevs will not apply .



There you have it! It is becoming somewhat clear that based on the shape of the data and if you are using empirical or Chebyshevs rule, data falls into some very predictable patters, maybe from that we can make some predictions about new data coming in...?

Section 7

CORRELATIONS AND COLLINEARITY

All of the work we do in statistical learning is based on the fact we can predict y based on x. If i eat 10,000(x) calories a day i will be fat(y) unless i am an Olympic swimmer apparently, so it does not always hold true, but with just one dependent and one independent variable, it would appear to be an easy answer. Now if i added physical activity to the mix, fat or not fat might be more accurate. Every now and then you will hear about a "study" that some new claim is made from, and the world falls apart for a few days talking about nothing else. My recent favorite news story is diet soda makes you fat, gives you cardiovascular disease, hypertension, metabolic syndrome and type II diabetes. Whether you believe that or not, and for the sake of argument the article does not mention level of activity per day, calories of food consumed per day, you know, lots of other stuff that could contribute. The study appears to make the claim that diet soda all by itself will cause all of these health problems. Peter Attia has started to write about the problems with these studies and the problems with them.

In short, an observational study means you are just watching and asking questions, not influencing, no treatment, no control. An observational study cannot be used to determine cause, you have surely heard correlation does not imply causation, this is never more true than in observational studies, and even more so for retrospective observational study, where you basically look at data from a prior observational study. You can find stuff, but you cannot conclude cause. Don't get me wrong they are needed an necessary for research to move forward, but causation cannot be determined from this.

Which brings me to these fun little anomalies

Spurious correlations are a hoot and a half. Spurious correlations is a mathematical relationship in which two or more events or variables are not causally related to each other, yet it may be wrongly inferred that they are. I can assure you that Nicolas Cage movies and pool drownings actually have nothing to do with each other but they do correlate.

I will not copy the website below, but take a look at some of the stuff that correlates;

http://www.tylervigen.com/spurious-correlations

Confounding and/or **Lurking** variable variable; i see these used interchangeably, so it is a bit difficult to separate them. That being said, it is a variable that correlates to the dependent and independent variable thus influencing the significance, if one is detected, it should be removed. You will typically see this by two variables that always trend together though they actually have no direct impact n each other, they just happen to trend together. There is a great alternative example here, just the top part of the page.

Collinearity or **Multicollinearity** ; I'm totally copying the definition from PSU, they have great stuff btw; "when it exists, it can wreak havoc on our analysis and thereby limit the research conclusions we can draw", "multicollinearity exists whenever two or more of the predictors in a regression model are moderately or highly correlated" In R you have to go slightly out of your way to determine collinearity using VIF, variation inflation factor, i may snag a highly collinear dataset in the next few posts to demonstrate it using the vif function against the model in R.

Who cares?

In this book i will be using data sets and methods where all of these factors need to be taken into consideration, I am going to uses election data form the last election and a variety of statistics form the counties to see if it is possible to determine how a count voted based on some demographics of that county. For instance, if a county has a high unemployment rate, will that county swing democrat or republican? Or is this a confounding relationship? What about the number of high shcool dropouts in a county, will this influence presidential selection?

If we happen to come up with a high correlation or even a prediction it does not necessarily mean there is causation.

Section 8

I'M SPURIOUSLY CONFOUNDED

Before i can move on i need to cover some tough problems for statistics and more specifically, regression.

All of the work we do in statistical learning is based on the fact we can predict y based on x. If i eat 10,000(x) calories a day i will be fat(y) unless i am an Olympic swimmer apparently, so it does not always hold true, but with just one dependent and one independent variable, it would appear to be an easy answer. Now if i added physical activity to the mix, fat or not fat might be more accurate. Every now and then you will hear about a "study" that some new claim is made from, and the world falls apart for a few days talking about nothing else. My most recent favorite post is diet soda makes you fat, gives you cardiovascular disease, hypertension, metabolic syndrome and type II diabetes. Whether you believe that or not, and for the sake of argument the article does not mention level of activity per day, calories of food consumed per day, you know, lots of other stuff that could contribute. The study appears to make the claim that diet soda all by itself will cause all of these health problems. Peter Attia has started to write about the problems with these studies and the problems with them.

In short, an observational study means you are just watching and asking questions, not influencing, no treatment, no control. An observational study cannot be used to determine cause, you have surely heard correlation does not imply causation, this is never more true than in observational studies, and even more so for retrospective observational study, where you basically look at data from a prior observational study. You can find stuff, but you cannot conclude cause. Don't get me wrong they are needed an necessary for research to move forward, but causation cannot be determined from this.

Which brings me to these fun little anomalies

Spurious correlations are a hoot and a half. Spurious correlations is a mathematical relationship in which two or more events or variables are not

causally related to each other, yet it may be wrongly inferred that they are. I can assure you that Nicolas Cage movies and pool drownings actually have nothing to do with each other but they do correlate.

I will not copy the website below, but take a look at some of the stuff that correlates;

http://www.tylervigen.com/spurious-correlations

Confounding and/or Lurking variable variable; i see these used interchangeably, so it is a bit difficult to separate them. That being said, it is a variable that correlates to the dependent and independent variable thus influencing the significance, if one is detected, it should be removed. You will typically see this by two variables that always trend together though they actually have no direct impact n each other, they just happen to trend together. There is a great alternative example here, just the top part of the page.

Collinearity or Multicollinearity ; I'm totally copying the definition from PSU, they have great stuff btw; "when it exists, it can wreak havoc on our analysis and thereby limit the research conclusions we can draw", "multicollinearity exists whenever two or more of the predictors in a regression model are moderately or highly correlated" In R you have to go slightly out of your way to determine collinearity using VIF, variation inflation factor.

Who cares?

Some of data sets i will be using all of these factors need to be taken into consideration, later in the book we are going to use election data from a prior election and a variety of statistics form the counties to see if it is possible to determine how a count voted based on some demographics of that county. For instance, if a county has a high unemployment rate, will that county swing democrat or republican? Or is this a confounding relationship? What about the number of high shcool dropouts in a county, will this influence presidential selection?

Chapter 6

SIMPLE LINEAR REGRESSION

I realize i have not made it very far into this book, but i am starting to understand why writers drink. It is possible that once you get through this book or others like it you will want to drink too. Don't do it, unless you can have just one.

I think it's difficult for a professor or teacher to know at exactly what point should linear regression be taught in a curriculum, it seems like it turns up everywhere calculus, algebra stats, modeling. It should be in all of them, but then the next question is do you need to know algebra, matrix algebra and linear algebra before knowing how to do a linear regression? I don't know to be honest. Having worked with SQL for most of my adult life I have had to know and use a tiny bit of all three and did not pay much attention to it or realize until I started formally beefing up my academics.

Regardless, the one thing I have heard from a few stats instructors is "don't worry about how its done or how it works, the software will take care of it for you", to be fair, these were not stats professors at the local beauty college, these were ivy league educated (I checked) professors and teachers saying this. Which, my problem is if I don't know how it works I will probably not truly understand it, ever. Depending on what you are doing a trivial knowledge may be sufficient, but what if it's not? If I am in an interview can I use the words "the software will do it for me" as the answer to a hard question?

Section 1

INTRODUCTION TO LM()

I have decided for this chapter to present it as a top-down approach to linear regression, that means i am not going to teach all of the minutia first, i am going to go straight to code perform a regression, a prediction and then explain what everything is.

In the next few pages, I will do my best to define Linear Regression in R using lm(). Your first question should be when do I use lm or linear regression to solve a problem. In short, you have a quantitative explanatory variable and a quantitative predictor. Yup, already lost you huh? Here is the deal, I am going to use the mtcars (motor trend cars 1974) dataset because everyone uses it and its super simple to understand. Let's start with a hypothesis, or an idea i want to test; if a car increases in weight does the gas mileage go up, down or stay the same? Stays the same would be the null hypothesis if you are keeping track, does not stay the same would be the alternative hypothesis. But, if you are not ready for hypo testing just move along don't sweat it and keep going.

One item to clear up is the difference in Simple Linear Regression and Multiple Linear Regression; Simple has one explanatory variable, Multiple has more than one. I know right? Your hair is blown back by that piece of complex knowledge.

To start with we need some data and luckily have that. A good starting point for any exploration of data is quite simply looking at the data, perform some basic descriptive statistics on the data and make sure you have some understanding of what it is.

At this point you have seen the output from the following functions in prior chapters, so i will not repeat them. But, do your own analysis of the two variabies we will be using.

```
data(mtcars)

head(mtcars)

View(mtcars)
```

```
summary(mtcars)


summary(mtcars$wt)

summary(mtcars$mpg)


str(mtcars)
```

If you are outside of the US the first thing you may want to do is convert mpg to liters per 100 kilometers so this data set makes more sense to you.

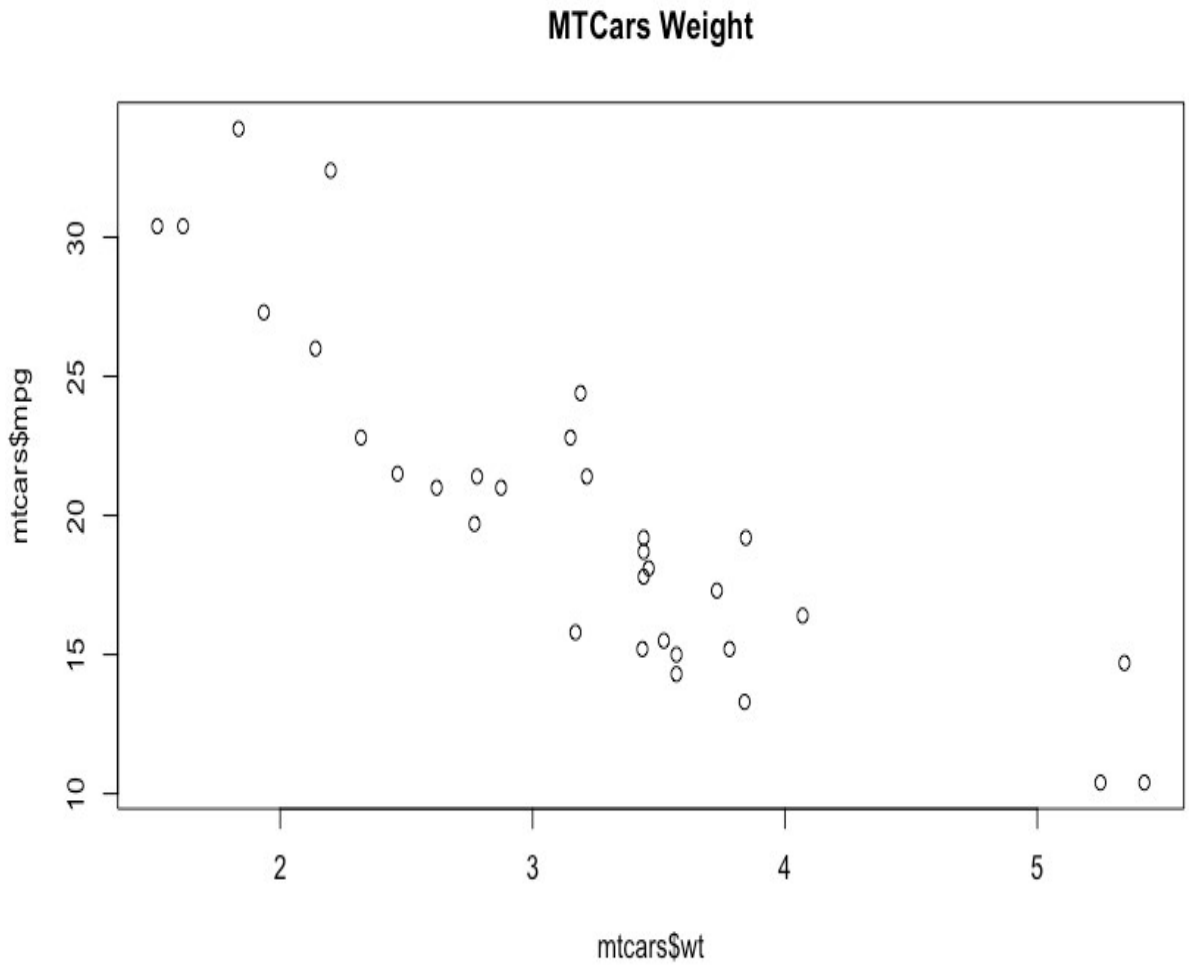
So if you must;

1 US gallon is equal to 3.785 liters
1 Mile is equal to 1.609 Kilometers

Liters per 100 kilometer = (100 * 3.785) / (1.609 * MPG)

```
# If you absolutely must convert, here you go

mtcars$lp100k <- (100 * 3.785) / (1.609 * mtcars$mpg)


plot(mtcars$mpg ~ mtcars$wt,

     main = "MTCars Weight");
```

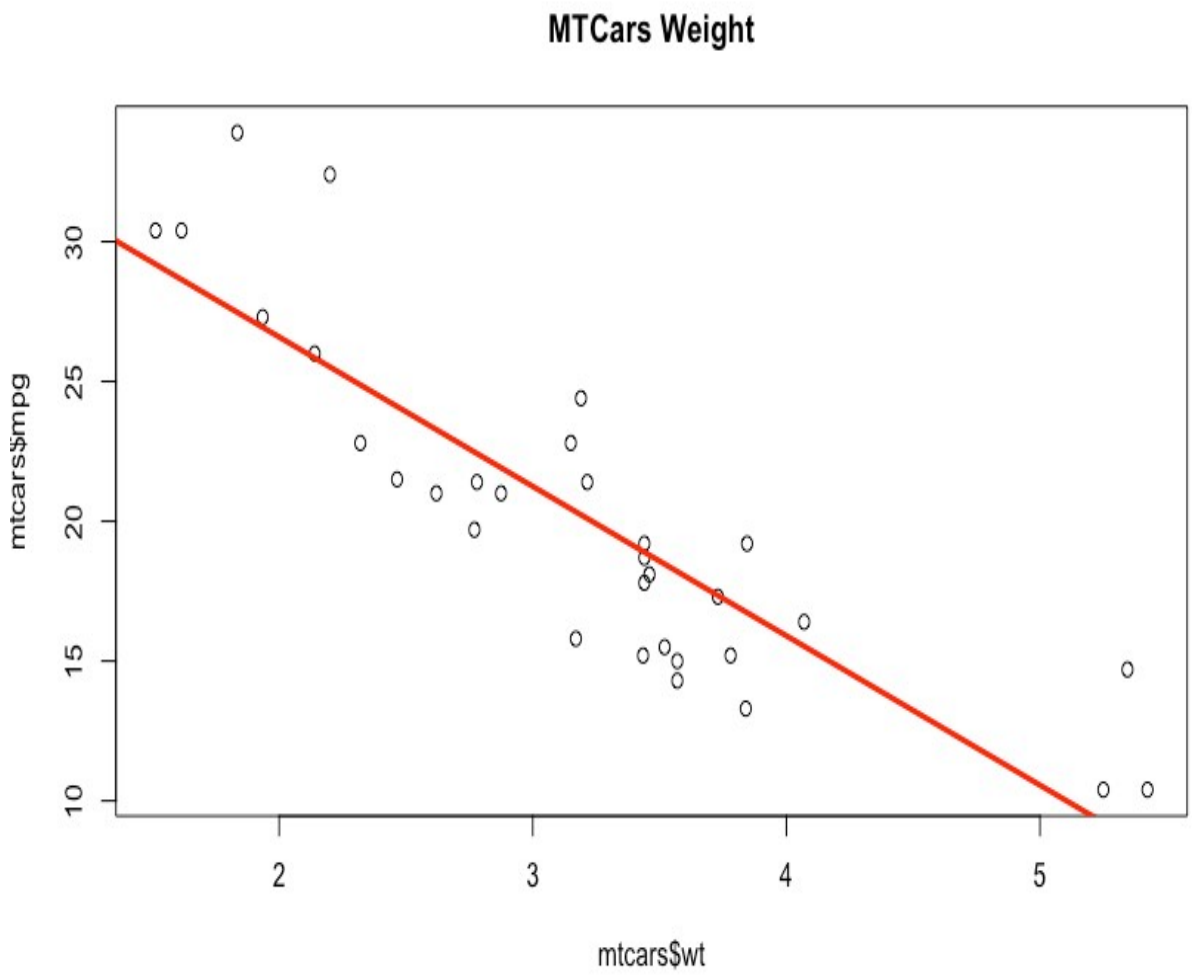
Continue to explore a bit to see if there appears to be a relationship between x and y, or weight and miles per gallon or liters per 100k. Specifically, we can start with an XY scatter plot to see if we can find anything interesting.



With very little we can get a scatter plot and just looking we can see there appears that as weight decreases, (weight of 2 = 2000 pounds) miles per gallon increase. Though jagged and scattered, it does look like something is here.

Now we can use a linear regression line to see what it will look like by adding the abline command and passing the results of an lm to it. I will explain how that line magic happens later on.

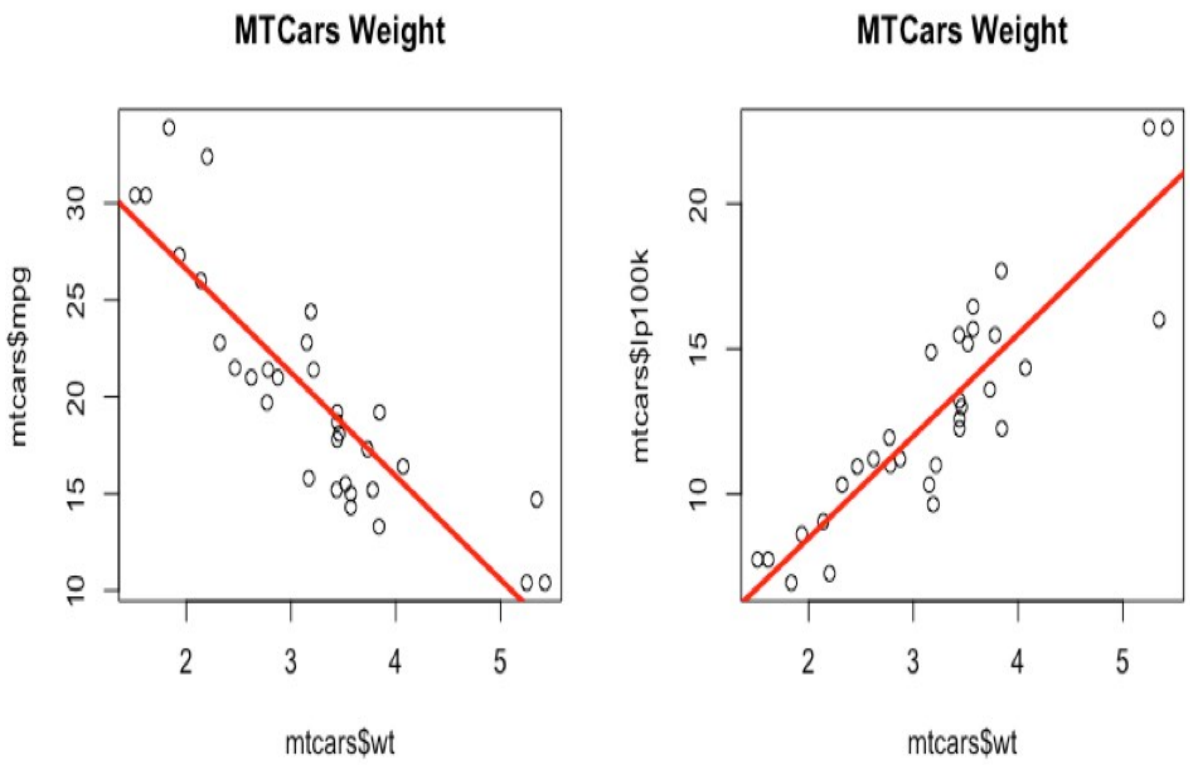
```
abline(lm(mpg ~ wt,data=mtcars),
       lwd=3,
       col="red")
```



For fun lets add the new variable for our European and Canadian friends. For this lets plot the two side by side using par(mfrow=c(rows,columns)).

```
par(mfrow=c(1,2))
plot(mtcars$mpg ~ mtcars$wt,
     main = "MTCars Weight")

abline(lm(mpg ~ wt,data=mtcars),
       lwd=3,
       col="red")

plot(mtcars$lp100k ~ mtcars$wt,
```

```
      main = "MTCars Weight");
abline(lm(lp100k ~ wt,data=mtcars),
       lwd=3,
       col="red")
```



Without thinking your first response is converting to liters gives me better gas mileage... NOPE... This is actually a prefect example of what happens when you convert a unit of measure to another unit of measure. We went form measuring the miles we travel using one gallon of fuel vs how many liters it takes to travel 100 kilometers, its completely reversed and at a different scale. The line and the dots are the same in both charts, but sort of reversed from each other. If you have some doubt you can add labels to the plot by adding "text(mtcars$wt, mtcars$lp100k, row.names(mtcars), cex=0.9, pos=4, col="red")" after the abline command. Be sure to change the y axis for mpg if you go back and forth.

But for now, our basic question was is there a relationship between the weight and miles per gallon, from a cursory look at the data, I would say yes, and there does appear to be a somewhat linear relationship.

The next step in our process is to see if we can fit a model to the data, that means can we take the data we have and use it to predict the miles per gallon of a vehi-

cle if we know the weight with some level of certainty?

Lets create a model... The following command will create a linear regression model using mpg as the predictor and wt(weight) as the explanatory variable. You can just as easily replace mpg with lp100k if you prefer.

```
mtcars.1 <- lm(mpg ~ wt, data=mtcars)
summary(mtcars.1)
```

If you did it correctly, there are no errors in the console and a new list in the global environment called mtcars.1. To get some useful meat out of this run summary() against mtcars.1.

```
> mtcars.1 <- lm(mpg ~ wt, data=mtcars)
> summary(mtcars.1)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

There is a lot of data there, we are not goin got worry about any of it right now, lets just predict something. Since we only have one explanatory variable we can pass in a list, in this case a list of just one value, 2000 pounds, which is 2 in our dataset.

```
predict(mtcars.1,list(wt=c(2.0)))
```

```
> predict(mtcars.1,list(wt=c(2.0)))
       1
26.59618
```

Okay! So this tells us that based on the data the model was trained with a vehicle that ways 2000 pounds is expected to get about 26.6mpg based on the 32 1974

vehicles we used to train the model. Which if you go back and look at the dataset you will see that is somewhere in the 2140 and 1935 pounds. Works as expected.

A little more on prediction. So, what else can we do? interval="prediction", or interval="confidence". These are very close but still different, the definition of each must be precise, and it trips me up.

Confidence: The confidence interval also called the mean interval asks what is the mpg of a car at a specific weight. You will notice we had a tighter response band between lower and upper in the output of confidence vs prediction. What you see in the numbers is within the 95% confidence interval that the mpg will be between 24.82389 28.36848 based on the model.

Prediction: While similar, the main difference is the standard error is taken into consideration as part of the calculation which allows for a wider band or upper and lower range of values.

You can read more about this prediction and confidence here. Sometimes i'm happy to say the software does it for me.

```
        predict(mtcars.1,list(wt=c(2.0)),
interval="prediction")

    predict(mtcars.1,list(wt=c(2.0)),
interval="confidence")
```

```
> predict(mtcars.1,list(wt=c(2.0)), interval="prediction")
       fit      lwr      upr
1 26.59618 20.12811 33.06425
> predict(mtcars.1,list(wt=c(2.0)), interval="confidence")
       fit      lwr      upr
1 26.59618 24.82389 28.36848
```

Tada! You have done your first linear regression and prediction. Easy right?

Section 2

INTRODUCTION TO THE MEASURES

Having done your first linear regression, there is actually a little bit of work that needs to be done to decide if the model actually is accurate. This is the art and the time that is spent on any model. We are lucky in that the data is already prepared, as in no data engineering is required. Knowing that lets look at the output of the summary again.

```
> mtcars.1 <- lm(mpg ~ wt, data=mtcars)
> summary(mtcars.1)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

What is all of that mess and does any of it actually matter? Yes. It matters, we will go through all of it. Two things i want to point out in this post is R-Squared and Coefficients.

R-Squared

First the R-Squared, as general rule of thumb, though there are dozens of exceptions to this rule, the closer to 1.0 the R-Squared is the better the model is at predicting something. You can only use the R-Squared when you have one explanatory variable, we have wt(weight) to predict mpg and nothing else, otherwise use adjusted r-squared, more on that later. The simplest definition of R-Squared; 75.8% of the variability in the data is accounted for by the model. Another way of saying it, R square is a way of measuring the relationship between x and y, or in this case mpg and wt. 1 is a perfect fit, if it is close to zero then x and y must have nothing to do with each other.

Coefficients

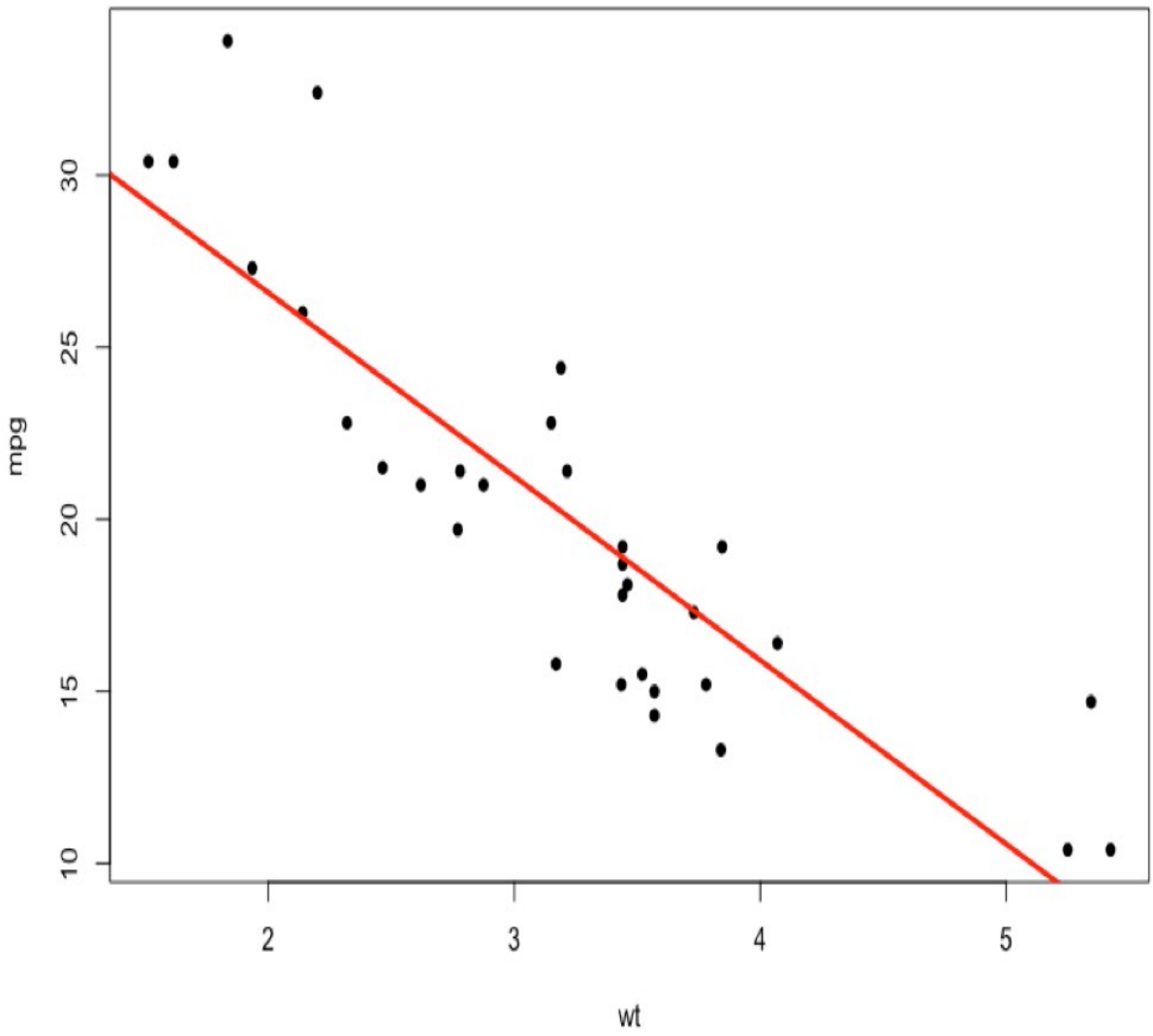
What does all this numeric jibber jabber mean? Focusing on just Estimate and the (intercept) and mtcars$wt. You have actually seen this already but were unaware of it. In the scatter plot where we had the line through the plot we ran lm() inside the abline function and a line magically appeared? So, what magic was that? if you run lm() and do not save it to a variable it will return and intercept coordinate and a slope coordinate. In this case 37.2851 as the intercept which means that the line will cross zero on the y axis at 37.2851 mpg and have a slope of -5.3445. If we take the original scatter plot and feed in just the numbers you will see the same line.

```
lm(mpg ~ wt, data=mtcars)
```

```
> lm(mpg ~ wt, data=mtcars)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Coefficients:
(Intercept)           wt
     37.285       -5.344
```

```
plot(mpg~wt,data=mtcars,
     cex=1,
```

```
      pch=16);
abline(37.2851, -5.3445,
       lwd=3,
       col="red"
```



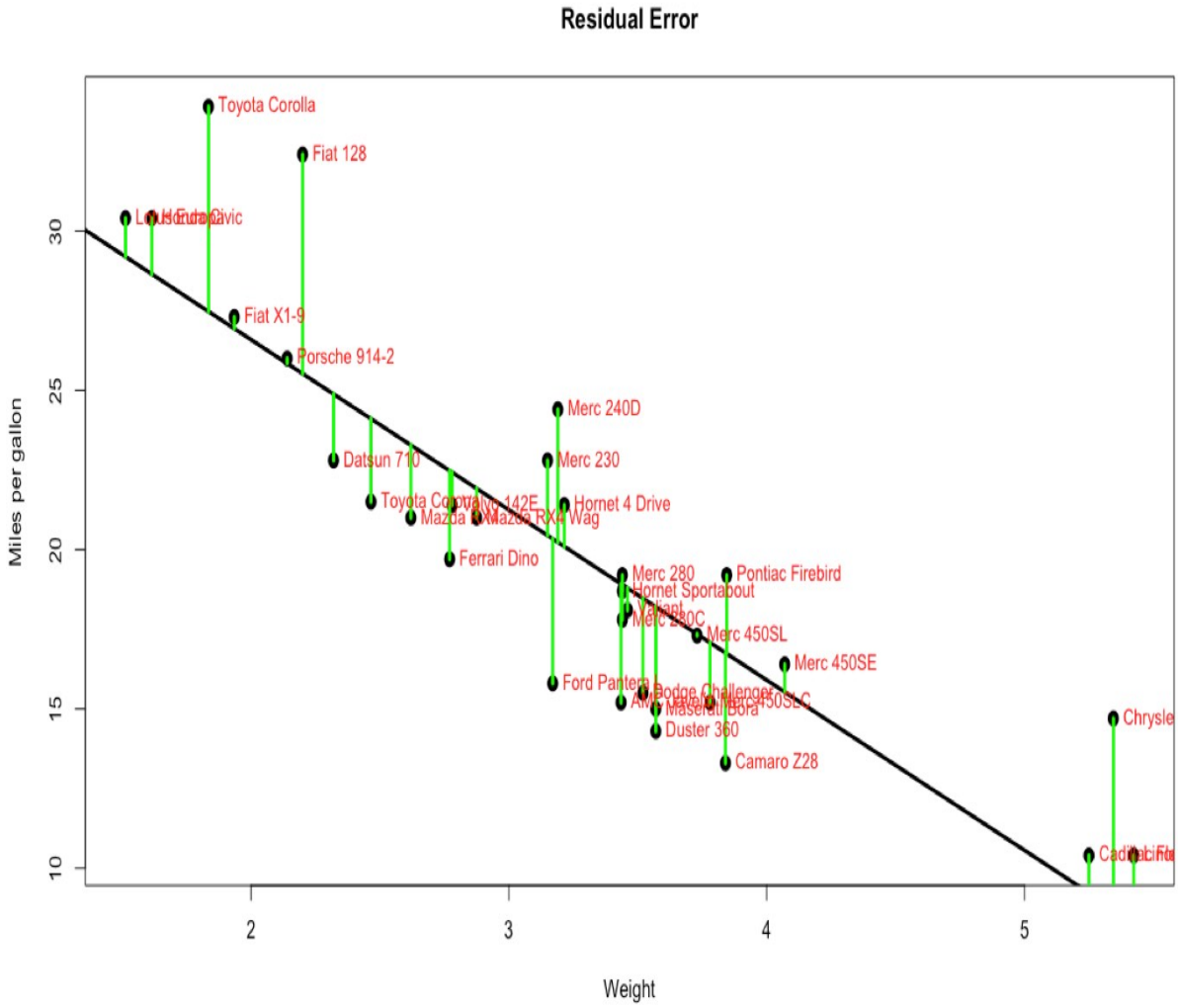
But, what does this really mean? Recall that wt is in thousands, 1.0 = 1,000 pounds, 2 = 2,000 pounds, 3.440 = 3,440 pounds etc. For the dataset they were all divided by 1000. What the scatter plot and lm tell us is that for every 1 unit

(1000 pound) increase in weight the mpg will decrease by 5.3445 miles per gallon on average based on the data we have provided.

Section 3

EVALUATING A MODEL

All the numbers, so many numbers, do we really need all of them…? If you want to make a an informed decision yes, you do. On the bright side they are not very hard to interpret, almost all of the numbers are related to just one number… The Error, more specifically the Residual Error. I'm going to blow your hair back again, its not an error, its nothing like and error, it should have never been called an error, its a difference. It is the difference between the line we ran through the middle of the scatter plot and the data points. Each point has a difference between the line and where the dot falls. Take a look at the visual below, the green line represents the distance between the line and the actual data point, thats our residual error. Its not hard to see that the larger the distance between the line and the data points the worse our model will perform. Not to mention the closer the data is to the the better.



Code for this is below if you want to adapt your own, I am 99% sure i grabbed this out of the ISLR at some point in the last two years, they show a sample on page 62, for lack of exact page and section of code just get your own copy, i consider ISLR a necessary resource. In fact, it is the next book you need to buy or download if you have not already.

```
plot(mtcars$mpg~mtcars$wt,
     cex=1.5,
     pch=16,
     xlab="Weight",  #X axis label
     ylab="Miles per gallon",
```

```
          main = "Residual Error") #Y axis label)

     abline(lm(mtcars$mpg ~ mtcars$wt),

            lwd=3,

            col="black")

     text(mtcars$wt, mtcars$mpg, row.names(mtcars), cex=0.9, pos=4,
col="red")


     mtcars.1 <- lm(mtcars$mpg ~ mtcars$wt)


     summary(mtcars.1)

     #Add some lines between the line and the actual to demonstrate
the error

     yhat <- predict(mtcars.1 , tfr = mtcars$wt)

     join <- function(i)


lines(c(mtcars$wt[i],mtcars$wt[i]),c(mtcars$mpg[i],yhat[i]),col="gr
een",lwd=3)

     sapply(1:32,join)
```

Going back to our output from the summary of our model we can start with the first section, Residuals.

One thing i want to point out in my scripts is how i qualify the variables when passed into lm(). The correct way, "**lm(mpg ~ wt, data=mtcars)**" the wrong way; "**lm(mtcars$mpg ~ mtcars$wt)**". I sometimes get lazy and just type it out without paying much attention, but, its not a good habit to get into, i have tried very hard to make sure every one in this book is the correct way. So do as i say and not as id do if you happen to see the wrong syntax and be sure to add data= when qualifying your dataset.

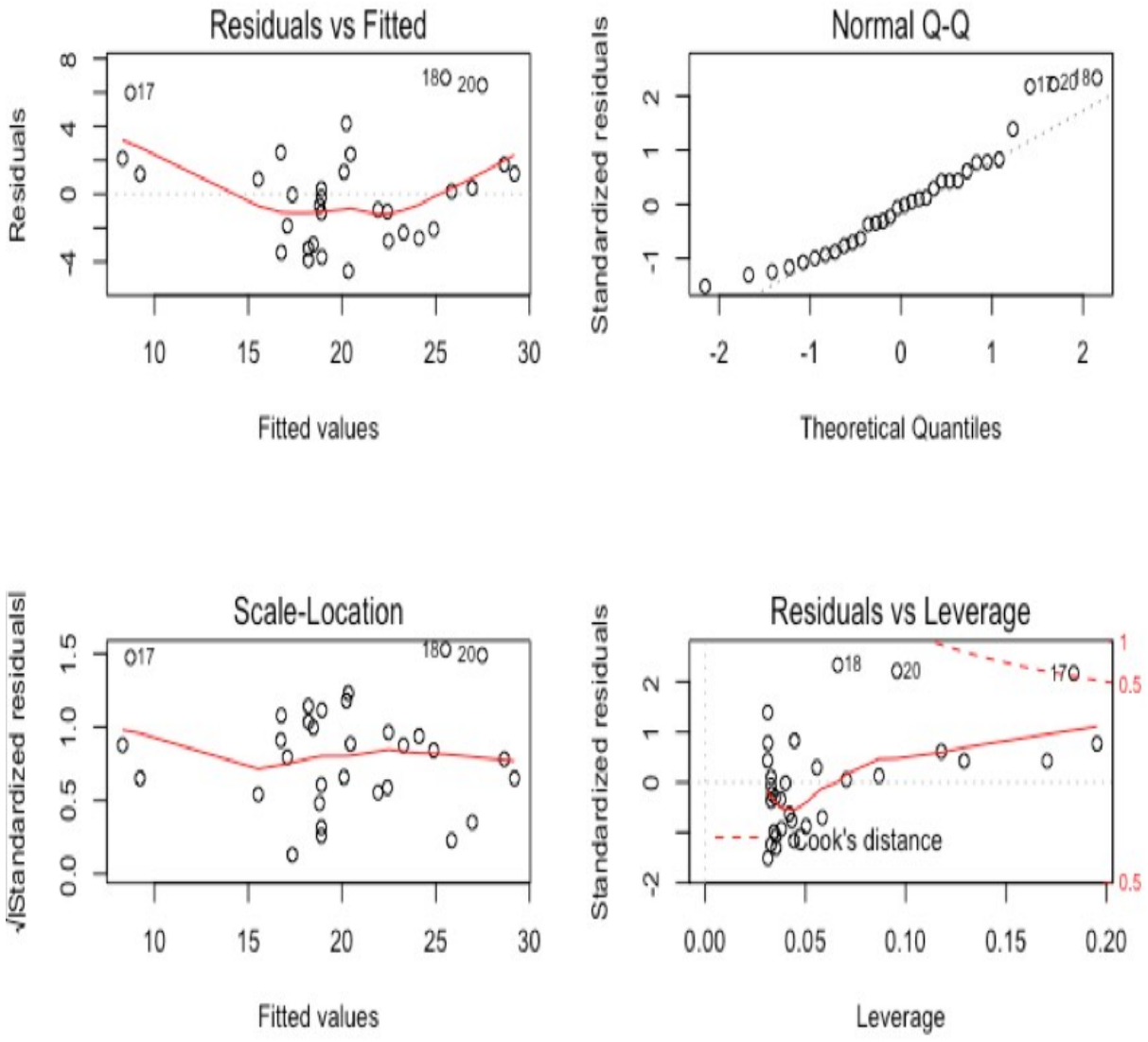
You can get a vague idea of the distribution of the residual errors by looking at a histogram or a plot of them, or plot the entire model with one com-

mand. If you want to know where that comes from just run the code below. Now another frustration, depending on who is teaching everyone has a different opinion about what is useful. I have included what to look for in each, regardless of opinions.

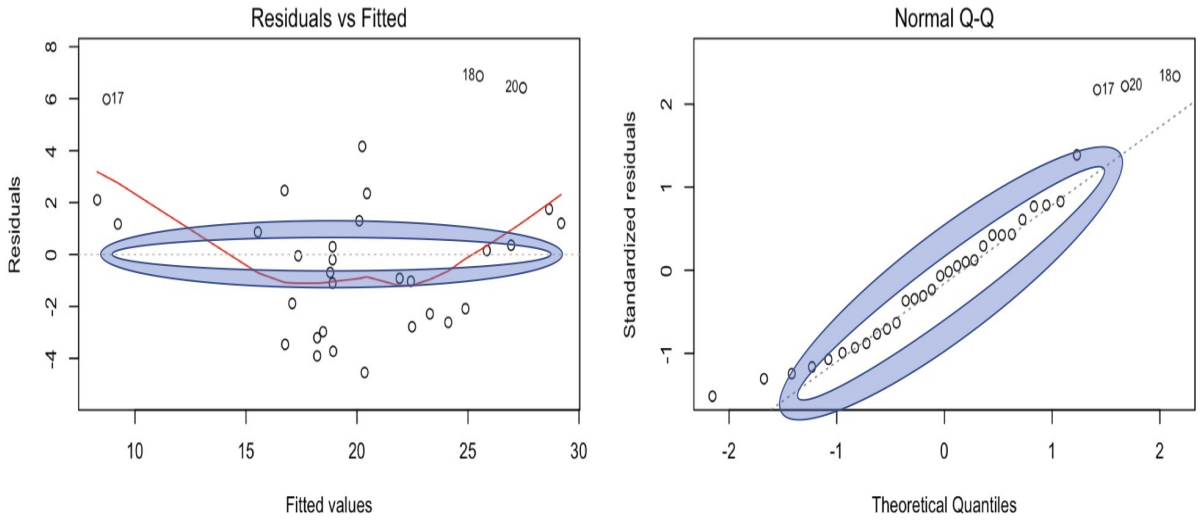
```
    # Make sure numbers are not hugely far away
from zero,

    # Be aware fo scale.

    summary(mtcars.1$residuals)


    # Scattered all about the plot is the generic
explanation.

    # If the data is clustered in one area the mod-
el will lean towards satisfying that data

    # but stink at predicting other data.


    plot(mtcars.1$residuals)


    # Generally looking for normal distrbution

    hist(mtcars.1$residuals)


    # hit enter in the console after each plot,

    # or set par(mfrow=c(2,2)) to see them all
together


    par(mfrow=c(2,2))
```

```
    plot(mtcars.1)


    # if you want to have some fun figure out the
mean of the residuals,

    # since we have scipen=999 you will never get
exact zero, but its zero.

    # Mean of the residuals will always be zero.


    mean(mtcars.1$residuals)
```

Form the plot(mtcars.1), the first two graphs are what to focus on, this is very subjective, but once you have done this with a few datasets it will make sense. You want the residual vs fitted to be as close to zero as possible, in this case the data is all over the place, the line is using something called smoothing to give you an idea where the data is concentrated or leaning, you will notice some of the points have numbers, those are outliers, which usually means more than two standard deviations away from the mean of the residual errors.



The Normal Q-Q should have all the points on or hugging the line, but ours looks a bit jagged and the far left and far right are way off the line indicating we will have trouble predicting low mpg cars, and very high mpg cars. In other words, based on the Residual Errors, we may not have the best data set or weight is a less than stellar explanatory variable, though clearly has some value.



Moving along with the output from summary(mtcars.1), with one change, if you run options(scipen=999) R will drop the scientific notation and show you the entire number, be careful with this as a computer has no concept of zero so some things you expect to be zero may now become an extremely small number.

```
> mtcars.1 <- lm(mpg ~ wt, data=mtcars)
> summary(mtcars.1)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858 < 0.0000000000000002 ***
wt           -5.3445     0.5591  -9.559       0.000000000129 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 0.0000000001294
```

The **Coefficient Estimate** was discussed briefly in the last section, essentially the intercept and slope, next is the **standard error**, in short it is the average expected error for a single prediction. ISLR page 65 discusses this, the average the prediction differs from the actual, the lower the better. Did i mention you need a copy of ISLR on hand?

T-Value is the Estimate divided by the **Standard error,** the farther from 1 the better. You can imagine based on the description of Standard error that if the estimate is 37 and the error is 37 our model is horrible, and t-value would be 1.

The last column is "**Pr(>|t|)**" (or **p-value**) here is what that means in english; The Probability of the absolute value of T occurring randomly in the

world. In other words, is this variable Weight useful in predicting mpg or is it nonsense. A value of <.05 is typically considered a good predictor of usefulness of the variable, but that too is subjective. For multiple regression all of these must be taken into consideration. **Signif. codes**: is your cheat sheet, R tries to help you by placing 1,2 or 3 stars next to a coefficient so you can easily tell if it is useful or not, always look at the actual p-value.

Next is the "**Residual standard error**: 3.046 on 30 degrees of freedom" or the **RSE**, RSE is the **Sum of Squared Errors/Degrees of freedom**. Wait, what? RSE is a quotient i have never heard of with a thing divided by another thing i have never heard of. So, remember when i said the software will do it for you, this could be one of those moments... So what i am going to do instead is in the next section show just these numbers and where they all come from, and then provide you an excel spreadsheet so if you want to plug in all of your own values and see what happens, you can.

Section 4

LM() JUST THE NUMBERS

In the last section i mentioned that most of the numbers come form the residual errors, thats not entirely true. You have a basic understanding of lm() you learned that R-square is the number to look at, that is based on residual error. You are also told to examine the p-value for each coefficient and for the entire model. P-value is a little bit harder to calculate, go search and find out for yourself. But in lieu of that i am going to provide the actual calculation for everything you may have seen reference in an lm.

N,n: Upper case N is population size, lower case n is the sample size.

Coefficients: Intercept is mpg in this case, the estimate is where the line crosses the y axis at Zero. mtcars$wt is the slope of the line.

Standard error: Imagine this as the confidence interval of the estimate, it is all so the expected error +/-. When we do the prediction this may make more sense.

T-Value: is the estimate/standard error the farther from 1 the better.

p-value: Usually <.05 is used and what you are looking for, but it may be .01 or .001.

Degrees of Freedom: Short uncomplicated, answer is the number of cases(rows) minus the number of coefficients.

Mean: Average of a variable or column

Variance: (sum(mean mpg – mpg) / n-1)

Standard Deviation: sqrt(variance)

Residuals: mpg – (slope + intercept * weight)

Error Squared: Residuals^2

Residual Standard Error RSE: variance / degrees of freedom

SSE or Sum of Squared Errors: Sum(Error Squared)

Multiple R-Squared: 1-SSE/(sum(mpg – mean(mpg))^2)

Adjusted R-Squared: 1-SSE/(sum(mpg – mean(mpg))^2)*(n-1)/(n-(1+1))

F-Statistic: should be far from 1, but how far depends on the units. This is a good measure of the entire model and useful for comparing one model to another.

P-Value for the entire model, again <.05 is the generic goal, also useful when comparing one model to another.

I have included the excel spread sheet on the book github under Chapter 6 if you really want to know how all of this works. If everything goes well these numbers will match what came out of the R summary of lm for mtcars with mpg as the predictor and weight as the explanatory variable.

n =	32
DF (Degrees of Freedom) =	30
MEAN	
MEAN MPG =	20.090625
MEAN WEIGHT =	3.21725
Variance	
Variance MPG =	36.324103
Variance Weight =	36.324103
Standard Deviation	
SD MPG=	6.026948
SD Weight =	6.026948
Slope =	-5.344472
Intercept =	37.285126
SSE =	278.3219
Residual Standard Error RSE =	3.0459
Multiple R-Squared =	0.7528
Adjusted R-Squared =	0.7446
F-Statistic =	91.375325

	mpg	MPG Var from mean	Residuals	Error Squared (Ei)	wt	Var from mean
Mazda RX4	21	0.826963	-2.282611	5.210311	2.62	0.826962891
Mazda RX4 Wag	21	0.826963	-0.919770	0.845978	2.875	0.826962891
Datsun 710	22.8	7.340713	-2.085952	4.351196	2.32	7.340712891
Hornet 4 Drive	21.4	1.714463	1.297350	1.683117	3.215	1.714462891
Hornet Sportabout	18.7	1.933838	-0.200144	0.040058	3.44	1.933837891
Valiant	18.1	3.962588	-0.693255	0.480602	3.46	3.962587891
Duster 360	14.3	33.531338	-3.905363	15.251857	3.57	33.53133789
Merc 240D	24.4	18.570713	4.163738	17.336715	3.19	18.57071289
Merc 230	22.8	7.340713	2.349959	5.522309	3.15	7.340712891
Merc 280	19.2	0.793213	0.299856	0.089914	3.44	0.793212891
Merc 280C	17.8	5.246963	-1.100144	1.210317	3.44	5.246962891
Merc 450SE	16.4	13.620713	0.866873	0.751469	4.07	13.62071289
Merc 450SL	17.3	7.787588	-0.050247	0.002525	3.73	7.787587891
Merc 450SLC	15.2	23.918213	-1.883024	3.545778	3.78	23.91821289
Cadillac Fleetwood	10.4	93.908213	1.173350	1.376749	5.25	93.90821289
Lincoln Continental	10.4	93.908213	2.103288	4.423819	5.424	93.90821289
Chrysler Imperial	14.7	29.058838	5.981074	35.773251	5.345	29.05883789
Fiat 128	32.4	151.520713	6.872711	47.234161	2.2	151.5207129
Honda Civic	30.4	106.283213	1.746195	3.049198	1.615	106.2832129
Toyota Corolla	33.9	190.698838	6.421979	41.241816	1.835	190.6988379
Toyota Corona	21.5	1.986338	-2.611004	6.817341	2.465	1.986337891
Dodge Challenger	15.5	21.073838	-2.972586	8.836269	3.52	21.07383789
AMC Javelin	15.2	23.918213	-3.726866	13.889533	3.435	23.91821289
Camaro Z28	13.3	46.112588	-3.462355	11.987904	3.84	46.11258789
Pontiac Firebird	19.2	0.793213	2.464367	6.073105	3.845	0.793212891
Fiat X1-9	27.3	51.975088	0.356426	0.127040	1.935	51.97508789
Porsche 914-2	26	34.920713	0.152043	0.023117	2.14	34.92071289
Lotus Europa	30.4	106.283213	1.201059	1.442543	1.513	106.2832129
Ford Pantera L	15.8	18.409463	-4.543151	20.640224	3.17	18.40946289
Ferrari Dino	19.7	0.152588	-2.780940	7.733627	2.77	0.152587891
Maserati Bora	15	25.914463	-3.205363	10.274350	3.57	25.91446289
Volvo 142E	21.4	1.714463	-1.027495	1.055746	2.78	1.714462891

Section 5

LM() P-VALUE AND HACKING

We need to spend some time on **p-value**. The calculation for p-value is a hot nightmare, not going to bother with it right now, if you need to know more about it you can find online calculators, but rarely an actual formula. Even the sites that will spill the beans on all the other formulas will resort to a t-distribution calculator for p-value. Though i may fall back in the future and spend some time on t-distribution, we shall see.

P-value has been the subject of controversy because all we have is general guidance on evaluating it. Most stats classes will state that it should be < .05 for the coefficient to be considered significant. More specifically, "The P-value is the probability of observing a sample statistic as extreme as the test statistic.". OR from ISLR page 67 p-value "a small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance, in the absence of any real association between the predictor and the response. Hence, if we see a small p-value, then we can infer that there is an association between the predictor and the response."

All sounds very simple, so whats the controversy? You can hack it! If we are using distribution of data and correlation between x and y, can i not remove rows of data to improve my odds of getting a low p-value? Or what if i had dozens or hundreds of columns and just wanted to find the ones that support my hypothesis? Its called data dredging.

Imagine for a moment you needed to test a new drug, it would not be surprising that if it is your drug you may want it to be successful in the trial, or you may lose funding, lose your job, maybe you have been working on this drug for ten years. Would you select patients that represent the average population of the consumer or would you select patients that would have the highest likelihood of success? What if your only trial subjects were professional guinea pigs? What if your results can not be reproduced? fivethir-

tyeight did a good write up on p-hacking, the very p-value we are using in linear regression.

Why does it matter to you? What if you work for one of these companies and you see something that is biased, not reproducible, or are data dredging, what will you do?

That being said, there is a school of thought that says the p-value should be thrown out as a measure of success, i tend to agree. Whats worse, what if the data dredge or bias becomes part of the production system that you sell to a consumer?

We will continue to use p-value going forward but will open up to more as well. Imagine creating 30 models and just compare the entire model to another model…?

Chapter 7

MULTIPLE LINEAR REGRESSION

The next few sections will be adding some more explanatory variables to see if we can get a better model from predicting mpg. In the REAL world you would never predict a vehicle mpg by weight alone, there are dozens if not hundred of other variables to consider. Lucky for us the mtcars dataset only has 11 variables to consider. The grand finale of this linear regression will be a real dataset we can play with from the EPA with thousands of rows and dozens of columns.

Section 1

LM() END TO END

First things first, data exploration! For this series i want to make the names slightly more meaningful, move the row name to an actual row and create the liters per 100 kilometers. None of this is absolutely required but they are all good data engineering tasks to learn.

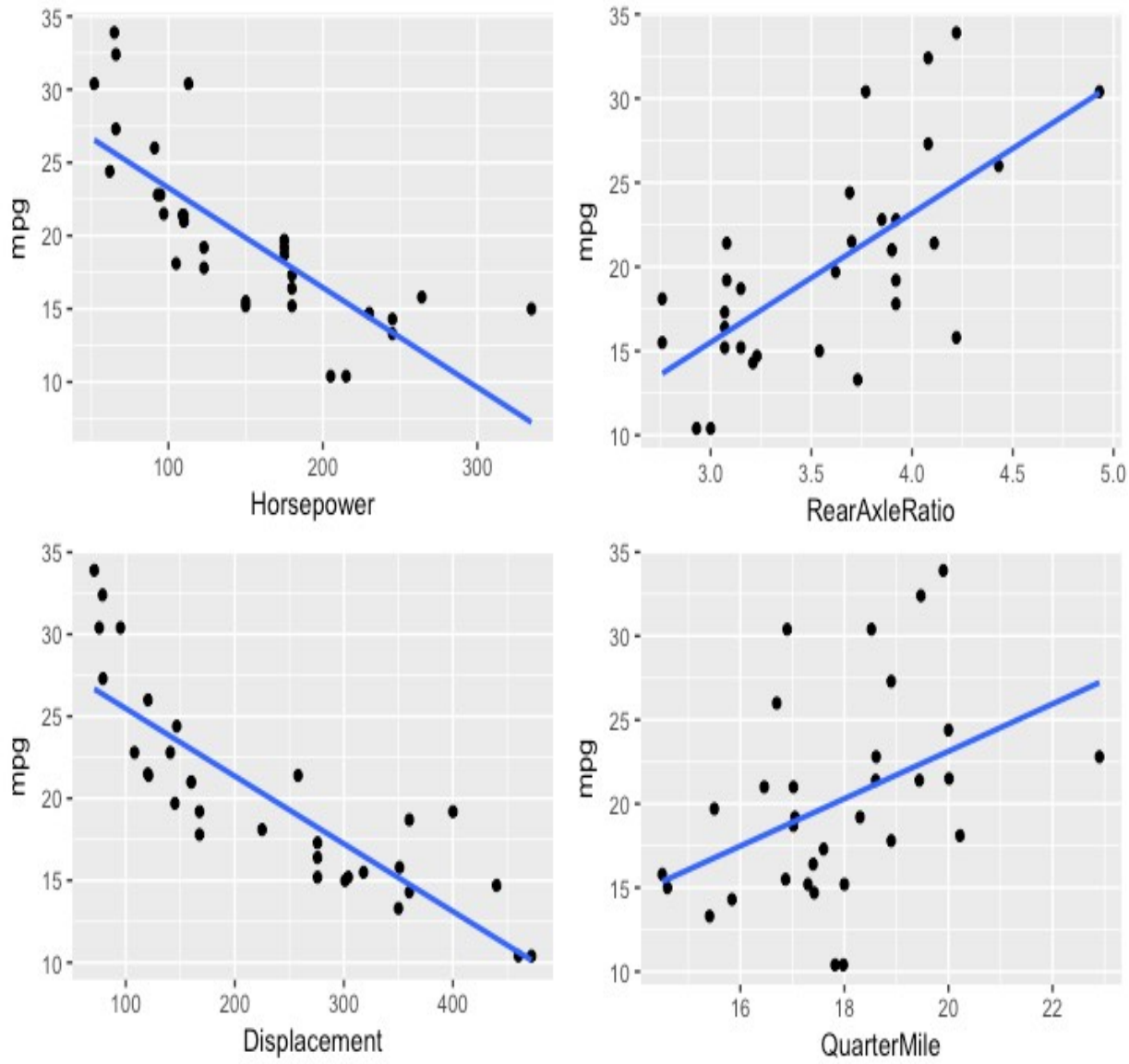
```
    data(mtcars)


    #rename the columns to something slightly more
meaningful
    names(mtcars) <- c("mpg", "Cylinders", "Displace-
ment", "Horsepower", "RearAxleRatio","Weight", "Quar-
terMile", "VSengine", "TransmissionAM", "Gears",
"Carburetors")


    # Create liters per 100 kilometers column
    mtcars$lp100k <- (100 * 3.785) / (1.609 *
mtcars$mpg)



    #fix the row names, i want them to be a column
    mtcars$Model <- row.names(mtcars)
    row.names(mtcars) <- NULL
    names(mtcars)
```

We will start with exploring a few simple variables, first is it linear? I am going to pull a ggplot trick here, in Base R graphics you can use "par(mfrow=c(2,2))" to get 4 graphs in one pane. GGPlot will not have it, so, awesome function called multiplot here.

```
##Explore
library(ggplot2)

p1 <- ggplot(mtcars,aes(x = Horsepower, y = mpg))
+
  geom_point() +
  geom_smooth(method='lm', se= F)

p2 <- ggplot(mtcars,aes(x = Displacement, y =
mpg)) +
  geom_point() +
  geom_smooth(method='lm', se= F)

p3 <- ggplot(mtcars,aes(x = RearAxleRatio, y =
mpg)) +
  geom_point() +
  geom_smooth(method='lm', se= F)

p4 <- ggplot(mtcars,aes(x = QuarterMile, y = mpg))
+
  geom_point() +
  geom_smooth(method='lm', se= F)
```

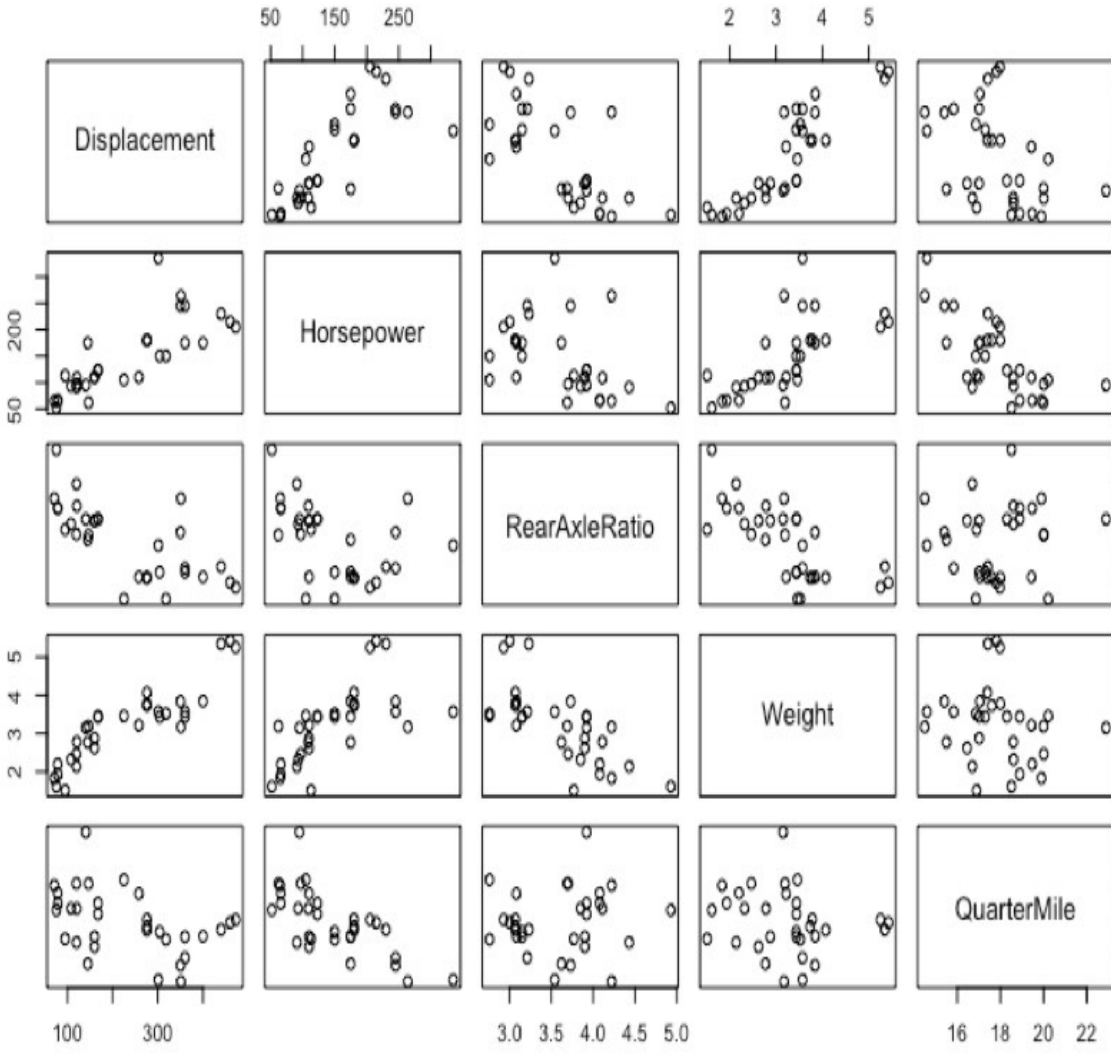
```
    #We have seen weight already, see the last several
posts.

    #ggplot(mtcars,aes(x = Weight, y = mpg)) +

    #  geom_point() +

    #  geom_smooth(method='lm', se= F)


    # for this to work you will need the function def-
inition linked above.

    multiplot(p1, p2, p3, p4, cols=2)
```

GGplot is not looking to shabby, eh? So what do we have? Horsepower, rear axle ratio, engine displacement, and quarter mile time. The first three, do seem to have a relatively decent linear relationship to mpg, rear axle ratio may be a bit of a stretch looks bad below 4.0 then goes worse. Quarter mile time looks terrible, at 18-19 second quarter mile time for instance could have a mpg of 10 to 30, not terribly linear, more random. For fun we will create a model with all of them and see what happens.



There are a couple ways to go about multiple scatter plots. One is pairs()

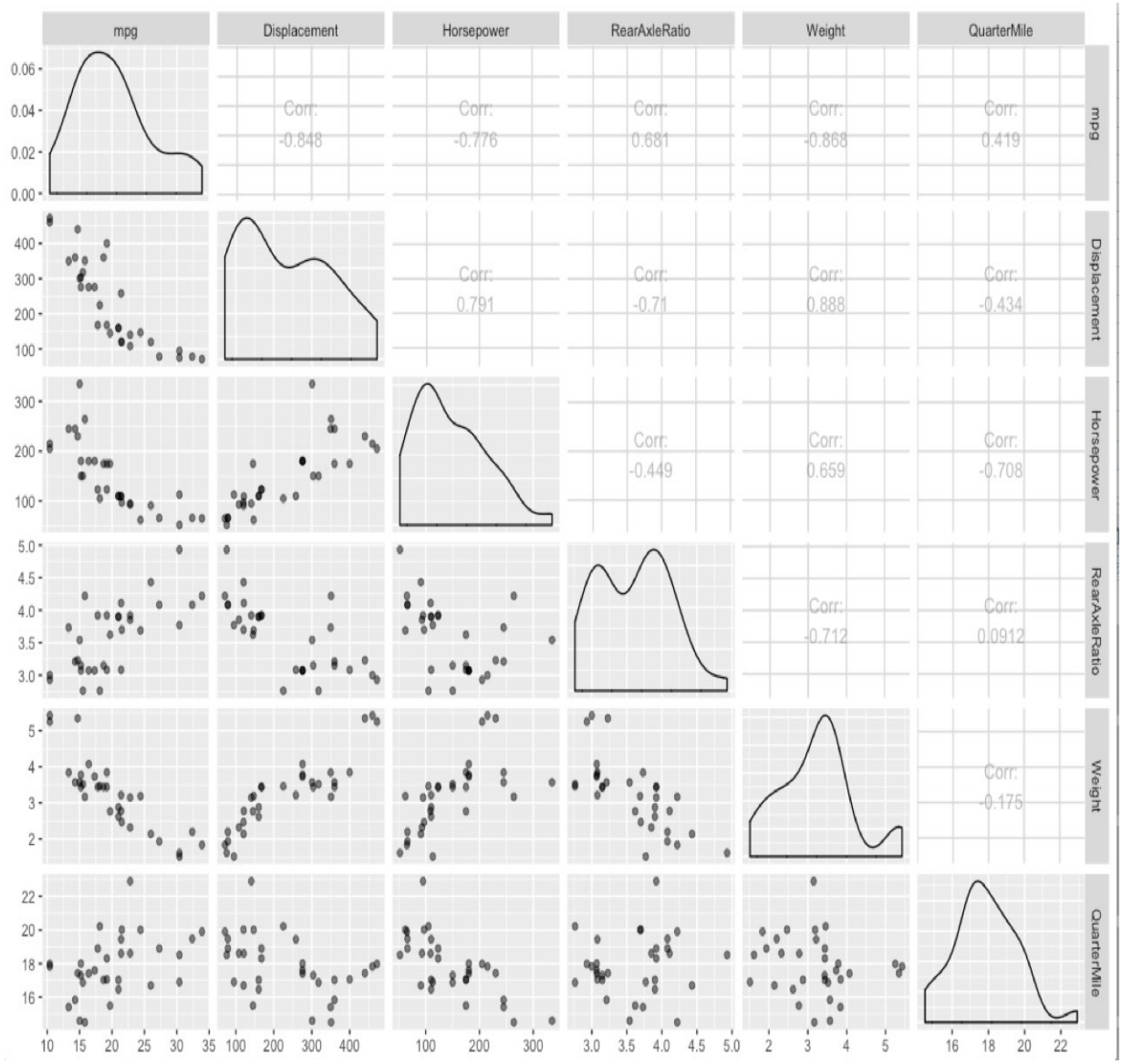
```
pairs(mtcars[3:7])
```




```
# more more scatter plot
#install.packages("GGally")
library(GGally)
ggpairs(mtcars[c(1,3:7)], aes(alpha = 0.9))
```

One thing i have not covered and probably should have by now is correlation, put simply it is the degree of relationship between two variables. You have seen this already in the mpg to weight regression, weight had a negative correlation to mpg, hence it went from the upper left of the scatter plot to the lower right. You

can see in the graphs above displacement also has a negative correlation to mpg while rear axle ratio and quarter mile have a positive correlation. Correlation is what is used to create the linear regression coefficient, more specifically the Pearson Correlation Coefficient. If you click on the image below in enlarge it you can see that each variable has a correlation to each other, if two explanatory variables are perfectly correlated you should drop one. You can also get an idea how well an explanatory variable will be at helping determine the predictor.



Another check is to see if the data is normally distributed, in other words is it mound shaped. I will save you the trouble of running this, its really not. This is a small data set, far too small to get everything perfect. The larger EPA dataset i

have my eye on will be a better test of this. We will move on with this, but be prepared for wonkyness.

```
## normally distributed
p1 <- ggplot(mtcars,aes(x = Horsepower)) +
  geom_histogram(bins=10)

p2 <- ggplot(mtcars,aes(x = Displacement)) +
  geom_histogram(bins=10)

p3 <- ggplot(mtcars,aes(x = RearAxleRatio)) +
  geom_histogram(bins=10)

p4 <- ggplot(mtcars,aes(x = QuarterMile)) +
  geom_histogram(bins=10)

#ggplot(mtcars,aes(x = Weight)) +
#   geom_histogram(bins=15)

multiplot(p1, p2, p3, p4, cols=2)
```

Lets run a new model with our new variables.

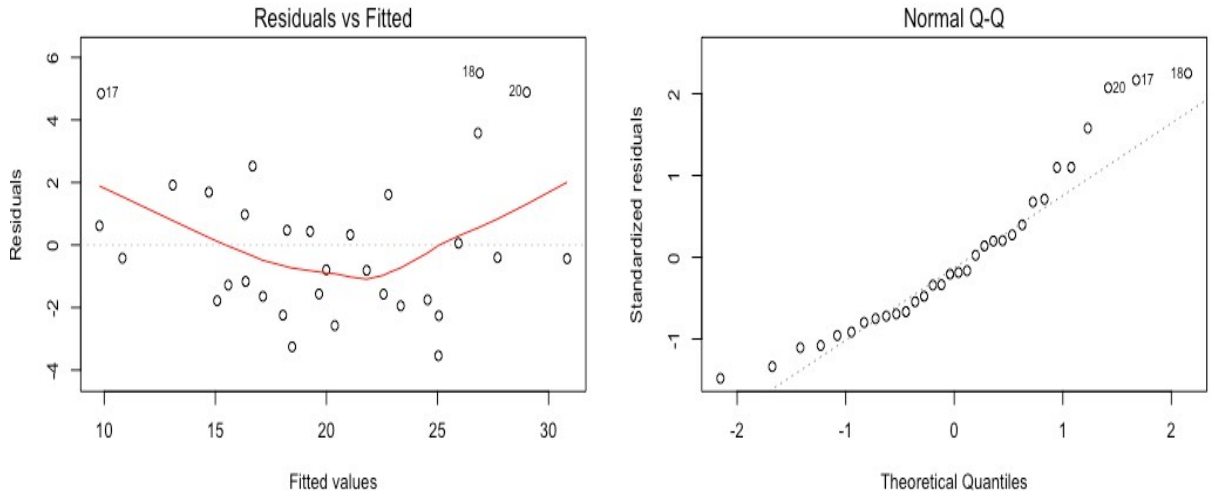
```
mtcars.1 <- lm(mpg ~ Horsepower + Displacement +
RearAxleRatio + QuarterMile + Weight, data=mtcars)
summary(mtcars.1)
```

And the results. Recall we are looking for a p-value for each coefficient of < .05, an Adjusted R-Squared as high as possible, closer to 1 the better. Residuals of constant variance and Normal QQ plot that follows the line. Though we were hoping that the relationship between Horsepower, rear axle ratio, engine displacement and mpg were okay-ish, the model disagrees.

```
> summary(mtcars.1)

Call:
lm(formula = mpg ~ Horsepower + Displacement + RearAxleRatio +
    QuarterMile + Weight, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5404 -1.6701 -0.4264  1.1320  5.4996

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.53357   10.96423   1.508  0.14362
Horsepower   -0.02060    0.01528  -1.348  0.18936
Displacement  0.00872    0.01119   0.779  0.44281
RearAxleRatio 2.01578    1.30946   1.539  0.13579
QuarterMile   0.64015    0.45934   1.394  0.17523
Weight       -4.38546    1.24343  -3.527  0.00158 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.558 on 26 degrees of freedom
Multiple R-squared:  0.8489,    Adjusted R-squared:  0.8199
F-statistic: 29.22 on 5 and 26 DF,  p-value: 0.0000000006892
```

```
par(mfrow=c(2,2))
```

```
plot(mtcars.1)
```



So the rule is, remove the worst performing p-value and run again. Displacement, you're fired!

```
mtcars.2 <- lm(mpg ~ Horsepower + RearAxleRatio + QuarterMile + Weight, data=mtcars)
```

```
summary(mtcars.2)
```

```
Call:
lm(formula = mpg ~ Horsepower + RearAxleRatio + QuarterMile +
    Weight, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5775 -1.6626 -0.3417  1.1317  5.4422

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   19.25970   10.31545   1.867 0.072785 .
Horsepower    -0.01784    0.01476  -1.209 0.237319
RearAxleRatio  1.65710    1.21697   1.362 0.184561
QuarterMile    0.52754    0.43285   1.219 0.233470
Weight        -3.70773    0.88227  -4.202 0.000259 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.539 on 27 degrees of freedom
Multiple R-squared:  0.8454,    Adjusted R-squared:  0.8225
F-statistic: 36.91 on 4 and 27 DF,  p-value: 0.0000000001408
```

Notice what happened, examining p-value and looking for < .05, everything but Weight became less significant. Considering how close they are, lets remove Horsepower since it is a tiny bit worse.

```
mtcars.3 <- lm(mpg ~ RearAxleRatio +QuarterMile + Weight, data=mtcars)

summary(mtcars.3)
```

```
> summary(mtcars.3)

Call:
lm(formula = mpg ~ RearAxleRatio + QuarterMile + Weight, data = mtcars

Residuals:
    Min      1Q  Median      3Q     Max
-4.1152 -1.8273 -0.2696  1.0502  5.5010

Coefficients:
             Estimate Std. Error t value    Pr(>|t|)
(Intercept)   11.3945     8.0689   1.412     0.16892
RearAxleRatio  1.6561     1.2269   1.350     0.18789
QuarterMile    0.9462     0.2616   3.616     0.00116 **
Weight        -4.3978     0.6781  -6.485 0.000000501 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.56 on 28 degrees of freedom
Multiple R-squared:  0.837,     Adjusted R-squared:  0.8196
F-statistic: 47.93 on 3 and 28 DF,  p-value: 0.00000000003723
```

Now, QuarterMile and Weight appear important, but RearAxleRatio is not, so lets drop it.

```
> summary(mtcars.4)

Call:
lm(formula = mpg ~ QuarterMile + Weight, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3962 -2.1431 -0.2129  1.4915  5.7486

Coefficients:
            Estimate Std. Error t value        Pr(>|t|)
(Intercept)  19.7462     5.2521   3.760        0.000765 ***
QuarterMile   0.9292     0.2650   3.506        0.001500 **
Weight       -5.0480     0.4840 -10.430 0.0000000000252 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.596 on 29 degrees of freedom
Multiple R-squared:  0.8264,    Adjusted R-squared:  0.8144
F-statistic: 69.03 on 2 and 29 DF,  p-value: 0.000000000009395
```

According to the data we have, they only important features as Quarter-Mile and Weight. Knowing a tiny bit about cars that does seem limiting. I can also say that removing the variables in a different order would have produced different results. Look below at what happens when you use just horsepower? This begs the question is p-value really the right thing to use?

```
mtcars.5 <- lm(mpg ~ Horsepower + Weight, data=mtcars)

summary(mtcars.5)
```

```
> summary(mtcars.5)

Call:
lm(formula = mpg ~ Horsepower + Weight, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept) 37.22727    1.59879  23.285 < 0.0000000000000002
Horsepower  -0.03177    0.00903  -3.519              0.00145
Weight      -3.87783    0.63273  -6.129           0.00000112
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,    Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 0.0000000000091
```

So lets do a prediction since we have gone through the trouble to do all of this.

```
Horsepower <- c(100,150,200)

Weight <- c(2.5,3.5,5.2)

newcars = data.frame(Horsepower,Weight)


predict(mtcars.5,newdata=newcars,interval="confi-
dence")
```

These predictions are reasonable, they are in line with a vehicle of 1974. Find out the HP in your vehicle and the weight try out the prediction and see what you

get. My vehicle is 395 hp, at 5700 pounds, the model does not accommodate that very well.

```
> predict(mtcars.5,newdata=newcars,interval="confidence")
       fit       lwr      upr
1 24.35540 23.159683 25.55111
2 18.88892 17.895283 19.88256
3 10.70796  8.448801 12.96712
```

That was however, relatively painful we had to create 5 models and review each one, what if we had had 100 variables? We still have variables in the dataframe we have not used yet, so, two more problems to solve in the next section.

Section 2

THE FORMULA

Remember how i said not math notation? How about a formula? I think this might me a tiny bit helpful is to realize that as many moving parts as there are in regression it all boils down to a pretty simple formula for calculating a prediction.

I am going to show the non-latex, non greek formula, you can find those in every book and blog, and if you're a beginner it will take some practice to be able to look at them and totally understand.

You are already familiar with the fact that we get an intercept and a slope, but how does that turn into a calculation to make a prediction? Linear regression is one of the easier ones to do by hand.

```
> lm(mpg ~ wt, data=mtcars)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Coefficients:
(Intercept)           wt
     37.285       -5.344
```

Our first model was predicting mpg using just weight, which looks like this;

```
intercept + slope*(value) = prediction
          =
   37.2851 + (-5.3445 )*wt = (mpg) ^
```

So, if it is this easy we should be able to get the same value as the predict function in R by using 2.0 (in thousand pounds) as our weight;

```
37.2851 + (-5.3445) * 2.0 = 26.5961
```

Easy huh? Lets do more;

```
3.0,4.0,9.0

37.2851 + (-5.3445) * 3.0 = 21.2516
   37.2851 + (-5.3445) * 4.0 = 15.9071
   37.2851 + (-5.3445) * 9.0 = -10.8154
```

Super easy huh?

So if we start adding more coefficients?

```
Coefficients:
             Estimate
(Intercept) 37.22727
Horsepower  -0.03177
Weight      -3.87783
---
```

$$(mpg)\ \hat{} = 37.22727 + (-3.87783)*Weight + (-0.03177)*Horsepower$$

Using the same variables from the prior section

```
   Horsepower <- c(100,150,200,395)

   Weight <- c(2.5,3.5,5.2,5.6)

37.22727 + (-3.87783)*Weight + (-0.03177)*Horsepower

   37.22727 + (-3.87783) * 2.5 + (-0.03177) * 100 = 24.35569

   37.22727 + (-3.87783) * 3.5 + (-0.03177) * 150 = 18.88936
```

```
37.22727 + (-3.87783) * 5.2 + (-0.03177) * 200 = 10.70855
```

```
37.22727 + (-3.87783) * 5.6 + (-0.03177) * 395 = 2.962272
```

If all of our math is correct these will be the same values from the prior section.

Lets back up for a moment and talk about the formula. Each coefficient can also be called the beta, and as you have learned you can have many, and can be denoted as follows. We did not use these specific values in the prior models, but i think you get the idea. The Y with the carrot symbol is called "y hat" and denotes a prediction estimate since we are using a sample.

"b zero" below will always be the intercept, everything after that is a coefficient, you can see that for 100 variables this could become quire a long formula. Keep in mind that while all of these may not have had a high p-vlaue and were eventually thrown out, they were all quantitative.

$$\hat{Y} = b_0 + b_1 + b_2 + b_3 + b_4 + b_5$$

$$\hat{Y} = b_0 Intercept + b_1 Displacement + b_2 \text{Horsepower} + b_3 \text{RearAxleRatio} + b_4 Weight + b_5 \text{QuarterMile}$$

$$\hat{Y} = 16.53357 + \mathbf{0.00872} * Displacement - \mathbf{0.02060} * Horespower + 2.01578 * RearAxleRatio - \mathbf{4.38546} * Weight + \mathbf{0.64015} * QuarterMile$$

To get ready for the next section we will need to learn a term called an indicator variable, this is what becomes of a qualitative/categorical/factor variable. For this example i will keep the formula short. Lets imagine we have a weight of 2000 pounds, 250 horsepower and a 6 cylinder engine. In the next blog post we will be using cylinder as categorical variable as it descries "which". R treats factors as indicators, meaning you will have a coefficient for all but the lowest factor in the model, the lowest is assumed to be zero and the result are built from that one, so here is what the model will look like;

$$\hat{Y} = b_0 + b_1 + b_2 + b_3 + b_4$$

$$\hat{Y} = b_0 \text{Intercept} + b_1 \text{Horsepower} + b_2 \text{Weight} + b_3 \text{IsCylinder6} + b_4 \text{IsCylinders8}$$

Now we cannot have all the coefficients turned on in this case, because we cannot have a vehicle with 4,6, and 8 cylinders, so we will have to perform the math with some of the indicator variables turned off. SO, if we have the variable, we multiply by 1, if we do not have the variable we multiply by zero.

Our Coefficients;

```
Coefficients:
             Estimate
(Intercept) 35.84600
Horsepower  -0.02312
Weight      -3.18140
Cylinders6  -3.35902
Cylinders8  -3.18588
```

Using our hypothetical car, Our Formula;

$$\hat{Y} = 35.84600 + (-0.02312) * \text{Horsepower} + (-3.18140) * \text{Weight} + (-3.35902)\ \text{IsCylinder6} * (1) + (-3.18588)\ \text{IsCylinders8} * (0)$$

$$\hat{Y} = 35.84600 + (-0.02312) * 250 + (-3.18140) * 2.0 + (-3.35902)\ \text{IsCylinder6} * (1) + (-3.18588)\ \text{IsCylinders8} * (0)$$

I am hoping, that mathematically that makes sense. In short a 4, 6, and 8 cylinder engine will all have different coefficients and since you can only have a car with one of those choices you must choose just one coefficient to use and turn off the other two you are not using. You will also see these referenced as dummy variables.

Section 3

INDICATOR OR DUMMY VARIABLES

So far we have been dealing with quantitative variables which ask how many or how much, the next is qualitative or categorical. Categorical usually asks which, and while it may be a number it would not make sense to perform math against it.

So, what do we have in the data set that is not math friendly and could be considered a category? Model could be, you cannot add or subtract and does answer the question of which.

Cylinders is, we only have three possible values and does describe a which or what.

VSengine has only two possible values, 0 or 1 indicating straight block engine config or v config.

TransmissionAM has only two possible values, automatic or manual.

Gears is also fixed, only three possible values.

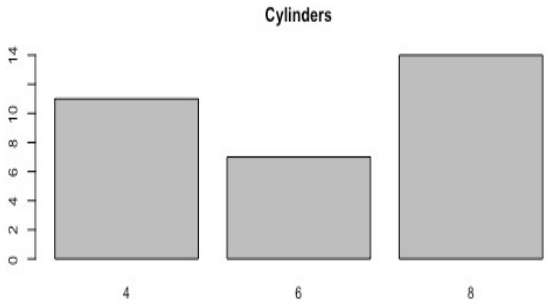
Carburetors is as well, there are 1,2,3,4,6,8 and it would not makes sense to perform any real math opeeration on carb.

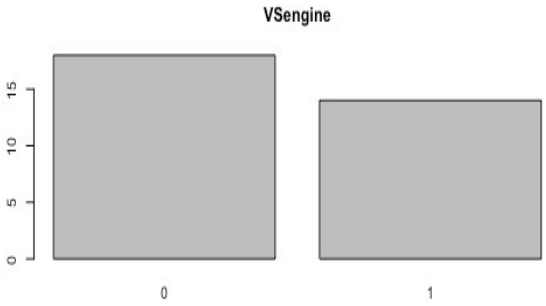
Great, so we have identified them, what do we do with them? Easy, tell R they are qualitative.

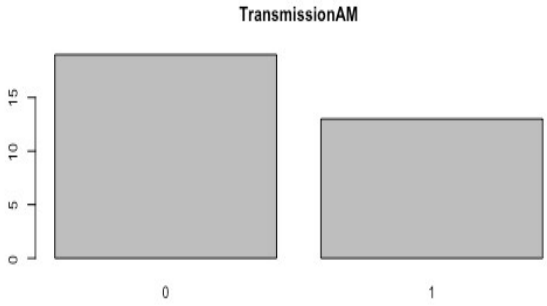
```
     # picking up where we left off, i will assume the mtcars is loaded
and column names have been changed.

     # create factors

     ColNames <-
c("Cylinders","VSengine","TransmissionAM","Gears","Carburetors")

     mtcars[ColNames] <- lapply(mtcars[ColNames], factor)



     summary(mtcars)

     str(mtcars)
```

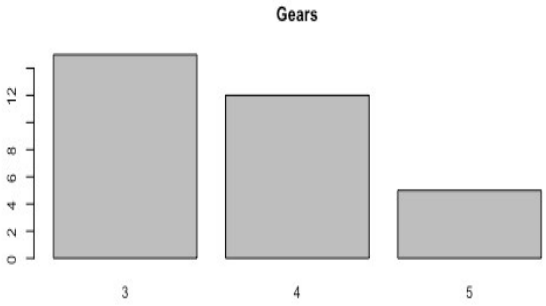
```
> str(mtcars)
'data.frame':   32 obs. of  13
 $ mpg           : num  21 21 2
 $ Cylinders     : Factor w/ 3
 $ Displacement  : num  160 160
 $ Horsepower    : num  110 110
 $ RearAxleRatio : num  3.9 3.9
 $ Weight        : num  2.62 2.
 $ QuarterMile   : num  16.5 17
 $ VSengine      : Factor w/ 2
 $ TransmissionAM: Factor w/ 2
 $ Gears         : Factor w/ 3
 $ Carburetors   : Factor w/ 6
 $ lp100k        : num  11.2 11
 $ Model         : chr  "Mazda
```

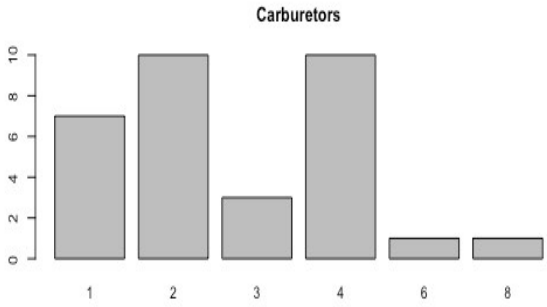
When you run summary and str you will see that the column types are now factors and indicate the number of levels. Summary will show the number of rows for each level. You will also notice that when you run base R graphics commands against factors you get slightly different behavior. Using plot below you will get a histogram.


Cylinders
0
2
4
6
8
10
14
4
6
8

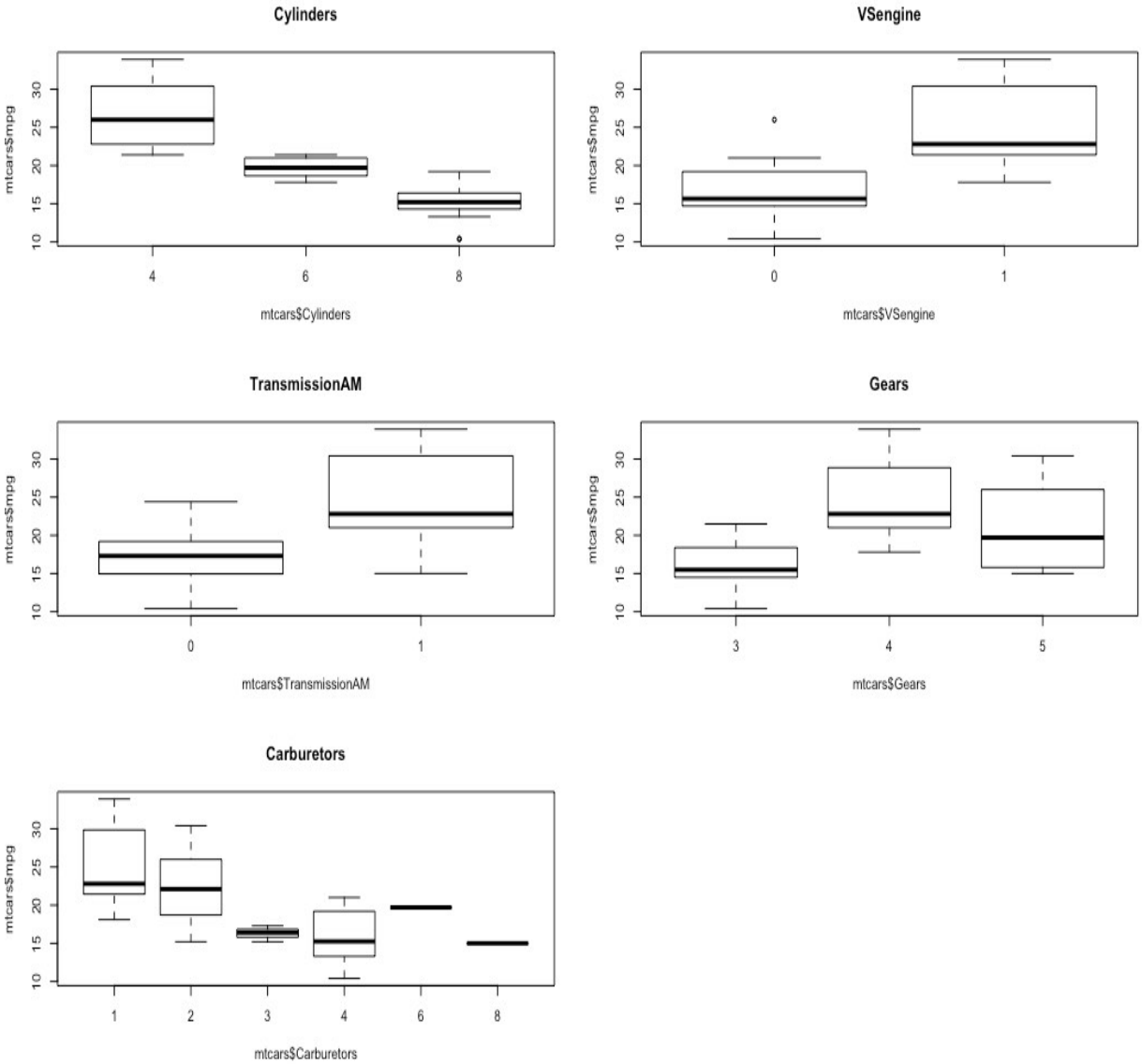

VSengine
0
5
10
15
0
1


TransmissionAM
0
5
10
15
0
1


Gears
0
2
4
6
8
12
3
4
5


Carburetors
0
2
4
6
8
10
1
2
3
4
6
8

For more plotting fun when you add the y axis to plot you will get a boxlot on a factor vs. a scatterplot.



In linear regression a factor is not treated the same as a quantitative variable, a factor actually becomes a thing called an indicator or dummy variable. As in, the coefficient for a 4 cylinder vs. a 6 cylinder vs. 8 cylinder can all be different but for a single calculation only one of them will be used. Also, each factor will show as an individual coefficient.

Take a look at the coefficients below;

```
mtcars.1 <- lm(mpg ~ Horsepower + Weight + Cylinders, data=mtcars)

summary(mtcars.1)
```

```
Call:
lm(formula = mpg ~ Horsepower + Weight + Cylinders, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.2612 -1.0320 -0.3210  0.9281  5.3947

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept) 35.84600    2.04102  17.563 0.00000000000000267 ***
Horsepower  -0.02312    0.01195  -1.934             0.063613 .
Weight      -3.18140    0.71960  -4.421             0.000144 ***
Cylinders6  -3.35902    1.40167  -2.396             0.023747 *
Cylinders8  -3.18588    2.17048  -1.468             0.153705
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.44 on 27 degrees of freedom
Multiple R-squared:  0.8572,    Adjusted R-squared:  0.8361
F-statistic: 40.53 on 4 and 27 DF,  p-value: 0.00000000004869
```

When a factor is added to the model each value in the factor will get a coefficient except the lowest, that one will have a coefficient of zero, so no need to actually add it and you can assume that if the value of the cylinder what we are trying to predict is 4, then none of the indicators will be turned on, or used in other words.

We have to perform the same diligence with a categorical variable as we do with a quantitative variable. While they may look funny in the model, in some packages when you ready to do a prediction we will have to create a column for every indicator, but not R. We will still need to look at the p-value to see if it is adding any value. However, one thing you must consider, if you keep one indicator variable from a factor you have to keep them all. For instance In the model mtcars.5 that was created above, Cylinders6 has a low p-value, but Cylinders8 has a p-value above .05 which we would typically throw out, but we cannot throw just one of them out, we have to keep them all or throw all of them out.

To show you how messy this can become lets add all of the categorical variables to the model.

```
Call:
lm(formula = mpg ~ Horsepower + Weight + Cylinders + VSengine +
    TransmissionAM + Gears + Carburetors, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0560 -1.3972  0.1046  0.7209  4.2525

Coefficients:
                Estimate Std. Error t value  Pr(>|t|)
(Intercept)     33.26496    5.62066   5.918 0.0000133 ***
Horsepower      -0.06125    0.03699  -1.656    0.1151
Weight          -2.21964    1.15988  -1.914    0.0717 .
Cylinders6      -2.07755    2.07939  -0.999    0.3310
Cylinders8       2.94014    4.84742   0.607    0.5517
VSengine1        2.17119    2.75804   0.787    0.4414
TransmissionAM1  1.73991    2.56492   0.678    0.5062
Gears4           0.52566    3.08339   0.170    0.8665
Gears5           2.54138    3.54365   0.717    0.4825
Carburetors2    -1.00834    2.06115  -0.489    0.6306
Carburetors3    -0.31245    2.88336  -0.108    0.9149
Carburetors4     0.27577    3.57353   0.077    0.9393
Carburetors6     1.09829    5.38720   0.204    0.8407
Carburetors8     2.95618    7.22200   0.409    0.6871
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.738 on 18 degrees of freedom
Multiple R-squared:  0.8802,    Adjusted R-squared:  0.7936
F-statistic: 10.17 on 13 and 18 DF,  p-value: 0.000008744
```

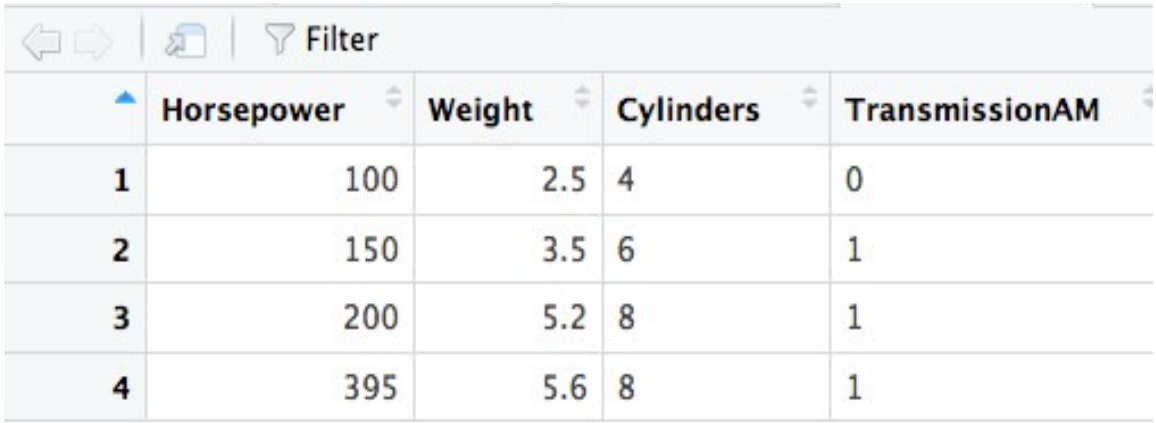
You will notice the model got a whole lot busier, and all the p-values are really bad except for weight! Even Horsepower is above .05 now. The very next blog post will get into how to deal with a model when you have this much variability, and we already know that adding and subtracting variables will improve then degrade the model depending on the order we do it. There is a better way! For now, i am going to jump to the end and show you the best model we can arrive at, then do a prediction.

First whats the least bad model?

```
Call:
lm(formula = mpg ~ Horsepower + Weight + Cylinders + TransmissionAM,
    data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9387 -1.2560 -0.4013  1.1253  5.0513

Coefficients:
                Estimate Std. Error t value          Pr(>|t|)
(Intercept)     33.70832    2.60489  12.940 0.000000000000773 ***
Horsepower      -0.03211    0.01369  -2.345           0.02693 *
Weight          -2.49683    0.88559  -2.819           0.00908 **
Cylinders6      -3.03134    1.40728  -2.154           0.04068 *
Cylinders8      -2.16368    2.28425  -0.947           0.35225
TransmissionAM1  1.80921    1.39630   1.296           0.20646
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.41 on 26 degrees of freedom
Multiple R-squared:  0.8659,    Adjusted R-squared:  0.8401
F-statistic: 33.57 on 5 and 26 DF,  p-value: 0.0000000001506
```

Recall what i said about indicator variables, if one is good and one is bad you will have to decide to keep all of it or none of it. In this case we are keeping Cylinders. TransmissionAM1, has a high p-value but if we remove it, something else will spike up, so its a balance of least worst. Adjusted R-square is high, p-value for model is low, f-statistic is far from one. Residual standard error of 2.41 relative to intercept is pretty low, which is good.

Now, for the prediction. This is where indicator variables become annoying in most software packages. R however, is extremely friendly and will handle the pivot and indicator variable for you behind the scene. Packages like SPSS will require you to create the indicator variable manually if a factor is present. Our data frame going in will look like this;


Filter

	Horsepower	Weight	Cylinders	TransmissionAM
1	100	2.5	4	0
2	150	3.5	6	1
3	200	5.2	8	1
4	395	5.6	8	1

```
Horsepower <- c(100,150,200,395)

Weight <- c(2.5,3.5,5.2,5.6)

Cylinders <- as.factor(c(4,6,8,8))

TransmissionAM <- as.factor(c(0,1,1,1))


newcars = data.frame(Horsepower,Weight,Cylinders,TransmissionAM)

predict(mtcars.3,newdata=newcars,interval="confidence")
```

```
> predict(mtcars.3,newdata=newcars,interval="confidence")
       fit       lwr      upr
1 24.25531 21.624384 26.88623
2 18.93087 16.282136 21.57961
3 13.94846  9.468533 18.42839
4  6.68839  1.249843 12.12694
```

The true test as always, much like any software is to test it over and over again. Make sure the training data you use represents the real world data that the model will be used to predict.

To give you an example the last case in the newcars dataframe, 395 horsepower, 5600 pounds, 8 cylinder engine, and automatic transmission has a predicted mpg of 6.7 but may be as low as 1.2mpg and as high as 12.1mpg. I can tell you that is not correct, that is my current vehicle, and gets around 15-20mpg depending on highway or city. But the model was trained with a small set of data form 1974, it cannot be used for anything outside of that time frame and car characteristics.

Section 4

MORE INDICATORS

More variables! For this one we are going to add all of the variables in their correct form in the data frame as qualitative or quantitative.

If you have not been keeping up with progress on the data set lets get it loaded and factors converted.

```
    data(mtcars)

    #rename the columns to something slightly more
meaningful

    names(mtcars) <- c("mpg", "Cylinders", "Displace-
ment","Horsepower", "RearAxleRatio", "Weight", "Quar-
terMile", "VSengine", "TransmissionAM", "Gears",
"Carburetors")


    # Create liters per 100 kilometers column

    mtcars$lp100k <- (100 * 3.785) / (1.609 *
mtcars$mpg)


    #fix the row names, i want them to be a column

    mtcars$Model <- row.names(mtcars)

    row.names(mtcars)

    ColNames <-
c("Cylinders","VSengine","TransmissionAM","Gears","Car
buretors")

    mtcars[ColNames] <- lapply(mtcars[ColNames],
factor)
```

So the data engineering, though minor, is done, lets go ahead and created a model with everything and see what happens. You can, if you like, run some analysis on all of the new columns using plot to see how the data looks, and if their appear to be any relationships.

The following is a bit ridiculous and i would only advise running this on smaller datasets, but its fun. On my machine, a several year old Mac takes about 1 minute torun, so be patient, it will give you status update in the console as it runs. I will let you review it on your own instead of putting the graphic here. Also note that if you run this before the categorical columns are converted to factors you will get correlations for those columns.

```
library(ggplot2)

library(GGally)

ggpairs(mtcars[1:12], aes(alpha = 0.5))
```

There are a couple of ways to pass the column names into the model, we need the following in the model "mpg", "Cylinders", "Displacement", "Horsepower", "RearAxleRatio", "Weight", "QuarterMile", "VSengine", "TransmissionAM", "Gears", "Carburetors", so we can list them individual as we have in the past, or cheat by slicing the dataframe. If we did not have the slice it would include model and lp100k which we do not want for this.

```
mtcars.1 <- lm(mpg ~ ., data=mtcars[1:11])

summary(mtcars.1)


#Or

#mtcars.1<- lm(mpg ~ Cylinders+ Displacement+
Horsepower+ RearAxleRatio+ Weight+ QuarterMile+ VSen-
gine+ TransmissionAM+ Gears+ Carburetors, data=mtcars)
```

```
Call:
lm(formula = mpg ~ ., data = mtcars[1:11])

Residuals:
    Min      1Q  Median      3Q     Max
-3.5087 -1.3584 -0.0948  0.7745  4.6251

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     23.87913   20.06582   1.190   0.2525
Cylinders6      -2.64870    3.04089  -0.871   0.3975
Cylinders8      -0.33616    7.15954  -0.047   0.9632
Displacement     0.03555    0.03190   1.114   0.2827
Horsepower      -0.07051    0.03943  -1.788   0.0939 .
RearAxleRatio    1.18283    2.48348   0.476   0.6407
Weight          -4.52978    2.53875  -1.784   0.0946 .
QuarterMile      0.36784    0.93540   0.393   0.6997
VSengine1        1.93085    2.87126   0.672   0.5115
TransmissionAM1  1.21212    3.21355   0.377   0.7113
Gears4           1.11435    3.79952   0.293   0.7733
Gears5           2.52840    3.73636   0.677   0.5089
Carburetors2    -0.97935    2.31797  -0.423   0.6787
Carburetors3     2.99964    4.29355   0.699   0.4955
Carburetors4     1.09142    4.44962   0.245   0.8096
Carburetors6     4.47757    6.38406   0.701   0.4938
Carburetors8     7.25041    8.36057   0.867   0.3995
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 2.833 on 15 degrees of freedom
Multiple R-squared:  0.8931,    Adjusted R-squared:  0.779
F-statistic:  7.83 on 16 and 15 DF,  p-value: 0.000124
```

One thing you will notice pretty quickly is using our standard of < .05 of a p-value, that nothing is significant. While we know for a fact that with less variables we do have some significant variables.

We have demonstrated that adding and subtracting a variable one at a time will change the model, sometimes for the better or the worse and that is what we will do here, but instead of creating each model plus or minus one variable we will let the software do it for us. Stepwise regression which is a form of subset selection. ISLR, page 204 defines Subset Selection as "This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.", and i really like that definition.

Two main forms of stepwise regression, f**orward stepwise** selection and **backward stepwise** selection.

Forward Stepwise selection starts with a model containing no predictors and then adds the predictors to the model one at a time. ISLR page 207 demonstrates the forward stepwise model as a variable is added and kept if it has the lowest RSS and highest R-squared.

So lets try it!

First we will try out a Forward Stepwise, scope is required and you will need to pass in each variable you want to be tested. Just as a normal regression the best and final model will be stored in cars.fwd1 which you can run summary on to get the results.

```
    ##stepwise regression forward

    cars.fwd1 <- step(lm(mpg~1,data=mtcars), direction
= "forward",

                      scope=(~Cylinders+ Displacement+
Horsepower+ RearAxleRatio+ Weight+ QuarterMile+ VSen-
gine+ TransmissionAM+ Gears+ Carburetors))
```

```
summary(cars.fwd1)
```

That was easy, and fast! Also notice that it kept a couple of variables that had a p-value > .05. This is because the benefit of the variable being there outweighed removing it, lower r-square and likely higher residual error. Also remember that since we are using indicator variables, or dummy variables like Cylinders and transmission, if you keep one of the indicators you have to keep them all, they are still on column of data in the original dataset.

```
Call:
lm(formula = mpg ~ Weight + Cylinders + Horsepower + TransmissionAM,
    data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9387 -1.2560 -0.4013  1.1253  5.0513

Coefficients:
                Estimate Std. Error t value          Pr(>|t|)
(Intercept)     33.70832    2.60489  12.940 0.000000000000773 ***
Weight          -2.49683    0.88559  -2.819           0.00908 **
Cylinders6      -3.03134    1.40728  -2.154           0.04068 *
Cylinders8      -2.16368    2.28425  -0.947           0.35225
Horsepower      -0.03211    0.01369  -2.345           0.02693 *
TransmissionAM1  1.80921    1.39630   1.296           0.20646
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.41 on 26 degrees of freedom
Multiple R-squared:  0.8659,    Adjusted R-squared:  0.8401
F-statistic: 33.57 on 5 and 26 DF,  p-value: 0.0000000001506
```

So that was fun, what about backward stepwise? The syntax is slightly different for backward stepwise, no scope command and you pass everything into lm(). Same principle as forward but in reverse.

```
cars.back1 <- step(lm(mpg ~ Cylinders+ Displacement+ Horsepower+ RearAxleRatio+ Weight+ QuarterMile+ VSengine+ TransmissionAM+ Gears+ Carburetors,data=mtcars), direction = "backward")

summary(cars.back1)
```

In this case, which does not happen every time, you can see that the same model was generated for forward and backwards. So it is reasonable to assume based on the data we have, this is the best we can do.

```
Call:
lm(formula = mpg ~ Cylinders + Horsepower + Weight + TransmissionAM
    data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-3.9387 -1.2560 -0.4013  1.1253  5.0513

Coefficients:
                Estimate Std. Error t value          Pr(>|t|)
(Intercept)     33.70832    2.60489  12.940 0.000000000000773 ***
Cylinders6      -3.03134    1.40728  -2.154           0.04068 *
Cylinders8      -2.16368    2.28425  -0.947           0.35225
Horsepower      -0.03211    0.01369  -2.345           0.02693 *
Weight          -2.49683    0.88559  -2.819           0.00908 **
TransmissionAM1  1.80921    1.39630   1.296           0.20646
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.41 on 26 degrees of freedom
Multiple R-squared:  0.8659,    Adjusted R-squared:  0.8401
F-statistic: 33.57 on 5 and 26 DF,  p-value: 0.0000000001506
```

Its fun when you find something new, and in R that can happen a LOT. MS SQL was frustrating in the fact that back in the day you would wait years for a single

feature to come out then another decade for MS to get it together and finish the feature. Column store was a great example of taking years to get a feature released incrementally over several versions. Keep in mind CCI(Column Store) is my absolute favorite thing ever done in SQL, and it happens to be in Hadoop as well as ORC and Vectorization, same guy pm'd and wrote both in case you are interested. Open source does not suffer from this so you can frequently get new features added or entirely new packages that solve problems and give you the ability to explore data and models in ways you never considered. I stumbled upon one such package recently.

OLSRR appears to have been released recently as of this writing, so i am pleased it did not live out in the wild for too long before i found it. The above section having been on forward and backward stepwise regression, OLSRR is the perfect continuation of this section. Not surprisingly the author used the exact same data set i did for his samples, mtcars. With that there is nothing i can really add. Here is the link to the Variable Selection Methods.

I like the detail that the package goes into when explaining stepwise. Stepwise is awesome, but one thing i skipped over that is the dimensionality problem that can occur with stepwise. I used a total of 10 variables in the trained model, that equates to 2^10 possible variations, which is a lot, and its super slow to calculate that many models. What if your model had 100 variables and 100,000 rows in the training set? You are not going to do that on your laptop, but this will happen and you will solve it, scale out is your friend in this case.

The more you read, the more you experiment and follow along, and the more data you try these features out on the better you will get and the better you will understand.

Chapter 8

REGRESSION WITH REAL DATA

You were warned! If you have ever sat in on a single data science talk you probably learned that the data engineering phase of a project will take 80% of your time. This is an anecdotal number, but my experience to date seems to reenforce this number. On average it will take about 80% of whatever time you have to perform the data engineering tasks. This blog is going to likely prove that, though you will not have had to do the actual work, just copy and paste the code and run it. You will however get an idea of the pain in the ass you are in for.

I am going to approach this post and the scripts exactly the way i came to the dataset, so i will remove rows, then learn something new and remove some more rows or maybe add them back. I could simply put the data engineering at the top, and not explain anything but that is not how the world will work. The second, third, forth, one hundredth time you do this you will have the scripts and knowledge. With any new dataset, curiosity and exploration will make the process of modeling much easier.

Section 1

DATA ENGINEERING

Here is the thing; A SQL Pro can typically do this work in T-SQL lickety-split, in R or Python you may be learning as you go. I always seem to know what i want to accomplish, but R frequently refuses to bend to my will. I will show you more than one way to do something, if i can find a solution using a tidyverse solution i will, but as of now i find dplyr syntax unfriendly and cumbersome. I look forward to the day that i find it as useful as the folks pushing it down my throat as the end all to data engineering. But, soon i will be going head first full time Python, that day may have to wait.

In continuing with the MPG problem i will use a 2018 EPA data set. Here is the csv i am using. Here is the link to the EPA website where more info can be found. This is the link to the pdf containing column descriptions and values, the data dictionary, if you will. It appears that it has not been updated since the 90's, so i am not sure if i have the wrong file or they really have not updated it since the 90's. You can derive some of the column values and meanings, but not all, you may need to do a bit of spelunking. I do have a copy of the data file i am using on my github site, just in case the file they publish gets updated and the wacky values you are about to see get fixed. I already have a pretty good idea of what columns to use since i want to keep this as close to the mtcars problem as possible, but you will soon find out there is more than one answer to MPG.

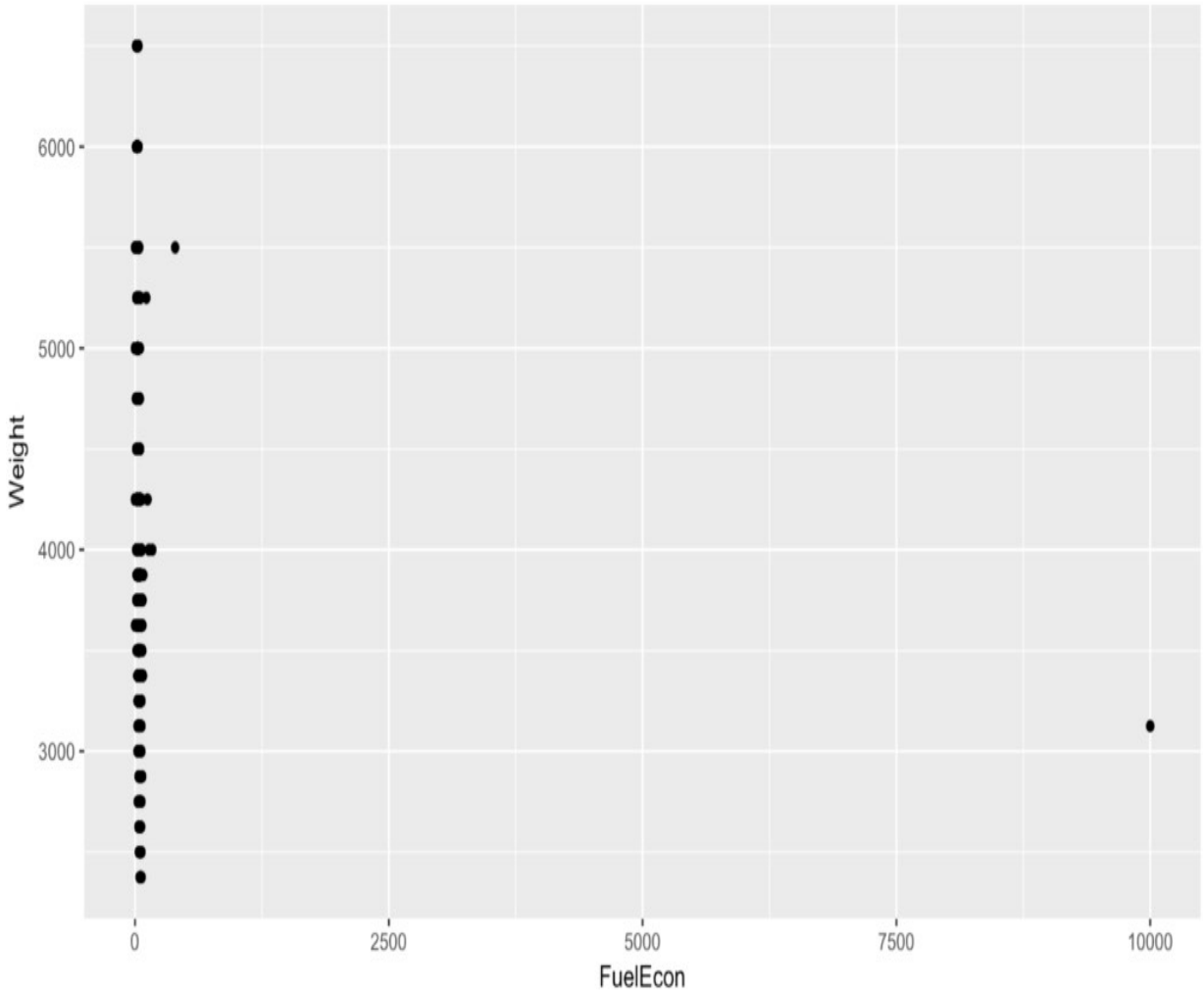
```
     # If you do not want to download the file, you can read it di-
rectly from the source into a dataframe.

     # epa <-
read.csv("https://www.epa.gov/sites/production/files/2017-10/18tst-
car.csv", stringsAsFactors=FALSE)


     #If the above does not work get it from my github site,
```

```
#
https://raw.githubusercontent.com/sqlshep/SQLShepBlog/master/data/18tstcar.csv


# or

# Set working dir if you download this locally

getwd()

setwd("/Users/AwesomeUser/data")


epa <- read.csv("18tstcar.csv",stringsAsFactors=FALSE)
```

Never be afraid to change column names while you are learning if you need to, however, if you are working with others or have a standard data dictionary for a production system be very careful, this is also a very good way to piss everyone off. I will change the names in this dataframe to come closer to what we were using in the last few mtcars examples.

```
# there are several ways to rename columns, here are three
ways to do it,

# choose your fav and stick with it



names(epa)[12] <-paste("Cylinders")

names(epa)[46] <-paste("FuelEcon")


colnames(epa)[colnames(epa)=="X..of.Gears"] <- "Gears"

colnames(epa)[colnames(epa)=="Axle.Ratio"] <- "AxleRatio"


library(dplyr)

epa <- dplyr::rename(epa, HorsePower = "Rated.Horsepower")
```

```
     epa <- dplyr::rename(epa, Weight =
"Equivalent.Test.Weight..lbs..")

     epa <- dplyr::rename(epa, Model =
"Represented.Test.Veh.Model")
```



The data exploration part of the project is very important, look at everything.

In case you are wondering when the data engineering shit show begins, its now. We have weight and mpg and a car that claims to get 10,000 mpg. My best advice is to treat it like a game, find and conquer weird data.

Now you have to determine what vehicle gets 10,000 mpg and is it cool enough for you to buy? No, i mean should you remove the row, fix the value, leave the value, and find out why you have a value that is so wrong. Bad data and missing data is something that has to be dealt with very strategically.

The following will produce any row with mpg > 9000 and the three rows for the Chevy Sonic, make sure you can isolate it.

```
filter(epa,   FuelEcon > 9000)
```

```
epa[epa$Test.Vehicle.ID == "184HV863DA" ,]
epa[epa$Test.Vehicle
```

According to Fuel Economy website the mpg should be 30 combined, so we can fix it by script or fix the original source. But, much like SQL be careful. In addition to wiping out data you did not mean to, once you make a modification it needs to be reproducible. If you need to go through 25 steps to modify a dataset do you want to do it one row at a time in excel or text editor? A better way is to script everything so it is easily reproducible and fast. In a given day on the data engineering work i will reload then main data file and run all of my modifications several times a day to make sure everything is working and reproducible. Don't assume you will remember everything if you need to start over, write it down, write it down in scripts.

Notice the total rows in the dataframe, and how many NA's (no value) we have. Then overwrite the dataframe OR or create a new one with everything less than 9999 mpg.

```
summary(epa$FuelEcon)
```

```
#  > summary(epa$FuelEcon)
#  Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
```

```
#  0.00    24.40    30.90    35.86    40.30 10000.00       12
```

```
NROW(epa)
```

```
#> NROW(epa)
# [1] 3509
```

```
epa <- (filter(epa, FuelEcon < 9999))
```

Notice below we have 13 less rows, the 12 NA's are gone and the 10,000mg car is gone. In my case this is fine, i am okay with deleting the 12 NA rows, but in the real world you may need to decide how to handle them? Why is there data missing? Do you need to leave the rows and fix the missing data issue either with imputation or actually getting the data? What method should you use to do this? In my case the first problem was bad data, so why do i have a car with 10,000 mpg? All of these need to be addressed by the data domain expert. In the end the answer may be to delete the row, but at some point this too will become a problem, i have worked with customers where 98% of the data had to be thrown out for a variety of reasons. It is possible throw out so much data that you no longer have a project, and this is a real problem you will encounter some day.

```
summary(epa$FuelEcon)
```

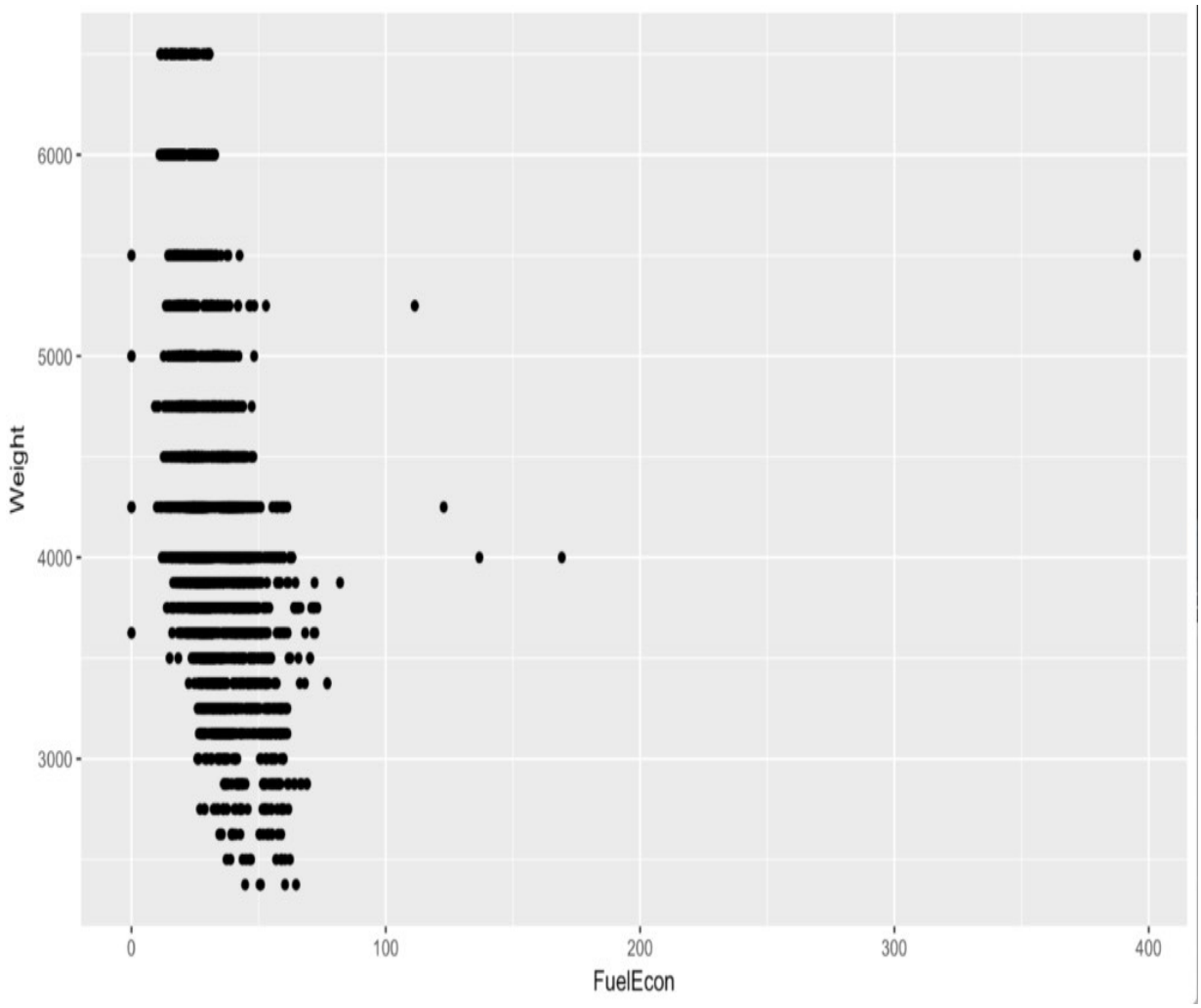
```
#  > summary(epa$FuelEcon)
#  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#  0.00   24.40   30.85   33.01   40.28  395.40
```

```
NROW(epa)
# > NROW(epa)
```

```
# [1] 3496
```

Lets look at our plot again.

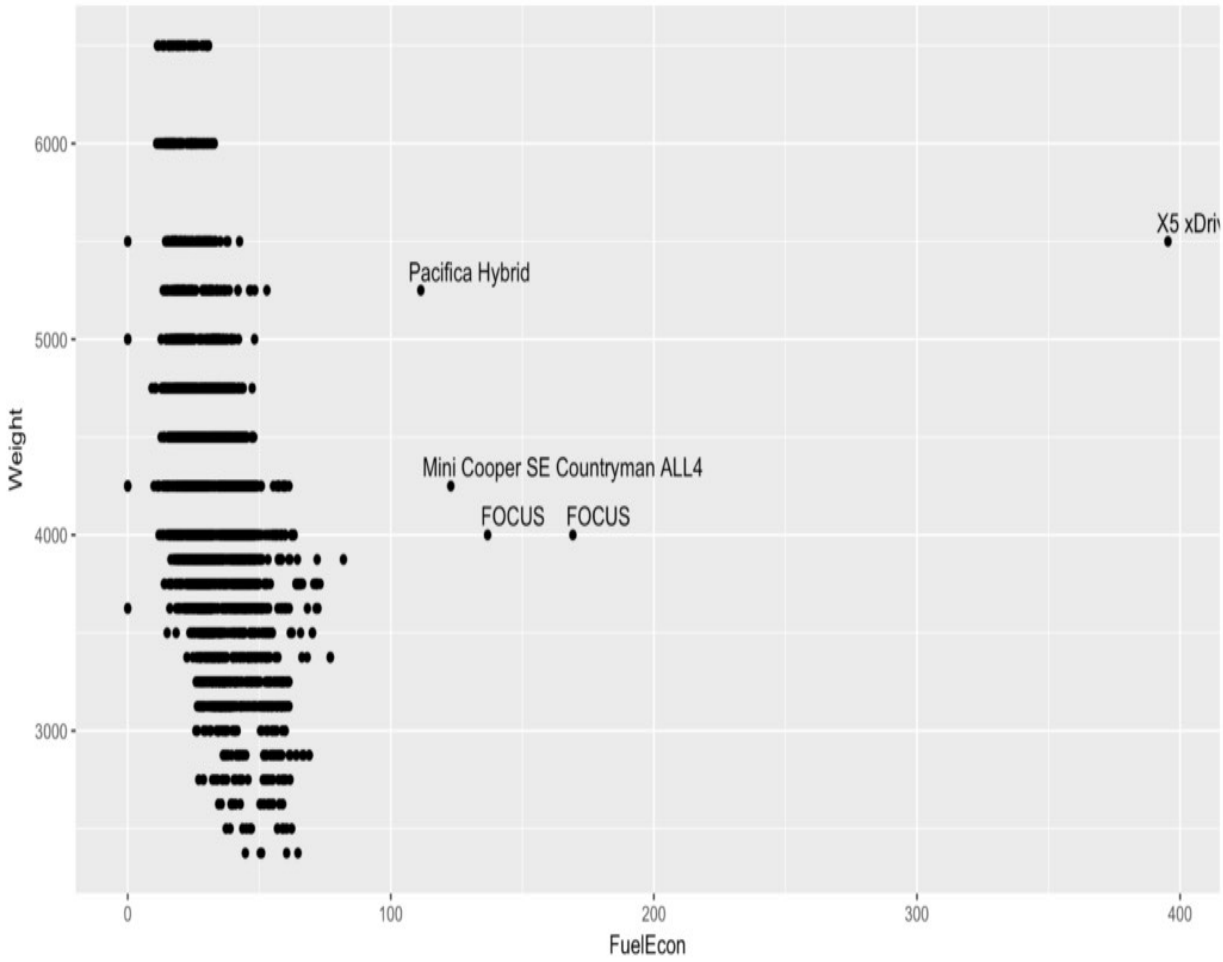
```
ggplot(epa,aes(FuelEcon,Weight))+
  geom_point()
```



Well, its better but now i have what are clear outliers and one in the 400 mpg range. Now, i once had a 3000 pound diesel car and it got no

where near 400 mpg, so i am going to assume this is either bad data or a Tesla. Lets add some labels to just the outliers.

```
     ggplot(epa, aes(FuelEcon,Weight)) +

       geom_point() +

       geom_text(data=subset(epa, FuelEcon >
90),aes(FuelEcon,Weight,label=Model), vjust=-.5, hjust=.10)



     # you can also use the following code to find more outliers,
though technically,

     # an outlier is > 2 standard deviations away from the mean,
see if you can calculate that.



     #filter(epa, FuelEcon > 100)

     #epa[epa$FuelEcon > 100
,c("Represented.Test.Veh.Make","Model","Test.Vehicle.ID","FuelEcon"
)]
```



Nope its a BMW X5 xDrive40e, and its not all electric, its hybrid and it does not get 395, its more around 24 - 56 mpg depending on "mode", so, bad data. From looking at the file and the codes in the file, it does not appear we have a way to identify a hybrid or all electric car with a binary. However, the EPA requires tests that are unique to electrics and hybrids. In the "Test Procedure Description" column you will find "Charge Depleting Highway" and "Charge Depleting UDDS" values, as well as "Test Fuel Type Description" of Electricity, though, a hybrid may also use tier 2 gasoline, so it is not mutually exclusive, meaning it can be tested for charge depletion and run on gasoline too. So in the interest of a petrol only mpg data set i think i will delete the electric and hybrid cars. Keep in mind, these could be kept in and used as an indicator/dummy variable to. If you were to used Hybrid as a variable it would likely have a pretty significant coeffi-

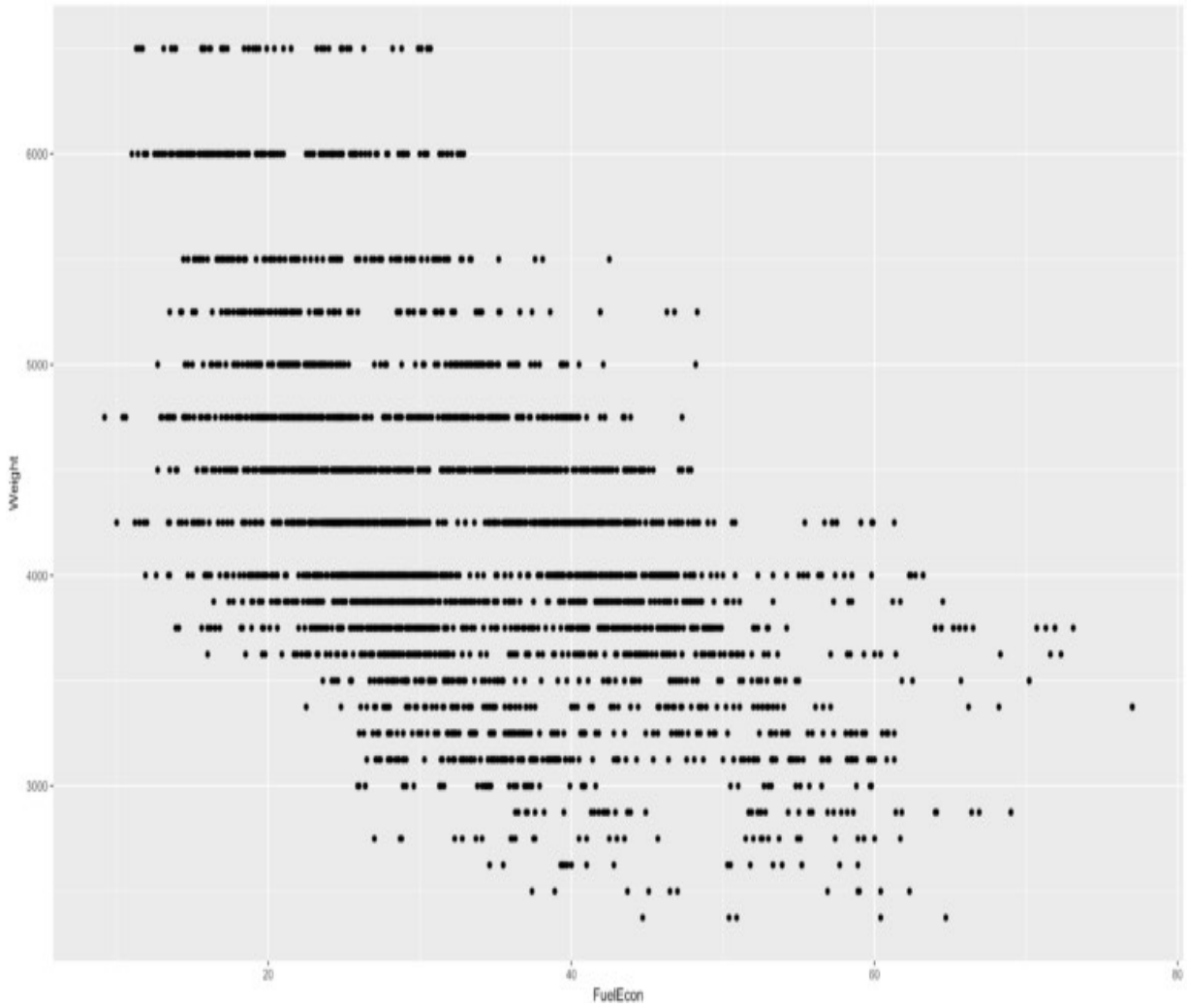
cient. It may be fun to take just the hybrids and create a separate model completely. Never be afraid to do that if necessary.

The Cruze is not hybrid but the data set claims 77mpg, which doing a little research even the best Diesel Cruze reports are around 55mpg and there is anecdotal evidence of hyper mile-ing to get up to 70, but i am sure the EPA did not do that. So, what do we do? I am leaving the Cruze, because for me to be critical of one and not all would be unfair. I am going to remove the electrics and hybrids and see where we end up.

```
View(filter(epa,   Test.Procedure.Description == "Charge Depleting Highway" | Test.Procedure.Description == "Charge Depleting UDDS"))
```

```
epa <- filter(epa,   Test.Procedure.Description != "Charge Depleting Highway" & Test.Procedure.Description != "Charge Depleting UDDS")
```

```
ggplot(epa,aes(FuelEcon,Weight))+
  geom_point()
```



This looks much nicer, and gets our dataset in a little better shape.

Notice the weights this time, they all seem to be rather perfect. The weight of a vehicle is very dynamic from vehicle to vehicle. You can run summary(as.factor(epa$Weight)) and see that all of the vehicles were rounded to within 250lbs of actual weight. Which is interesting because this opens the possibility of using this as a factor or an indicator variable in the regression model. But, we will deal with weight shortly.

Lets take a closer look at a couple of cars.

```
View(filter(epa,   Represented.Test.Veh.Make == "BENTLEY" &
Model == "Continental GT" ))
```

You will notice that we have multiple rows for each car, if you have not realized that look a little closer. Each row represents a different EPA test, and there are at least 5 test run on each vehicle. You can learn more about the process of EPA testing here. There are a couple of ways to approach this, you can leave all of the tests in, make the test a factor and add it to the model so that you have to indicate which test you want predictions from, which is what i will do. You could also split out the test into different models, i may do that later to see if it makes a difference in model accuracy.

The following will display the number of cars in each test, it is clearly imbalanced.

```
table(epa$Test.Procedure.Description)
```

I want to be cognizant of the data i am removing, that being said this is still a toy training set just for fun. We may need to remove rows since a higher number of rows of one vehicle type can skew the model toward that type of vehicle performance. For instance, if i have 100 vehicles in the training data set that get 15mpg and 10 that get 40mpg, and that does not represent real world, the model will be heavily influenced by the 100 vehicles that get 15 mpg thus skewing all predictions. I would want the model to be as accurate as possible for the 40mpg as it is for the 15mpg, now, maybe the data will exist to help differentiate that, like and 8 cylinder engine vs. a three cylinder, and a weight of 8000 pounds vs 2000 pounds, but then again maybe not... In this dataset with Vehicle.ID == "53XECSE012" the only significant difference is the fuel, E85 Ethanol and gasoline. So what would be the appropriate thing to do? The Suburban has 58 rows, and the Civic, Vehicle.ID == "EGAA1M" has 40 rows. In the real world, this may need to be dealt with.

The following code will show the highest number of tests per vehicle, though you can also see that there is a minor configuration change for each

test though perhaps not so significant that it should result in a training row for a model...? This is where the data domain expert is useful in helping decide if the vehicles with minor changes should be included or excluded.

```
k<-table(epa$Test.Vehicle.ID)
head(sort(k,decreasing = TRUE))

View(filter(epa, epa$Test.Vehicle.ID == "53XECSE012"))

k1<-filter(epa, epa$Test.Vehicle.ID == "53XECSE012")
table(k1$Model,k1$Test.Fuel.Type.Description)
table(k1$Test.Fuel.Type.Description,round(k1$FuelEcon))
```

I do want to remove the police interceptors.

```
# Police
View(filter(epa, epa$Police...Emergency.Vehicle. == "Y"))
table(epa$Police...Emergency.Vehicle.)
epa <- filter(epa,epa$Police...Emergency.Vehicle. != "Y" )
```

Well, that was awesome, we dropped a few rows but still have a long way to go. If you have browsed the data you will see that there are still a lot of test for a single vehicle that seem very very close to each other, what wee need is a good old fashion distinct to get the almost duplicates out. Lucky for us there is one, distinct from dplyr will allow us to specify the columns we want distinct tested against while still allowing us to present the entire row. As a SQL guy all i can say is where have you been all my life.

```
# Review distinct
```

```
View(distinct(epa,Vehicle.Manufacturer.Name
                ,Veh.Mfr.Code
                ,Represented.Test.Veh.Make
                ,Model
                ,Test.Vehicle.ID
                ,Test.Veh.Displacement..L.
                ,Drive.System.Description
                ,AxelRatio
                ,Weight
                ,Test.Fuel.Type.Cd
                , .keep_all = TRUE))


# Reload dataframe with less values.
epa <- (distinct(epa,Vehicle.Manufacturer.Name
                ,Veh.Mfr.Code
                ,Represented.Test.Veh.Make
                ,Model
                ,Test.Vehicle.ID
                ,Test.Veh.Displacement..L.
                ,Drive.System.Description
                ,AxelRatio
                ,Weight
                ,Test.Fuel.Type.Cd
                , .keep_all = TRUE) )
```

My dataset is now down to 1/3 of what it was, recall what i said about throwing all of your data out, we won't get that far but i think you see how easy it can happen.

At some point, or any point we could start throwing columns that will not be part of the training set. Remember that most of the columns are emissions test results and all we are looking for is mpg. Now what may be fun later on is to take this same dataset and predict what our emissions could be based on a configuration. So far we have kept the entire 67 variables, lets start dumping them.

So, what do we want to keep?

While it may not matter at this point, but, i am going to create a dew data frame to work with, epaMpg.

```
# trim down the columns for the mdel training set

View(epa[c(4,5,10,11,12,14,15,16,18,22,23,34,35,36,37,46)])


epaMpg <-
epa[c(4,5,10,11,12,14,15,16,18,22,23,34,35,36,37,46)]


names(epaMpg)
```

Lets convert the columns to factors that need to be factors.

```
epaMpg$Cylinders <- as.factor(epaMpg$Cylinders)

epaMpg$Tested.Transmission.Type.Code <-
as.factor(epaMpg$Tested.Transmission.Type.Code)

epaMpg$Gears <- as.factor(epaMpg$Gears)

epaMpg$Test.Procedure.Cd <-
as.factor(epaMpg$Test.Procedure.Cd)

epaMpg$Drive.System.Code <-
as.factor(epaMpg$Drive.System.Code)
```

```
     epaMpg$Test.Fuel.Type.Cd <-
as.factor(epaMpg$Test.Fuel.Type.Cd)
```

For fun, if you want to do something somewhat silly, run ggpairs against the dataset and review the results, notice the difference between factors and quantitative columns.

```
     ggpairs(epaMpg[c(3:6,14)])

     ggpairs(epaMpg[c(8:12,14)])
```

This is one version of the data engineering, this process is iterative, and can go on for days, even for a small dataset. A question to answer is essential, and a data domain expert is essential to help make sense of the data.

Section 2

TECHNIQUES FOR BUILDING A MODEL

From the last post, we have a dataset, now lets do something with it.

To make life easier on you i am providing the data i am using, its already engineered, more could be done, i still have more than one test per vehicle per procedure, but i'm leaving it for now. You can if you want, narrow it down even further by selecting just one test procedure type per vehicle. See last post to get the details on what that means.

So, lets load the data, convert factors.

```
epaMpg <- read.csv("epaMpg.csv",stringsAsFactors=FALSE)


summary(epaMpg)


epaMpg$Cylinders <- as.factor(epaMpg$Cylinders)

epaMpg$Tested.Transmission.Type.Code <-
as.factor(epaMpg$Tested.Transmission.Type.Code)

epaMpg$Gears <- as.factor(epaMpg$Gears)

epaMpg$Test.Procedure.Cd <- as.factor(epaMpg$Test.Procedure.Cd)

epaMpg$Drive.System.Code <- as.factor(epaMpg$Drive.System.Code)

epaMpg$Test.Fuel.Type.Cd <- as.factor(epaMpg$Test.Fuel.Type.Cd)


epaMpg <- epaMpg[!is.na(epaMpg$Cylinders),]
```

You should have in the neighborhood of 1034 rows if you used my dataset, yours may vary. Next, lets stuff it into a model with abandon and see what we get.

```
     # scipen will just blow out the  scientific notation, i like to see
the number.

     options(scipen = 999)


     # this should be familiar by now.

     epaMpg.1 <- lm(FuelEcon ~ HorsePower + Cylinders + Tested.Transmis-
sion.Type.Code + Gears + Drive.System.Code + Weight + AxleRatio +
Test.Procedure.Cd + Test.Fuel.Type.Cd,data=epaMpg)


     summary(epaMpg.1)
```

We have a pretty significant model now, lots of data, lots of numbers, Adjusted R2 is 83.45% which is not bad at all, Residual standard error at 3.863 is relatively low as it is in units of MPG.

We also have a lot of variables that look like they need to be ditched, so we could go through and drop one, rerun the model check the p-values, drop another, rerun the model check the p-values, drop another rerun the model check the p-values. See a pattern developing there? There is a better way.

```
Call:
lm(formula = FuelEcon ~ HorsePower + Cylinders + Tested.Transmission.Type.Code +
    Gears + Drive.System.Code + Weight + AxleRatio + Test.Procedure.Cd +
    Test.Fuel.Type.Cd, data = epaMpg)

Residuals:
    Min      1Q  Median      3Q     Max
-12.269  -2.135  -0.362   1.679  35.642

Coefficients:
                                 Estimate  Std. Error t value            Pr(>|t|)
(Intercept)                    57.0616606   2.8499184  20.022 < 0.0000000000000002 *
HorsePower                     -0.0273784   0.0022750 -12.034 < 0.0000000000000002 *
Cylinders4                     -1.3473339   0.8650077  -1.558             0.119647
Cylinders5                     -3.5685099   2.2433454  -1.591             0.111993
Cylinders6                     -2.9190855   1.0066331  -2.900             0.003816 *
Cylinders8                     -2.1139974   1.2349966  -1.712             0.087256 .
Cylinders10                    -3.3155847   1.7083699  -1.941             0.052566 .
Cylinders12                    -2.0297250   1.6479356  -1.232             0.218361
Cylinders16                    19.2303649   4.1704683   4.611 0.00004527875048301 *
Tested.Transmission.Type.CodeAM 5.0355407   0.7930823   6.349 0.00000000328376418 *
Tested.Transmission.Type.CodeAMS 2.1633027  0.6172016   3.505             0.000477 *
Tested.Transmission.Type.CodeCVT 9.1913362  1.8996749   4.838 0.00001516597284493 *
Tested.Transmission.Type.CodeM  1.5955652   0.5128438   3.111             0.001916 *
Tested.Transmission.Type.CodeSA 0.4532099   0.3584960   1.264             0.206455
Tested.Transmission.Type.CodeSCV 2.6946730  1.2725912   2.117             0.034468 *
Gears4                          1.2057656   3.3337331   0.362             0.717663
Gears5                         -2.3410162   1.9100833  -1.226             0.220636
Gears6                         -0.9215991   1.8011557  -0.512             0.608995
Gears7                          0.3294529   1.7971839   0.183             0.854587
Gears8                          1.8205785   1.8052673   1.008             0.313469
Gears9                          0.3494847   1.8469646   0.189             0.849958
Gears10                         1.4371185   2.0086156   0.715             0.474482
Drive.System.CodeA              1.6764237   0.9433822   1.777             0.075868 .
Drive.System.CodeF              2.3911084   0.8935246   2.676             0.007572 *
Drive.System.CodeP              2.1938795   3.0475834   0.720             0.471771
Drive.System.CodeR              1.4686776   0.8672272   1.694             0.090667 .
Weight                         -0.0029101   0.0002655 -10.960 < 0.0000000000000002 *
AxleRatio                      -1.8184933   0.2569875  -7.076 0.00000000002794686 *
Test.Procedure.Cd3             16.3309290   2.2930669   7.122 0.00000000002040522 *
```

```
Test.Procedure.Cd11              -8.2451258   3.0086480  -2.740              0.006245 *
Test.Procedure.Cd21               3.8038721   2.3216450   1.638              0.101647
Test.Procedure.Cd31               3.5274984   2.3367218   1.510              0.131465
Test.Procedure.Cd35               4.3578604   3.3472843   1.302              0.193249
Test.Procedure.Cd90               2.5231667   2.3697630   1.065              0.287255
Test.Procedure.Cd95              -0.1284533   2.4282263  -0.053              0.957822
Test.Fuel.Type.Cd23              -6.7558996   3.3338905  -2.026              0.042987 *
Test.Fuel.Type.Cd26               1.4175989   2.8512510   0.497              0.619169
Test.Fuel.Type.Cd27              -1.3916295   2.8168523  -0.494              0.621389
Test.Fuel.Type.Cd38             -17.1985801   2.1003761  -8.188 0.00000000000000807 *
Test.Fuel.Type.Cd61              -9.0496357   1.9844630  -4.560 0.00005746169317608 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.863 on 994 degrees of freedom
Multiple R-squared:  0.8408,    Adjusted R-squared:  0.8345
F-statistic: 134.6 on 39 and 994 DF,  p-value: < 0.00000000000000022
```

So, what do we do? Froward and Backward stepwise regression... You can get a load of useful information on page 207 pf the ISLR, i will let you read all of it. In short, Forward stepwise starts with no predictors and adds the rest one at a time until all of them are in the model. The algorithm will use a combination of R2 AIC, BIC and Mallows C(p) to determine which ones to keep. the final best model will be in "epaMpg.fwd1" in this case. Be sure to scroll up through the console to review the output, it is interesting to see what it does to get to the end.

```
##stepwise regression forward

epaMpg.fwd1 <- step(lm(FuelEcon~1,data=epaMpg), direction =
"forward",

                    scope=(~HorsePower + Cylinders + Tested.Transmis-
sion.Type.Code + Gears + Drive.System.Code + Weight + AxleRatio +
Test.Procedure.Cd + Test.Fuel.Type.Cd))


summary(epaMpg.fwd1)
```

Surprisingly, forward is not enough, it is possible to run the model in both directions, as in, start with no variables and add one at a time, and start with all

variables eliminate the worst one at a time and get different models. For this reason there is also a backward regression.

```
epaMpg.back1 <- step(lm(FuelEcon ~ HorsePower + Cylinders + Tested.Transmission.Type.Code + Gears + Drive.System.Code + Weight + AxleRatio + Test.Procedure.Cd + Test.Fuel.Type.Cd,data=epaMpg), direction = "backward")

summary(epaMpg.back1)
```

Well, we have three models, they are all filled with jibber jabber, is there anyway i can compare them without looking at every variable side by side? Yes, yes there is.

ANOVA, which actually is another statistical learning thing called analysis of variance that will usually tell you the difference in means among groups, but in R, it will also tell you the difference in models.

```
anova(epaMpg.1,epaMpg.back1,epaMpg.fwd1)
```

So, what to look for? Lowest RSS, lowest p-value, of all the models this will get you to a starting place to focus in a single model to start working with. In our case, the model i created with all variables and the one generated by the forward stepwise regression as well as the backward stepwise regression all came out to be the same. So, we have a model!

```
> anova(epaMpg.1,epaMpg.back1,epaMpg.fwd1)
Analysis of Variance Table

Model 1: FuelEcon ~ HorsePower + Cylinders + Tested.Transmission.Type.Code +
    Gears + Drive.System.Code + Weight + AxleRatio + Test.Procedure.Cd +
    Test.Fuel.Type.Cd
Model 2: FuelEcon ~ HorsePower + Cylinders + Tested.Transmission.Type.Code +
    Gears + Drive.System.Code + Weight + AxleRatio + Test.Procedure.Cd +
    Test.Fuel.Type.Cd
Model 3: FuelEcon ~ HorsePower + Test.Procedure.Cd + Weight + Gears +
    Test.Fuel.Type.Cd + Cylinders + Tested.Transmission.Type.Code +
    AxleRatio + Drive.System.Code
  Res.Df   RSS Df         Sum of Sq F Pr(>F)
1    994 14834
2    994 14834  0 0.000000000000000
3    994 14834  0 0.000000000007276
```

I don't want to get into variable selection cost when dumping all variables into a model and then attempting every variation over every variable, but you can guess its a lot, and its expensive. To give you a place to read up on this, check out page 207 in the ISLR as stated above, it discusses the cost. 2p number of possible models, p being equal the number of variables, it adds up fast. we have 9 in ours, so 29 models to test if we test everything, but only 512... And as stated in the ISLR forward and backward stepwise only test a subset of that, math is on page 208 of ISLR, i will not be a better person for it, so i am not going to bother diving deep into it and explaining it to you.

Point is, what if i do want to test all 512 variations? There is a package i mentioned couple of posts back that will do this and give us a lot of data, olsrr with the function ols_all_subset()

So, lets try one out! We already have a model created in epaMpg.1, so we will use that one, but remember as it is stated in the variable selection documentation that running this will test 2p variable combinations, so 20 variables will be 220 = 1,048,576 combinations. So, when you run ols_all_subset, it will test all variables, so it will be much much slower.

```
ols <- ols_all_subset(epaMpg.1)
```

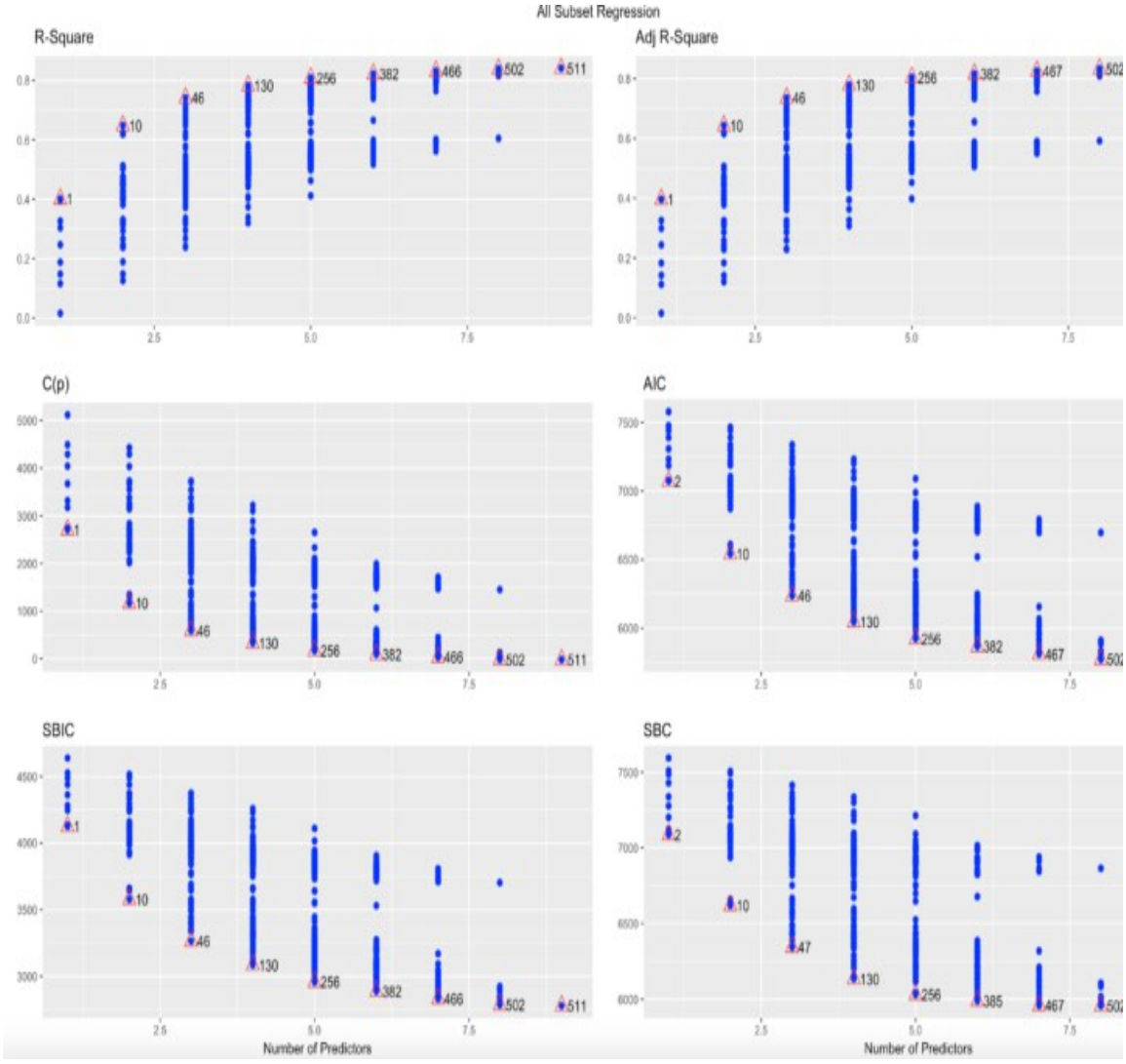
```
View(ols)
```

	mindex	n	predictors	rsquare	adjr	predrsq	cp	aic	sbic	sbc
511	511	9	HorsePower Cylinders Tested.Transmission.Type.Cod...	0.8407647	0.8345170	0.8274047	-20.00000	5770.373	2776.809	5972
506	502	8	HorsePower Cylinders Tested.Transmission.Type.Cod...	0.8390909	0.8334478	0.8263091	-11.55186	5773.185	2785.347	5956
504	503	8	HorsePower Cylinders Tested.Transmission.Type.Cod...	0.8327433	0.8263556	0.8198017	28.07252	5819.191	2824.652	6016
507	504	8	HorsePower Cylinders Tested.Transmission.Type.Cod...	0.8317170	0.8263373	0.8195454	34.47898	5813.516	2830.868	5981

The View() will allow you to view the results in the R console, you can sort and review the data at a minimum you will see that the lowest AIC and the highest R-square is the model we have chose with all variables. The variables tested for each configuration are also listed so you can get an idea of what is best, from this you can create queries to include or exclude criteria if you have a larger dataset.

You can also plot ols;

```
plot(ols)
```


All Subset Regression
R-Square
Adj R-Square
C(p)
AIC
SBIC
SBC
Number of Predictors
Number of Predictors

Section 3

PREDICTION

Lets do some prediction.

This series started because the regression and data from the mtcars test data provided was useless as predicting the mpg of my truck, looking at the dataset it is not hard to figure the dataset is useless for any modern vehicle, to be fair it was not meant to be.

I admit the way the data is now, there will be some overfit for some parameters, as that is not the purpose of this exercise i am not going to worry about it. I think this will be a very good dataset to review in the future to deal with overfit issues.

If you have missed the code in the last couple sections, this will get you to now;

```
epaMpg <- read.csv("epaMpg.csv",stringsAsFactors=FALSE)


summary(epaMpg)


epaMpg$Cylinders <- as.factor(epaMpg$Cylinders)

epaMpg$Tested.Transmission.Type.Code <-
as.factor(epaMpg$Tested.Transmission.Type.Code)

epaMpg$Gears <- as.factor(epaMpg$Gears)

epaMpg$Test.Procedure.Cd <- as.factor(epaMpg$Test.Procedure.Cd)

epaMpg$Drive.System.Code <- as.factor(epaMpg$Drive.System.Code)

epaMpg$Test.Fuel.Type.Cd <- as.factor(epaMpg$Test.Fuel.Type.Cd)


epaMpg <- epaMpg[!is.na(epaMpg$Cylinders),]


# scipen will just blow out the  scientific notation, i like to see
the number.

options(scipen = 999)
```

```
# this should be familiar by now.

epaMpg.1 <- lm(FuelEcon ~ HorsePower + Cylinders + Tested.Transmis-
sion.Type.Code + Gears + Drive.System.Code + Weight + AxleRatio +
Test.Procedure.Cd + Test.Fuel.Type.Cd,data=epaMpg)


summary(epaMpg.1)
```

First we need to create a data frame with our "New Data". The dataset and the value types need to match what was originally passed into the model, so there is a little bit of work to get this right the first time.

```
HorsePower <- c(395,1500,70)

Cylinders <- as.factor(c(8,16,4))

Tested.Transmission.Type.Code <- as.factor(c("A","SA","CVT"))

Gears <-as.factor(c(8,8,1))

Drive.System.Code <- as.factor(c(4,"A","F"))

Weight <- c(5500,4750,2800)

AxleRatio <-c(3.21,3.64,3.2)

Test.Procedure.Cd <- as.factor(c(90,90,90))

Test.Fuel.Type.Cd <- as.factor(c(61,61,61))


newcars =
data.frame(HorsePower,Cylinders,Tested.Transmission.Type.Code,Gears,Dri-
ve.System.Code,Weight,AxleRatio,Test.Procedure.Cd,Test.Fuel.Type.Cd)


View(newcars)
```

Using our new dataframe lets run a prediction

```
predict(epaMpg.1,newdata=newcars,interval="confidence")
```

From this we get the the estimate and the 95% confidence interval of the prediction, imagine it as the possible range.

```
> predict(epaMpg.1,newdata=newcars,interval="confidence")
       fit       lwr      upr
1 17.58447 15.433917 19.73501
2 12.20592  6.632648 17.77918
3 44.88640 43.435505 46.33729
>
```

And there you have it!

Chapter 9

LOGISTIC REGRESSION

I learned logistic regression using SPSS, which if you have no plans to use SPSS in life its sort of a waste of time. My professor said “i don’t know how to do that in SPSS” a lot, professor also said look it up on google a lot too. I expected better.

So what is logistic regression?

Section 1

THE BASICS

In Short its a way to predict a qualitative variable from qualitative and quantitative variables. It falls under the statistical learning domain of classification, there are more in there as well, but lets focus on one at a time.

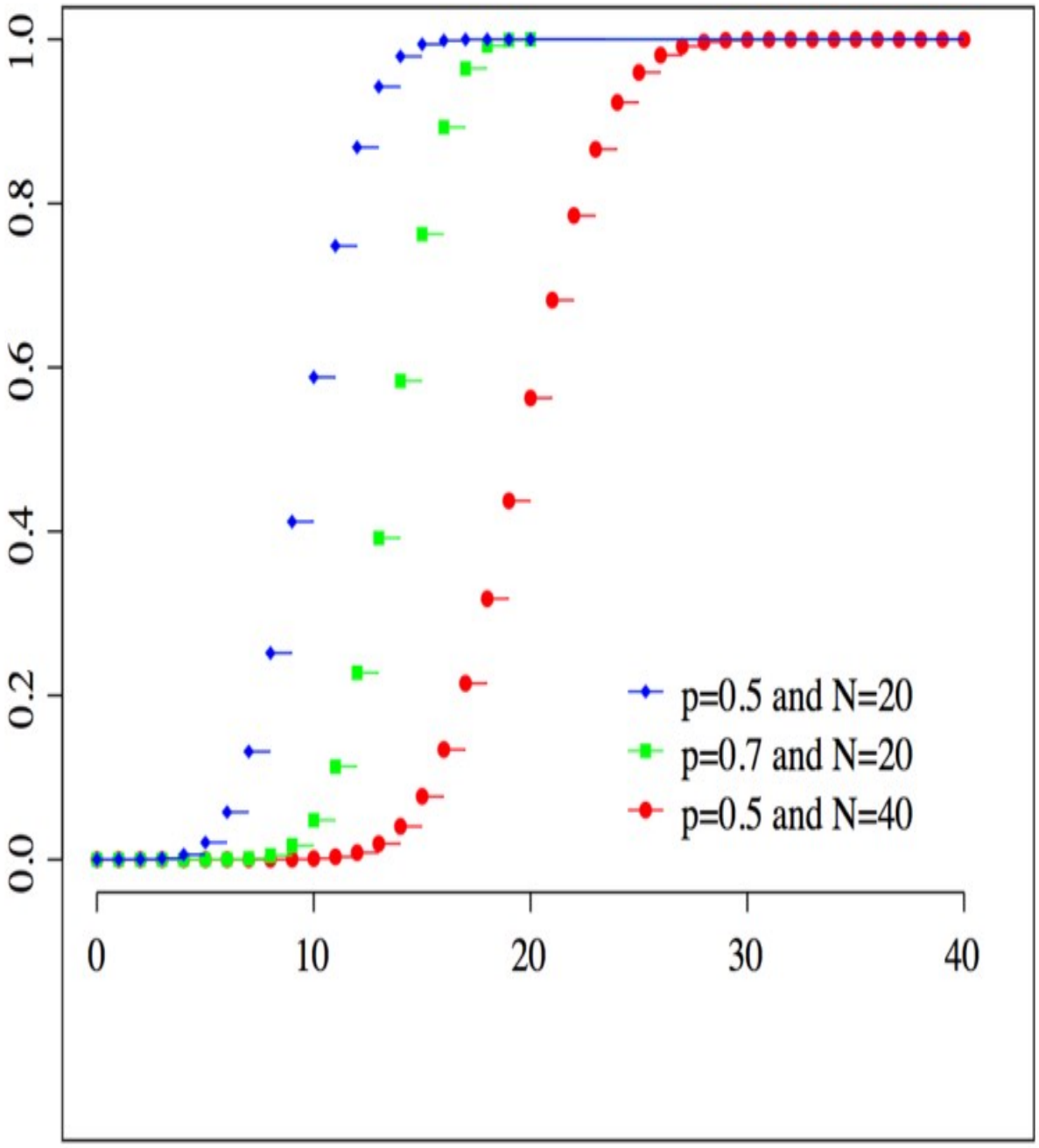
When would i use it? Well, lets start with something simple, is my lawn brown or green right now, and you are not allowed to look at it. The predictor is binary, only two values, we can simplify it to is Green = "Yes" or "No". Lets imagine some explanatory variables that could help us answer the question of green, yes or no, and you cannot look at your lawn, you are on the beach in Bermuda and there is no webcam for your lawn. What variables might be useful in predicting this?

1. Temperature for the last 30 days
2. Rain or lawn watering for last 30 days
3. Type of grass, winter or seasonal grass
4. Month of year, December-March is likely not green
5. Climate Region of the US
6. Brown or Green (our predictor from past observations)

So logically, it is somewhat easy to see that if it is January in the North East, the temperature is an average of 25 for the last 30 days, rain (snow) or no rain my grass will be brown. So, ideally we can feed this into a model and it can give us a yes or no if my grass is green. Make sense?

Hopefully you understand the concept, from data tell me yes or no, 0 or 1, Hillary or Trump (spoilers), brown or green. This is called binomial, but here is the deal with a binomial, while the end result is a straight binary answer, getting there is not, its an evolution or a curve. Think about your

grass again, in most places you do not wake up one day with brown grass, though when fall hits it sure seems that way, its a progression, the temperature starts to drop every day, the rain my slow or turn to snow, December may be approaching. What will happen in the data is a slow curve from my grass is green, to maybe its green to maybe its brown to its brown. The fancy term is binomial distribution, demonstrated by cumulative distribution shown below, thank you Wikipedia.



Don't worry about the p-value and the n, but consider the lower left is yes my grass is green, and the top right is my grass is brown, as time passes, temperature drops and we enter winter, we move a long the line this tells us the probability that the grass is still green, until it is not, as you can well imagine there is a middle part basically between .2 and .8 on the y axis that falls into the category of we are not sure. For the purpose of our imaginary

dataset, those numbers are the probability of the grass being green, in our case, green is zero and brown is 1. Anything in between is a probability of green or brown. So, not only can we get a yes or no, more importantly we get a probability. But, it is up to you to decide 0 or 1, if we decide that anything less than .3 is green and anything above .3 is brown, that will be the answer. While the input data did have a yes or no, we can move prediction of that yes or no up and down with the probability.

In math speak jibber jabber, p(y|x) What is the probability of y given x has occurred. What is the probability of green grass given the temperature has been 25 degrees for 30 days? You can take a probability class to be bombarded by this so i will save you from it for now.

Another example, which is a very popular example, hence the reason i avoided it first is, should you be given a loan? Since its you the answer is always yes, if you are asking your grandmother, if you are asking a bank, not so fast. Kaggle has some datasets on this, feel free to do your own exploration, i am undecided as of this writing if i will blog one or not. But, think about the loan, the outcome is yes or no, the inputs for bank to decide are credit score, employed yes or no, income, number of current open credit accounts, own or rent a home, though that is tied up in credit score, it will be really high if you own over rent.

Section 2

LOGISTIC REGRESSION 1

[MSM spotlights Donald Trump vs. Hillary Clinton and Bernie Sanders]

I am going to be using a file that is on my github site that actually came form about a dozen different sources, it has been whittled down to 54 variables and you will see the the data science process that it will become even smaller. As this is a toy dataset for demonstration purposes only, i am not going to reference too many of the sources as they are across many years

so ***making a decision based on this data would be a very bad idea***. So, demo, toy data only!

First things first, lets load the data and then we will jump ahead and look at some pictures.

```
setwd("/Users/Data")

getwd()


data.Main <-
read.csv("USA.dataAll.csv",stringsAsFactors=FALSE,header=TRUE)


#install.packages("car")

library(car)


#install.packages("ggplot2")

library(ggplot2)
```

Data Cleanup, get rid of the NA's. Missing data is a complicated thing, i am choosing zeros over imputation, mean, median or mode, or you deleting the row. When you do this, choose wisely.

```
#remove all NAs and infinites

is.na(data.Main) <- sapply(data.Main, is.infinite)

data.Main[is.na(data.Main)] <-0


#New York County has two entries, only need one.

data.Main <- data.Main[!(data.Main$X == "1864"),]
```

Check out your variable names, go play around, i have covered data discovery and investigation in prior blog posts, so i will not do it here.

```
names(data.Main[6:26])

options(scipen=999)
```

I am going to jump way ahead to demonstrate what logistic regression is and how it is visualized. The following is a binomial distribution cumulative distribution function generated by the glm or general linear model with binomial being passed as the family of algorithm to use. Notice that we are passing Winner a 0 or 1 value, and lg_population which is a log of population just so the data is a little more friendly in the graph or in fancy math terms, allow the distribution to resemble something more normal, also know as Normal Distribution.

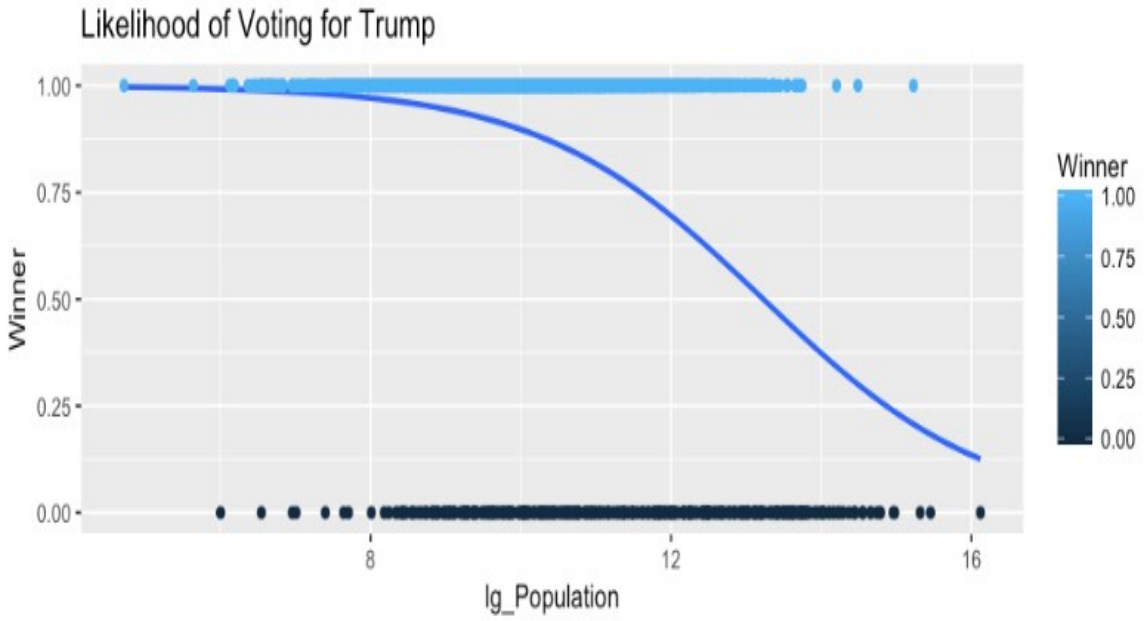
What we are inevidibly looking for here is the nice curve seen below and not a straight line. When we start looking at P-Values you will notice that the lack of a curve and high p-values tend to correlate, thus the variable would add no value to the model.

```
ggplot(data.Main, aes(lg_Population, Winner,colour=Winner)) +

  stat_smooth(method="glm",
method.args=list(family="binomial"), se=FALSE)+

  geom_point() +

  ggtitle("Likelihood of Voting for Trump")



#if you want to see the difference , No log and add text for
outliers,

ggplot(data.Main, aes(Population, Winner,colour=Winner)) +

  stat_smooth(method="glm",
method.args=list(family="binomial"), se=FALSE)+
```

```
        geom_point() +

        geom_text(data=subset(data.Main, Population >
3500000),aes(Population,Winner,label=county_name),
angle=90,hjust=-.1,) +

        ggtitle("Likelihood of Voting for Trump")
```

In short, the binomial glm shows the proportion of a county that voted for Trump crossed with population. So we can see that anything on the far left of the graph had a proportion (y axis) of nearly 1, (nearly 100%) of the county voted for Trump, this far left also means that these counties had a very small population based on the X axis. This indicates that the lower the population of a county, the more likely they are to vote for Trump, as we move to the write you will see the line dropping and moving right which indicates that as population of a county increases a smaller proportion is likely to vote for Trump.

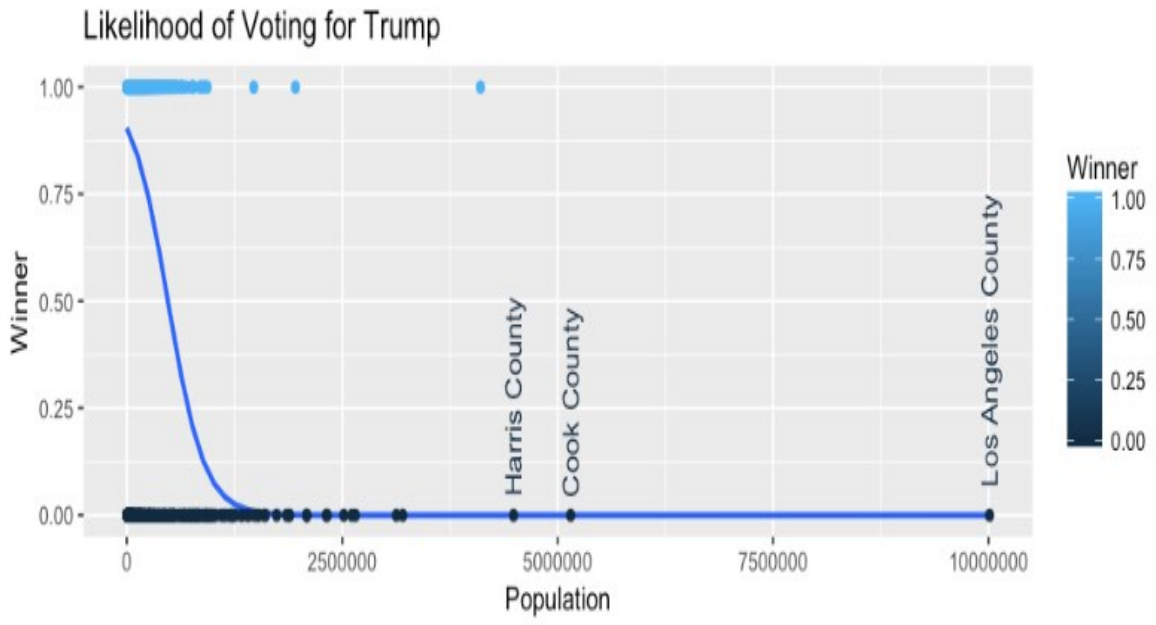
Log Population



If the log is not used you can see the data appears to skew, but that is because we have a population outlier, Los Angeles County. But the curve is

evident. This would indicate that for Logistic Regression we may have something here we can use for a prediction.

Population



The ggplot R code for all of the interesting columns is located on my github site in a Jupyter notebook for R. Please download and play with it. And if you do not have R setup for Jupyter notebooks, great instructions here.

Section 3

LOGISTIC REGRESSION 2

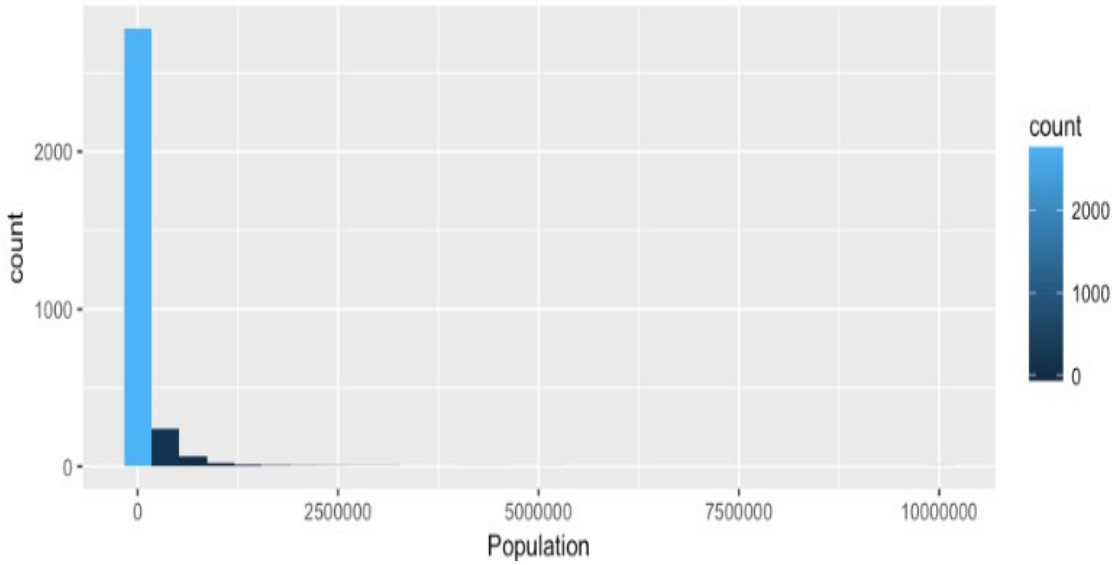
Woo Hoo here we go, in this section we will predict a president, sort of, we are going to do it knowing who the winner is so technically we are cheating. But, the main point is to demonstrate the model not the data. The ISLR has an great section on Logistic Regression, though i think the data chose was terrible. I would advise walking through it then finding a dataset that has a 0 or 1 outcome. Stock market data for models sucks, i really hate using it and really try to avoid it.

Using the dataset and data engineering from the prior blog post we can start. Or grab the Jupyter Notebook for R from here.

Before we dive deeper lets revisit normal distribution and log using the population data.

```
ggplot(data=data.Main, aes(Population)) +
  geom_histogram(aes(fill=..count..))
```

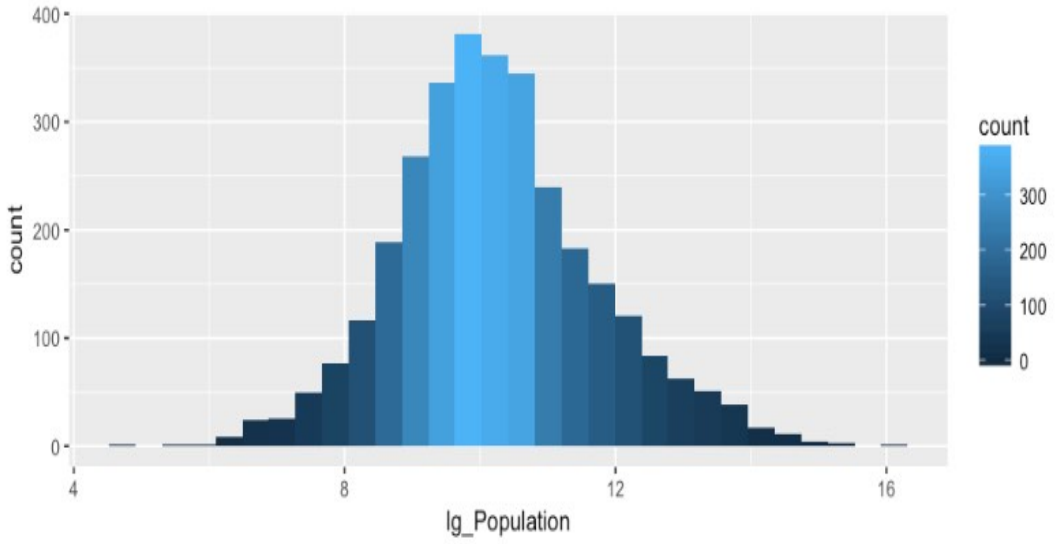
Looking at the Histogram we can see the data is not normally distributed, almost 3000 counties are piled up in the 0, which we know is not true, but this is what happens when you have an imbalance of values in the data, in this case right skew. The solution is to apply a log normal form of the data by using is log value. log(population).



You will notice in the data set provided there is a log form of all the numeric variables already provided with the prefix of lg_.

```
ggplot(data=data.Main, aes(lg_Population)) +
  geom_histogram(aes(fill=..count..))
```

By some voodoo or magic (not really) our population is now normally distributed, which will allow us to use it in a model.



Most of the data is pretty easy to understand by the column names, the only oddity is the RUC_* columns which if you have been keeping up you will notice are already setup as indicator variables. If a county has an RUC=1 only the RUC_1 will have a "1" in it, the other indicator variable columns will be "0". SO what the heck is RUC? The short answer is, a categorical variable that indicates size range of a county. A value of one indicates a county of greater than 1,000,000 where a value of 9 indicates less than 2500.

Lets throw everything into the model and see what happens! Notice we are using glm, and family = binomial, this is R peak for logistic regression.

```
elect.glm <- glm(Winner~Population + PovertyPercent + EDU_LessHS-
Diploma + EDU_HSDiploma + EDU_SomeCollegeorAS + EDU_BSorHigher + Unem-
ploymentRate + Married + Divorced + GINI_Coeff + HHMedianIncome + HH-
MeanIncome + PerCapitaPI + Diabetes + Inactivity + Obesity + Uninsured +
OpiodRx + USDANaturalAmenityRank + RUC_1 + RUC_2 + RUC_3 + RUC_4 + RUC_5
+ RUC_6 + RUC_7 + RUC_8, data=data.Main,  family=binomial)

summary(elect.glm)
```

The worst thing that can possibly happen in your data life is to try and conceive of a model only to find out all of your variables are pretty much useless. Though if you recall from the linear regression posts, the worst thing we can do is dump all the variables into a model as a first step, and second worse thing is to assume they are all bad but 1, population.

The problem is we need to either add the variables one at a time, or remove them one at a time. Lucky for us, stepwise is still a thing for glm.

```
Call:
glm(formula = Winner ~ Population + PovertyPercent + EDU_LessHSDiploma +
    EDU_HSDiploma + EDU_SomeCollegeorAS + EDU_BSorHigher + UnemploymentRate +
    Married + Divorced + GINI_Coeff + HHMedianIncome + HHMeanIncome +
    PerCapitaPI + Diabetes + Inactivity + Obesity + Uninsured +
    OpiodRx + USDANaturalAmenityRank + RUC_1 + RUC_2 + RUC_3 +
    RUC_4 + RUC_5 + RUC_6 + RUC_7 + RUC_8, family = binomial,
    data = data.Main)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3173   0.0413   0.1257   0.3023   2.8839

Coefficients:
                          Estimate    Std. Error z value            Pr(>|z|)
(Intercept)           -3.0605483167  2.8084149306  -1.090               0.2758
Population            -0.0000007458  0.0000003502  -2.130               0.0332 *
PovertyPercent         0.0074820486  0.0242800006   0.308               0.7580
EDU_LessHSDiploma     -0.1756081825  0.0351444079  -4.997  0.0000005830146087 ***
EDU_HSDiploma         -0.0339608351  0.0319459073  -1.063               0.2877
EDU_SomeCollegeorAS   -0.0256203247  0.0329167010  -0.778               0.4364
EDU_BSorHigher        -0.2122255986  0.0339445980  -6.252  0.0000000004049258 ***
UnemploymentRate      -0.1910196361  0.0442648158  -4.315  0.0000159326381475 ***
Married                0.2593069297  0.0197616871  13.122 < 0.0000000000000002 ***
Divorced               0.0055928017  0.0394787958   0.142               0.8873
GINI_Coeff            -0.0001067559  0.0017435083  -0.061               0.9512
HHMedianIncome        -0.0000357307  0.0000253588  -1.409               0.1588
HHMeanIncome           0.0000333079  0.0000219708   1.516               0.1295
PerCapitaPI           -0.0000066789  0.0000070177  -0.952               0.3412
Diabetes              -0.2563534790  0.0547025953  -4.686  0.0000027817086023 ***
Inactivity             0.2412814223  0.0321161952   7.513  0.0000000000000579 ***
Obesity               -0.0055328205  0.0290415463  -0.191               0.8489
Uninsured              0.0033694496  0.0190530481   0.177               0.8596
OpiodRx                0.0228897248  0.0373153205   0.613               0.5396
USDANaturalAmenityRank -0.0568450043  0.0764883627  -0.743               0.4574
RUC_1                 -1.0177570102  0.4620370645  -2.203               0.0276 *
RUC_2                 -0.7648336162  0.4265916511  -1.793               0.0730 .
RUC_3                 -0.0538446203  0.4335291116  -0.124               0.9012
RUC_4                 -0.2538360888  0.4660260936  -0.545               0.5860
RUC_5                 -0.7006634217  0.5351938401  -1.309               0.1905
RUC_6                 -0.3626424535  0.3924413871  -0.924               0.3555
```

```
RUC_7               -0.2890190450  0.3914550410  -0.738             0.4603
RUC_8               -0.4095626944  0.4682714828  -0.875             0.3818
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2716.6  on 3150  degrees of freedom
Residual deviance: 1223.5  on 3123  degrees of freedom
AIC: 1279.5

Number of Fisher Scoring iterations: 7
```

Lets do a forward stepwise log regression and see what happens.

```
elect.fwd <- step(glm(Winner~1,data=data.Main,  family=binomial),
direction = "forward",scope=(~Population + PovertyPercent + EDU_LessHS-
Diploma + EDU_HSDiploma + EDU_SomeCollegeorAS + EDU_BSorHigher + Unem-
ploymentRate + Married + Divorced + GINI_Coeff + HHMedianIncome + HH-
MeanIncome + PerCapitaPI + Diabetes + Inactivity + Obesity + Uninsured +
OpiodRx + USDANaturalAmenityRank + RUC_1 + RUC_2 + RUC_3 + RUC_4 + RUC_5
+ RUC_6 + RUC_7 + RUC_8 ))

summary(elect.fwd)
```

Wowza, we certainly have more than one useful variable now. As discussed in prior posts, RUC_* is a dummy variable, we cannot keep just two of them, its all or nothing.

```
Call:
glm(formula = Winner ~ Married + EDU_BSorHigher + EDU_LessHSDiploma +
    Inactivity + Diabetes + Population + UnemploymentRate + RUC_1 +
    RUC_2, family = binomial, data = data.Main)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2635   0.0432   0.1267   0.3013   2.8456

Coefficients:
                     Estimate    Std. Error z value            Pr(>|z|)
(Intercept)       -5.5988579665  1.0349844677  -5.410        0.0000000632 ***
Married            0.2440360596  0.0142914370  17.076 < 0.0000000000000002 ***
EDU_BSorHigher    -0.1772806293  0.0135665533 -13.067 < 0.0000000000000002 ***
EDU_LessHSDiploma -0.1431146549  0.0154143124  -9.285 < 0.0000000000000002 ***
Inactivity         0.2474317128  0.0275938799   8.967 < 0.0000000000000002 ***
Diabetes          -0.2705205383  0.0461561091  -5.861        0.0000000046 ***
Population        -0.0000007191  0.0000003326  -2.162            0.030597 *
UnemploymentRate  -0.1885404154  0.0408878593  -4.611        0.0000040043 ***
RUC_1             -0.7851107656  0.2358855701  -3.328            0.000874 ***
RUC_2             -0.5116584137  0.2107039804  -2.428            0.015169 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2716.6  on 3150  degrees of freedom
Residual deviance: 1232.7  on 3141  degrees of freedom
AIC: 1252.7

Number of Fisher Scoring iterations: 7
```

```
elect.back <- step(glm(Winner~Population + PovertyPercent + EDU_-
LessHSDiploma + EDU_HSDiploma + EDU_SomeCollegeorAS + EDU_BSorHigher +
UnemploymentRate + Married + Divorced + GINI_Coeff + HHMedianIncome +
HHMeanIncome + PerCapitaPI + Diabetes + Inactivity + Obesity + Uninsured
+ OpiodRx + USDANaturalAmenityRank + RUC_1 + RUC_2 + RUC_3 + RUC_4 +
```

```
RUC_5 + RUC_6 + RUC_7 + RUC_8 ,data=data.Main,  family=binomial), direc-
tion = "backward")

     summary(elect.back)
```

Different order, but same values.

```
Call:
glm(formula = Winner ~ Population + EDU_LessHSDiploma + EDU_BSorHigher +
    UnemploymentRate + Married + Diabetes + Inactivity + RUC_1 +
    RUC_2, family = binomial, data = data.Main)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2635   0.0432   0.1267   0.3013   2.8456

Coefficients:
                     Estimate    Std. Error z value            Pr(>|z|)
(Intercept)       -5.5988579665  1.0349844677  -5.410        0.0000000632 ***
Population        -0.0000007191  0.0000003326  -2.162            0.030597 *
EDU_LessHSDiploma -0.1431146549  0.0154143124  -9.285 < 0.0000000000000002 ***
EDU_BSorHigher    -0.1772806293  0.0135665533 -13.067 < 0.0000000000000002 ***
UnemploymentRate  -0.1885404154  0.0408878593  -4.611        0.0000040043 ***
Married            0.2440360596  0.0142914370  17.076 < 0.0000000000000002 ***
Diabetes          -0.2705205383  0.0461561091  -5.861        0.0000000046 ***
Inactivity         0.2474317128  0.0275938799   8.967 < 0.0000000000000002 ***
RUC_1             -0.7851107656  0.2358855701  -3.328            0.000874 ***
RUC_2             -0.5116584137  0.2107039804  -2.428            0.015169 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2716.6  on 3150  degrees of freedom
Residual deviance: 1232.7  on 3141  degrees of freedom
AIC: 1252.7

Number of Fisher Scoring iterations: 7
```

Let's see what happens when we use the log versions of the variables? To save some time, lets just slam this into a backward stepwise to see what we get.

```
elect_lg.glm.back <- step(glm(Winner~lg_Population + lg_PovertyPer-
cent + lg_EDU_LessHSDiploma + lg_EDU_HSDiploma + lg_EDU_SomeCollegeorAS
+ lg_EDU_BSorHigher + lg_UnemploymentRate + lg_Married + lg_Divorced +
lg_GINI_Coeff + lg_HHMedianIncome + lg_HHMeanIncome + lg_PerCapitaPI +
lg_Diabetes + lg_Inactivity + lg_Obesity + lg_Uninsured + lg_OpioidRx +
RUC_1 + RUC_2 + RUC_3 + RUC_4 + RUC_5 + RUC_6 + RUC_7 + RUC_8, data=da-
ta.Main,  family=binomial), direction = "backward")

summary(elect_lg.glm.back)
```

Notice this time we have 13 variables in the model that were deemed significant, so, changing the variable to log normal did make a pretty big difference. Recall what we learned from a prior section about p-value, still true. As a rule we are looking for a p-value of less than 0.05 to claim it is significant.

```
Call:
glm(formula = Winner ~ lg_Population + lg_PovertyPercent + lg_EDU_HSDiploma +
    lg_EDU_SomeCollegeorAS + lg_EDU_BSorHigher + lg_UnemploymentRate +
    lg_Married + lg_HHMeanIncome + lg_Diabetes + lg_Inactivity +
    lg_OpioidRx + RUC_1 + RUC_3, family = binomial, data = data.Main)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2531   0.0492   0.1392   0.3154   2.7152

Coefficients:
                      Estimate Std. Error z value            Pr(>|z|)
(Intercept)          -17.46271    2.63923  -6.617 0.00000000036760376 ***
lg_Population          -0.24155    0.06549  -3.688            0.000226 ***
lg_PovertyPercent      -1.94851    0.27776  -7.015 0.00000000002298362 ***
lg_EDU_HSDiploma        1.88690    0.48138   3.920 0.00008640659553640 ***
lg_EDU_SomeCollegeorAS  2.85560    0.45573   6.266 0.00000000370438805 ***
lg_EDU_BSorHigher      -1.25755    0.34148  -3.683            0.000231 ***
lg_UnemploymentRate    -1.49922    0.29897  -5.015 0.00000531247802712 ***
lg_Married              9.73539    0.74318  13.100 < 0.0000000000000002 ***
lg_HHMeanIncome        -3.18246    0.38812  -8.200 0.00000000000000241 ***
lg_Diabetes            -2.55360    0.51348  -4.973 0.00000658883800489 ***
lg_Inactivity           6.22383    0.66513   9.357 < 0.0000000000000002 ***
lg_OpioidRx             0.58560    0.19515   3.001            0.002693 **
RUC_1                  -0.34611    0.23340  -1.483            0.138097
RUC_3                   0.58851    0.24028   2.449            0.014315 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2716.6  on 3150  degrees of freedom
Residual deviance: 1263.9  on 3137  degrees of freedom
AIC: 1291.9

Number of Fisher Scoring iterations: 7
```

Though the model is looking pretty awesome i am going to remove RUC as it is technically a duplicate of population, in other words, i will correlate to the point it is adding no more value. So lets look at hour new model.

```
elect_lg.glm <- glm(Winner ~ lg_Population + lg_PovertyPercent + lg_EDU_HSDiploma + lg_EDU_SomeCollegeorAS + lg_EDU_BSorHigher + lg_UnemploymentRate + lg_Married + lg_HHMeanIncome + lg_Diabetes + lg_Inactivity + lg_OpioidRx, family = binomial, data = data.Main)
```

```
summary(elect_lg.glm)
```

Next section we will go through each of the remaining variables and talk about the other data spit out from the model.

Section 4

MODEL EVALUATION

Make It a More Confusing Matrix, Please!

If the title of the thing you are using is called confusion, stop and start over. Never will you need more proof that statistics is deliberately screwing with you and trying to keep you away until you disassemble a confusion matrix. In lieu of the name, lets give it some new names;

Bewilderment matrix, disorientation matrix, agitation matrix, befuddling matrix, perplexity matrix, i think you get the point…

So what it is it?

True Positive	False Negative	Accuracy	Precision
4253	5050	0.626	0.625
False Positive	**True Negative**	**Recall**	**F1 Score**
2557	8493	0.457	0.528

	Prediction	
	Positive	**Negative**
Actually True	**True Positive**	**False Negativ**
Actually False	**False Positive**	**True Negativ**

True Positive: Top left, In a prediction, these are the ones the model got right. For instance you are many many months pregnant and for some reason you decide to take a pregnancy test and it comes back positive, as it should, yay model!

True Negative: In the lower right are also the ones the model got right. For instance you are a 38 year old male and decide to take a pregnancy test and it comes back negative, as expected, yay model! Bit ridiculous but you get the point.

False Negative: This is one of two ways the model can blow it, a false negative is when the test result is actually true, but the prediction decided it was false. For instance you really are many months pregnant and the pregnancy test comes back negative. This is also known as a Type II error.

False Positive: Lastly is the false positive in the lower left of my matrix, this one indicates that the model predicted a true when it should have predicted a false, for instance, you are a 38 year old male, you take a pregnancy test and it comes back positive. Once again, a ridiculous example, but a biologically male human probably would not test positive for being pregnant. Now, that being said, it is possible for a male human to test positive on a pregnancy test for a specific tumor, while the test was inaccurate for pregnancy it may mean you need to get something checked out. Tumor notwithstanding and focusing just on a pregnancy test this is a Type 1 Error.

None of that appears to be terribly confusing, once you look at it, it makes a good bit of sense. But, i think we can make it worse. There is another two by two grid we have not looked at an di will say in three semesters of stats it was not discussed, though as i have stated before the stats professors goal is to typically get you through he class not cover everything extensively.

Accuracy This is also called the misclassification error, or accuracy. it is the (False Positives + False Negatives) / sample size. So;

$$(2557+5050) / (4253+5050+2557+8493) = 0.3737533$$

To get the Accuracy subtract from 1.

$$1 - 0.3737533 = 0.6262467 \text{ (Accuracy)}$$

Precision This is where confusion may start to kick in, the short version is; Precision is the proportion of positives that are classified correctly. In our case,

True Positive / (True Positive + False Positive) so;
4253 / (4253 + 2557) = 0.6245228 (Precision)

Recall Is answering the question of how many are actually classified correctly.

True Positive / (True Positive + False Negative)
4253 / (4253 + 5050) = 0.4571644 (Recall)

F1 Score is the harmonic mean of precision and recall, thus believed to be a good summary of the models ability to predict.

F1 Score = 2((Precision * Recall) / (Precision + Recall)) so;

2 * ((0.6245228 * 0.4571644)/(0.6245228 + 0.4571644)) = 0.5278968

Low and behold it worked and i did not have to use math notation. If you want to see that there are plenty of sites to confuse more.

Section 5

LOGISTIC REGRESSION 3

In Logistic Regression 2 we created a model, quite blindly i might add. What do i mean by that? I spent a lot of time getting the single data file ready and had thrown out about 50 variables that you never had to worry about. If you are feeling froggy you can go to the census and every government website to create your own file with 100+ variables. But, sometimes its more fun to tale your data and slam it into a model and see what happens.

What still needs done is to look for collinearity in the data, in the prior section i removed the RUC variables form the model since the will change exactly with the population, the only thing they may add value to is if i wanted to use a factor for population vs. the actual value.

If you are more interested in Python i have created a notebook using the same dataset but with the python solution here.

So, where we left off. You will find a new Jupyter notebook here with the same name as the blog, feel free to download it and the prior ones.

```
elect_lg.glm <- glm(Winner ~ lg_Population + lg_PovertyPercent + lg_EDU_HSDiploma + lg_EDU_SomeCollegeorAS + lg_EDU_BSorHigher + lg_UnemploymentRate + lg_Married + lg_HHMeanIncome + lg_Diabetes + lg_Inactivity + lg_OpioidRx, family = binomial, data = data.Main)

summary(elect_lg.glm)
```

Notice all of hour p-values are less than .05 which is what we truly desire in life, what we do not know is if any of the remaining variables are collinear, so how mighty we figure that out? vif() THe generic rule of thumb for the vif is an variable that has a value greater than two is suspect. Seriously, thats it, its suspect, suspect of what you may ask, suspect of being collinear.

```
Call:
glm(formula = Winner ~ lg_Population + lg_PovertyPercent + lg_EDU_HSDiploma +
    lg_EDU_SomeCollegeorAS + lg_EDU_BSorHigher + lg_UnemploymentRate +
    lg_Married + lg_HHMeanIncome + lg_Diabetes + lg_Inactivity +
    lg_OpioidRx, family = binomial, data = data.Main)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2271   0.0507   0.1407   0.3200   2.6626

Coefficients:
                     Estimate Std. Error z value            Pr(>|z|)
(Intercept)           -17.0767     2.6280  -6.498      0.000000000815 ***
lg_Population          -0.2542     0.0626  -4.061      0.0000488072135 ***
lg_PovertyPercent      -1.7349     0.2584  -6.715      0.000000000188 ***
lg_EDU_HSDiploma        2.0626     0.4759   4.334      0.0000146501892 ***
lg_EDU_SomeCollegeorAS  2.9949     0.4528   6.615      0.000000000373 ***
lg_EDU_BSorHigher      -1.1135     0.3354  -3.320             0.00090 ***
lg_UnemploymentRate    -1.5169     0.2976  -5.097      0.0000003452531 ***
lg_Married              9.9277     0.7350  13.507 < 0.000000000000002 ***
lg_HHMeanIncome        -3.4160     0.3770  -9.060 < 0.000000000000002 ***
lg_Diabetes            -2.7026     0.5103  -5.296      0.0000001182063 ***
lg_Inactivity           6.1692     0.6615   9.326 < 0.000000000000002 ***
lg_OpioidRx             0.6079     0.1957   3.107             0.00189 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2716.6  on 3150  degrees of freedom
Residual deviance: 1274.8  on 3139  degrees of freedom
AIC: 1298.8

Number of Fisher Scoring iterations: 7
```

```
x<-vif(elect_lg.glm)

data.frame(x)

vif(elect.glm) > 2
```

```
x[vif(elect.glm) > 2]
```

```
1
2  #vif(elect_lg.glm) > 2
3  x[vif(elect_lg.glm) > 2]
4
```

lg_PovertyPercent	2.62673881828767
lg_EDU_HSDiploma	2.99135546245548
lg_EDU_BSorHigher	4.62040164015897
lg_Married	3.87791985449251
lg_HHMeanIncome	6.22308591030368
lg_Diabetes	3.36645966586608
lg_Inactivity	4.26789713265908

There are a few that are above two, but quite frankly it still a pretty useless definition. SO lets dig a little deeper.

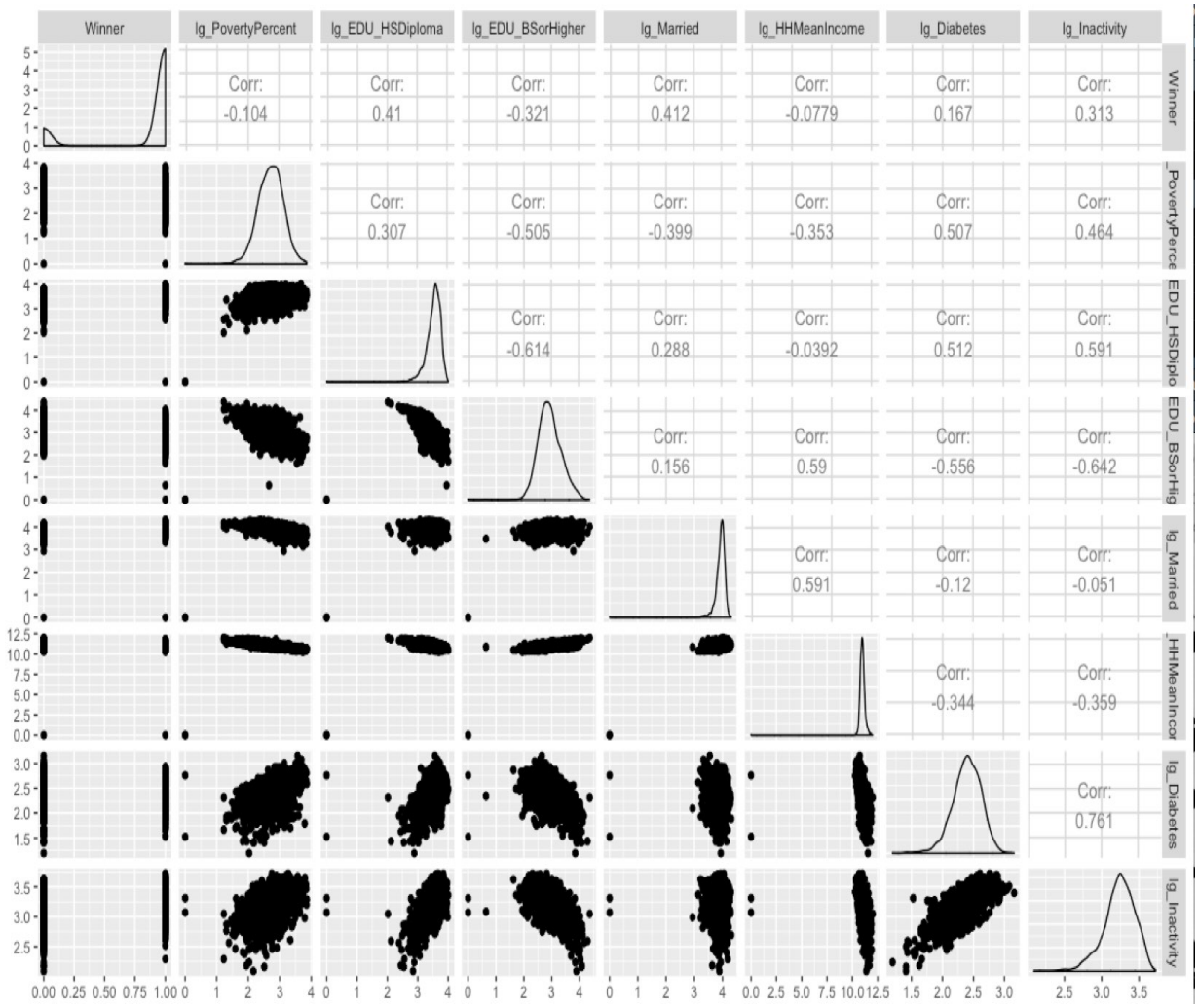
Lets create a vector of column names that ahve a value of greater than 2 and run them through ggpairs and see what some scatterplots and correlations look like.

```
cols = c("Winner",names(x[vif(elect_lg.glm) > 2]))

print(cols)


#This will take a minute or two

ggpairs(data=data.Main,columns=cols)
```

You will have to zoom in to get a good look at this, but what we can see that a few of theses have a positive correlation, Inactivity, EDU_BSorHigher HHMeanIncome have a pretty high vif, and looking at the correlations also have pretty high values.

Lets unpack this; EDU_HSDiploma indicates the percentage of the county that completed high school but went no further and correlates to Winner with a .41 which means i positively correlates indicating counties with a high EDU_HSDiploma percentage typically vote at a higher rate for Trump. Also, counties that have a higher percentage of Married couples also correlated at .412. While not are main interest you can see that there is a correlation in counties with inactivity(lg_Inactivity) and diabetes (lg_Diabetes), not really a shocker that those two correlate.

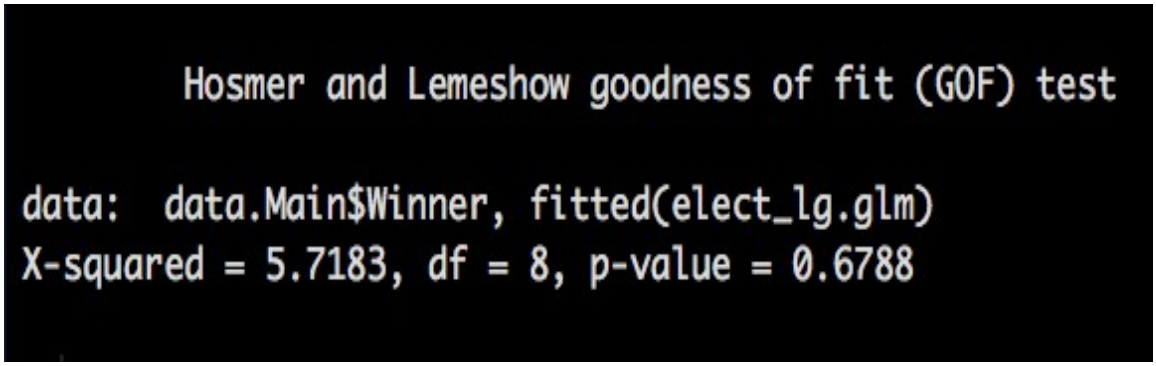
So the question becomes if you have variables that are correlated do you leave them in or take them out? This is where the causation, correlation, and spurious correlation come into play. You have to think about it and make a decisions. Me? I am going to leave everything in, as is.



I am not going to go crazy on this next section of model evaluation, logistic regression is an annoying one that determine a rock solid model really is a bit of voodoo. What we can do after our data engineering, and correlation test is perform a Hosmer-Lemeshow goodness of fit test which will spit out a guess what? A P-Value, but wait, i am going to blow your mind, the p-value in this case needs to be above .05.

```
install.packages("ResourceSelection")

library(ResourceSelection)


hoslem.test(elect_lg.glm$y, fitted(elect_lg.glm), g=10)
```

A p-value of .6788 is not disappointing at all considering we are looking for anything above .05. However, one thing Hosmer-Lemeshow does not take into account is overfitting, so that is still a concern.



The last item to do in our evaluation is to check the confusion matrix.

```
elect.probs = predict(elect_lg.glm, type="response")

elect.probs[1:10]


data.Hold <-  data.Main


elect.pred=rep("Trump",3151)
```

```
elect.pred[elect.probs < .5] = "Hillary"


data.Hold$Winner[data.Hold$Winner == '1'] ="Trump"
data.Hold$Winner[data.Hold$Winner == '0'] ="Hillary"
table(elect.pred,data.Hold$Winner)
```

If you have forgotten what a confusion matrix is,

	Prediction	
	Positive	Negative
Actually True	True Positive	False Negativ
Actually False	False Positive	True Negativ

What we are interested in are the TP and TN this gives our accuracy of the model (309+2589)/3151 = 0.919708 accuracy. Little creepy huh? It would appear that base don this data the model was able to predict the outcome of the election with 92% certainty, keeping in mind that we trained and tested on the same dataset, so technically it should be a nearly perfect fit. In the next post i will pullout a train and test set and see if it is still so good...

However, if there area a high number of true negatives, 2589 in this case that can skew the accuracy, so a better measure is F1-Score, Remember from last section F1 Score = 2((Precision * Recall) / (Precision + Recall)), so now we need to determine the precision and recall.

True Positive / (True Positive + False Positive) = Precision

309/ (309+179) = .633

True Positive / (True Positive + False Negative) = Recall

309 / 309+ 74 = .807

F1 Score = 2(.633*.807) /(.633+.807) = .710

Meh, when looking at the F1 Score, it would appear our model is only able to generalize well about 71% of the time. Hard to say if i could predict an election from that.