# Everything You Need to Know about ggplot2

*Brandon Huang*

## Table of Contents

### Preparation

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

## Introduction

ggplot2 is a data visualization package in R written by Hadley Wickham. To load ggplot2 into R, use `library(ggplot2)`. The "gg" in ggplot2 stands for grammar of graphics. Wickham has a book called *Grammar of Graphics* dedicated to the theory and concepts behind the grammar of graphics. Wickham also has a collection of packages in R, called tidyverse, in which ggplot2 is part of.

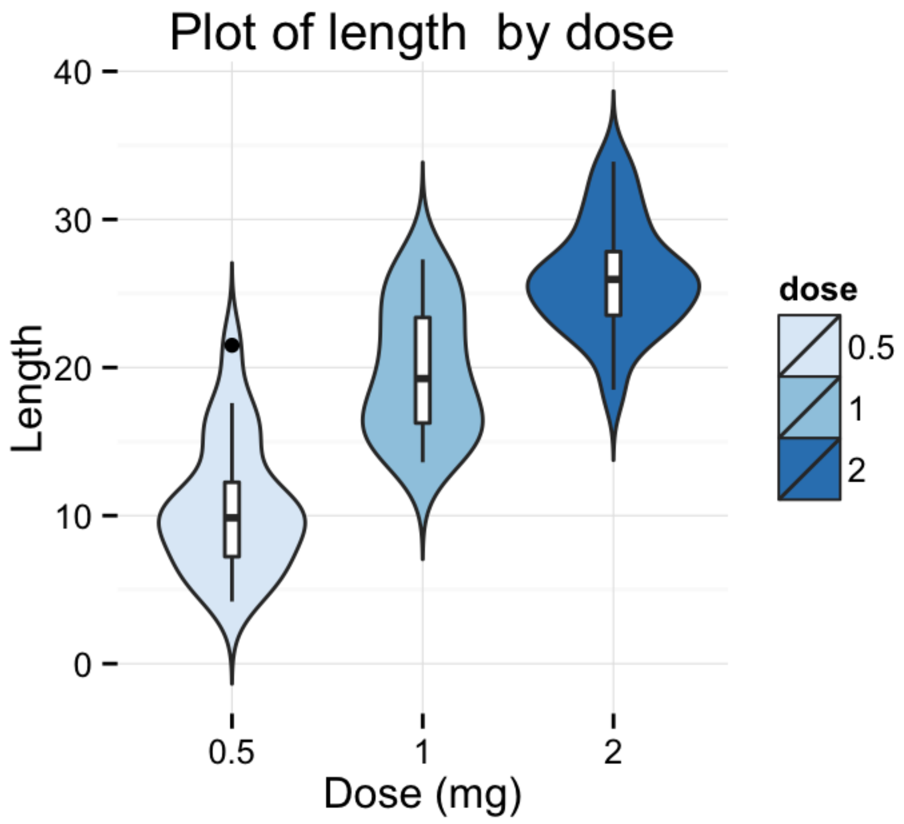
The main function in ggplot2 is `ggplot()`. `ggplot()` allows the user to input a dataset to be plotted through ggplot2's aesthetic tools (we will go into more aesthetic tools later on). Plotting with `ggplot()` is more versatile than the base `plot()` function because plotting a graph using `ggplot()` is very broken down. In essence, each part of the graph (for `ggplot()`), uses a different ggplot2 function. For example, in `plot`, the axis labels are coded inside the `plot()` function while ggplot has a separate axis label function for both x and y labels; that is: `xlab()`, `ylab()`. Another example is that `plot()` automatically draws a scatterplot while ggplot uses `geom_point()` to denote that a scatterplot is to be drawn.

I will showcase the advantages of `ggplot()` compared to `plot` in more depth as we continue onwards. We will further break down `ggplot()`'s intricacies in order to fully comprehend the utility of `ggplot()`. When used correctly, ggplot can greatly shorten the time it takes for your audience to comprehend your graphs.

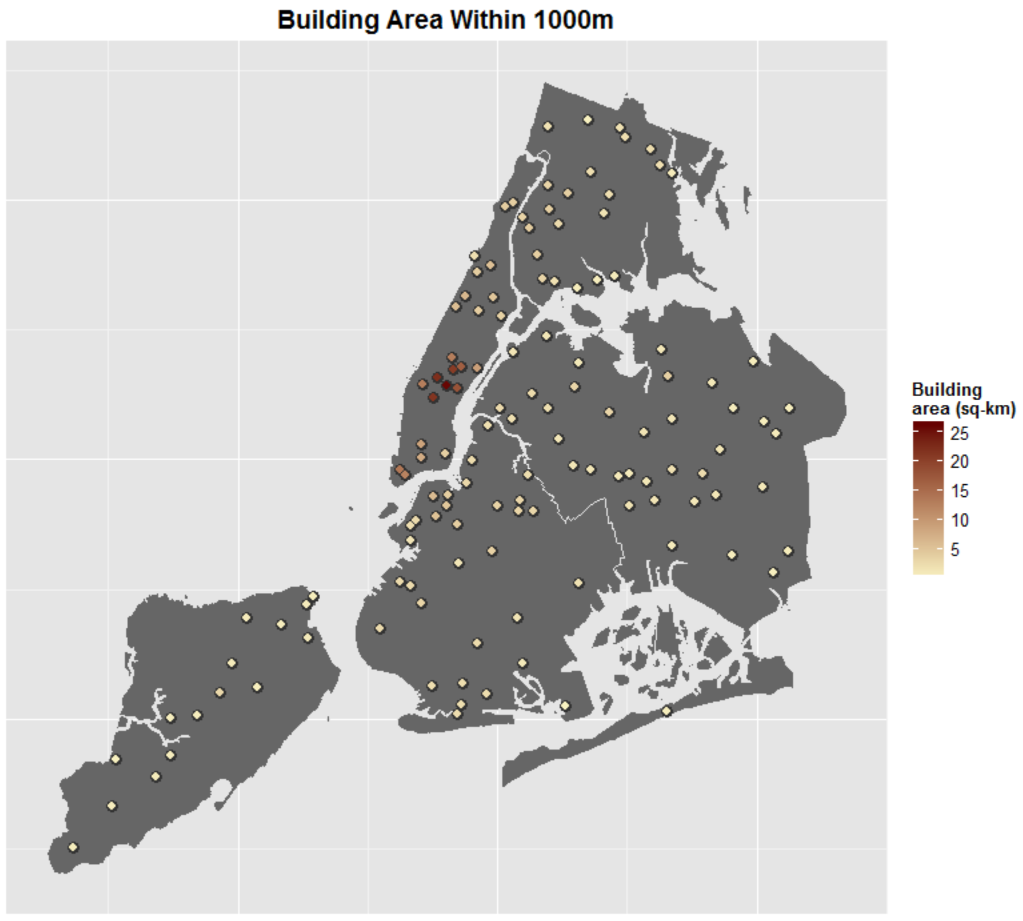
## ggplot2

### The Fundamentals

The most important ggplot2 function to know is `ggplot()`. `ggplot()` is the basis for all graphs because the function accesses the dataset. A typical `ggplot()` looks like `ggplot(<data>)`. Next, we want to add the x and y data using the `aes()` parameter. It is important to note that this does not add the x and y points. In order to add the x and y points, we must add `geom_point()` to `ggplot()`. Besides `geom_point` for scatterplots, the other different types of graphs include histograms, bar charts, bubble charts, and even violin charts.



Or you can draw maps with longitude and latutude datasets. In this map, the shades of red help visualize the building area. This could be very helpful to real estate agents!



Map.

## The Structure

The structure of a ggplot is as follows: `ggplot()` + `<graph type function (ex. geom_point(), geom_histogram(), etc)>` + `xlab()` +

`ylab()` + `labs()` + `<additional functions (ex. geom_smooth(), facet_wrap())>` I will now thoroughly demonstrate each aspect and explain how each function contributes to the graph.

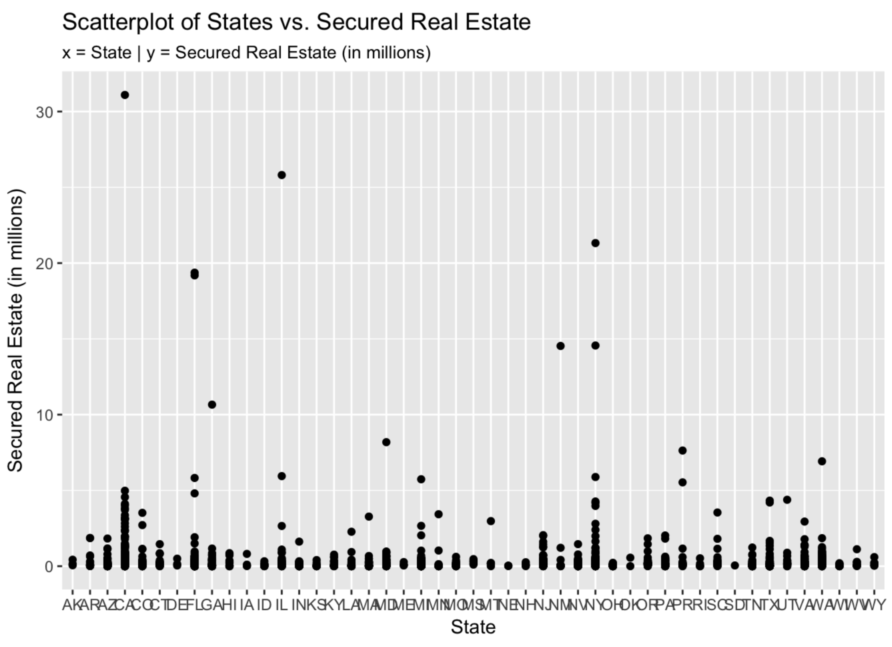
## Example of the Structure

### 1. geom_point

- I will use Chapter 7 Bankrupty data for the year of 2016 to demonstrate the efficiency of ggplot graphs in decreasing comprehension time for the reader of these graphs. The reader will be able to quickly draw information from these graphs because of the simplicity and the detailed labels. This first graph demonstrates each state with their respective secured real estate (in millions). Secured real estate is money secured by creditors from bankrupt clients. This first graph helps us determine which state has the most money secured. This is very valuable information to real estate agents and creditors.

```
#Preparing the Dataset for ggplot

setwd("/Users/brandon/Desktop/stat133/Post/post01/Data")

#chapter 7 bankruptcy data
bankruptcy2016 <- read.csv(file = "ch7data2016.csv", stringsAsFactors = FALSE)

#we want the ones with secured real estate
bankruptcy2016 <- filter(filter(bankruptcy2016, VAR24 != 0))

#secured real estate money in millions
bankruptcy2016$VAR24 <- bankruptcy2016$VAR24/1000000

#case open duration in hundreds
bankruptcy2016$VAR7 <- bankruptcy2016$VAR7/100
```

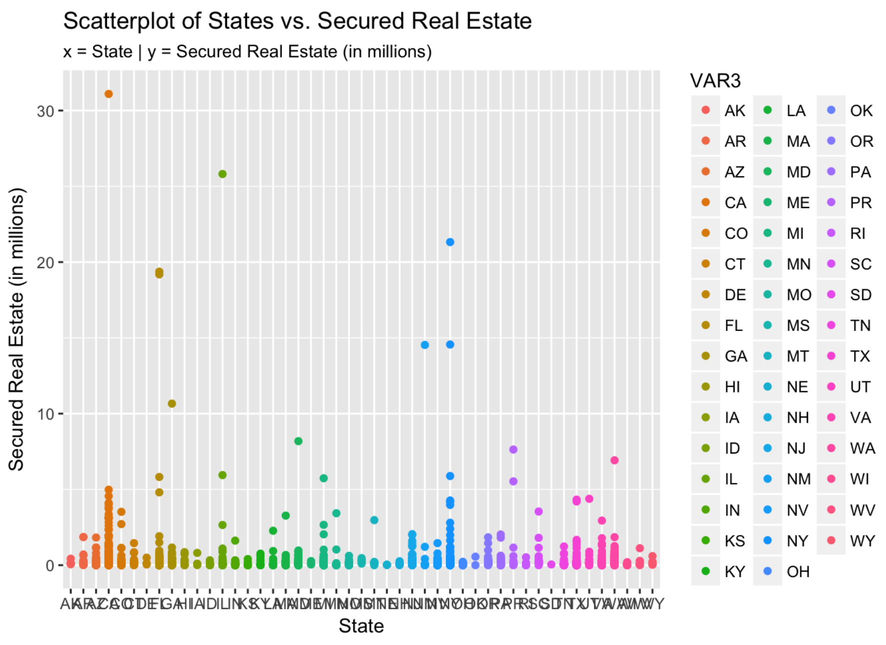
```
#scatterplot of states vs secured real estate
ggplot(bankruptcy2016, aes(x = VAR3, y = VAR24)) +
  geom_point() +
  labs(title = "Scatterplot of States vs. Secured Real Estate",
       subtitle = 'x = State | y = Secured Real Estate (in millions)') +
  xlab(label = 'State') +
  ylab(label = 'Secured Real Estate (in millions)')
```



### 2. Denoting a Respective color for each State

- We can further add to `geom_point` by adding in `aes(color = VAR3)`. VAR3 is the column for states. By doing this, we can denote a certain color for each state. As you can see, the labels are too smug and we cannot tell which state is which. However, by labeling each state a certain color, we can easily tell the difference between the states. The take away from this graph is that it answers which state has the most secured real estate money by creditors. It shows which states have more collected bankruptcy products. This is very useful for business people who would like to enter the bankruptcy real estate market. :

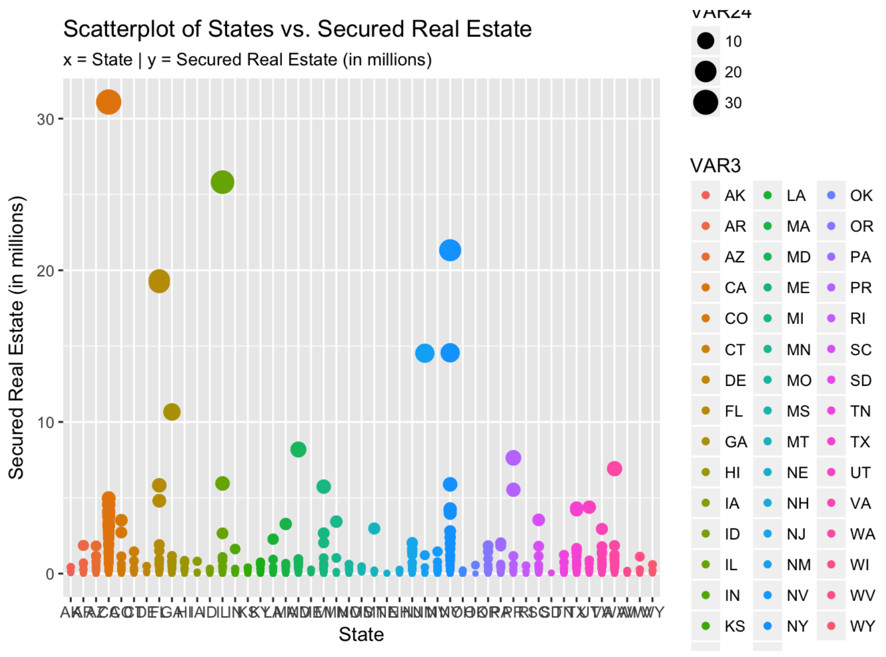
```
ggplot(bankruptcy2016, aes(x = VAR3, y = VAR24)) +
geom_point(aes(color = VAR3)) +
labs(title = "Scatterplot of States vs. Secured Real Estate",
subtitle = 'x = State | y = Secured Real Estate (in millions)') +
xlab(label = 'State') +
ylab(label = 'Secured Real Estate (in millions)')
```



## 3. Denoting a Relative Size for the Amount of Secured Real Estate

- Furthermore, we can change the size of the dots accordindly to the amount of money. In essence, the more money, the bigger the size of the dot. Since VAR24 is Secured Real Estate (in millions), we want to set the size in aes in geom_point equal to VAR24. This helps the reader quickly understand the graph.

```
ggplot(bankruptcy2016, aes(x = VAR3, y = VAR24)) +
geom_point(aes(color = VAR3, size = VAR24)) +
labs(title = "Scatterplot of States vs. Secured Real Estate",
subtitle = 'x = State | y = Secured Real Estate (in millions)') +
xlab(label = 'State') +
ylab(label = 'Secured Real Estate (in millions)')
```



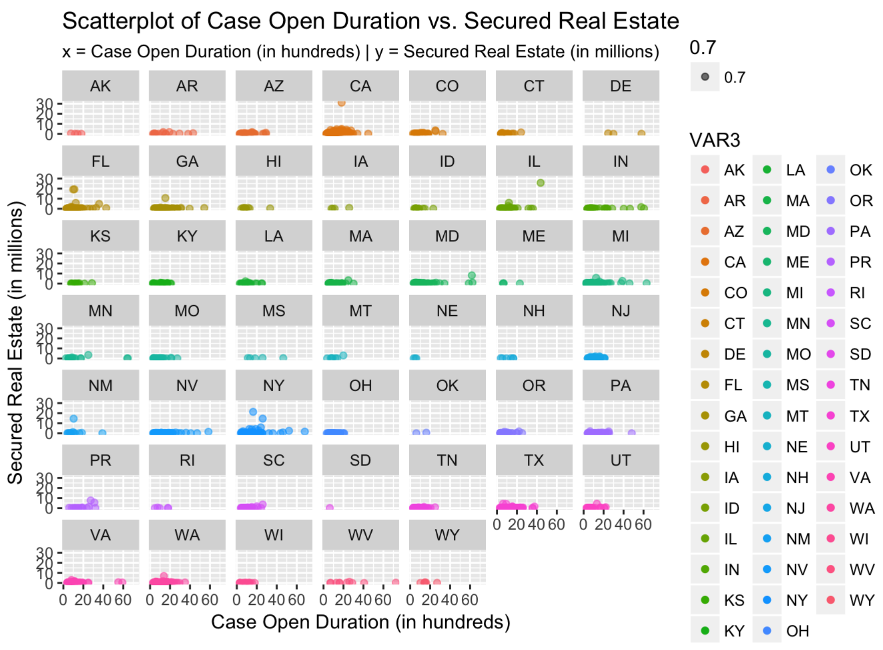
## 4. Labeling the x-axis and y-axis

- As you might have already guessed, the xlab and ylab components denote the x and y axis.

## 5. Grouping Graphs by each State

- With `facet_wrap`, we can easily plot each state with their appropriate dataset. This will create 50 graphs for 50 states. Each graph will answer the reader's question of, "does longer case open duration increase the secured real estate" and the reader will be able to answer this question for all 50 states.

```
ggplot(bankruptcy2016, aes(x = VAR7, y = VAR24)) +
facet_wrap(~ VAR3) +
geom_point(aes(color = VAR3, alpha = 0.7)) +
labs(title = "Scatterplot of Case Open Duration vs. Secured Real Estate",
subtitle = 'x = Case Open Duration (in hundreds) | y = Secured Real Estate (in millions)') +
xlab(label = 'Case Open Duration (in hundreds)') +
ylab(label = 'Secured Real Estate (in millions)')
```



## Utilization

As you can see from above, plotting datasets help us visualize our questions. The first plot helps answer the question, which state has the most bankruptcy properties secured by creditors. To reiterate, this information helps creditors and real estate agents filter out which states are a profitable market and which states are not a profitable markets.

By adding in `geom_point(aes(color = <column>))`, we are able to visualize each state by their appropriate color. These data visualization methods help others understand the data because of its clear visualization. More so, `geom_point(aes(size = <column>))` increases comprehension of a graph. Others are able to quickly locate which states have the largest amount of secured real estate because of the size difference in dots.

Simply put, the utilization of data visualization tools (like `ggplot`) are infinite. In the examples of different `ggplot` codes, I will demonstrate other uses of `ggplot`. These will include time series, animating a ggplot to further increase comprehension of a certain variable from a data set, barplots, and even more scatter plots.

The aim of more examples is to show that `ggplot`'s break down of functions to create a graph maximizes a graph's customizability. And because of the maximization of customizability, the time of comprehension of complex graph substantially decreases. Others are able to quickly absorb information from a complex dataset because of `ggplot`'s efficient data visualization.

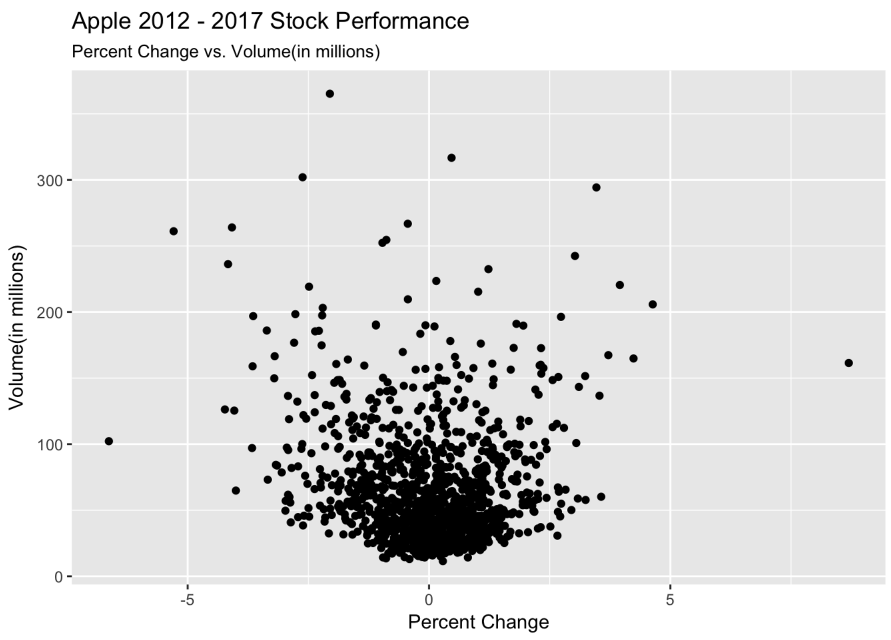
# Sample Plots

## 1. Scatter Plot

In the stock market, when there is a huge price increase on a certain stock, the volume increases too. And if the price keeps increasing but the volume does not, then the price goes down until the volume increases again to a peak. But do not just take my word for it, let us confirm this! Below I have `read.csv`, or imported, the dataset of the S&P 500 into a dataset called stocks. The S&P 500 is an index that contains 500 U.S. stocks. Amongst the stocks is Apple denoted by its stock symbol "AAPL".

By using `dplyr`, I have filtered out 'AAPL' from the list of 500 stocks and I have calculated the percent change of the stock. We will now use ggplot to answer the question, "Does a higher percent change, whether negative or positive, equate to an increase in volume?"

```r
#set wd to data folder
setwd("/Users/brandon/Desktop/stat133/Post/post01/Data")

#import 5 years S&P 500 dataset
stocks <- read.csv(file = 'all_stocks_5yr.csv', stringsAsFactors = FALSE)

#create subset apple data set
aapl <- stocks %>%
  filter(Name == 'AAPL') %>%
  mutate(percent_change = (Close - Open)/ Open * 100, Volume_mil = Volume/ 1000000)

#plot percent change vs. volume traded
ggplot(aapl, aes(x = aapl$percent_change, y = aapl$Volume_mil)) +
  geom_point() +
  labs(title = 'Apple 2012 - 2017 Stock Performance', subtitle = 'Percent Change vs. Volume(in millions)') +
  xlab(label = 'Percent Change') +
  ylab(label = 'Volume(in millions)')
```

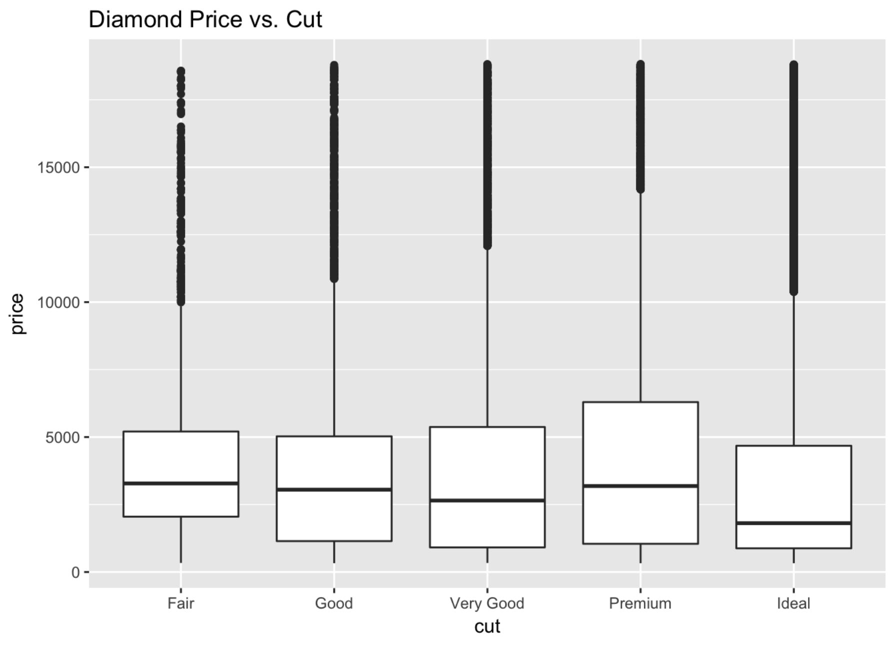


Clearly, we can see that a higher percent change, whether negative or positive, does equate to a higher move in volume. The closer the percent change is to 0, the less volume there is. Very interesting!

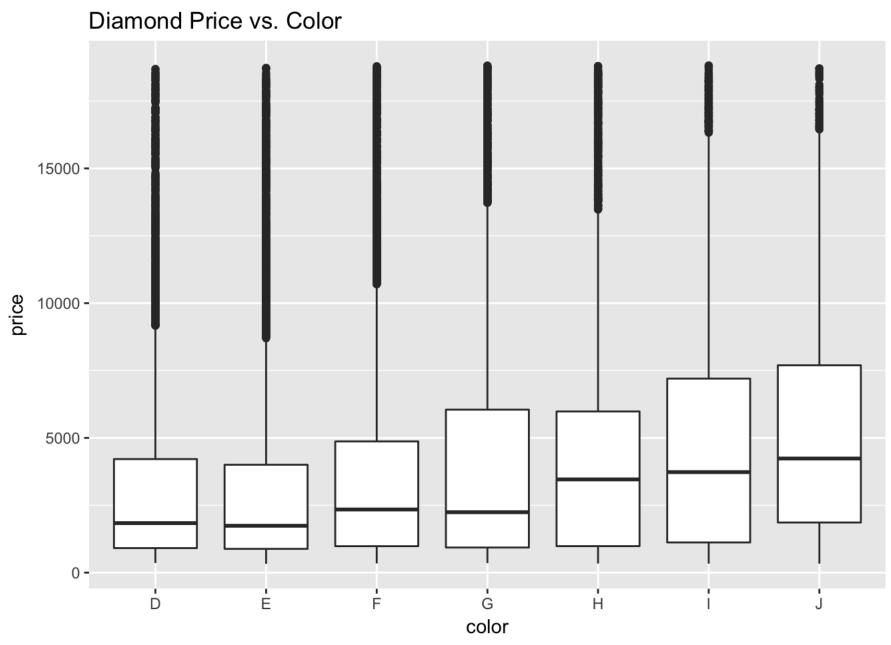
## 2. Boxplots

Beneath, I have graphed diamond price compared to cut, color, and clarity. Diamond is a dataset that comes with R. The boxplot helps us easily compare which aspects of a diamond contributes to the price's decrease or increase. Boxplots are a very good tool to compare between different elements of an object. This helps us observe abnormalities in datasets. In the case of diamonds, higher quality does not equate to hgiher price as seen below.

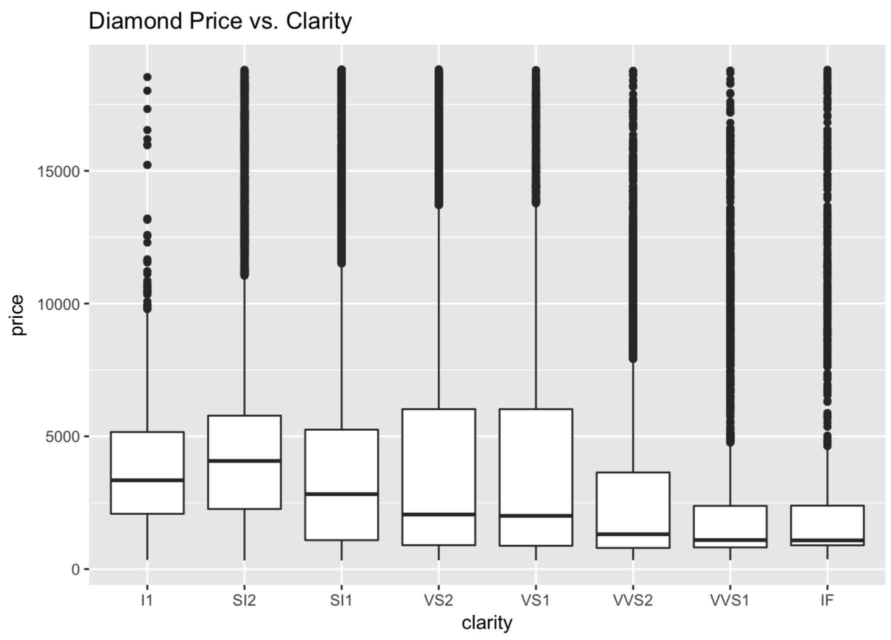
```
ggplot(diamonds, aes(cut, price)) +
  geom_boxplot() +
  xlab(label = 'cut') +
  ylab(label = 'price') +
  labs(title = 'Diamond Price vs. Cut')
```



```
ggplot(diamonds, aes(color, price)) +
         geom_boxplot() +
         xlab(label = 'color') +
         ylab(label = 'price') +
         labs(title = 'Diamond Price vs. Color ')
```



```r
ggplot(diamonds, aes(clarity, price)) +
         geom_boxplot() +
         xlab(label = 'clarity') +
         ylab(label = 'price') +
         labs(title = 'Diamond Price vs. Clarity')
```



## 3. Correlogram

The correlogram shows the correlation between variables in a matrix. The colors are also adjusted so that the negative numbers are in hues of orange and the positive numbers are in hues of green. I mainly wanted to show the correlogram because there are many other gg additions that the R user can download from github. As you can see, the first line of the code reads, “devtools”. These extra ggplot tools are created by other developers and posted on github for every user to download. Beside this correlogram, there are tools on github that allow an user to animate their ggplot and showcase the scatterplot for each year like a GIF.

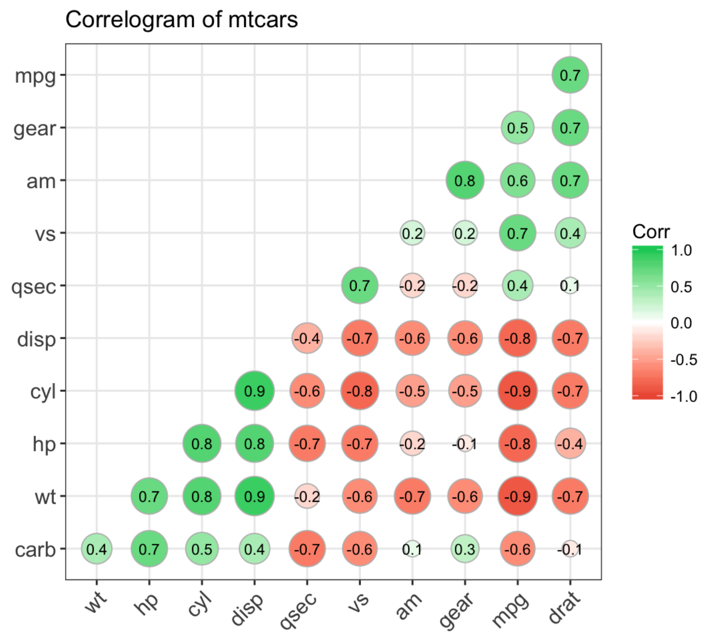
By exploring the web for more functionalities of ggplot, a user can increase their ggplot efficiency by tailoring graphs to datasets. I find this very interesting because of the endless possibilities of ggplot. These are graphs that were never taught in school yet are so useful in explaining data sets and visualizing data. The code below is from the link and I encourage you to look through the list for even cooler ggplot graphing tools!

Work cited for the code below: http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Animated%20Bubble%20Plot

```r
devtools::install_github("kassambara/ggcorrplot")
```

```
## Skipping install of 'ggcorrplot' from a github remote, the SHA1 (edbbaa41) has not changed since last install.
##   Use `force = TRUE` to force installation
```

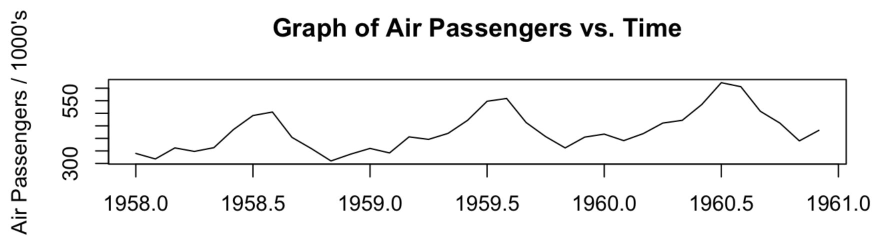
```r
library(ggplot2)
library(ggcorrplot)

data(mtcars) #correlation matrix
corr <- round(cor(mtcars), 1)

ggcorrplot(corr, hc.order = TRUE, #plot
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("tomato2", "white", "springgreen3"),
           title="Correlogram of mtcars",
           ggtheme=theme_bw)
```

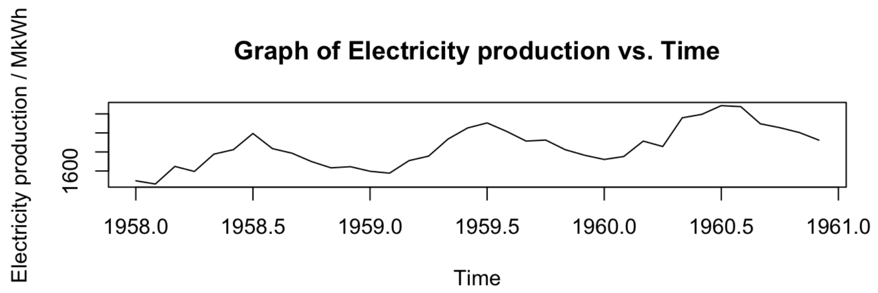


## Weakness of ggplot: Time Series

In the beginning, I thought that ggplot would be able to handle time serie graphs. However, when I tried to input the following base:plot as a ggplot, the following message appeared, “Error: ggplot2 doesn’t know how to deal with data of class mts/ts/matrix”. The error message appeared because ggplot can only handle data frames.

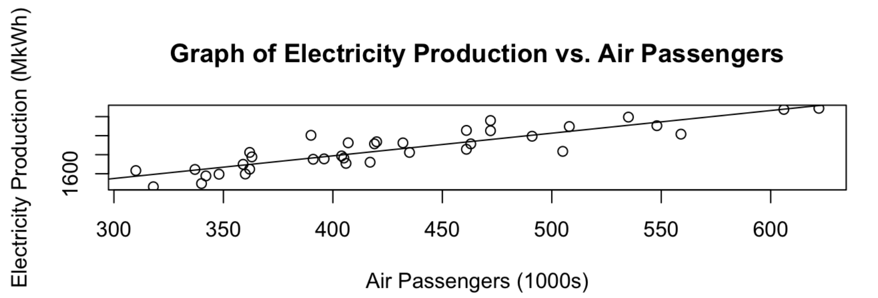
```r
setwd("/Users/brandon/Desktop/stat133/Post/post01/Data")
cbe <- read.table(file = 'cbe.dat', header = T) #create matrix cbe
elec_ts <- ts(cbe$elec, start = 1958, freq = 12) #create ts for electricity

data(AirPassengers) #acquire airpassengers
AP <- AirPassengers
ap_elec <- ts.intersect(AP, elec_ts) #time series: intersect air passengers and electricity


layout(1:2) #graph layout 1 column 2 row
plot(ap_elec[ , 1], ylab = "Air Passengers / 1000's", main = 'Graph of Air Passengers vs. Time') #plot x = time, y
= air passengers
plot(ap_elec[ , 2], ylab = 'Electricity production / MkWh', main = 'Graph of Electricity production vs. Time') #pl
ot x = time, y = electricity
```





```r
#plot of electricity production and air passengers
plot(x = as.vector(ap_elec[ , 1]),
y = as.vector(ap_elec[ , 2]),
ylab = 'Electricity Production (MkWh)',
xlab = 'Air Passengers (1000s)',
main = 'Graph of Electricity Production vs. Air Passengers')

#linear regression line
abline(reg = lm(ap_elec[ , 2] ~ ap_elec[ , 1]))

#correlation between electricity and air passengers
cor(ap_elec[ ,1], ap_elec[ , 2])
```

```
## [1] 0.8841668
```



As shown above, the regular base:plot function easily plots the correlation between air passengers and the onset of electricity. I included these graphs to demonstrate that ggplot is not all inclusive. There are graphs that ggplot cannot handle. I hope this code demonstrated one weakness of ggplot. I wanted to showcase every element of ggplot, even its weaknesses so my reader can fully understand every aspect of ggplot. Although this is a weakness, I am curious if there are ways to convert a time series to a data frame so that ggplot can graph time series through data frames.

## Conclusion

To reiterate, ggplot uses aesthetic properties to customize a graph according to size, color, and many more! The aes function allows the user to fully customize each point on the graph unlike the base:plot function. Even more, ggplot can group each row and graph numerous graphs to showcase the intricacies of each grouped row.

In this blog post, I used dplyr to clean through data sets so that I can have clean graphs. dplyr is an amazing tool to go through data and get more information regarding the data set. For example, the aapl data was compiled through using dplyr and mutating the data set to draw more information. I strongly suggest using both tools to maximize the information one is able to acquire from a dataset. Furthermore, dplyr can enhance facet wraps by its group_by function and summarizing the dataset to acquire new information for the facet wraps in ggplot.

I found data visualization to be the most interesting topic so far because of its capabilities in helping others understand data through a beautiful portrayal of the data. I found this to be very useful in explaining to others the meaning of the data and many times, no explanation is even required! Once I have a full grasp of the shiny app, I will be able to bring my graph customization to a whole new level. The widgets inside the shiny app will allow me to switch inputs on command and I will be able to attain different graphs showcasing different generatedinformation very conveniently.

I hope this blog post has demonstrated:

1. ggplot's customization
2. utilizing ggplot to the fullest
3. the endless possibilities of ggplot
4. ggplot is not perfect

Thank you for reading the post! :)

---

## References:

---

1. "R for Data Science" by Hadley Wickham and Garrett Grolemund
2. Violin Chart image:http://www.sthda.com/sthda/RDoc/figure/ggplot2/ggplot2-violin-plot-logo-data-visualization-1.png
3. Chapter 7 Bankruptcy Dataset: https://www.justice.gov/ust/bankruptcy-data-statistics/chapter-7-trustee-final-reports
4. S&P stocks: https://www.kaggle.com/camnugent/sandp500
5. Map image: http://zevross.com/blog/wp-content/uploads/2014/07/map.png
6. dataset of diamonds from R
7. "Introductory Time Series Analysis" by Paul S.P. Cowpertwait and Andrew V. Metcalfe
8. dataset of mtcars
9. cbe dataset: http://www.maths.adelaide.edu.au/emac2009/
10. dataset of air passengers
11. correlogram package: https://github.com/kassambara/ggcorrplot