Efficient and Eye-pleasing Data with ggplot

Kevin Lee

10/18/2017

Introduction

In the business world, data can be immensely large. Companies that strive to grow place high importance in scalability. In a data table of hundreds or even thousands of entries, it is hard to convey or present information. At a meeting, no one will pull up the entire data table/frame to support their findings or to convey a trend over time. This is where data visualization enters the spotlight.

Data visualization is a modern visual communication that allows for the creation of visual representation of data. By transforming data into a visual medium, it allows people to communicate the information clearly and effectively. And personally, I believe it makes data… fun!

I will be showing you some examples of data visualization using ggplot2!

Background

What is ggplot? ggplot is a data visualization package for R to create statistical graphics with a set of components and wrappers. ggplot's goal is to essentially make working with graphics easier. ggplot2, created by Hadley Wickham, is an implementation of Leland Wilkinson's Grammer of Graphics. It has grown to become one of the most used R packages for data visualization. ggplot has an immense community of people working to improve it and provide better functionality.

Instruction / Examples

First, we load the package ggplot2

```
# load package ggplot2
library(ggplot2)
```

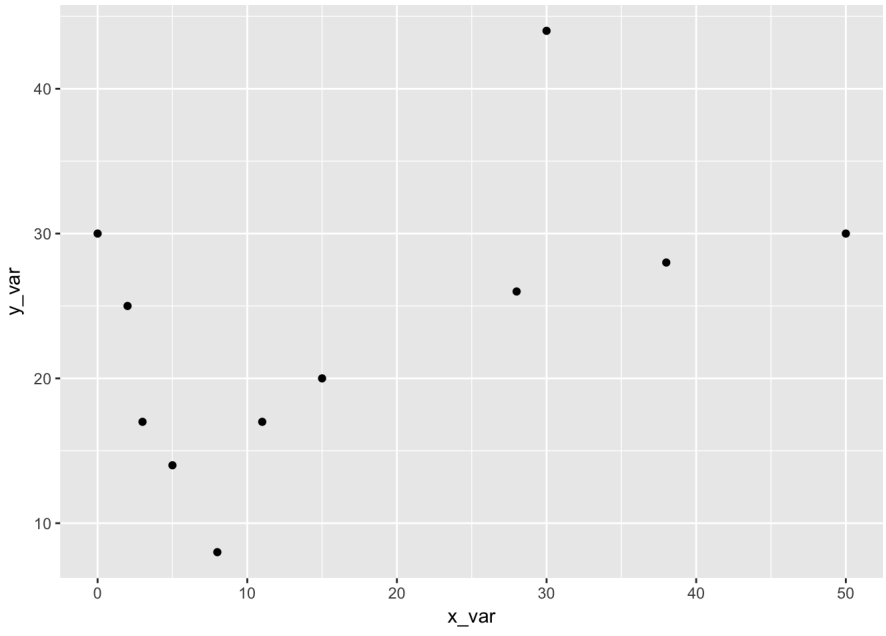
and create a example data frame to plot!

```
# 11 rows with x_var, y_var and a letter
dat <- data.frame(x_var = c(0, 2, 3, 5, 8, 11, 15, 28, 30, 38, 50), y_var = c(30, 25, 17, 14, 8, 17, 20, 26, 44, 2
8, 30), letters = c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K"))
```

Scatterplot

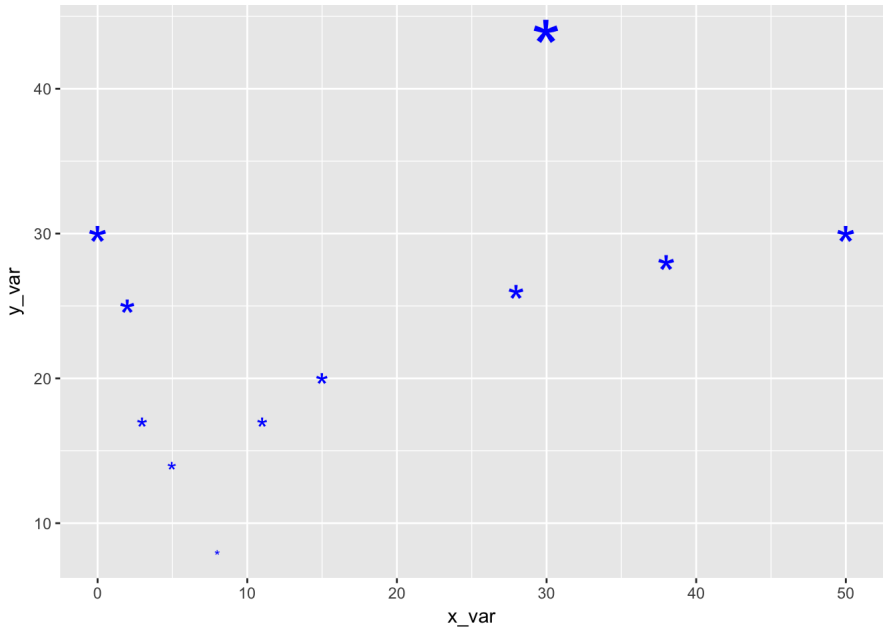
Let's start off with the basic scatterplot

```
# basic scatterplot
ggplot(dat, aes(x = x_var, y = y_var)) + geom_point()
```



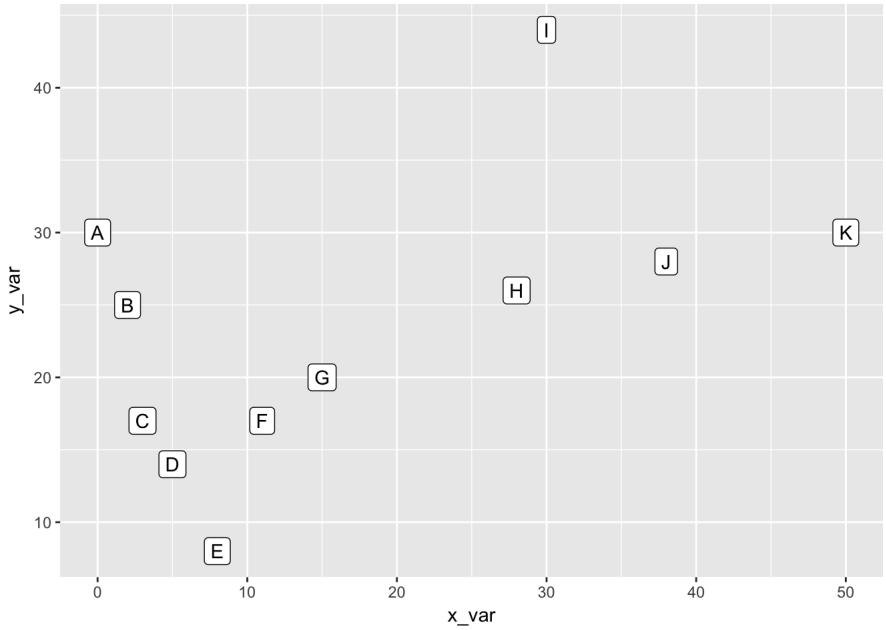
`geom_point` adds scattered points to represent the data in our plot. We have parameters to change its shape, size, color, and more! The color parameter can take a string of the color or a hex. The shape parameter can also take a string of the character you want as the point or a number mapped to a shape in the ggplot shape dictionary.

```
# plot with a different color and shape with size relative to the value of the y variable
ggplot(dat, aes(x = x_var, y = y_var)) + geom_point(color = 'blue', shape = '*', size = dat$y_var/3)
```



We can even place labels on the points to make it clear what they represent. Below I plot the graph with labels for each point according to the letter it represents.

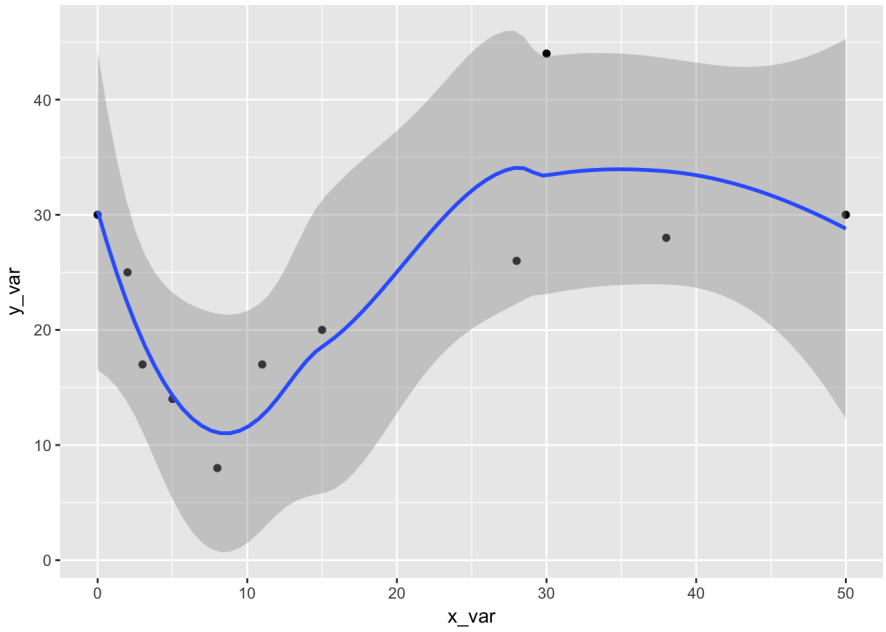
```
# plot with labels
ggplot(dat, aes(x = x_var, y = y_var)) + geom_label(aes(label = dat$letter))
```



To analyze patterns in the plot, we can place a smooth line through the data. The smoothing method can use 'lm', 'glm', 'gam', 'loess', 'rlm'.. - `lm` is linear smooths - `glm` is generalised linear smooths - `gam` is used for more than 1000 observations otherwise `loess` is used - `loess` is local smooths - `rlm` is robust regression

```
# plot with scatter points and a loess smooth line
ggplot(dat, aes(x = x_var, y = y_var)) + geom_point() + geom_smooth()
```

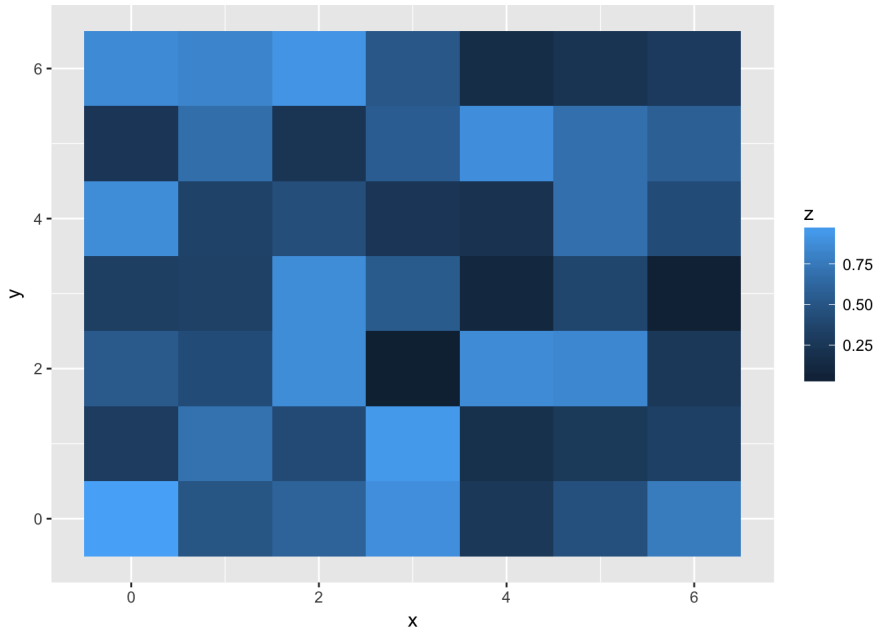
```
## `geom_smooth()` using method = 'loess'
```



Geometric Tile and Heatmaps

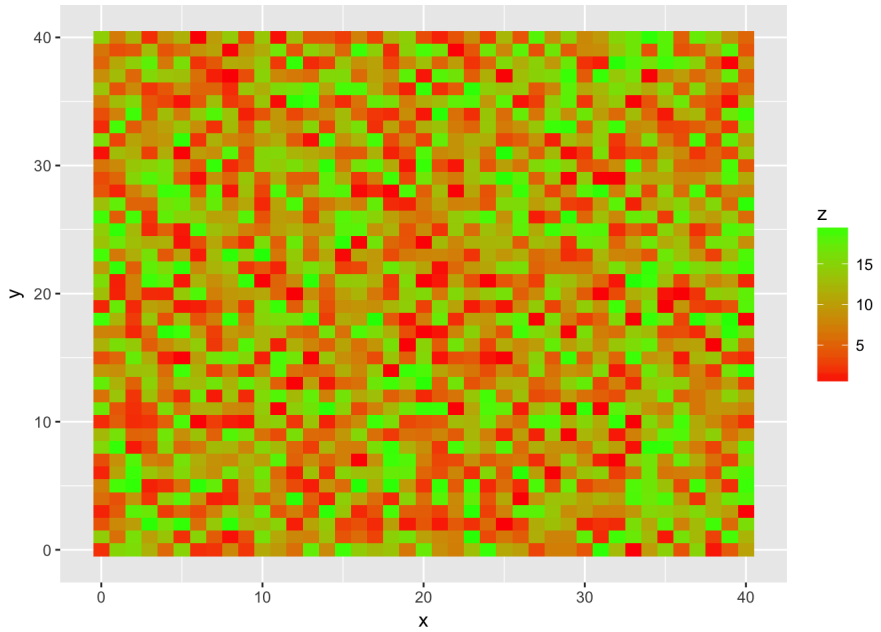
Let's move away from simple scatterplots and check out geom_tile! Below is an example using geom_tile to display different numeric values with a range of color in a grid fashion. This can be useful when the data is conceptually easy to understand in a grid visualization and if the data ranges in an interval.

```
# create grid
df <- expand.grid(x = 0:6, y = 0:6)
# uniform distribution on interval 0:1
df$z <- runif(nrow(df), min = 0, max = 1)
# plot the grid and fill according to the interval between 0 and 1
ggplot(df, aes(x, y, fill = z)) + geom_tile()
```



You can get colorful with this and even bigger for a bigger heat map. A heatmap is a table that visually has colors instead of numbers. It allows the user to see the lows and highs of a data with the intensity of color.

```
# create grid
df <- expand.grid(x = 0:40, y = 0:40)
# uniform distribution on interval 0:1
df$z <- runif(nrow(df), min = 0, max = 20)
# plot
ggplot(df, aes(x=x, y=y, fill=z)) + scale_fill_gradient(low = "red", high = "green") + geom_tile()
```

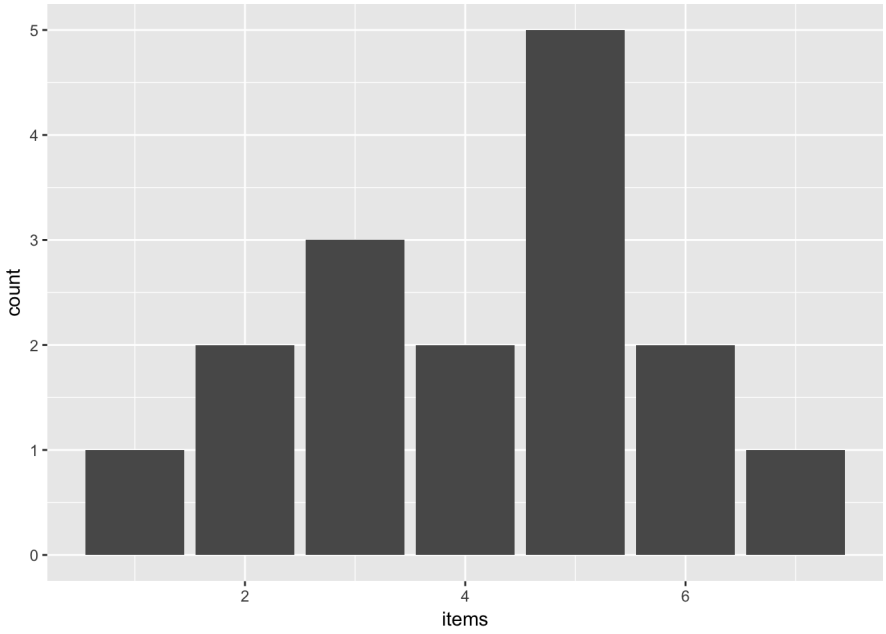


With a heatmap, our goal to improve data visualization is met if we have the right kind of data to use it for. One popular use of heatmaps is for geographic data sets. The user will be able to see which area has a higher intensity of the selected variable. This illustrates that data can be visualized by multiple types of plots but there are certain plots that better convey the intended meaning. If this was not the case, there probably would not be such a variety of different data plots!

Barplot

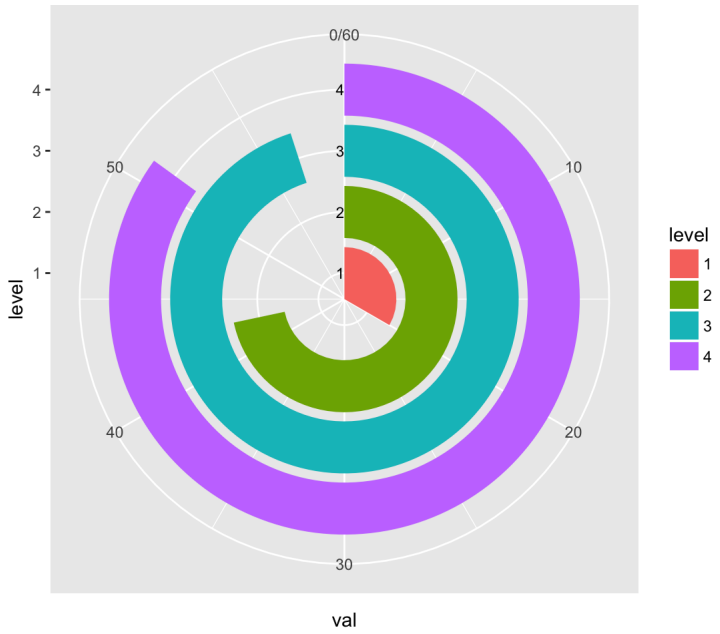
Now we will move on to barplots. A barplot is a simple chart that is usually used when a user wants to view data according to the occurance of it. The standard barplot looks something like this:

```
# create a data frame
dat <- data.frame(items = c(1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 5, 5, 5, 6, 6, 7))
# plot using geom_bar
ggplot(dat, aes(x = items)) + geom_bar()
```



We can design it in various ways. We will try to do a circular orientation barplot.

```
# data frame
dat <- data.frame(level = c("1", "2", "3", "4"), val = c(20, 43, 57, 51))
# plot the barplot
ggplot(data = dat, aes(x = level, y = val, fill = level)) + geom_bar(width = 0.85, stat = "identity") +
  # polar plot
  coord_polar(theta = "y") +
  # increase ylim
  ylim(c(0,60)) +
  # group labels
  geom_text(data = dat, hjust = 1, size = 3, aes(x = level, y = 0, label = level))
```



First we create a data frame to plot. Then, like the simple barplot, we plot with `geom_bar`; however, we use `coord_polar` to use a polar plot for the circular look. We set the `ylim` to a higher value and put down labels for each bar. This is nice to look at and a cool way to look at a barplot. However, this bring up the point where data visualization should make it clear what the data is trying to represent. With this circular barplot, can we really say we improved the way we conveyed this data from the regular vertical barplot? This representation shows how far ggplot can push data visualization, but we have to keep in mind our goal when creating these plots.

Themes

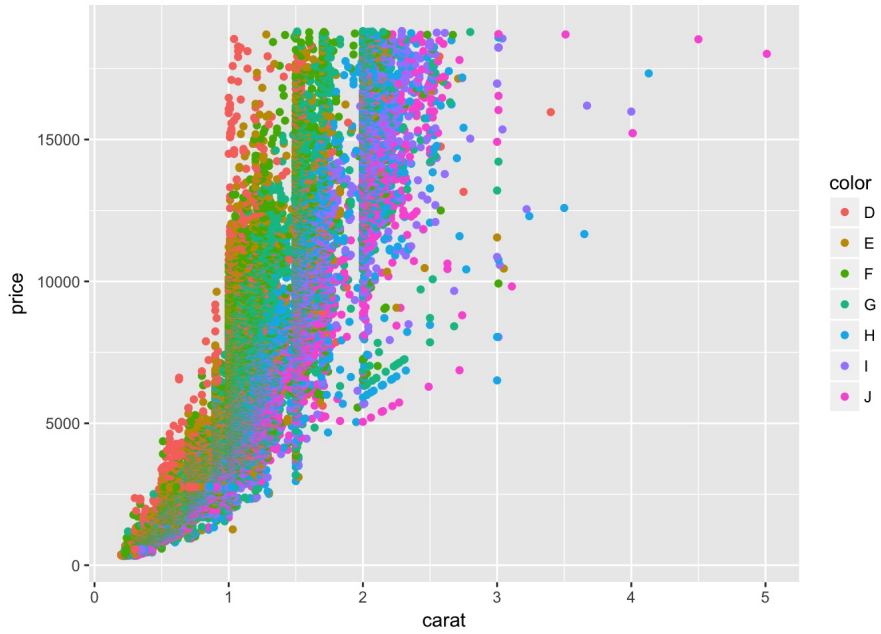
On the topic of appearance of different plots and such, ggplot offers themes for users to use on their plots. Depending on the data visualization you are going for, it might help to stray from the default theme.

First we will read from a csv for the data we will plot in this section.

```
data <- read.csv("/Users/Kevin/stat133/stat133-hws-fall17/post01/data/diamonds.csv")
```

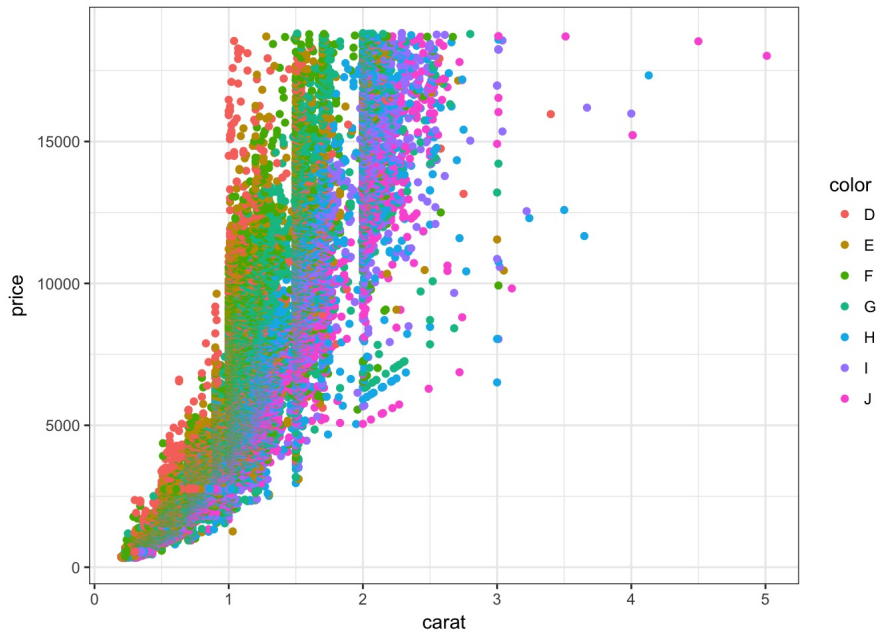
Each of the following themes are the themes included in ggplot. `theme_grey`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_grey()
```



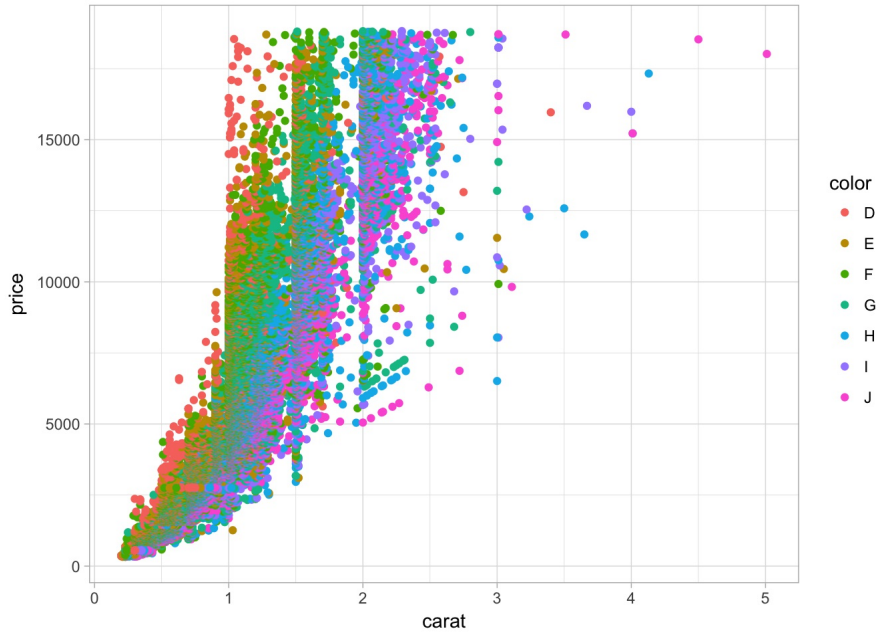
`theme_bw`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_bw()
```



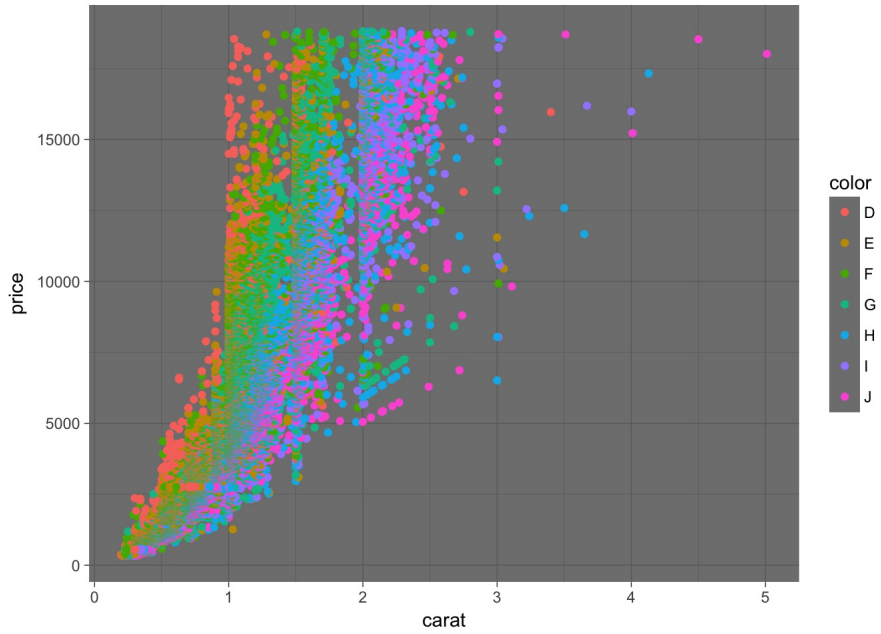
`theme_light`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_light()
```



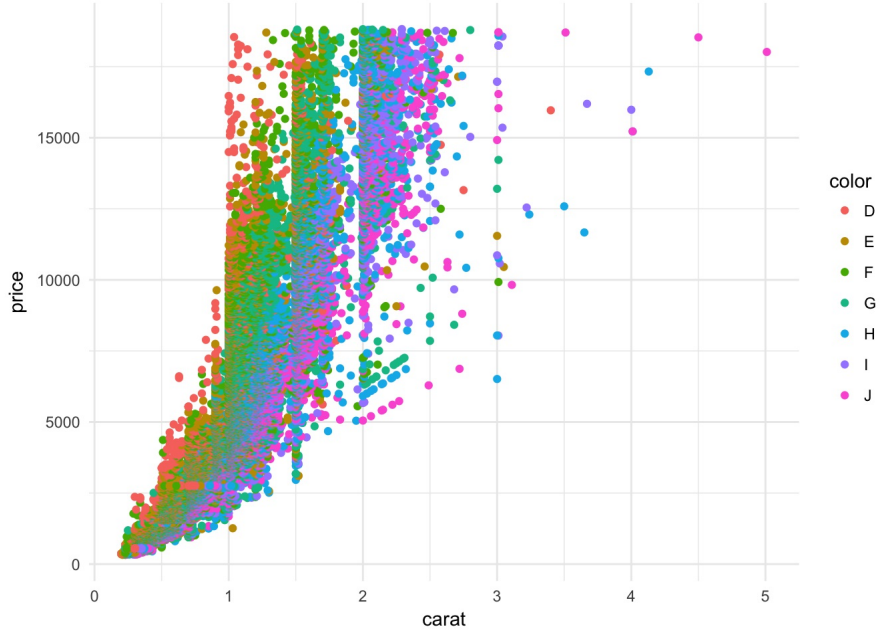
`theme_dark`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_dark()
```



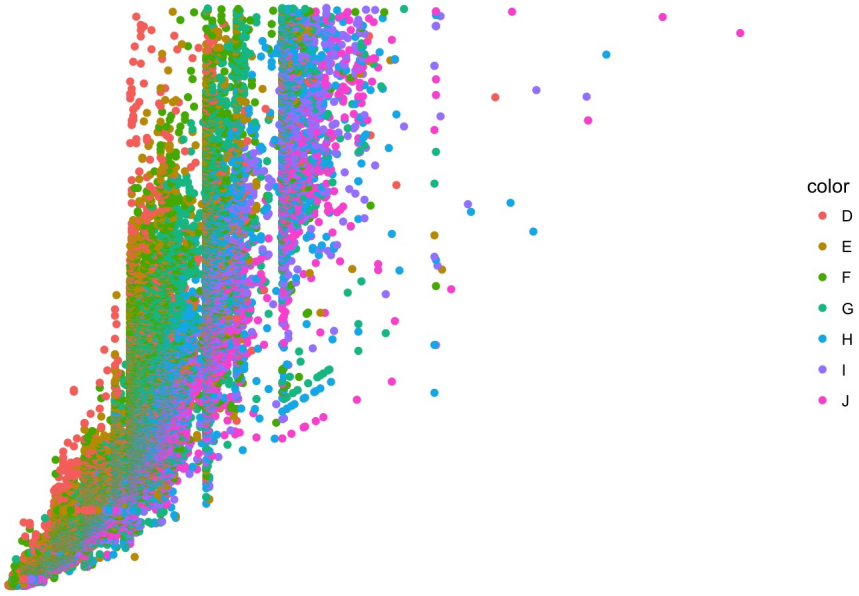
`theme_minimal`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_minimal()
```



`theme_void`

```
ggplot(data=data, aes(carat, price)) + geom_point(aes(color = color)) + theme_void()
```



As you can see, there are even different appearances ggplot can take in the plot background as well. Some of the different themes might just be up to personal taste. `theme_dark` might accentuate colors of your plot better than `theme_linedraw` might and a user might prefer to present their data with that theme. It can also be a way to simply stand out and present the fact that a user took the time to work on the data visualization. And resources online provide many more themes to expand ggplot's flexibility and usage to another level.

Conclusion and Reflection

So far in the course, we have had multiple instances where we had to plot our manipulated data frames. At first, I simply thought of it as problems or just instruction we are supposed to follow. However, we should not think of it in these terms which is easy to fall for in a class. If an instructor or GSI tells a student to plot something, the student might not understand what the plotting allows for in the real world.

Data visualization is as important as manipulating and organizing the data. To illustrate this point, software developers write code and create programs to be used. Others will need to be able to understand what the developer did to be able to work with it or to use it. Otherwise, it is useless! To solve this problem, there are community agreements on code style, APIs written for functions, comments in the file, and other ways to convey the purpose of their work. In all fields, other users need to be able to understand. Data visualization helps in this aspect by converting thousands of lines of data into a plot/chart that can convey the relation and importance of all those lines in a single view.

I have also learned that data visualization in itself is an art and can be quite difficult to do by yourself. While writing my post, I had to view multiple resources and references to get the plots just the way I imagined it in my mind. The R online community aspect is key in creating beautiful and informative plots. There is always someone who thought of the same problem and a group of people working to help.

References

- *ggplot from yhat*. 2014, http://ggplot.yhathq.com/. Accessed 19 Oct. 2017.
- Wickham, Hadley. ggplot2, 2013, http://ggplot2.org/. Accessed 19 Oct. 2017.
- *Data Visualization*. Wikipedia, 2017, https://en.wikipedia.org/wiki/Data_visualization. Accessed 19 Oct. 2017.
- ggplot2 Quick Reference *Software and Programmer Efficiency Research Group*. Sape Research Group, 2015, http://sape.inf.usi.ch/quick-reference/ggplot2/shape. Accessed 19 Oct. 2017.
- *ggplot2*. tidyverse, v 2.2.1.9000, http://ggplot2.tidyverse.org/reference. Accessed 19 Oct. 2017.
- Yau, Nathan. *Flowing Data*. FlowingData, 2007, https://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/. Accessed 19 Oct. 2017.
- *RDocumentation*. 2017, https://www.rdocumentation.org/packages/ggplot2/versions/2.2.1/topics/theme. Accessed 19 Oct. 2017
- *RDocumentation*. 2017, https://www.rdocumentation.org/packages/ggplot2/versions/2.2.1. Accessed 19 Oct. 2017