

post01-sylvia-feng

Sylvia Feng

Introduction





As internet becomes increasingly common around the world, its importance and helpfulness in daily life are also recognized by people. With the spread of internet comes the abundance of data. Realizing how much information they contain, we start to address and utilize these data for greater application in business and many other aspects of life. However, non-data-analysts could hardly acquire any important message directly from the mass, unorganized raw data. This is why we need data visualization techniques to make them more readable and user-friendly.

What is data visualization?

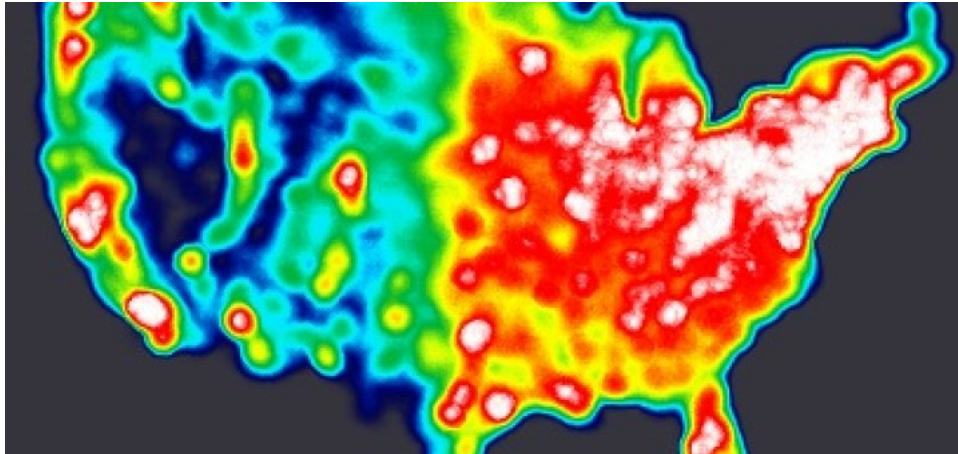
It is actually very self-explanatory that **data visualization** means transforming text-based data into visuals like charts, graphs that help people better understand the data. Some patterns, correlations and trends might be more obvious in visuals than in text.

Now you probably can think of typical data visuals like pie charts, bar charts, but the data visualization tools today are growing more sophisticated. For example:

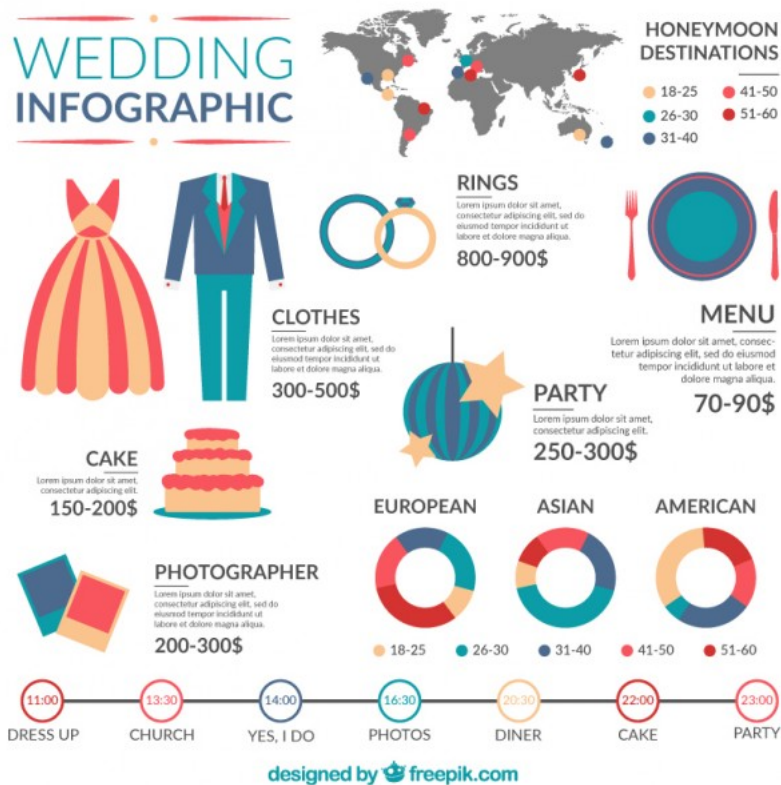
- **sparklines** are widely used in stock market and many other analysis aiming to show the general trend of data;

Name	Income	Income per Period
Austria	200.00	
France	246.67	
Germany	255.83	
UK	250.83	

- **heat maps** can tell us the magnitude of data by color, generally used in weather forecast;



- **infographics** make use of icons and colors to replace words and numbers and deliver message in a direct, captivating way.



Data Visualization in R

R is a powerful tool in visualizing data. So far in class, we have learned multiple types of visuals to display NBA data. Here is a compiled list of all the graphs that we have touched on and some other basic data visualization methods that are commonly used:

##1. Histogram First, let's import a data set to work on:

```
nba <- read.csv(file = "../nba2017-players.csv")
```

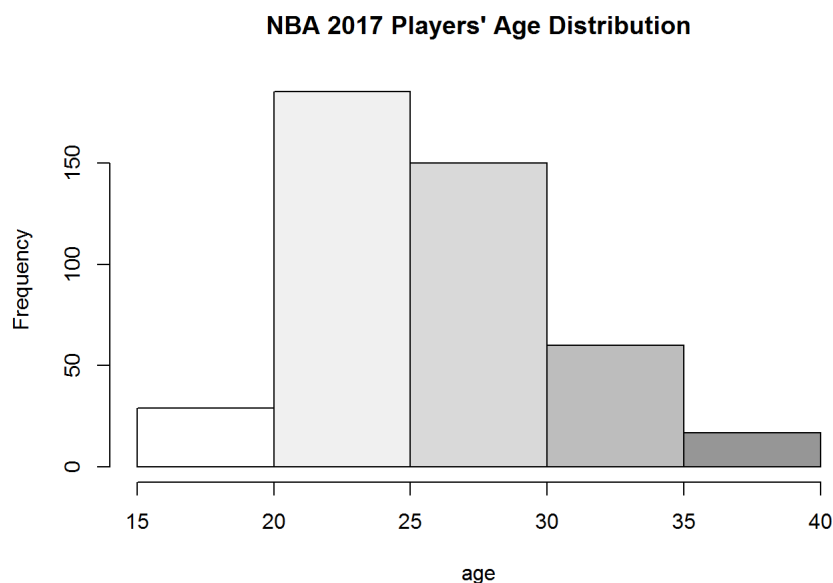
Next, I would like to download a package to add different color to the graph:

```
library(RColorBrewer)
```

RColorBrewer is a package from CRAN that allows user to choose sensible colour schemes for figures in R. This package allows us to further improve the aesthetics of the visualization methods upon what we have done on class.

Now, let's create a histogram to see the age distribution of the 2017 NBA players, contained in this data set.

```
hist(nba$age, breaks = 5, xlab = "age", main = "NBA 2017 Players' Age Distribution", col = brewer.pal(8, "Greys"))
```



As you can see, in this histogram,

age is divided in 5 ranges and older age is represented by darker color.

We can easily tell from the graph that we have over 160 players in the age group 20-25, being the largest age group in NBA 2017 players. Second is between 25-30 with around 150 players, and 35-40 has the least people. This histogram gives us a direct view of the age distribution of the 2017 NBA players, including max, min and approximate frequency. Imagine if we didn't visualize the data set - we might need to manually

count every player!

2. Bar Chart

In this section, I want to first import a new data set of 2017 NBA players with data grouped by teams, so that we can reduce the number of bars on the bar chart and have a clearer visualization.

Here I also load a package "dplyr" for data organization purpose.

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

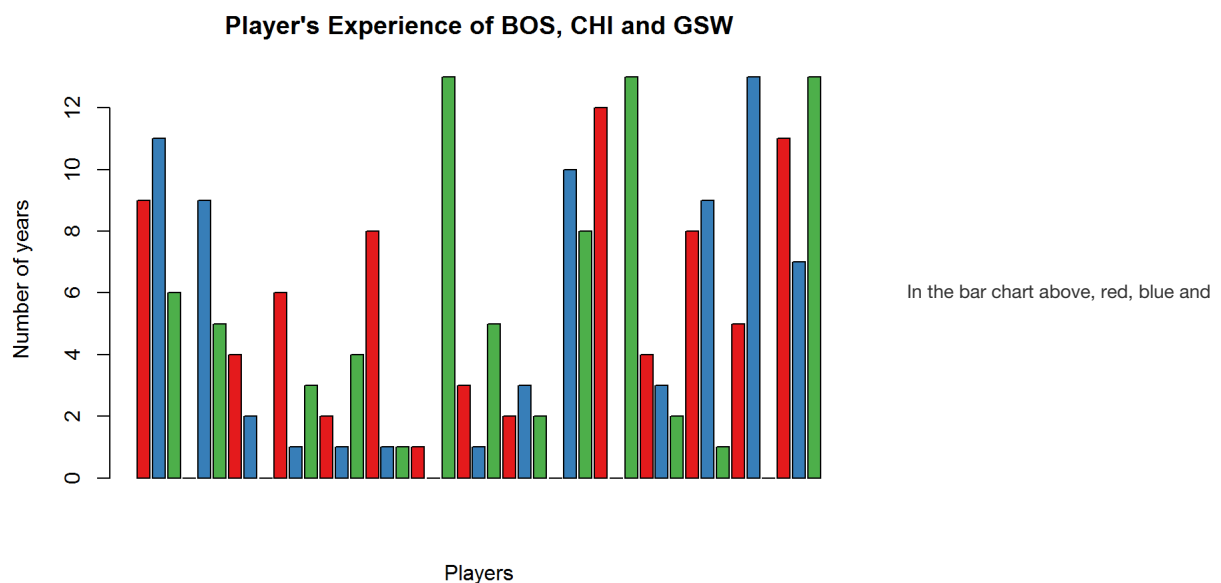
```
##  
## Attaching package: 'dplyr'
```

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

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

```
team <- read.csv(file = "../hw03/data/nba2017-teams.csv")  
a <- filter(nba, team == "BOS" | team == "CHI" | team == "GSW")
```

```
A <- barplot(a$experience, main = "Player's Experience of BOS, CHI and GSW", col = brewer.pal(3, "Set1"), xlab = "Players", ylab = "Number of years")
```



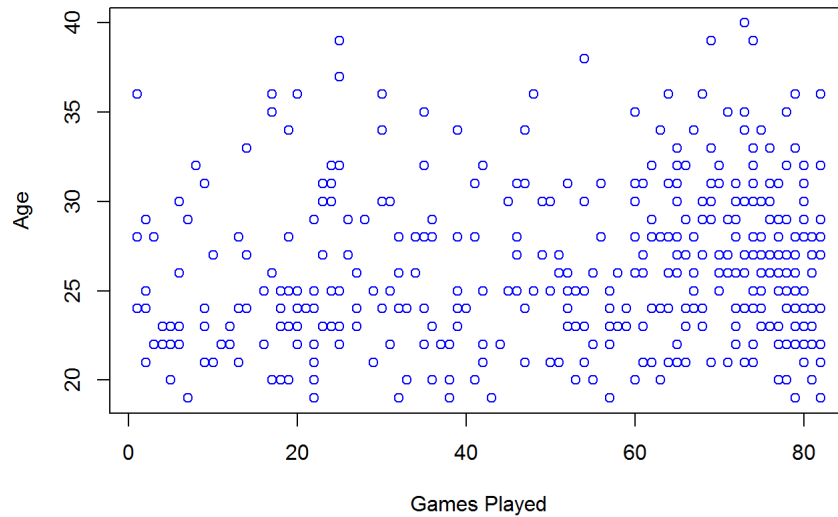
green represent BOS, CHI and GSW, respectively. Without directly pointing out the name of the team, we can easily tell from the graph by the color it uses.

3. Scatterplot

Scatterplot is simpler way of putting all the data on a graph. It gives us a rough idea of the spread of data, i.e. whether they are more concentrated in one part, or they are more spread out. For instance, look at the scatterplot below between Players' age and the total games they played.

```
plot(nba$games, nba$age, xlab = "Games Played", ylab = "Age", main = "Relationship between Age and Games Played", col = 4)
```

Relationship between Age and Games Played



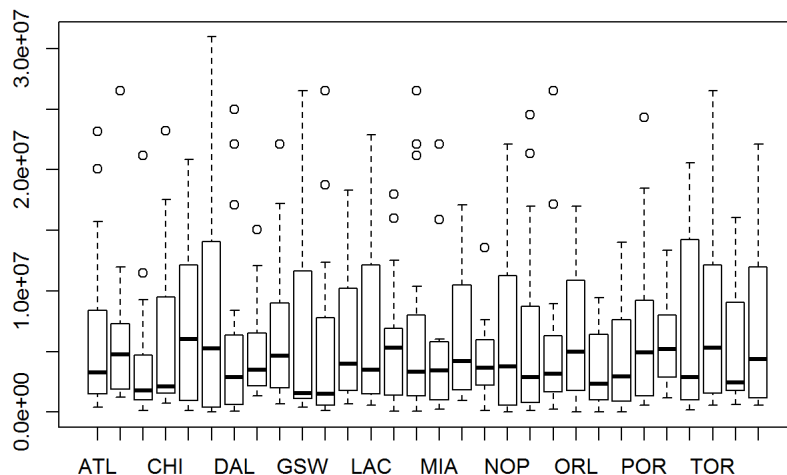
We can see that most players are

between 20-30 years old with 60-80 games played in their career life. This way of visualizing data is still less processed and might not be as direct as looking at a bar chart or histogram.

4. Box Plot

Box Plot is also another way to see the distribution of data. Especially, it tells us more exact statistics including median, 25th percentile, 75th percentile, range and outliers.

```
boxplot(nba$salary~nba$team)
```

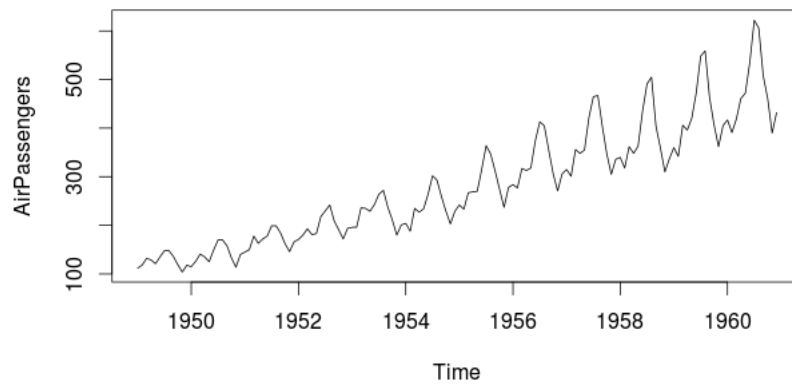


This box plot of salary distribution by

teams in NBA gives us a visualization of what is the typical salary of player at each team, and which team's salary is more spread out, and if any team has specially high-paid players. It simplifies raw data but also save key measurements of data.

5. Line Chart

Line Charts are commonly preferred when we are to analyze a trend spread over a time period. Furthermore, line plot is also suitable to plots where we need to compare relative changes in quantities across some variable (like time). For instance, the line chart below is showing the trend in the number of air passengers around the time period 1950-1960.

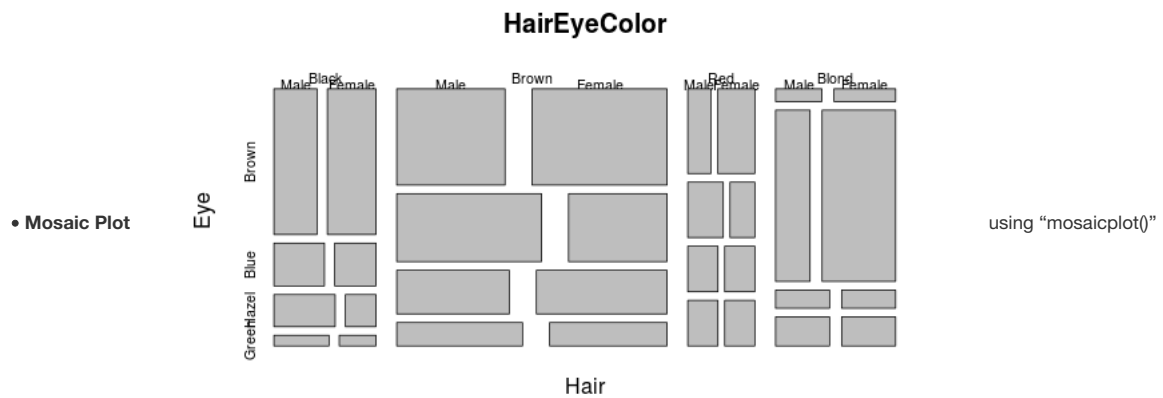


From this line chart, we can clearly see

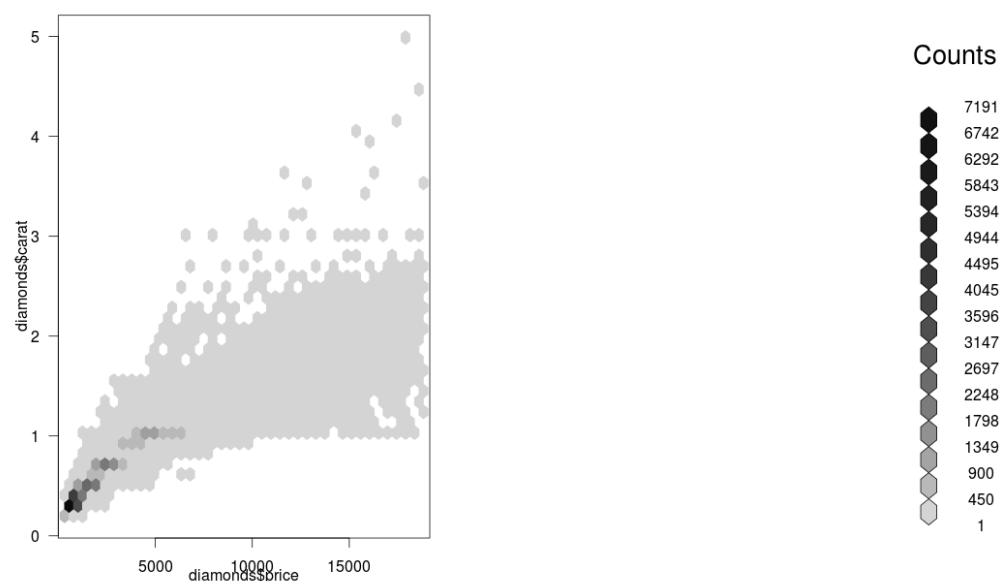
that there is a generally increasing trend of air passengers, without need to actually look at the numbers.

Advanced Visualization

After going over some basic data visualization methods, there are a lot of other more complicated ones. Due to the limitation of my laptop capacity, I am sorry but fail to execute the exact advanced packages and complex codes should be used. However, I would like to demonstrate briefly how to approach these visualization, and provide an introduction for those who are interested. Through downloading packages on R, we can manage to draw graphs like:

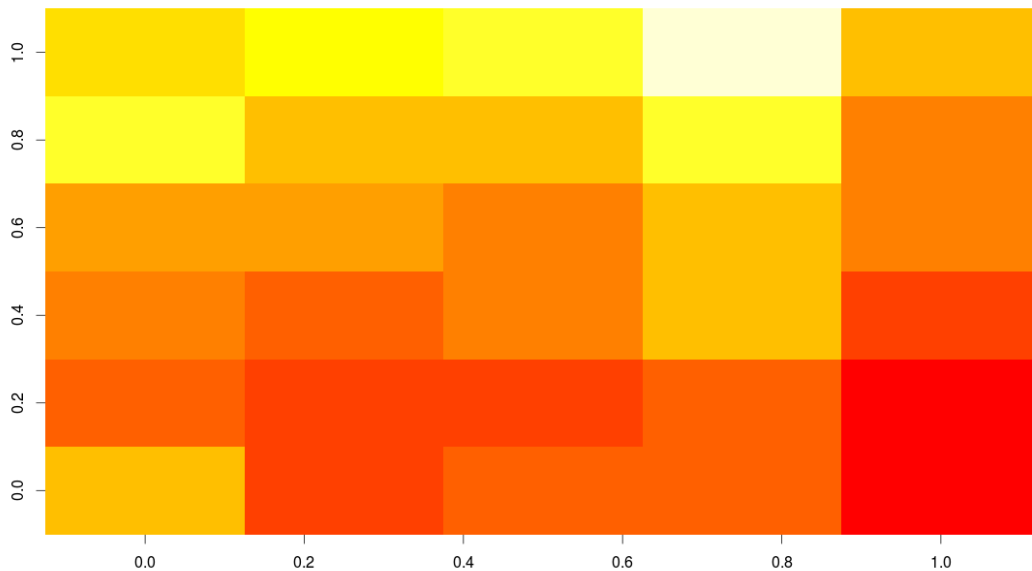


• Hexagon Binning



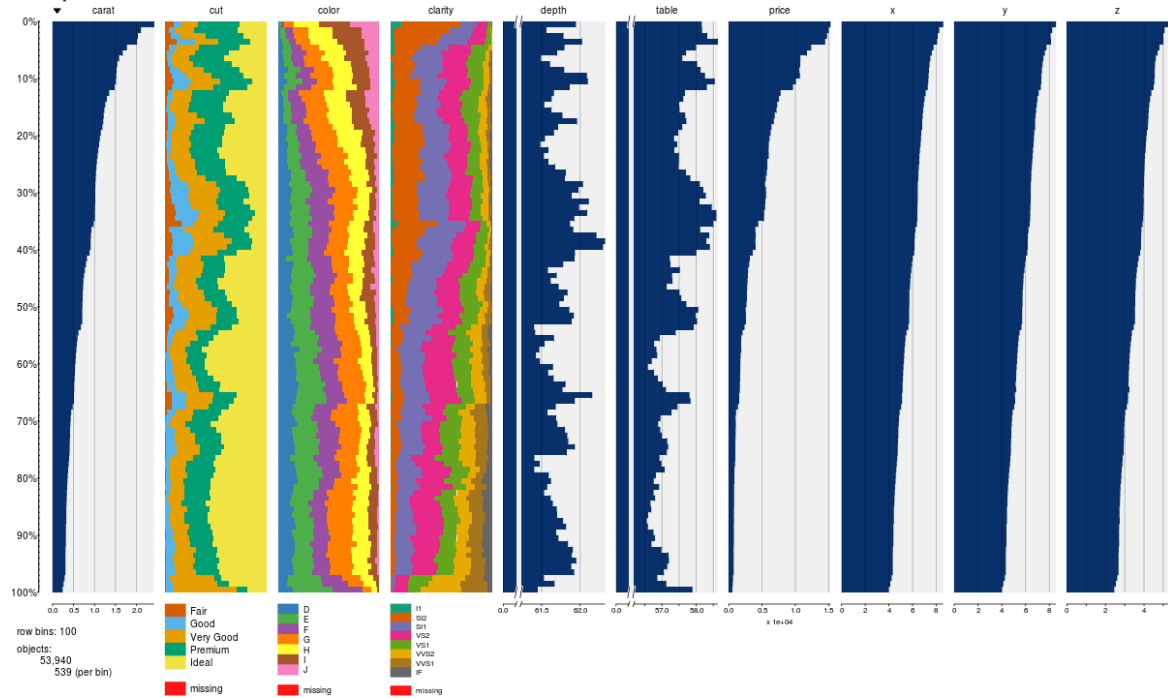
using functions in **hexbin** package

• Heat Map



using “heatmap()”

• table plot

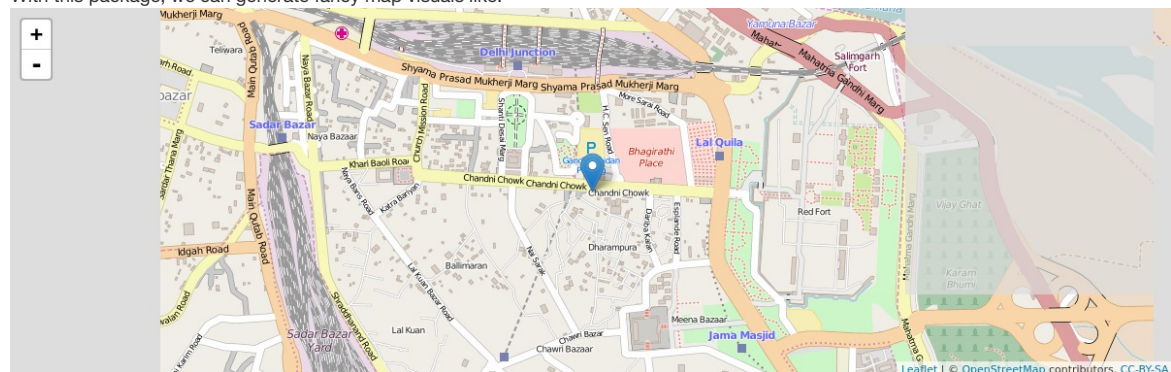


to summarize data using tableplot function from “tableplot” package , etc.

More complex techniques include:

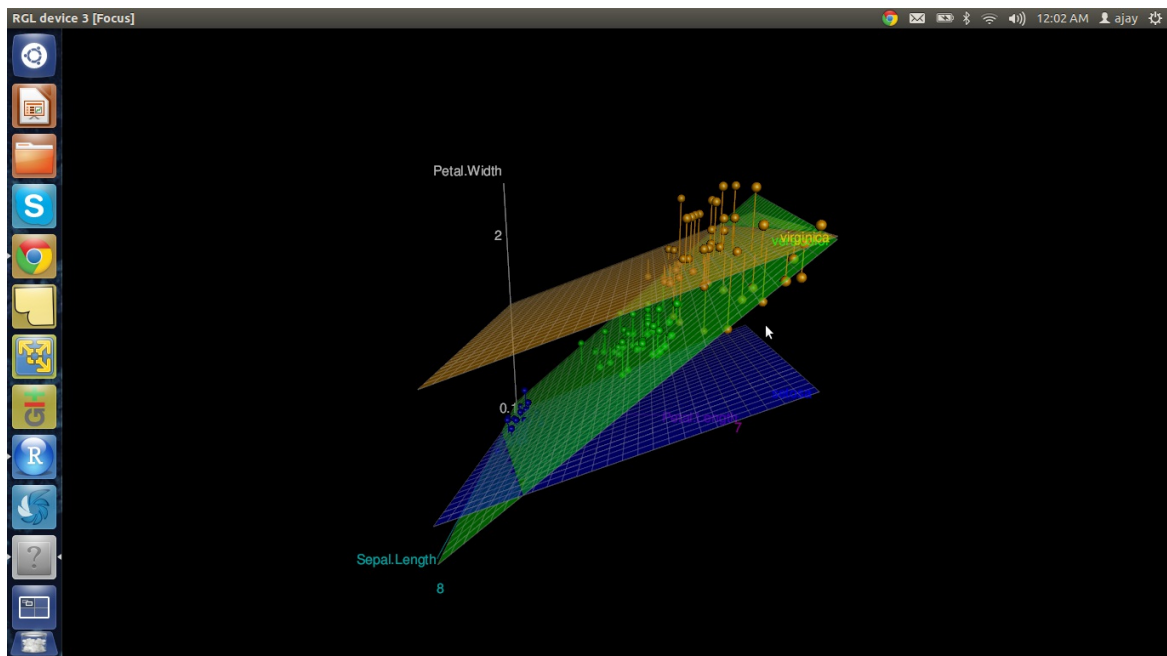
• Map Visualization: Some open-source Javascript libraries offer Leaflet for interactive maps.

With this package, we can generate fancy map visuals like:

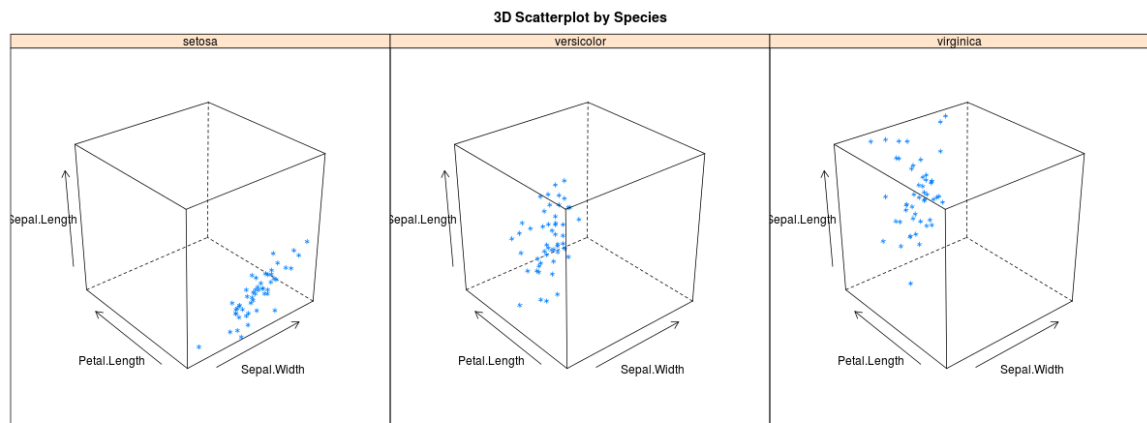


• 3D Plot: One of the easiest ways of impressing someone by R’s capabilities is by creating a 3D graph. We use the package R Commander which acts as Graphical User Interface (GUI). Here are the steps:

1. Simply install the Rcmdr package
2. Use the 3D plot option from within graphs



You can also make 3D graphs using Lattice package . Lattice can also be used for xyplots.



Conclusion

In this post, I have discussed the various forms of data visualization inside and outside of classroom of stat133 by covering the basic to some advanced levels of charts and graphs. They are all useful to display the data using R Programming, and that is what makes R the best data visualization software in the world. While Python may make progress with seaborn and ggplot, nothing beats the sheer immense number of packages in R for statistical data visualization.

There are much more complex and delicate data visualization techniques today that are widely employed by data analysts and statisticians. If you would like to find out a more comprehensive and interactive demonstration of what have been cover is this post, there is a short video that you might want to look at on [Data Visualization Using R](#).