Uber Data Analysis through Visualisations in R

(STEP-BY-STEP PROJECT DOCUMENTATION INCLUDING SOURCE CODE )

Milan (21BCS4362)

2022-07-1

Data storytelling is an important component of Machine Learning through which companies are able to understand the background of various operations. With the help of visualization, companies can avail the benefit of understanding the complex data and gain insights that would help them to craft decisions. This is more of a data visualization project that will guide you towards using the ggplot2 library for understanding the data and for developing an intuition for understanding the customers who avail the trips.

## Importing necessary libraries

In the first step of our R project, we will import the essential packages that we will use in this uber data analysis project. Some of the important libraries of R that we will use are –

* **ggplot2** :- This is the backbone of this project. ggplot2 is the most popular data visualization library that is most widely used for creating aesthetic visualization plots.
* **ggthemes** :- This is more of an add-on to our main ggplot2 library. With this, we can create better create extra themes and scales with the mainstream ggplot2 package.
* **lubridate** :- Our dataset involves various time-frames. In order to understand our data in separate time categories, we will make use of the lubridate package.
* **dplyr** :- This package is the lingua franca of data manipulation in R.
* **tidyr** :- This package will help you to tidy your data. The basic principle of tidyr is to tidy the columns where each variable is present in a column, each observation is represented by a row and each value depicts a cell.
* **DT** :- With the help of this package, we will be able to interface with the JavaScript Library called – Datatables.
* **scales** :- With the help of graphical scales, we can automatically map the data to the correct scales with well-placed axes and legends.

library(ggplot2)  
library(ggthemes)  
library(lubridate)

##   
## Attaching package: 'lubridate'

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

library(dplyr)

##   
## 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

library(tidyr)  
library(tidyverse) # metapackage of all tidyverse packages

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ tibble 3.1.7 ✔ stringr 1.4.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1  
## ✔ purrr 0.3.4

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ lubridate::as.difftime() masks base::as.difftime()  
## ✖ lubridate::date() masks base::date()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ lubridate::intersect() masks base::intersect()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ lubridate::setdiff() masks base::setdiff()  
## ✖ lubridate::union() masks base::union()

library(DT)  
library(scales)

##   
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':  
##   
## discard

## The following object is masked from 'package:readr':  
##   
## col\_factor

## Creating vector of colors to be implemented in our plots

In this step of data science project, we will create a vector of our colors that will be included in our plotting functions. You can also select your own set of colors.

colors = c("#CC1011", "#665555", "#05a399", "#cfcaca", "#f5e840", "#0683c9", "#e075b0")  
colors

## [1] "#CC1011" "#665555" "#05a399" "#cfcaca" "#f5e840" "#0683c9" "#e075b0"

## Read the data from each time-frame

Now, we will read several csv files that contain the data from April 2014 to September 2014. We will store these in corresponding data frames like apr\_data, may\_data, etc. After we have read the files, we will combine all of this data into a single dataframe called ‘data\_2014’.

Then, in the next step, we will perform the appropriate formatting of Date.Time column. Then, we will proceed to create factors of time objects like day, month, year etc.

apr\_data <- read.csv("uber-raw-data-apr14.csv")  
may\_data <- read.csv("uber-raw-data-may14.csv")  
jun\_data <- read.csv("uber-raw-data-jun14.csv")  
jul\_data <- read.csv("uber-raw-data-jul14.csv")  
aug\_data <- read.csv("uber-raw-data-aug14.csv")  
sep\_data <- read.csv("uber-raw-data-sep14.csv")  
  
data\_2014 <- rbind(apr\_data,may\_data, jun\_data, jul\_data, aug\_data, sep\_data)

The data contains the columns Date.Time which is a factor, Latitude and Longitudes which are double and Base which is factor. we will format the datetime into a more readable format using the Date Time conversion function.

data\_2014$Date.Time <- as.POSIXct(data\_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S")  
  
data\_2014$Time <- format(as.POSIXct(data\_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S"), format="%H:%M:%S")  
  
data\_2014$Date.Time <- ymd\_hms(data\_2014$Date.Time)  
  
data\_2014$day <- factor(day(data\_2014$Date.Time))  
data\_2014$month <- factor(month(data\_2014$Date.Time, label = TRUE))  
data\_2014$year <- factor(year(data\_2014$Date.Time))  
data\_2014$dayofweek <- factor(wday(data\_2014$Date.Time, label = TRUE))

data\_2014$hour <- factor(hour(hms(data\_2014$Time)))  
data\_2014$minute <- factor(minute(hms(data\_2014$Time)))  
data\_2014$second <- factor(second(hms(data\_2014$Time)))

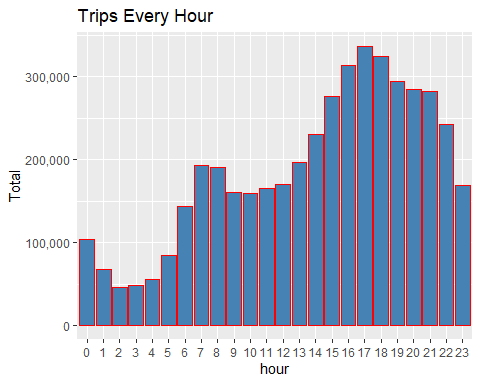
# Data Visualisation

## Plotting the trips by the hours in a day

In the next step or R project, we will use the ggplot function to plot the number of trips that the passengers had made in a day. We will also use dplyr to aggregate our data. In the resulting visualizations, we can understand how the number of passengers fares throughout the day. We observe that the number of trips are higher in the evening around 5:00 and 6:00 PM.

hour\_data <- data\_2014 %>%  
 group\_by(hour) %>%  
 dplyr::summarize(Total = n())

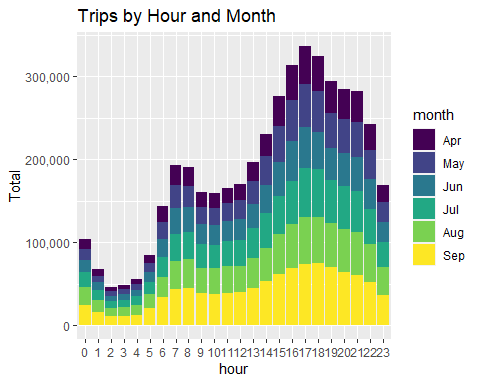
ggplot(hour\_data, aes(hour, Total)) +   
 geom\_bar( stat = "identity", fill = "steelblue", color = "red") +  
 ggtitle("Trips Every Hour") +  
 theme(legend.position = "none") +  
 scale\_y\_continuous(labels = comma)



month\_hour <- data\_2014 %>%  
 group\_by(month, hour) %>%  
 dplyr::summarize(Total = n())

## `summarise()` has grouped output by 'month'. You can override using the  
## `.groups` argument.

ggplot(month\_hour, aes(hour, Total, fill = month)) +   
 geom\_bar( stat = "identity") +  
 ggtitle("Trips by Hour and Month") +  
 scale\_y\_continuous(labels = comma)

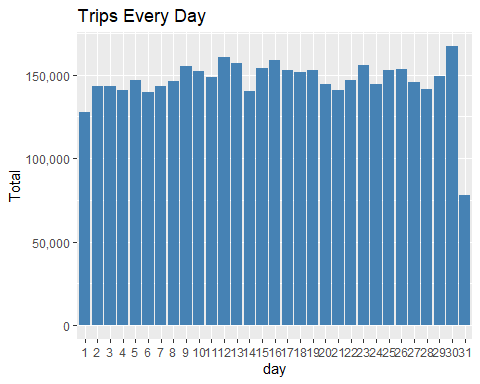


## Plotting data by trips during every day of the month

In this section of DataFlair R project, we will learn how to plot our data based on every day of the month. We observe from the resulting visualization that 30th of the month had the highest trips in the year which is mostly contributed by the month of April.

day\_group <- data\_2014 %>%  
 group\_by(day) %>%  
 dplyr::summarize(Total = n())

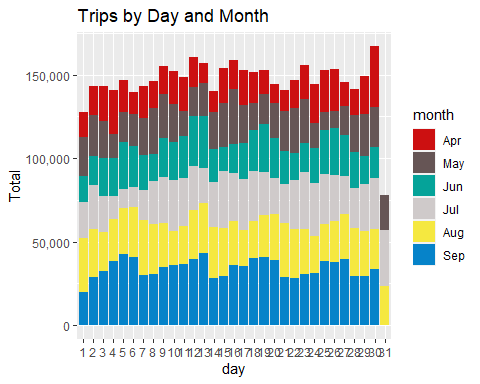
ggplot(day\_group, aes(day, Total)) +   
 geom\_bar( stat = "identity", fill = "steelblue") +  
 ggtitle("Trips Every Day") +  
 theme(legend.position = "none") +  
 scale\_y\_continuous(labels = comma)



day\_month\_group <- data\_2014 %>%  
 group\_by(month, day) %>%  
 dplyr::summarize(Total = n())

## `summarise()` has grouped output by 'month'. You can override using the  
## `.groups` argument.

ggplot(day\_month\_group, aes(day, Total, fill = month)) +   
 geom\_bar( stat = "identity") +  
 ggtitle("Trips by Day and Month") +  
 scale\_y\_continuous(labels = comma) +  
 scale\_fill\_manual(values = colors)

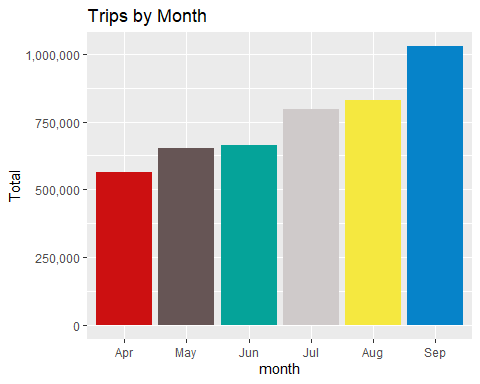


## Number of Trips taking place during months in a year

In this section, we will visualize the number of trips that are taking place each month of the year. In the output visualization, we observe that most trips were made during the month of September. Furthermore, we also obtain visual reports of the number of trips that were made on every day of the week.

month\_group <- data\_2014 %>%  
 group\_by(month) %>%  
 dplyr::summarize(Total = n())

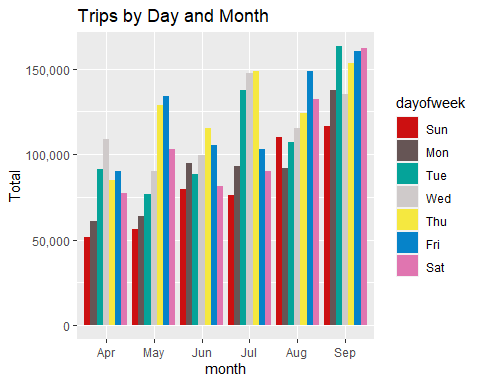
ggplot( month\_group, aes(month, Total, fill = month)) +   
 geom\_bar( stat = "identity") +  
 ggtitle("Trips by Month") +  
 theme(legend.position = "none") +  
 scale\_y\_continuous(labels = comma) +  
 scale\_fill\_manual(values = colors)



month\_weekday <- data\_2014 %>%  
 group\_by(month, dayofweek) %>%  
 dplyr::summarize(Total = n())

## `summarise()` has grouped output by 'month'. You can override using the  
## `.groups` argument.

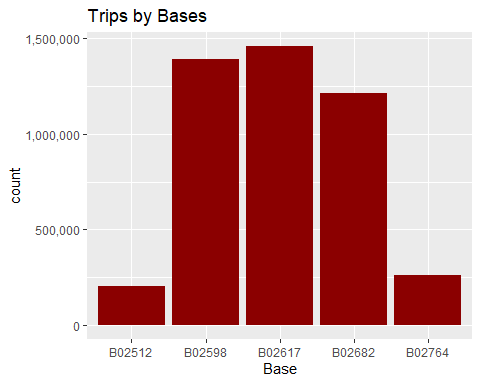
ggplot(month\_weekday, aes(month, Total, fill = dayofweek)) +   
 geom\_bar( stat = "identity", position = "dodge") +  
 ggtitle("Trips by Day and Month") +  
 scale\_y\_continuous(labels = comma) +  
 scale\_fill\_manual(values = colors)



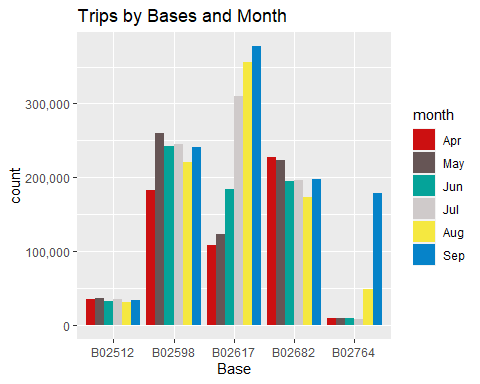
## Finding out the number of Trips by bases

In the following visualization, we plot the number of trips that have been taken by the passengers from each of the bases. There are five bases in all out of which, we observe that B02617 had the highest number of trips. Furthermore, this base had the highest number of trips in the month B02617. Thursday observed highest trips in the three bases – B02598, B02617, B02682.

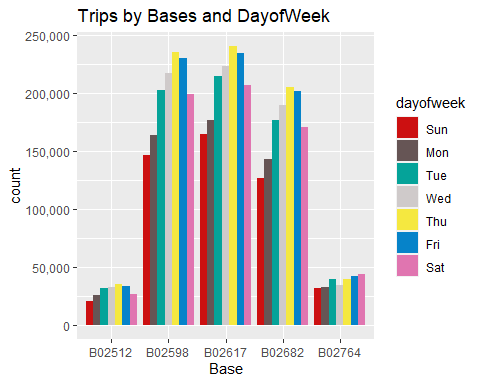
ggplot(data\_2014, aes(Base)) +   
 geom\_bar(fill = "darkred") +  
 scale\_y\_continuous(labels = comma) +  
 ggtitle("Trips by Bases")



ggplot(data\_2014, aes(Base, fill = month)) +   
 geom\_bar(position = "dodge") +  
 scale\_y\_continuous(labels = comma) +  
 ggtitle("Trips by Bases and Month") +  
 scale\_fill\_manual(values = colors)



ggplot(data\_2014, aes(Base, fill = dayofweek)) +   
 geom\_bar(position = "dodge") +  
 scale\_y\_continuous(labels = comma) +  
 ggtitle("Trips by Bases and DayofWeek") +  
 scale\_fill\_manual(values = colors)



## Creating a Heatmap visualization of day, hour and month

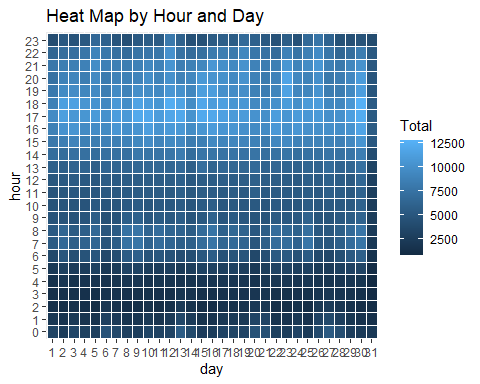
In this section, we will learn how to plot heatmaps using ggplot(). We will plot five heatmap plots –

* First, we will plot Heatmap by Hour and Day.
* Second, we will plot Heatmap by Month and Day.
* Third, a Heatmap by Month and Day of the Week.
* Fourth, a Heatmap that delineates Month and Bases.
* Finally, we will plot the heatmap, by bases and day of the week.

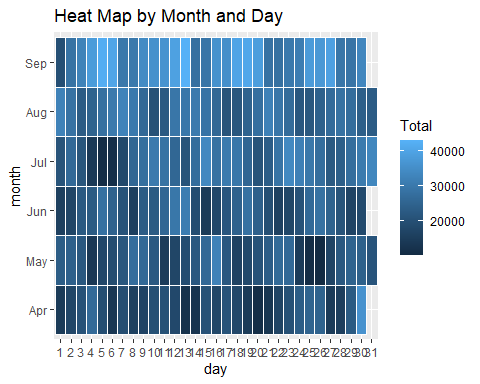
day\_and\_hour <- data\_2014 %>%  
 group\_by(day, hour) %>%  
 dplyr::summarize(Total = n())

## `summarise()` has grouped output by 'day'. You can override using the `.groups`  
## argument.

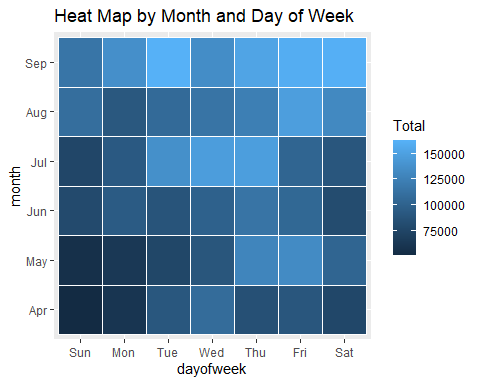
ggplot(day\_and\_hour, aes(day, hour, fill = Total)) +  
 geom\_tile(color = "white") +  
 ggtitle("Heat Map by Hour and Day")



ggplot(day\_month\_group, aes(day, month, fill = Total)) +  
 geom\_tile(color = "white") +  
 ggtitle("Heat Map by Month and Day")



ggplot(month\_weekday, aes(dayofweek, month, fill = Total)) +  
 geom\_tile(color = "white") +  
 ggtitle("Heat Map by Month and Day of Week")

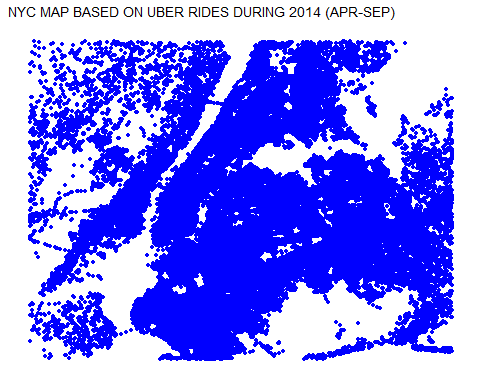


## Creating a map visualization of rides in New York

In the final section, we will visualize the rides in New York city by creating a geo-plot that will help us to visualize the rides during 2014 (Apr – Sep) and by the bases in the same period.

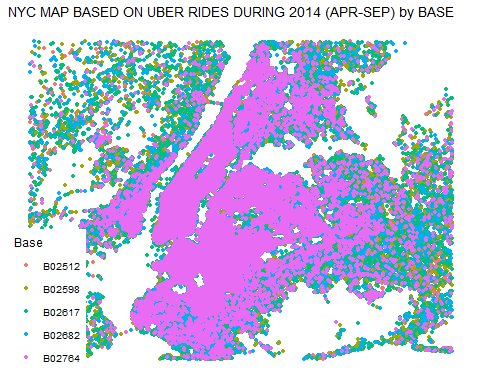
min\_lat <- 40.5774  
max\_lat <- 40.9176  
min\_long <- -74.15  
max\_long <- -73.7004  
  
ggplot(data\_2014, aes(x=Lon, y=Lat)) +  
 geom\_point(size=1, color = "blue") +  
 scale\_x\_continuous(limits=c(min\_long, max\_long)) +  
 scale\_y\_continuous(limits=c(min\_lat, max\_lat)) +  
 theme\_map() +  
 ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)")

## Warning: Removed 71701 rows containing missing values (geom\_point).



ggplot(data\_2014, aes(x=Lon, y=Lat, color = Base)) +  
 geom\_point(size=1) +  
 scale\_x\_continuous(limits=c(min\_long, max\_long)) +  
 scale\_y\_continuous(limits=c(min\_lat, max\_lat)) +  
 theme\_map() +  
 ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP) by BASE")

## Warning: Removed 71701 rows containing missing values (geom\_point).



# Summary

At the end of the Uber data analysis R project, we observed how to create data visualizations. We made use of packages like ggplot2 that allowed us to plot various types of visualizations that pertained to several time-frames of the year. With this, we could conclude how time affected customer trips. Finally, we made a geo plot of New York that provided us with the details of how various users made trips from different bases.