Uber Data Analysis - STAT5000 Project

Vidit Vivek Sharma

03/12/2021

Uber Pickup Data Analysis

Here in this project, I will analyze the Uber Pickups in New York City dataset. This project will use basic data representation and transformation packages like tidyverse and ggplot. 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 helps them to guide you towards understanding the data and for developing an intuition for understanding the customers who avail the trips.

What is established with this report.

1. Data understanding with the help of various visualizations.
2. Data Prediction on the basis of present data.

Data Sources Used

In this analysis, I have used the Uber Pickup data for the city of New York for the months of April 2019 to the month of September 2019. I got this data from Kaggle.

It consists of the date and time of the pickup in one column and also contains the longitude and latitude data of the pickup. To make the report reproducible, I have uploaded the datasets to github thus this code can be executed from anywhere and the code will execute without any errors.

Link to the datasets: 6 files [https://github.com/vidit-sharma/Uber-Data-Analysis (https://github.com/vidit-sharma/Uber-Data-Analysis)]

Libraries used in this analysis

The first step is to import the essential packages that will be used in this uber data analysis project.

1. **ggplot2**: ggplot2 is the most popular data visualization library that is most widely used for creating aesthetic visualization plots.
2. **ggthemes**: Extra Themes, Scales, and Geoms for 'ggplot2'.
3. **tidyverse**: load This package to use tidyverse functions
4. **lubridate**: This R package makes it easier to work with dates and times
5. **dplyr**: Data Manipulation
6. **tidyr**: This package will help you to tidy your data.
7. **DT**: Data tables in JS
8. **scales**: With the help of graphical scales, we can automatically map the data to the correct scales with well-placed axes and legends.
9. **leaflet**: This package helps in creating interactive Web maps with JavaScript.

```
library(ggplot2)
library(ggthemes)
```

```
## Warning: package 'ggthemes' was built under R version 4.1.1
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## Warning: package 'tibble' was built under R version 4.1.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

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

```
library(dplyr)
library(tidyr)
library(DT)
```

```
## Warning: package 'DT' was built under R version 4.1.1
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

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

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(leaflet)
```

```
## Warning: package 'leaflet' was built under R version 4.1.2
```

Loading the Datasets

Next step is to load the 6 .csv files in R from Github into the corresponding tables named apr_data, may_data, jun_data, jul_data, aug_data, and sep_data.

```
#Loading the data sets from github to R

url_in <- "https://github.com/vidit-sharma/Uber-Data-Analysis/raw/main/uber-raw-data-"
file_names <- c("apr14.csv",
                "may14.csv",
                "jun14.csv",
                "jul14.csv",
                "aug14.csv",
                "sep14.csv")
urls <- str_c(url_in,file_names)

apr_data <- read_csv(urls[1], show_col_types = FALSE)
may_data <- read_csv(urls[2], show_col_types = FALSE)
jun_data <- read_csv(urls[3], show_col_types = FALSE)
jul_data <- read_csv(urls[4], show_col_types = FALSE)
aug_data <- read_csv(urls[5], show_col_types = FALSE)
sep_data <- read_csv(urls[6], show_col_types = FALSE)
```

Merging the Datasets

Now we have to combine these datasets as we will be performing the operations as a whole on the entire data. Also we have to change the name of the column name of the first column from ‘Date.Time’ to ‘Date_Time’ as this will make performing operations on the column possible.

```
#combining all 6 csv files to form one data set

combined_data <- rbind(apr_data, may_data, jun_data,
                       jul_data, aug_data, sep_data)
cat("The dimensions of the data are:", dim(combined_data))
```

```
## The dimensions of the data are: 4534327 4
```

```
colnames(combined_data)[1] <- "Date_Time"

head(combined_data)
```

Date_Time <chr>	Lat <dbl>	Lon <dbl>	Base <chr>
4/1/2014 0:11:00	40.7690	-73.9549	B02512
4/1/2014 0:17:00	40.7267	-74.0345	B02512
4/1/2014 0:21:00	40.7316	-73.9873	B02512
4/1/2014 0:28:00	40.7588	-73.9776	B02512
4/1/2014 0:33:00	40.7594	-73.9722	B02512
4/1/2014 0:33:00	40.7383	-74.0403	B02512

6 rows

Next, the format of the Date_Time has to be transformed into a more readable format using the Date Time conversion function.

```
#convert Date_Time to desired format
combined_data$Date_Time <- as.POSIXct(combined_data$Date_Time,
                                      format="%m/%d/%Y %H:%M:%S")

combined_data$Time <- format(as.POSIXct(combined_data$Date_Time,
                                        format = "%m/%d/%Y %H:%M:%S"),
                                        format="%H:%M:%S")

combined_data$Date_Time <- ymd_hms(combined_data$Date_Time)

head(combined_data)
```

Date_Time <dttm>	Lat <dbl>	Lon <dbl>	Base <chr>	Time <chr>
2014-04-01 00:11:00	40.7690	-73.9549	B02512	00:11:00
2014-04-01 00:17:00	40.7267	-74.0345	B02512	00:17:00
2014-04-01 00:21:00	40.7316	-73.9873	B02512	00:21:00
2014-04-01 00:28:00	40.7588	-73.9776	B02512	00:28:00
2014-04-01 00:33:00	40.7594	-73.9722	B02512	00:33:00
2014-04-01 00:33:00	40.7383	-74.0403	B02512	00:33:00

6 rows

As I need to perform operations and visualization on the date and time, thus next I proceeded to create factors of date and time objects like day, month, year and second, minute, hour.

```
# Create new factor columns for month, day, year, and dayofweek
combined_data$day <- factor(day(combined_data$Date_Time))
combined_data$month <- factor(month(combined_data$Date_Time, label=TRUE))
combined_data$year <- factor(year(combined_data$Date_Time))
combined_data$dayofweek <- factor(wday(combined_data$Date_Time, label=TRUE))

# Create new factor columns for second, minute, and hour
combined_data$second = factor(second(hms(combined_data$Time)))
combined_data$minute = factor(minute(hms(combined_data$Time)))
combined_data$hour = factor(hour(hms(combined_data$Time)))

head(combined_data)
```

Date_Time <dttm>	Lat <dbl>	Lon <dbl>	Base <chr>	Time <chr>	… <fct>	mo… <ord>	y… <fct>	dayofwe… <ord>	sec… <fct>
2014-04-01 00:11:00	40.7690	-73.9549	B02512	00:11:00	1	Apr	2014	Tue	0
2014-04-01 00:17:00	40.7267	-74.0345	B02512	00:17:00	1	Apr	2014	Tue	0
2014-04-01 00:21:00	40.7316	-73.9873	B02512	00:21:00	1	Apr	2014	Tue	0
2014-04-01 00:28:00	40.7588	-73.9776	B02512	00:28:00	1	Apr	2014	Tue	0

Date_Time <dttm>	Lat <dbl>	Lon <dbl>	Base <chr>	Time <chr>	… <fct>	mo… <ord>	y… <fct>	dayofwe… <ord>	sec… <fct>
2014-04-01 00:33:00	40.7594	-73.9722	B02512	00:33:00	1	Apr	2014	Tue	0
2014-04-01 00:33:00	40.7383	-74.0403	B02512	00:33:00	1	Apr	2014	Tue	0

6 rows | 1-10 of 12 columns

Visualization of Data Using GGPLOT

For the visualization, I have created a table for the colors used to represent different months on the plot.

```
#define the colors for use in later visualization
colors = c("#CC3300", "#666666", "#05a399", "#cfcaca",
           "#FFFF00", "#3366CC", "#000066")
colors
```

```
## [1] "#CC3300" "#666666" "#05a399" "#cfcaca" "#FFFF00" "#3366CC" "#000066"
```

Plotting data by trips during months in a year

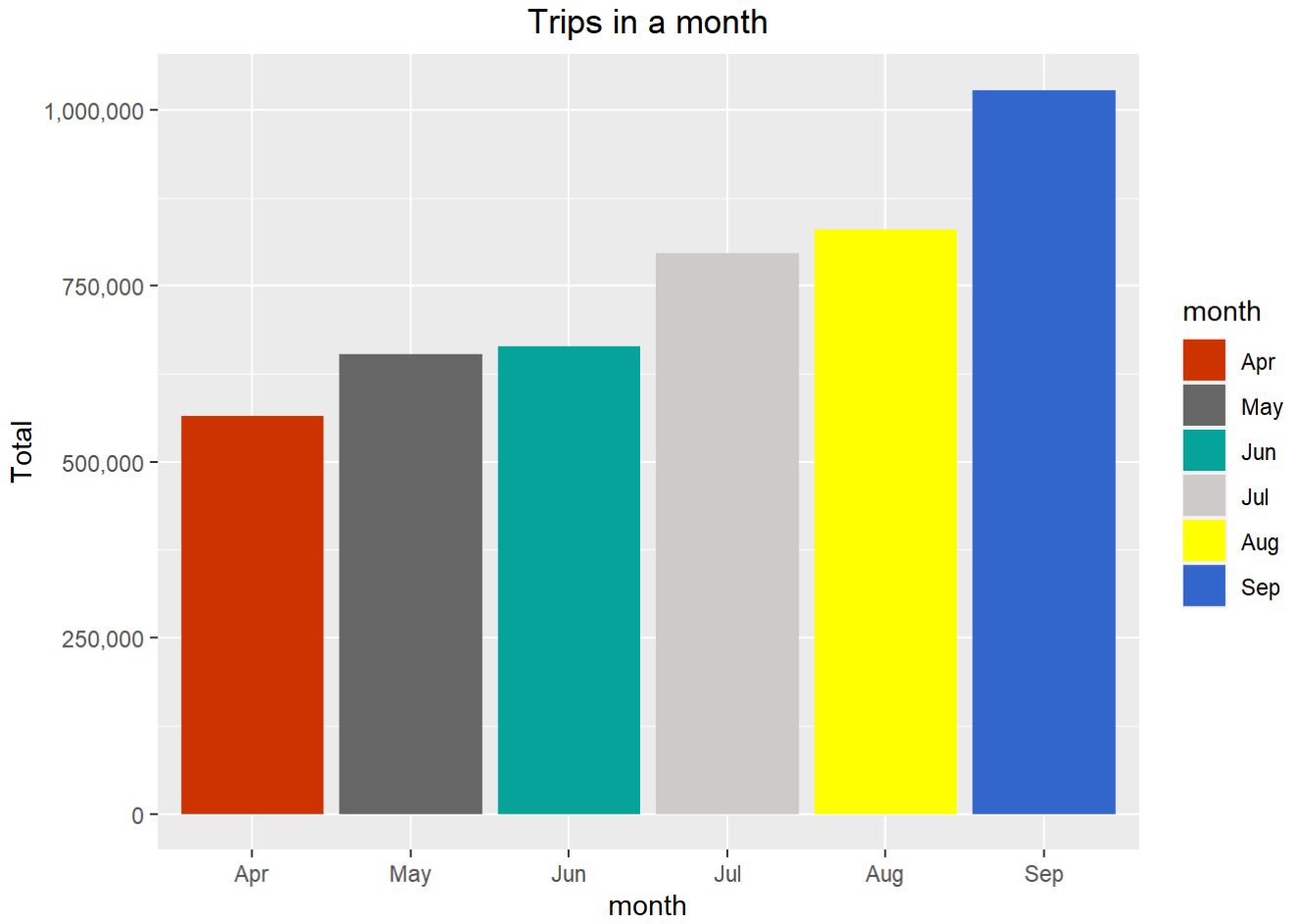
Next I decided to find the total number of trips that took place in each month . So here I have visualized the number of trips that took place each month of the year and it helps in observing the most trips that took place in each month.

```
month_data <- combined_data %>%
  group_by(month) %>%
  dplyr::summarize(Total = n())

head(month_data)
```

month <ord>	Total <int>
Apr	564516
May	652435
Jun	663844
Jul	796121
Aug	829275
Sep	1028136

6 rows

```
ggplot(month_data, aes(month, Total, fill = month)) +
  geom_bar(stat = "Identity") +
  ggtitle("Trips in a month") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)
```



From the above plot, we can clearly see a trend that number of trips are increasing every month thus September has the maximum number of trips per month.

Plotting the trips by hours of the day

Here I use the ggplot function to plot the number of trips that the passengers had made on hourly basis. Through this visualization, we can see that overall which time of the day had the maximum and minimum number of trips.

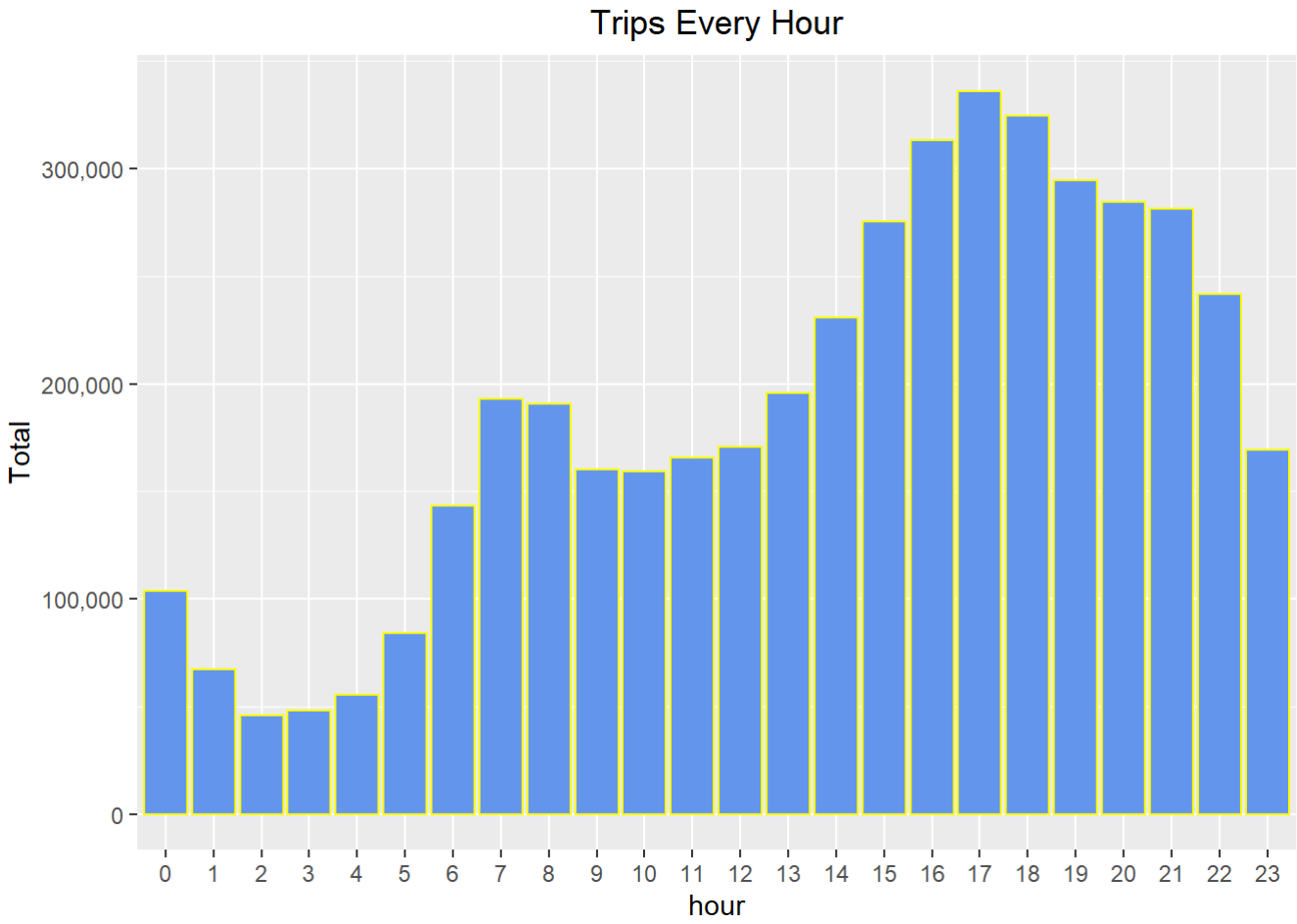
```
# summarized the data by hour
hourly_data <- combined_data %>%
  group_by(hour) %>%
  dplyr::summarize(Total = n())

head(hourly_data)
```

hour <fct>	**Total** <int>
0	103836

hour <fct>	Total <int>
1	67227
2	45865
3	48287
4	55230
5	83939

6 rows

```
ggplot(hourly_data, aes(hour, Total)) +
  geom_bar(stat="identity",fill="cornflowerblue",color="yellow") +
  ggtitle("Trips Every Hour") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels=comma)
```



From the above plot, we can see that the maximum number of trips took place between 5pm to 6pm and the minimum number of trips took place between 2am to 3am.

In the next plot, I have used different colors to represent different months to plot the number of trips that the passengers had made on hourly basis.

```
# summarized the data by month and hour
month_hour_data <- combined_data %>%
  group_by(month, hour) %>%
  dplyr::summarize(Total = n())
```

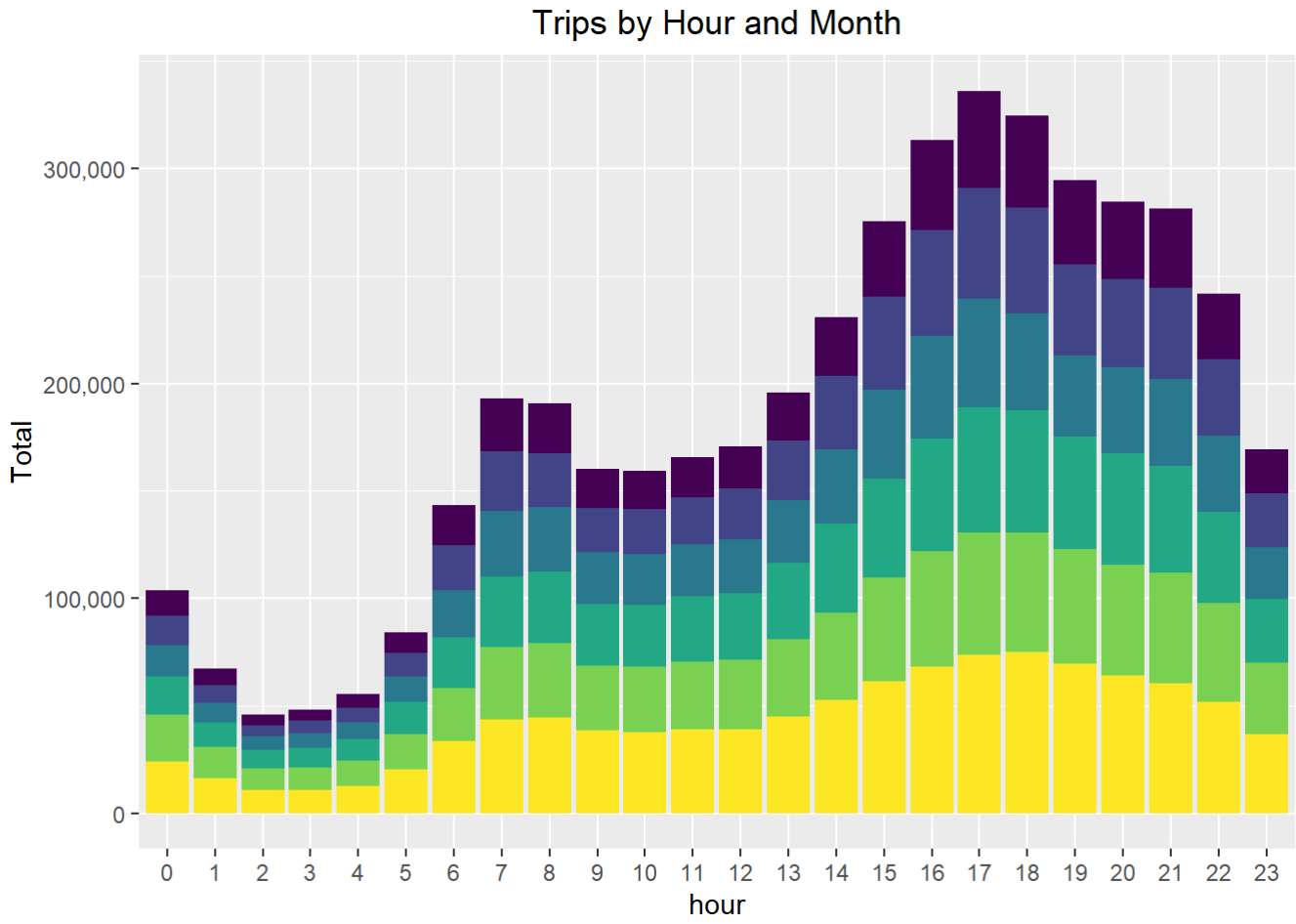
```
## `summarise()` has grouped output by 'month'. You can override using the `.groups` argumen
t.
```

```
head(month_hour_data)
```

month <ord>	hour <fct>	Total <int>
Apr	0	11910
Apr	1	7769
Apr	2	4935
Apr	3	5040
Apr	4	6095
Apr	5	9476

6 rows

```
ggplot(month_hour_data, aes(hour, Total, fill=month)) +
  geom_bar(stat = "identity") +
  ggtitle("Trips by Hour and Month") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = comma)
```



Plotting data by trips during every day of the month

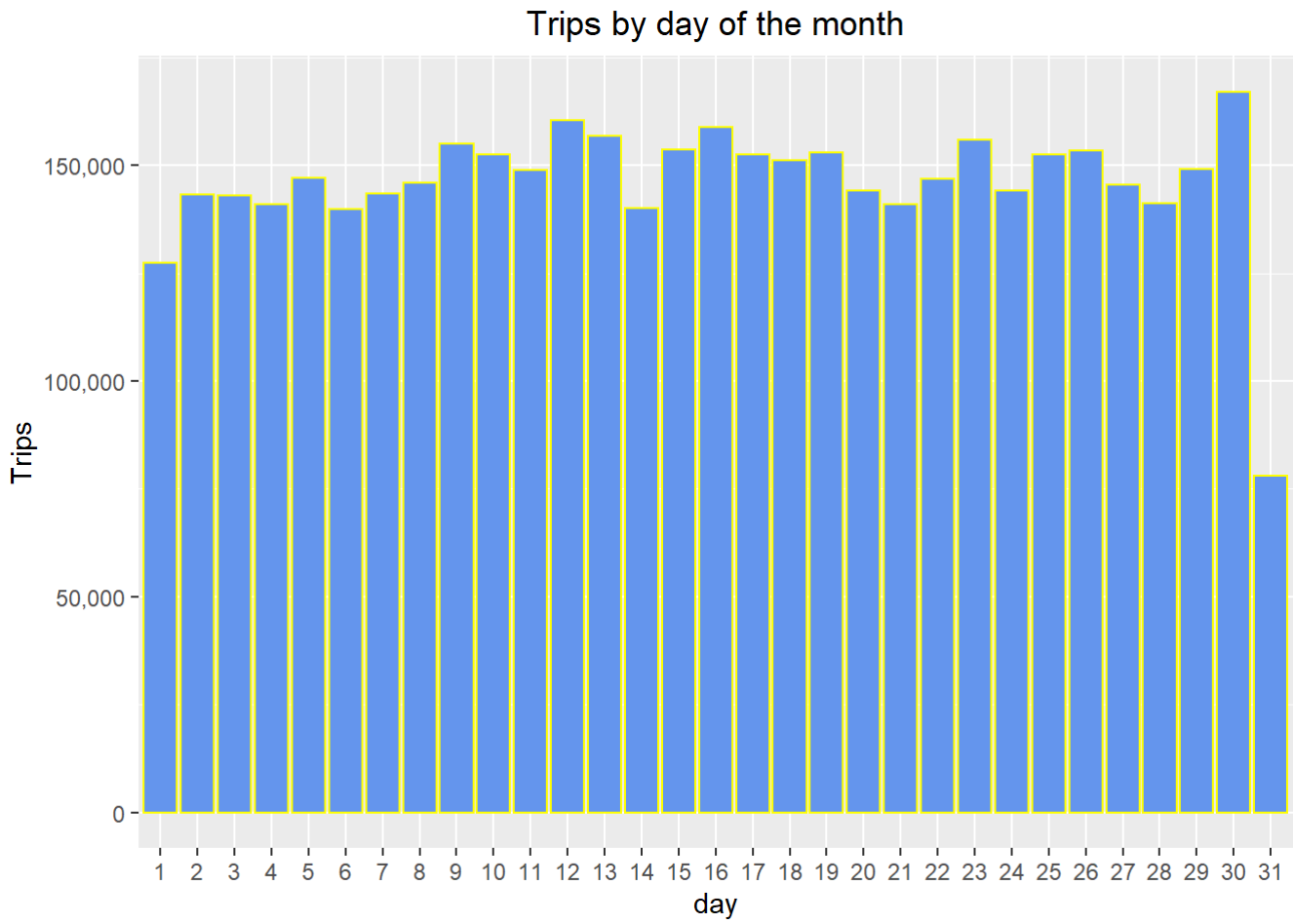
Here I use the ggplot function to plot the number of trips that the passengers had made on date basis. Through this visualization, we can see that overall which day of the month had the maximum and minimum number of trips.

```
# summarized data by day of the month
day_data <- combined_data %>%
  group_by(day) %>%
  dplyr::summarize(Trips = n())

head(day_data)
```

day <fct>	Trips <int>
1	127430
2	143201
3	142983
4	140923
5	147054
6	139886

6 rows

```
# Plot the data for the day
ggplot(day_data, aes(day, Trips)) +
  geom_bar(stat = "identity", fill="cornflowerblue",color="yellow") +
  ggtitle("Trips by day of the month") +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = comma)
```



From the above plot, we can see that the maximum number of trips took place on the 30th of the month and the minimum number of trips took place on 1st of the month.

In the next plot, I have used different colors to represent different months to plot the number of trips that the passengers had made on different days of the week.This plot can be used to find the trend of how number of trips of the days of week change over different months.

```
# summarized data by day of the week and month
day_month_data <- combined_data %>%
  group_by(dayofweek, month) %>%
  dplyr::summarize(Trips = n())
```

```
## `summarise()` has grouped output by 'dayofweek'. You can override using the `.groups` argu
ment.
```

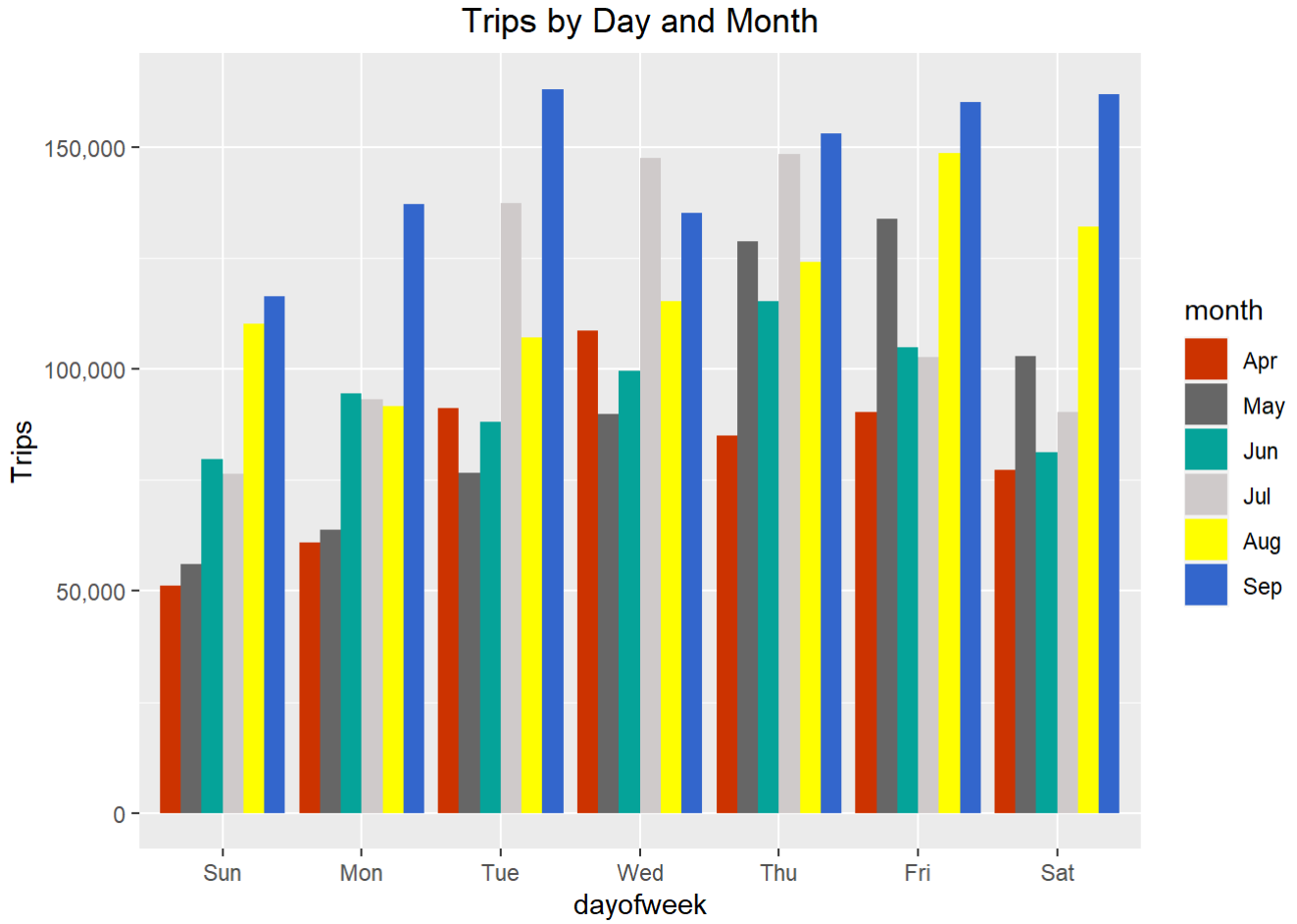
```
head(day_month_data)
```

	dayofweek	month	Trips
	<ord>	<ord>	<int>

dayofweek <ord>	month <ord>	Trips <int>
Sun	Apr	51251
Sun	May	56168
Sun	Jun	79656
Sun	Jul	76327
Sun	Aug	110246
Sun	Sep	116532

6 rows

```
# Plot the summarized data
ggplot(day_month_data, aes(dayofweek, Trips, fill = month)) +
  geom_bar(stat = "identity", aes(fill = month), position = "dodge") +
  ggtitle("Trips by Day and Month") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)
```

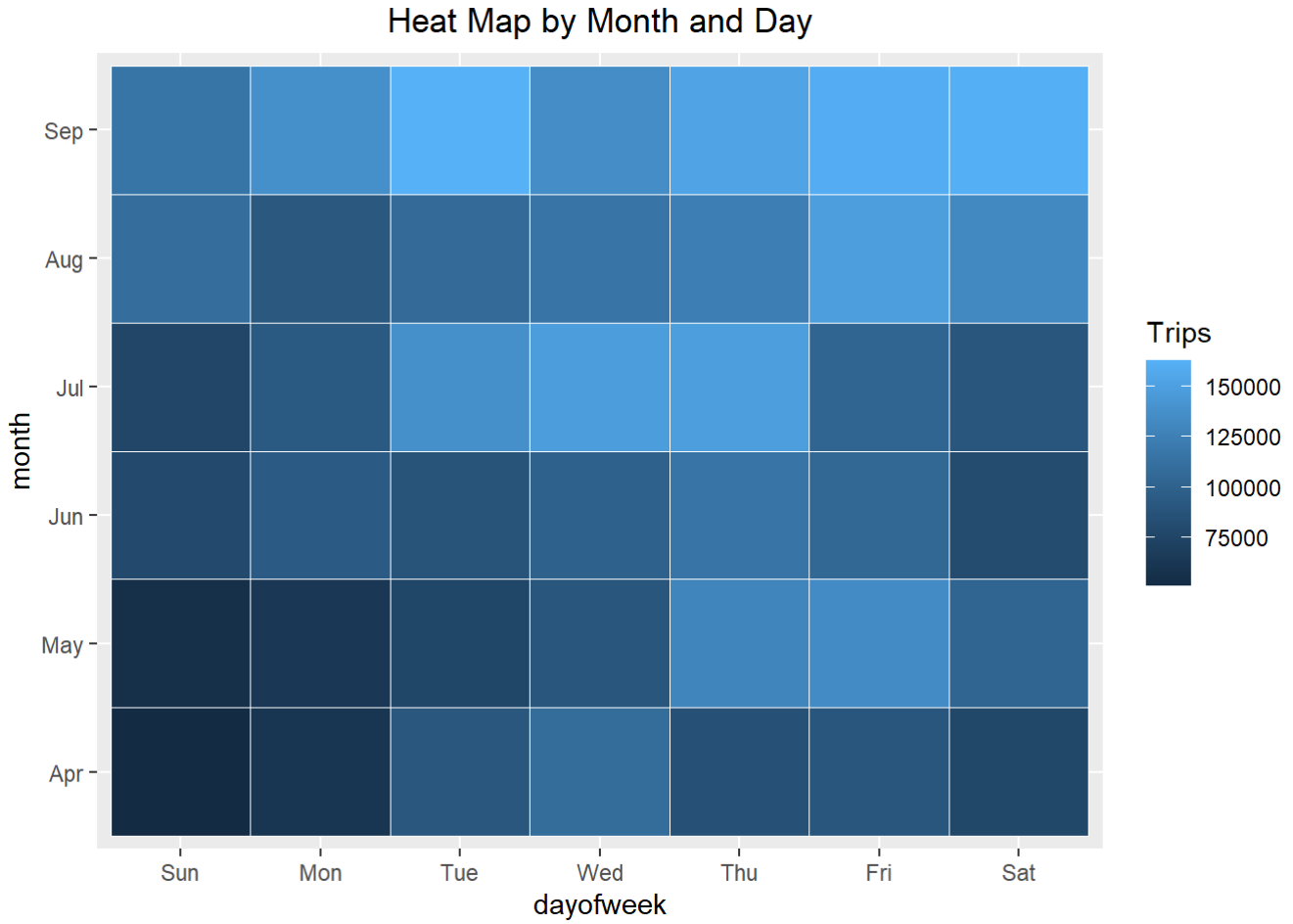


From the above plot, we can see that the Tuesdays of the month of September had the maximum number of the trips.

Plotting heat-map by month and day of the week

Next I used ggplot to create a heat-map for month vs day of the week. This gives us a comparative view of which day of the week and which month had what number of trips helping us compare between them

```
ggplot(day_month_data, aes(dayofweek, month, fill = Trips)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Day")+
  theme(plot.title = element_text(hjust = 0.5))
```



From the above plot, we can see that the Tuesdays of the month of September had the maximum number of the trips.

Plotting heat-map by hour and date of the month

Next I used ggplot to create a heat-map for comparison of number of trips for each hour of the day vs each date of the month.

```
day_hour_data <- combined_data %>%
  group_by(day, hour) %>%
  dplyr::summarize(Total = n())
```

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

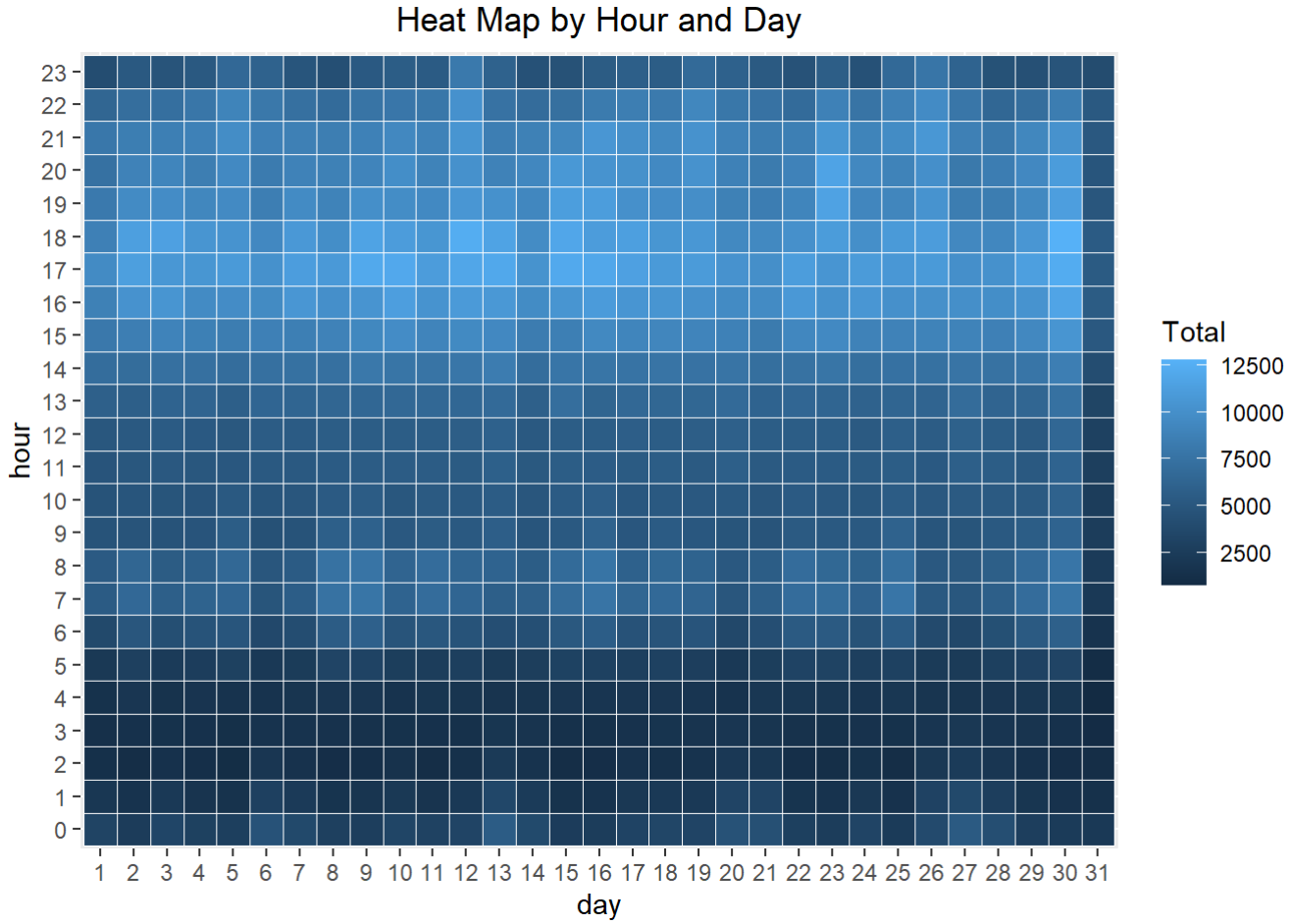
```
head(day_hour_data)
```

day	hour	Total
<fct>	<fct>	<int>

day <fct>	hour <fct>	Total <int>
1	0	3247
1	1	1982
1	2	1284
1	3	1331
1	4	1458
1	5	2171

6 rows

```
# Plot a heat map using ggplot
ggplot(day_hour_data, aes(day, hour, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Hour and Day")+
  theme(plot.title = element_text(hjust = 0.5))
```



Here we see that the hours for the date as 31st as the least among the rest of the dates and this is as expected as April, June and September months have only 30 days.

Plotting heat-map by Month and Day

Lastly to confirm the above output, I used ggplot to create a heat-map for comparison of number of trips for each month vs each date of the month.

```
# Summarize data by month and day
month_day_data <- combined_data %>%
  group_by(month, day) %>%
  dplyr::summarize(Trips = n())
```

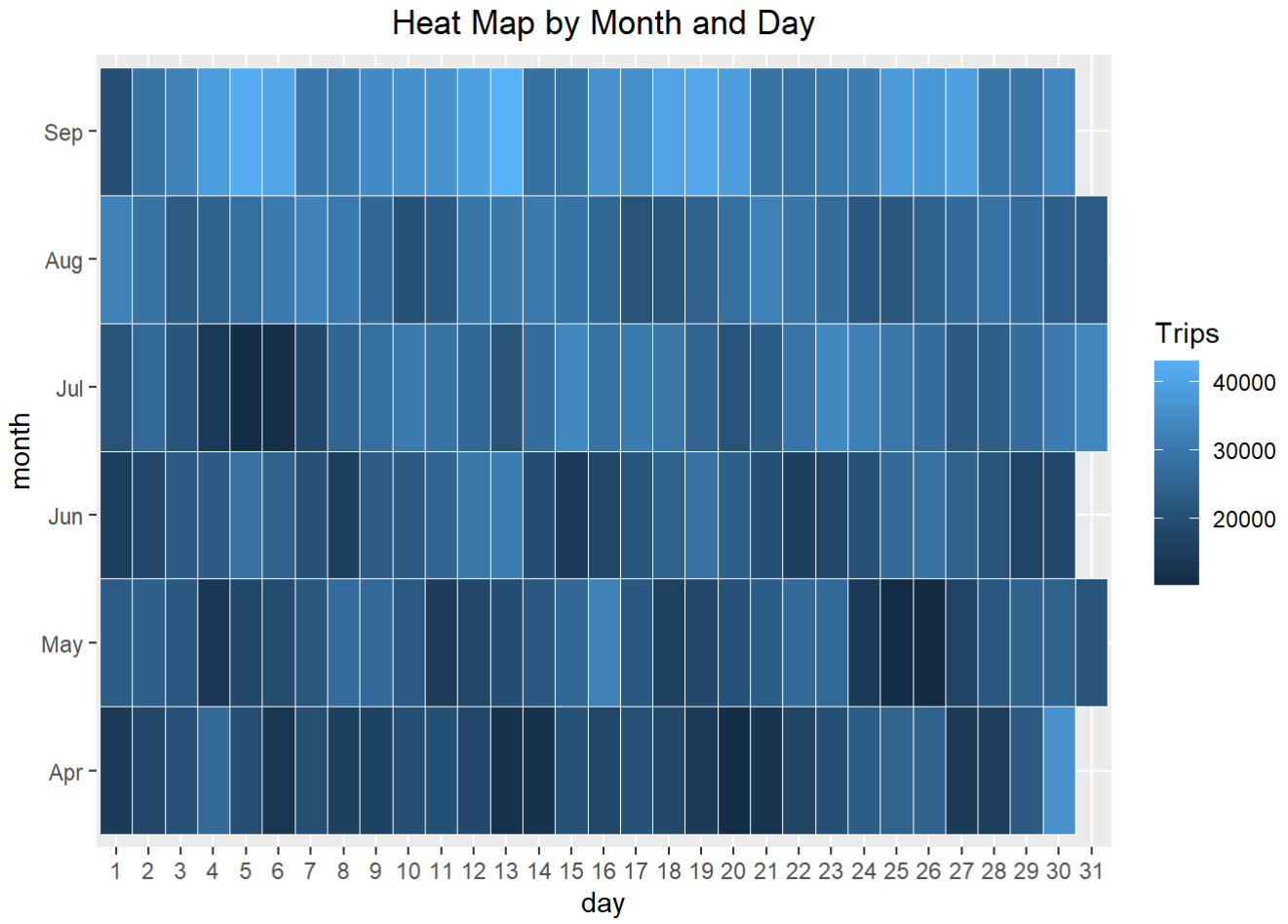
```
## `summarise()` has grouped output by 'month'. You can override using the `.groups` argumen
t.
```

```
head(month_day_data)
```

month <ord>	day <fct>	Trips <int>
Apr	1	14546
Apr	2	17474
Apr	3	20701
Apr	4	26714
Apr	5	19521
Apr	6	13445

6 rows

```
# Plot a heat-map using ggplot
ggplot(month_day_data, aes(day, month, fill = Trips)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Day")+
  theme(plot.title = element_text(hjust = 0.5))
```



As expected April, June and September months do not have any entry for 31st as these months have only 30 days.

Map visualizations of rides in New York

Here I have used two method to show map visualization, one is with ggplot and the other with leaflet.

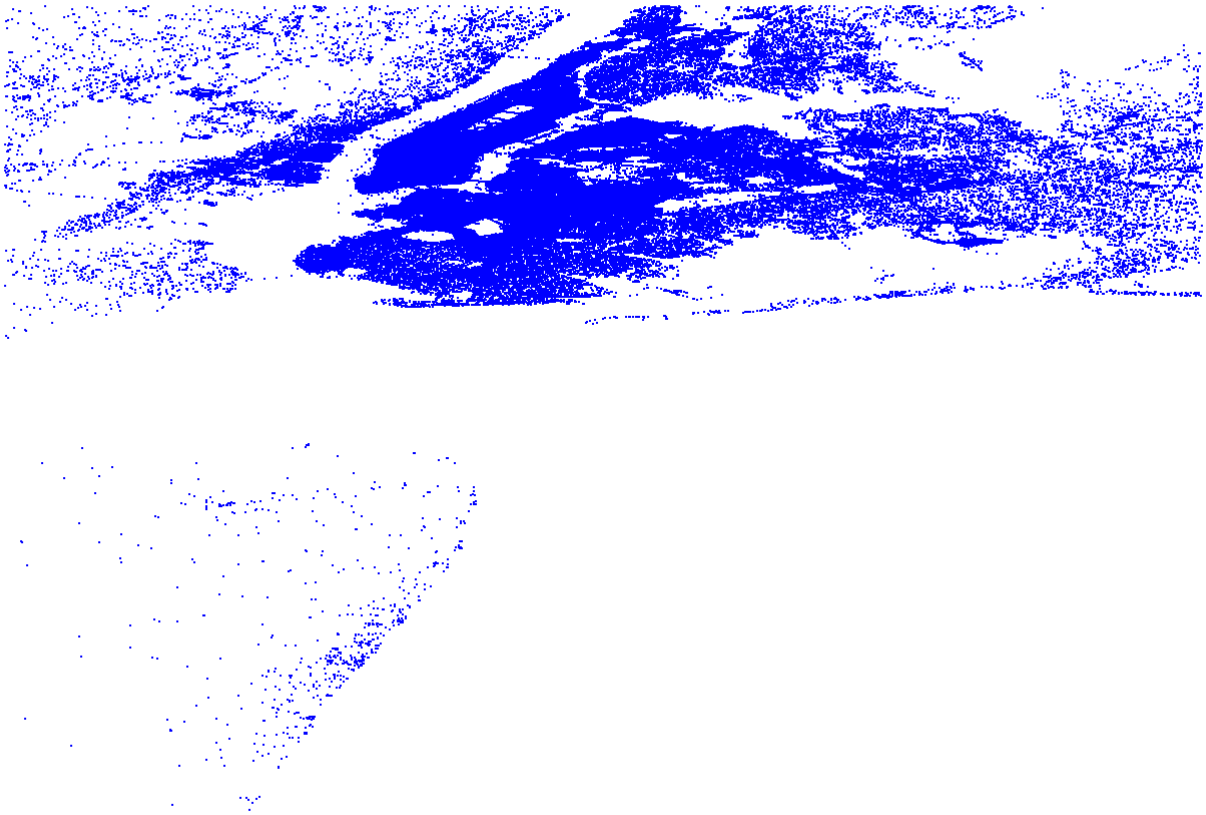
Using ggplot for Map visualization

In this section of visualization, I have visualized the rides in New York city by creating a geo-plot with longitude on the x-axis and latitude on the y-axis. This helps us to find the highest density of rides in the form of latitude and longitude.

```
# Set Map Constants
min_lat <- 40
max_lat <- 40.91
min_long <- -74.15
max_long <- -73.7004

ggplot(combined_data, aes(x=Lon, y=Lat)) +
  geom_point(size=0.02, 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 70180 rows containing missing values (geom_point).
```

NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)

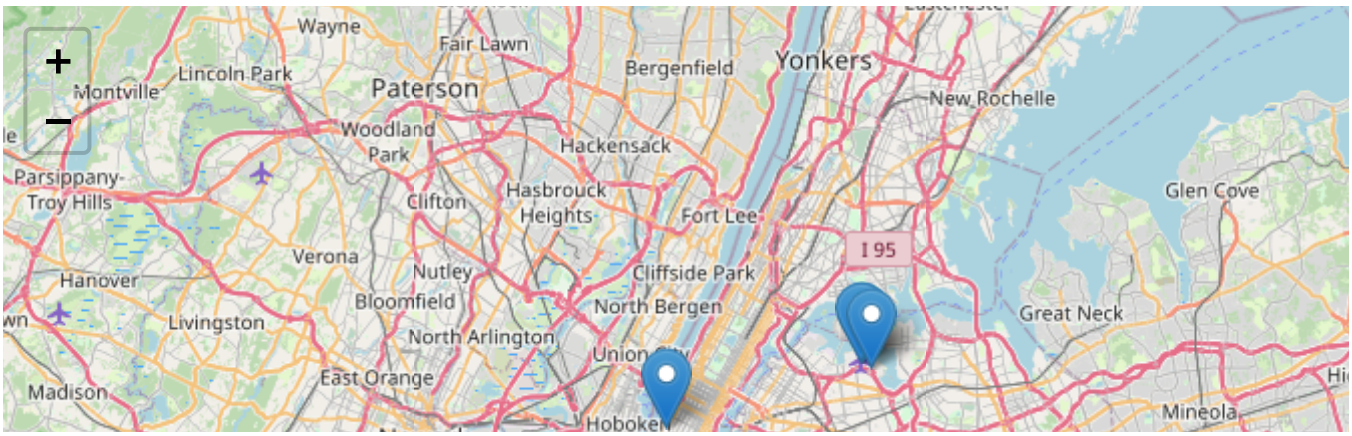


The above visualization is quite helpful but it is limited as it is just on the grid and does not have an actual map.

Using leaflet for Map visualization

Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. This package helps in plotting the rides on a map which can be much better in the sense of understanding than a normal grid plot. This process is much faster if you know what needs to

```
data_sorted <- combined_data %>%
  mutate(Lat_3 = round(Lat, 3),Lon_3 = round(Lon, 3)) %>%
  count(Lat_3, Lon_3, sort = TRUE) %>%
  head()

leaflet(data_sorted) %>%
  addTiles() %>%
  setView(-74.00, 40.71, zoom = 10) %>%
  addMarkers(~Lon_3, ~Lat_3)
```



Model the data for data prediction

Here I have created a 'lm' model that will be used to predict the trips data for the upcoming months. lm model is an linear regression model thus helping in prediction based on the previous values.

```
num<-c(4,5,6,7,8,9)

month_data$month_num <- num

mod<- lm(Total ~ month_num,data=month_data)
summary(mod)
```

```
##
## Call:
## lm(formula = Total ~ month_num, data = month_data)
##
## Residuals:
##      1      2      3      4      5      6
##  21716  24467 -49293  -2184 -54199  59494
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   202126      80254   2.519  0.06545 .
## month_num      85169      11941   7.132  0.00204 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49950 on 4 degrees of freedom
## Multiple R-squared:  0.9271, Adjusted R-squared:  0.9089
## F-statistic: 50.87 on 1 and 4 DF,  p-value: 0.002044
```

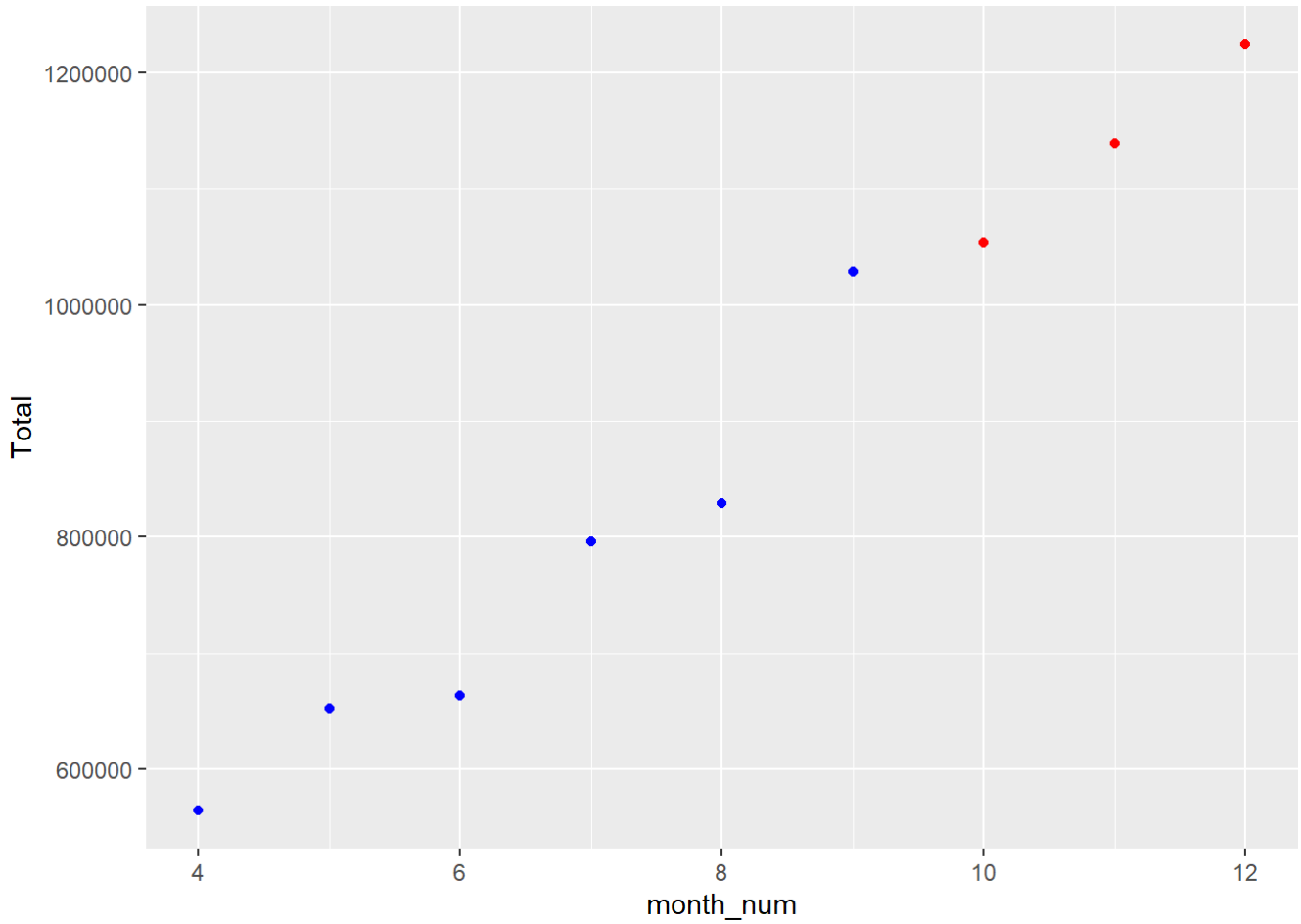
```
new_month<- data.frame(month = c("Oct","Nov", "Dec"),
                       Total=c(predict(mod, data.frame(month_num = 10)),
                               predict(mod, data.frame(month_num = 11)),
                               predict(mod, data.frame(month_num = 12))),
                       month_num=c(10,11,12))

new_month_data <- rbind(month_data,new_month )

new_month_data
```

month <ord>	Total <dbl>	month_num <dbl>
Apr	564516	4
May	652435	5
Jun	663844	6
Jul	796121	7
Aug	829275	8
Sep	1028136	9
Oct	1053811	10
Nov	1138979	11
Dec	1224148	12

9 rows

As in the above code we see that the model is created and has predicted values for the month of October, November and December.

```
ggplot() +
  geom_point(data=new_month_data[new_month_data$month_num<=9,],
             aes(x = month_num, y = Total), color = "blue") +
  geom_point(data=new_month_data[new_month_data$month_num>9,],
             aes(x = month_num, y = Total), color = "red")
```



Here in the above plot, Blue points represents the value from our data and Red points represent the prediction values of the month of October(10), November(11) and December(12).

Github Link

I have also uploaded the report in rmd and knitted format at the provided link: [https://github.com/vidit-sharma/Uber-Data-Analysis.git (https://github.com/vidit-sharma/Uber-Data-Analysis.git)]

Conclusion

At the end of this Uber data analysis project, we have observed how various data visualizations tell us about the trips during various time of the day or different days of the week. We made use of packages like ggplot2 and leaflet that allowed us to plot various types of visualizations that pertained to several time-frames and conclude how time affected customer trips.