Case_Study_1

Cem Demirci

Case Study : How Does a Bike-Share Navigate Speedy Success?

Introduction

In this case study, we are working as a junior data analyst working for a fictional, Cyclistic. We will follow these steps to answer the business questions.

- Ask
- Prepare
- Process
- Analysis
- Share
- Act

as taught in the course.

Scenario

The director of the marketing team, Lily Moreno, believes that maximizing the number of annual members is important for company's future success. Therefore, our team wants to understand **how casual riders and annual members use Cyclistic bikes differently**. Using these insights, a new marketing strategy will be used to convert causal riders into annual members.

ASK

A Clear Statement of the Business Task

Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. It is believed that there is no need for any other marketing campaign, casual riders are already aware of the Cyclistic program, therefore the goal is to convert casual riders into annual members using insights.

PREPARE

1. Where is your data located?

The data is uploaded in RStudio interface, where we can use r programming language as well as other programming languages to easily manipulate related documents and information regarding to bike ride data. I have decided to use r programming for this analysis.

2. How is data organised?

The data is segregated by quarters of years 2013 to 2018 and months including the years from 2020 to 2022 as CSV files.

3. Are there issues with bias or credibility in this data? Does your data ROCCC?

Since the data is directly collected from the customers, it seems that there is no issue regarding to bias credibility. The data is reliable, original, comprehensive, current, and also cited therefore the rocccs.

4. How are you addressing licensing, privacy, security, and accessibility?

The data has been made available by Motivate International Inc. under this license (https://ride.divvybikes.com/data-license-agreement). No personal identifiable information including credit card numbers provided regarding to riders in the files. The data set do not contain private information, therefore provides privacy and security on riders.

5. How did you verify the data's integrity?

- All data has same number of columns with identical columns.
- The data is consistent across the years and provided as csv file type.
- The data contains all the required information regarding to rides.

6. How does it help you answer your question?

The data provides all the necessary information regarding to rides including dates, times, bike types, customer types, start and end locations and so on. Using this information, I have created new attributes to further analyze existing data. This allowed me to see relationships between riders (annual and casual riders). I believe that these outcomes will be useful the make final business decisions.

7. Are there any problems with the data?

There were NA values, which I have removed. Also, there were test rides of company as well which are removed too.

PROCESS

1. What tools are you choosing and why?

I have downloaded files between 04.2020 and 03.2021 for this case study. Since there were enormous ride data, literally millions of rows, it would be hard to make necessary adjustment in Microsoft Excel. Therefore, I have decided to manipulate the data using r programming language within RStudio.

2. Have you ensured your data's integrity?

- The data is complete: It contains all the required components for analysis.
- The data is consistent: Files are organized in an equal number of columns and same data types
- Its credibility was proven above. It is also trustworthy.

3. What steps have you taken to ensure that your data is clean?

- I have concatenated all twelve csv files into one data frame.
- I have checked if there are duplicated values. There were not.
- I have removed all the NA values.
- I have checked if there are irrelevant values and removed them all.
- I have checked unique variables to see misspelling errors.

4. How can you verify that your data is clean and ready to analyze?

- Changed names of some columns for better readability.
- mutate() for creating new attributes using existing ones and remove irrelevant values out of it.
- duplicate() for removing duplicated values.
- na.omit() for removing NA values
- count() to check further irrelevant values or attributes.
- filter() to see any missing or irrelevant values.
- subset() for checking test rides of company.
- format() for formatting dates into days, months and hours.
- factor() for reordering months and days.

5. Have you documented your cleaning process so you can review and share those results?

Here are the steps I have covered during the cleaning process:

First thing is first. Let's load the required packages.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6      ✓ purrr   0.3.4
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.4.1
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(ggplot2)
```

I also required additional packages during the analysis phase so I would like to include them here as well. But note that these would not be necessary for your own study.

```
library(dplyr)
library(readr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following objects are masked from 'package:base':
```

```
##
##     date, intersect, setdiff, union
library(data.table)
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
library(chron)
##
## Attaching package: 'chron'
## The following objects are masked from 'package:lubridate':
##
##     days, hours, minutes, seconds, years
library(ggmap)
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
## Please cite ggmap if you use it! See citation("ggmap") for details.
library(maps)
##
## Attaching package: 'maps'
## The following object is masked from 'package:purrr':
##
##     map
library(patchwork)
```

1. Downloaded all the relevant csv files into same file. After that, concatenate csv files into one data frame.

```
data_all <- list.files(path = "C:/Users/cem_d/Desktop/Google_Data_Analytics_Certificate/Case_Study_1/tripdata_202004_202103",
                       pattern = "*.csv", full.names = TRUE) %>%
  lapply(read_csv) %>%
  plyr::rbind.fill()
```

```
## Rows: 84776 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 200274 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 343005 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 551480 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 622361 Columns: 13
## ── Column specification ────────────────────────────────────────────────────
```

```
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 532958 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 388653 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 259716 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 131573 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
```

```
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 96834 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 49622 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 228496 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2. Remove NA values.

```
data_all<-na.omit(data_all)
```

3. Remove empty rows or columns.

```
data_all_1 <- remove_empty(data_all, which = c("rows", "cols"))
count(filter(data_all_1, start_station_name==" "),
      start_station_name, start_station_id, end_station_name, end_station_id)
## [1] start_station_name start_station_id   end_station_name   end_station_id
## [5] n
## <0 rows> (or 0-length row.names)
```

checked.

5. Check rideable_type values.

```
data_all_1 %>%
  count(rideable_type)
##   rideable_type       n
## 1  classic_bike  318614
## 2   docked_bike 2554291
## 3 electric_bike  421786
```

6. Rename column names for readability.

```
data_all_2 <- rename(data_all_1, rider_type=member_casual, start_time=started_at,
                     end_time=ended_at, bike_type=rideable_type)
```

7. Create new column for duration of rides.

```
data_all_3 <- data_all_2 %>%
  mutate(duration_ride=as.double(end_time-start_time)/60)
```

8. Check if the duration is smaller than 0.

```
nrow(subset(data_all_3, duration_ride < 0))
## [1] 10454
```

9. Remove all the rows where duration ride is smaller than 0.

```
data_all_4<-data_all_3[!(data_all_3$duration_ride<0),]
```

10. Create new columns (day, month, time) for future analysis.

```
data_all_4$day<-format(data_all_4$start_time,"%a")
```

```
data_all_4$month<-format(data_all_4$start_time,"%b-%Y")


data_all_4$hour<-as.integer(format(data_all_4$start_time,"%H"))
```

11. Check if there are any duplicated rows.

```
data_all_4<-data_all_4[!duplicated(data_all_4$ride_id),]
```

12. I have recognized some test rides made by the company. Lets see how many…

```
nrow(subset(data_all_4, start_station_name %like% "TEST"))
## [1] 3195
nrow(subset(data_all_4, end_station_name %like% "TEST"))
## [1] 3488
```

13. Removing those test rides from the data set as well.

```
data_all_4<-data_all_4[!(data_all_4$start_station_name %like% "TEST"),]
data_all_4<-data_all_4[!(data_all_4$end_station_name %like% "TEST"),]
```

14. I have to reorder days and months so that these will be seen as ordered in visualizations.

```
data_all_4$month<-factor(data_all_4$month,levels = c("Apr-2020", "May-2020", "Jun-2020",
                                        "Jul-2020","Aug-2020", "Sep-2020", "Oct-2020",
                            "Nov-2020", "Dec-2020", "Jan-2021", "Feb-2021", "Mar-2021"))


data_all_4$day<-factor(data_all_4$day,levels = c("Mon","Tue","Wed","Thu","Fri","Sat","Sun"
))
```

15. Done with cleaning process. Creating the final document as data_bike.

```
data_bike<-data_all_4
```

ANALYZE & SHARE

Now, our data is ready for analysis. The analysis will help us to gain insights on how the causal riders and annual members are differ.

1. Let's see the number of casual riders and annual member and compare them.

```
df<-data_bike %>%
  group_by(rider_type) %>%
  summarize(count=length(rider_type),
```

```
            percentage_of_total=(length(rider_type)/nrow(data_bike))*100)


df
```

```
## # A tibble: 2 × 3
##   rider_type   count percentage_of_total
##   <chr>        <int>               <dbl>
## 1 casual     1344725                41.0
## 2 member     1935982                59.0
```

From the created table, we can see that %59 of the total riders are the actual members and %41 of them are casual ones. Lets plot the relationship between them.

```
plot_1<-ggplot(df, aes(x=rider_type,y=percentage_of_total,color=rider_type,fill=rider_type
))+
  geom_bar(stat = "identity")+
  geom_text(aes(label=round(percentage_of_total,digits = 2)),vjust=1.5,size=4,color="black
")+
  labs(y="% of Total",x="Rider Type",title = "Percentage of Bike Riders")+
  theme(legend.position = "none")


plot_1
```

2. Now lets compare riders by months.

```
df2<-data_bike %>%
  group_by(rider_type, month) %>%
  summarise(number_of_riders=length(month),as_percentage=length(month)/nrow(data_bike)*100) %>%
  arrange(desc(number_of_riders))
```

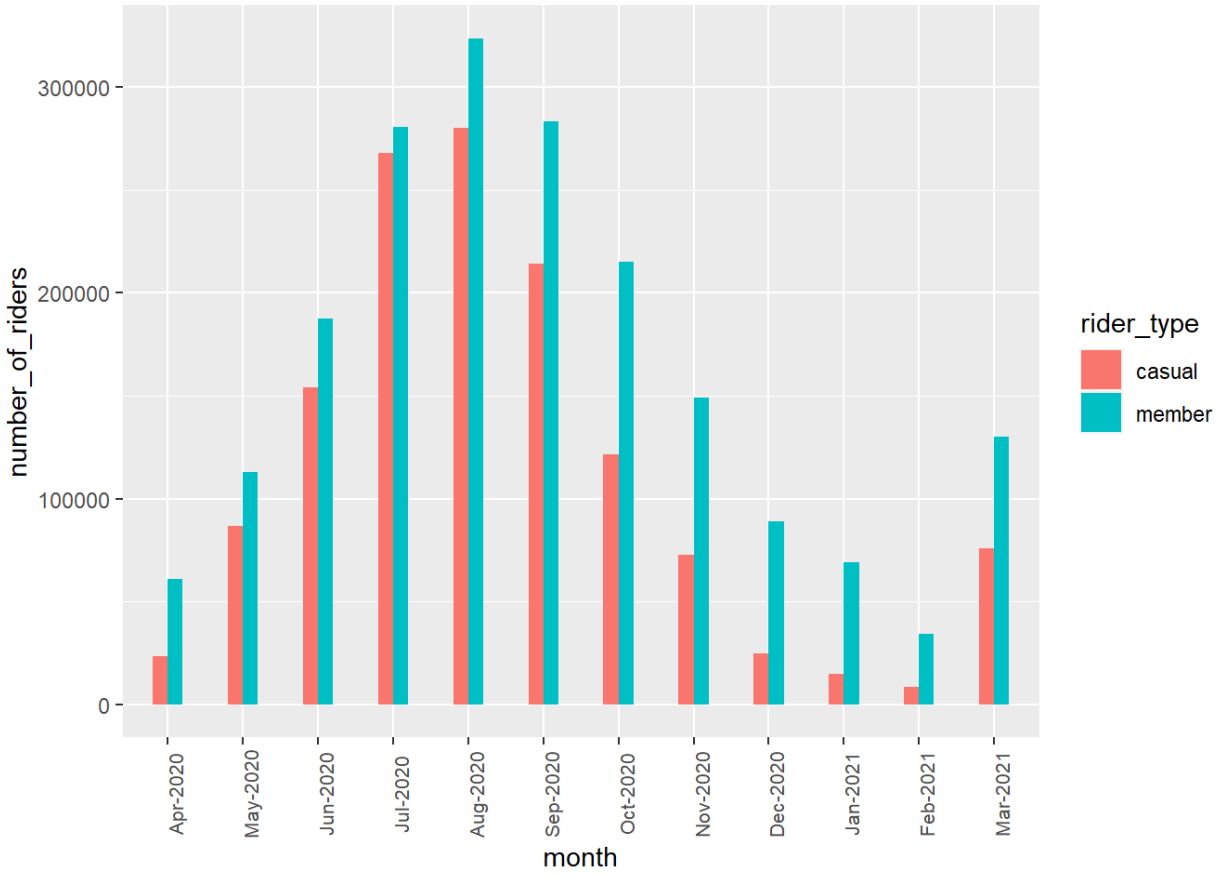
```
## `summarise()` has grouped output by 'rider_type'. You can override using the
## `.groups` argument.
```

```
df2
```

```
## # A tibble: 24 × 4
## # Groups:   rider_type [2]
##    rider_type month    number_of_riders as_percentage
##    <chr>      <fct>               <int>         <dbl>
##  1 member     Aug-2020           323720          9.87
##  2 member     Sep-2020           283537          8.64
##  3 member     Jul-2020           280525          8.55
##  4 casual     Aug-2020           280473          8.55
##  5 casual     Jul-2020           267943          8.17
##  6 member     Oct-2020           215047          6.55
##  7 casual     Sep-2020           214066          6.52
##  8 member     Jun-2020           187708          5.72
##  9 casual     Jun-2020           154176          4.70
## 10 member     Nov-2020           149076          4.54
## # … with 14 more rows
```

```
plot_2<-ggplot(df2,aes(x=month,y=number_of_riders, fill=rider_type))+
  geom_col(width = 0.4,position = position_dodge(width = 0.4))+
  theme(axis.text.x=element_text(size=8, angle = 90))+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))


plot_2
```



The plot shows us that both casual riders and annual members mostly ride during the summer season and rides are dramatically decreased for both users through autumn and cold season. It is clearly customers prefer better weather quality to ride bikes. Therefore, in order to convert casual riders into annual members, all the marketing campaigns should be launched in between spring and summer seasons in order to attract more casual members.

3. I have also compared riders by the days of the week.

```
df3<-data_bike %>%
  group_by(rider_type,day) %>%
  summarise(number_of_riders=length(day),
            percentage_by_day=length(day)/nrow(data_bike)*100) %>%
  arrange(desc(number_of_riders))
```

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

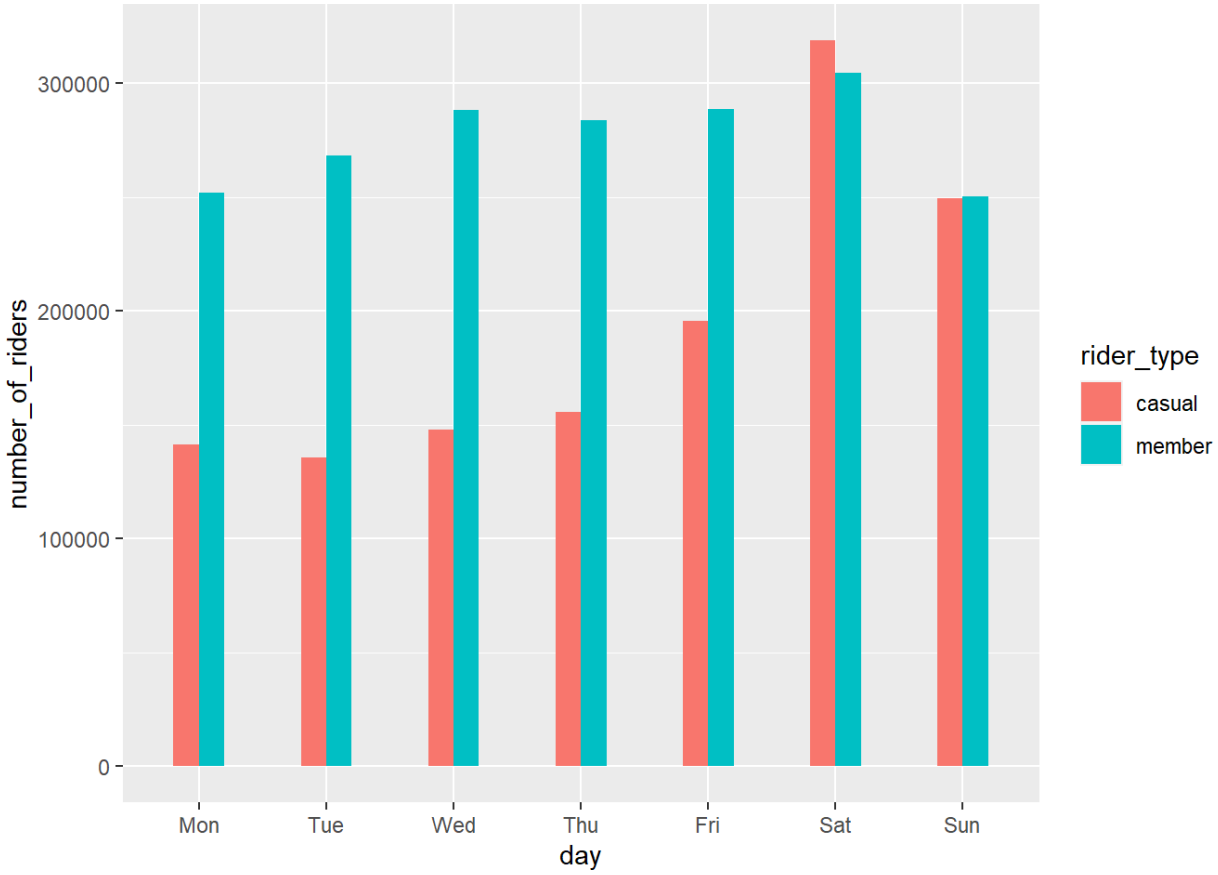
```
df3
```

```
## # A tibble: 14 × 4
## # Groups:   rider_type [2]
##    rider_type day   number_of_riders percentage_by_day
##    <chr>      <fct>            <int>             <dbl>
##  1 casual     Sat             318853              9.72
##  2 member     Sat             304579              9.28
##  3 member     Fri             288868              8.81
```

```
##  4 member      Wed              288316              8.79
##  5 member      Thu              283693              8.65
##  6 member      Tue              268200              8.18
##  7 member      Mon              251881              7.68
##  8 member      Sun              250445              7.63
##  9 casual      Sun              249623              7.61
## 10 casual      Fri              195762              5.97
## 11 casual      Thu              155592              4.74
## 12 casual      Wed              147798              4.51
## 13 casual      Mon              141472              4.31
## 14 casual      Tue              135625              4.13
```

```
plot_3<-ggplot(df3,aes(x=day,y=number_of_riders,fill=rider_type))+
  geom_col(width = 0.4,position = position_dodge(width = 0.4))+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))


plot_3
```



From the above graph, casual members are mostly busy on Saturdays and don't use bike services in weekdays compared to weekends. Interestingly, the trip number of annual riders is almost constant through days.

4.What about comparing annual members and casual riders on a hourly basis? Let's see what we got here…

```r
df4<-data_bike %>%
  group_by(rider_type,hour) %>%
  summarise(number_of_trips=n())
```

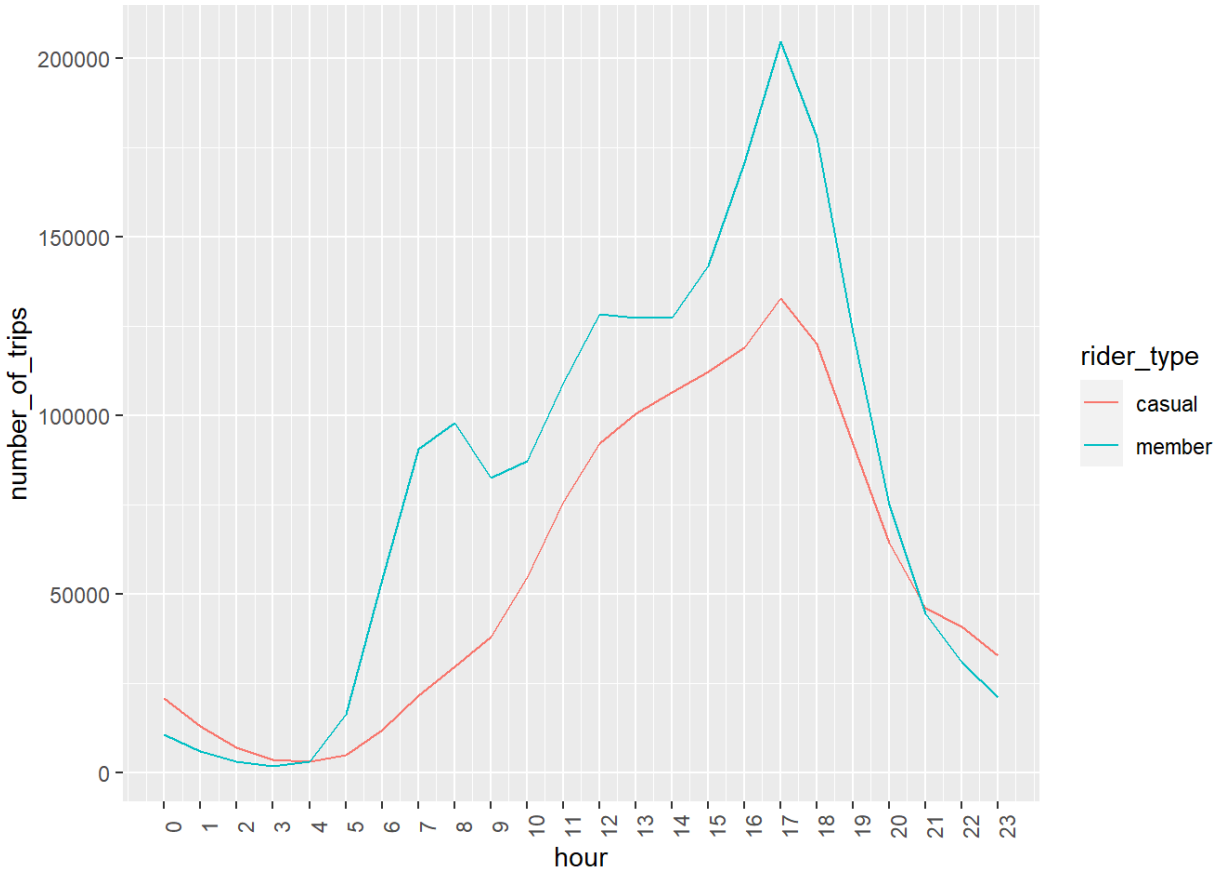
```
## `summarise()` has grouped output by 'rider_type'. You can override using the
## `.groups` argument.
```

```r
df4
```

```
## # A tibble: 48 × 3
## # Groups:   rider_type [2]
##    rider_type  hour number_of_trips
##    <chr>      <int>           <int>
##  1 casual         0           20948
##  2 casual         1           13022
##  3 casual         2            7150
##  4 casual         3            3760
##  5 casual         4            3059
##  6 casual         5            4921
##  7 casual         6           12110
##  8 casual         7           21753
##  9 casual         8           29778
## 10 casual         9           38031
## # … with 38 more rows
```

```r
plot_4<-ggplot(df4,aes(x=hour,y=number_of_trips,color=rider_type))+
  geom_line()+
  theme(axis.text.x = element_text(angle = 90))+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))+
  scale_x_continuous(breaks = c(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
                                19,20,21,22,23))


plot_4
```



For the members, there are two distinct peak hours 8 AM and 5 PM which is also coinciding with the peak demand of casual riders. Seems like office workers are the majority of the people demanding bike share services in morning and evening hours.

5. Now, let's divide hours into mornings, evenings and afternoons.

```
df5<-data_bike %>%
  mutate(day_time=ifelse(hour<12,"Morning",
                  ifelse(hour>=12 & hour<19,"Afternoon", "Evening"))) %>%
  group_by(rider_type,day_time) %>%
  summarise(number_of_trips=length(rider_type)) %>%
  arrange(desc(number_of_trips))
```

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

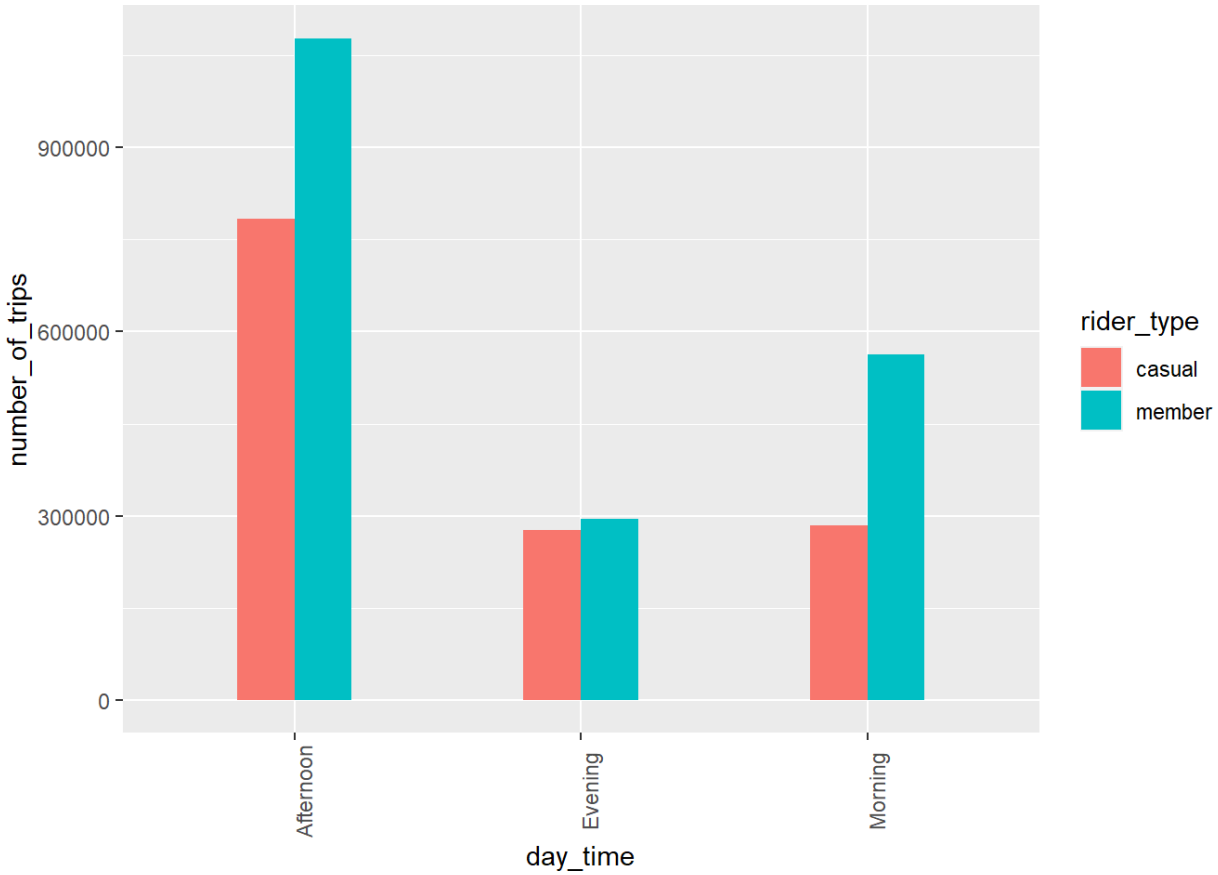
```
df5
```

```
## # A tibble: 6 × 3
## # Groups:   rider_type [2]
##   rider_type day_time  number_of_trips
##   <chr>      <chr>               <int>
## 1 member     Afternoon         1077957
## 2 casual     Afternoon          783151
## 3 member     Morning            563165
```

```
## 4 member      Evening            294860
## 5 casual      Morning            285068
## 6 casual      Evening            276506
```

```
plot_5<-ggplot(df5,aes(x=day_time,y=number_of_trips,fill=rider_type))+
  geom_col(width = 0.4,position = position_dodge(width = 0.4))+
  theme(axis.text.x = element_text(angle = 90))+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))

plot_5
```



All the members prefer riding bikes afternoon mostly.

6. Let's compare how different bike types are used by customers.

```
df6<-data_bike %>%
  group_by(rider_type,bike_type) %>%
  summarise(number_of_trips=n()) %>%
  arrange(desc(number_of_trips))
```

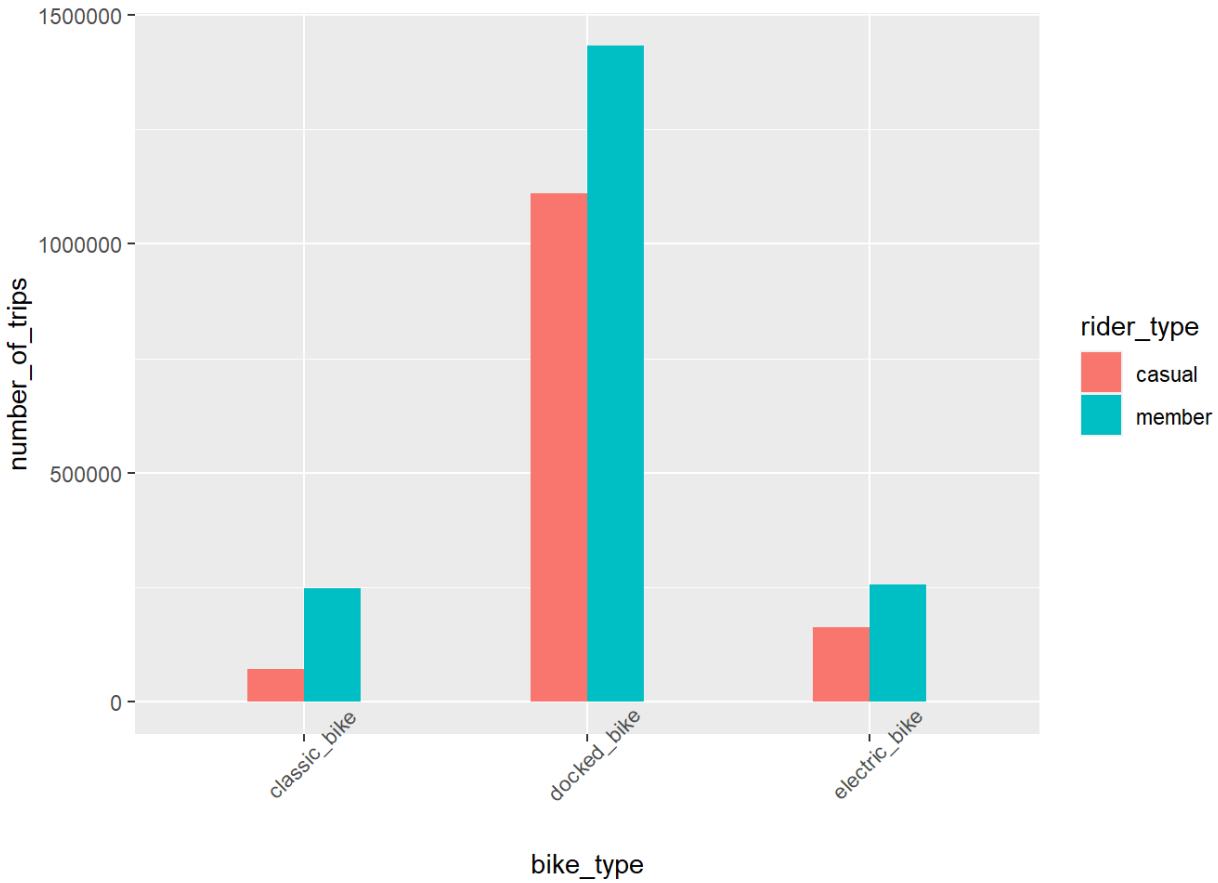
```
## `summarise()` has grouped output by 'rider_type'. You can override using the
## `.groups` argument.
```

```
df6
```

```
## # A tibble: 6 × 3
## # Groups:   rider_type [2]
##   rider_type bike_type     number_of_trips
##   <chr>      <chr>                   <int>
## 1 member     docked_bike           1432363
## 2 casual     docked_bike           1111003
## 3 member     electric_bike          255438
## 4 member     classic_bike           248181
## 5 casual     electric_bike          163290
## 6 casual     classic_bike            70432
```

```
plot_6<-ggplot(df6,aes(x=bike_type,y=number_of_trips,fill=rider_type))+
  geom_col(width = 0.4,position = position_dodge(width = 0.4))+
  theme(axis.text.x = element_text(angle = 45))+
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))


plot_6
```



Apparently, all customers tend to ride docked bike type over the others. As it can be seen, classic bikes and electric bikes almost shares the same amount of usage for actual members which is significantly less compared to docked bike usage. Seems like customers prefer docked bikes over the others may be due to easy use of this bike type since it offers more comfort during ride. Still, we need more data related to fleet to make more relevant assumptions.

7. Let's see most popular locations for starting a bike ride! First creating a new data frame for locations. Then choosing top 10 famous locations for rides as shown below. I will also add Chicago map to better visualize density of start locations and end locations.

7.1. Start Locations…

```
location_data<-data_bike %>%
  group_by(start_station_name, start_lat, start_lng) %>%
  summarise(count=length(start_station_name)) %>%
  arrange(desc(count))
```

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

```
location_data
```

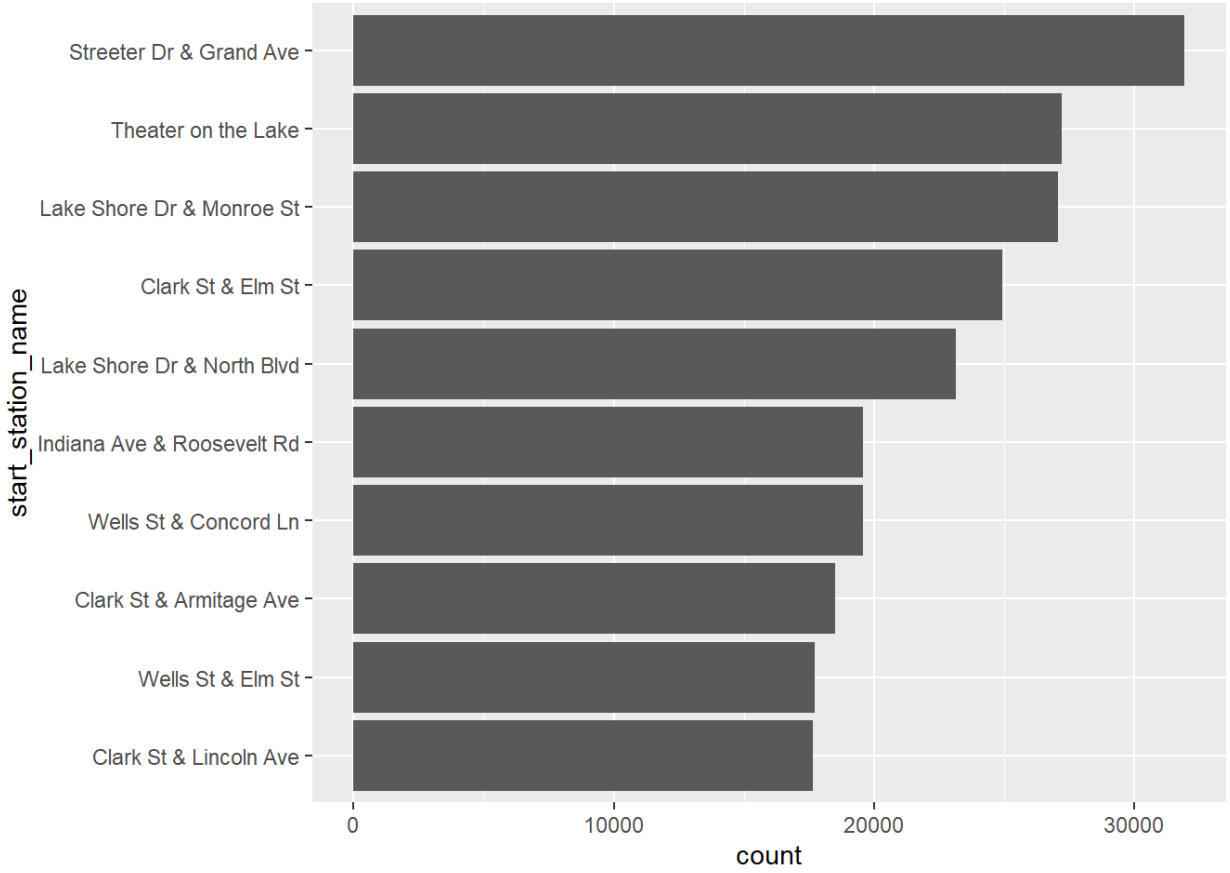
```
## # A tibble: 420,089 × 4
## # Groups:   start_station_name, start_lat [278,826]
##    start_station_name         start_lat start_lng count
##    <chr>                          <dbl>     <dbl> <int>
##  1 Streeter Dr & Grand Ave         41.9     -87.6 31930
##  2 Theater on the Lake             41.9     -87.6 27207
##  3 Lake Shore Dr & Monroe St       41.9     -87.6 27087
##  4 Clark St & Elm St               41.9     -87.6 24934
##  5 Lake Shore Dr & North Blvd      41.9     -87.6 23131
##  6 Indiana Ave & Roosevelt Rd      41.9     -87.6 19588
##  7 Wells St & Concord Ln           41.9     -87.6 19569
##  8 Clark St & Armitage Ave         41.9     -87.6 18521
##  9 Wells St & Elm St               41.9     -87.6 17714
## 10 Clark St & Lincoln Ave          41.9     -87.6 17658
## # … with 420,079 more rows
```

```
popular_locations<-head(location_data,10)


popular_locations$start_station_name<-factor(popular_locations$start_station_name,
                                             levels = popular_locations$start_station_name
[order(popular_locations$count)])


locations<-ggplot(popular_locations,aes(x=count,y=start_station_name))+
  geom_bar(stat = "identity")


locations
```

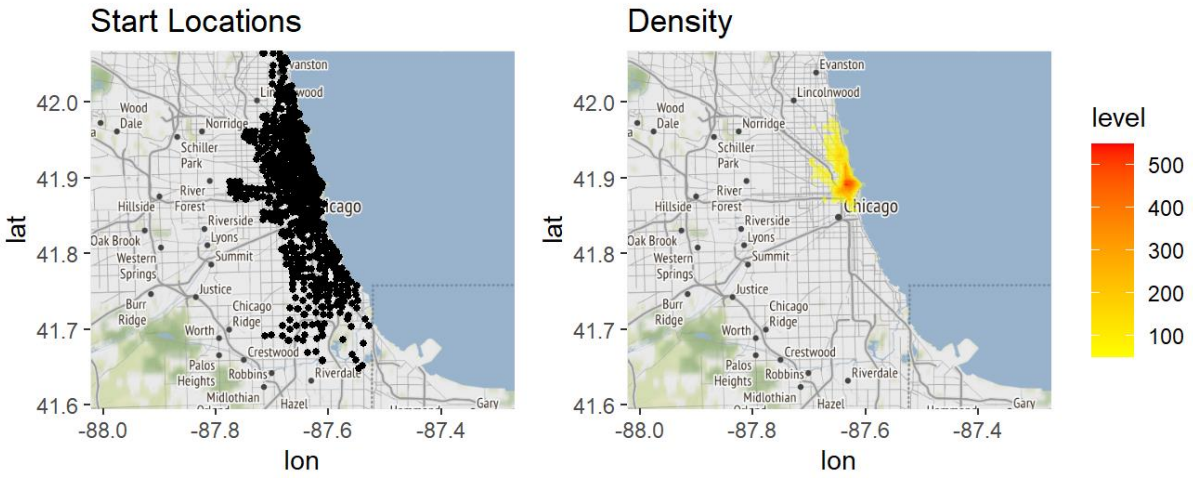



```
chicago <- get_stamenmap(bbox = c(left = -88.0225, bottom = 41.5949,
                                  right = -87.2713, top = 42.0677),
                         zoom = 10)
## Source : http://tile.stamen.com/terrain/10/261/379.png
## Source : http://tile.stamen.com/terrain/10/262/379.png
## Source : http://tile.stamen.com/terrain/10/263/379.png
## Source : http://tile.stamen.com/terrain/10/261/380.png
## Source : http://tile.stamen.com/terrain/10/262/380.png
## Source : http://tile.stamen.com/terrain/10/263/380.png
## Source : http://tile.stamen.com/terrain/10/261/381.png
## Source : http://tile.stamen.com/terrain/10/262/381.png
## Source : http://tile.stamen.com/terrain/10/263/381.png
chicago_map <- ggmap(chicago)

a<-chicago_map+geom_point(aes(x = start_lng, y = start_lat), data = location_data,
                          size = 1, alpha=1)+labs(title = "Start Locations")

b<-chicago_map+stat_density_2d(aes(x=start_lng, y=start_lat, fill = ..level..),
                               data = location_data, geom = "polygon", alpha =.5)+
  scale_fill_gradient(low = "yellow", high = "red")+labs(title = "Density")
```

```
a+b
```



7.2. End Locations…

```
location_data_1<-data_bike %>%
  group_by(end_station_name, end_lat,end_lng) %>%
  summarise(count=length(end_station_name)) %>%
  arrange(desc(count))
```

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

```
location_data_1
```

```
## # A tibble: 420,075 × 4
## # Groups:   end_station_name, end_lat [324,168]
##    end_station_name           end_lat end_lng count
##    <chr>                        <dbl>   <dbl> <int>
##  1 Streeter Dr & Grand Ave       41.9   -87.6 34269
##  2 Theater on the Lake           41.9   -87.6 28905
##  3 Lake Shore Dr & Monroe St     41.9   -87.6 26734
##  4 Clark St & Elm St             41.9   -87.6 24822
##  5 Lake Shore Dr & North Blvd    41.9   -87.6 23804
```

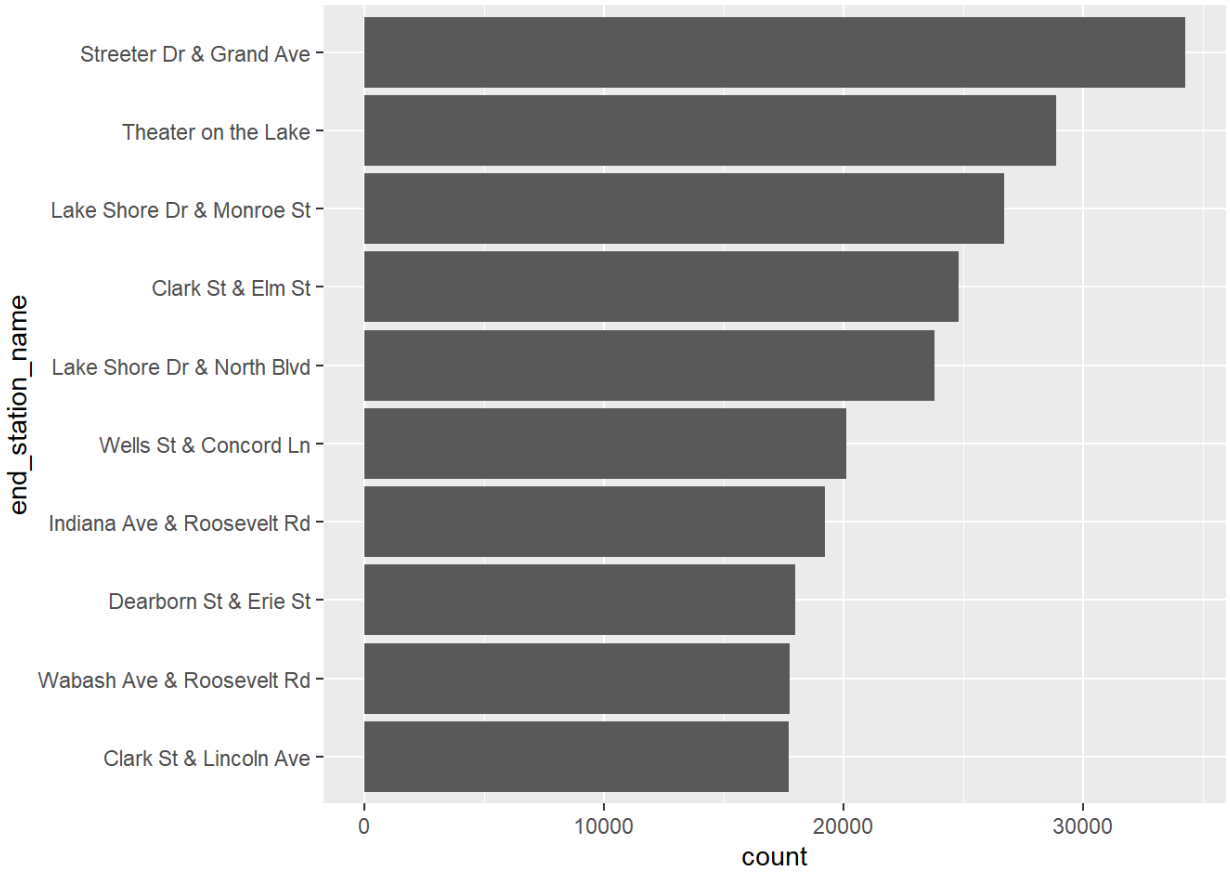
```
##  6 Wells St & Concord Ln          41.9   -87.6 20113
##  7 Indiana Ave & Roosevelt Rd     41.9   -87.6 19235
##  8 Dearborn St & Erie St          41.9   -87.6 18010
##  9 Wabash Ave & Roosevelt Rd      41.9   -87.6 17775
## 10 Clark St & Lincoln Ave         41.9   -87.6 17731
## # … with 420,065 more rows
```

```
popular_locations_2<-head(location_data_1,10)

popular_locations_2$end_station_name<-factor(popular_locations_2$end_station_name,
                                             levels = popular_locations_2$end_station_name
[order(popular_locations_2$count)])

locations_2<-ggplot(popular_locations_2,aes(x=count,y=end_station_name))+
  geom_bar(stat = "identity")

locations_2
```



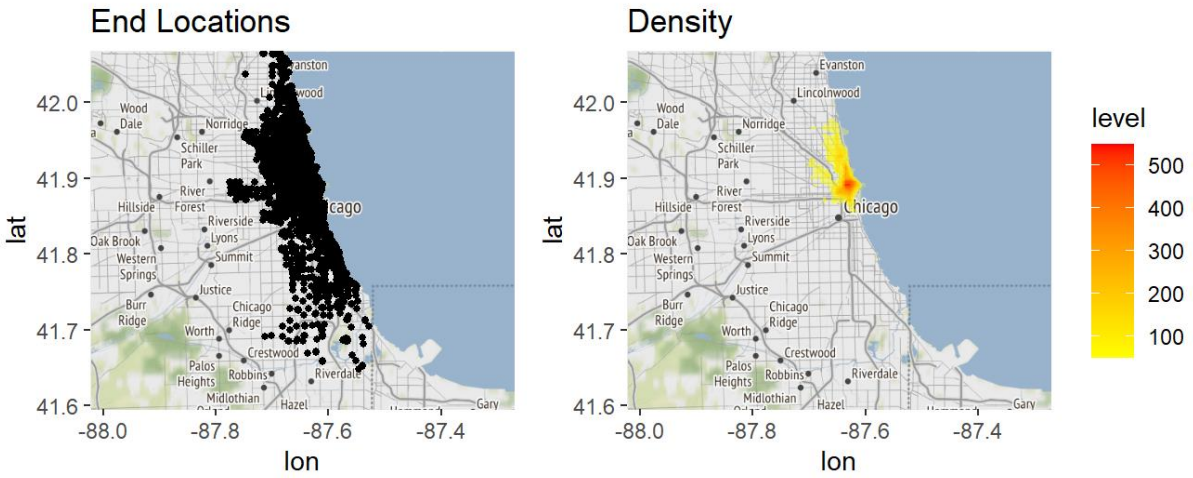

```
chicago <- get_stamenmap(bbox = c(left = -88.0225, bottom = 41.5949,
                                  right = -87.2713, top = 42.0677),
                         zoom = 10)
```

```
chicago_map <- ggmap(chicago)

c<-chicago_map+geom_point(aes(x = end_lng, y = end_lat), data = location_data_1,
                          size = 1, alpha=1)+labs(title = "End Locations")

d<-chicago_map+stat_density_2d(aes(x=end_lng, y=end_lat, fill = ..level..),
                               data = location_data_1, geom = "polygon", alpha =.5)+
  scale_fill_gradient(low = "yellow", high = "red")+labs(title = "Density")

c+d
```



8. Finally, let's see most popular routes people choose.

```
travel<-data_bike %>%
  select(start_station_name,start_lat,start_lng,end_station_name,end_lat,end_lng,rider_ty
pe)


travel$route<-paste(travel$start_station_name,travel$end_station_name,sep="--->")


travel<-travel %>%
  group_by(route,rider_type) %>%
  summarise(count=length(route)) %>%
```

```
  arrange(desc(count))
```

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

```
travel
```

```
## # A tibble: 223,676 × 3
## # Groups:   route [140,006]
##    route                                                rider_type count
##    <chr>                                                <chr>      <int>
##  1 Streeter Dr & Grand Ave--->Streeter Dr & Grand Ave   casual      6077
##  2 Lake Shore Dr & Monroe St--->Lake Shore Dr & Monroe St casual    5926
##  3 Millennium Park--->Millennium Park                   casual      4990
##  4 Buckingham Fountain--->Buckingham Fountain           casual      4839
##  5 Indiana Ave & Roosevelt Rd--->Indiana Ave & Roosevelt Rd casual  3878
##  6 Michigan Ave & Oak St--->Michigan Ave & Oak St       casual      3570
##  7 Fort Dearborn Dr & 31st St--->Fort Dearborn Dr & 31st St casual  3154
##  8 Michigan Ave & 8th St--->Michigan Ave & 8th St       casual      3131
##  9 Shore Dr & 55th St--->Shore Dr & 55th St             casual      2990
## 10 Theater on the Lake--->Theater on the Lake           casual      2863
## # … with 223,666 more rows
```

According to the table created, casual costumers highly prefer places where museums, theaters, parks, piers, lakes and any other attractions are located. Therefore, it would be beneficial to attract more annual riders by offering discounts on tickets for casual riders who are willing to convert their memberships.

ACT

Based on my analysis;

- The population of annual members are more than casual members. %59 of the customers are annual members whereas %41 of them are casual riders. According to past 12 months' data, comparing these values, almost half of the whole costumers are casual members.
- The number of customers using the bike share services increases through summer season and decreases through fall season. The members are more in percent than casual rider.
- Casual riders prefer using bike share services more during weekends and members are use them constantly over entire week.
- Members are cycling mostly at 8 AM and 5 PM, during the rush hours, and casual members tend to use services mostly at 5 PM.
- Both customers are choosing afternoons for riding.
- Docked bikes are being used more compared to other types by all customer types.
- Casual members prefer places where most of the attractions are located.

MY RECOMMENDATIONS

- Offer discounts in summer seasons to attract more casual riders.
- Increasing price for casual riders could be an effective strategy to convert them into annual members.
- Offer discounted prices during non busy hours to target casual riders.
- Provide additional offers for people who are registering in fall season.

- Provide discounts or special offers in the afternoons for casual members who are willing to convert their membership. This would attract them to convert their membership.
- Limit the distance of casual members can travel.

ADDITIONAL DATA REQUIRED

- Age and gender of customers: This information could be used to make further study on how to attract more customers based on different profiles.
- Occupation of customers: This information could be used to attract more people under same occupation.
- Address information: To examine how location parameters would affects the demand on bike share services.

RESORUCES

- GitHub
- Kaggle
- Stackoverflow