Data analysis of a bike sharing company

Pratik Kanade

24/05/2021

Introduction

The main aim of this analysis is to find the key differences for various aspects in usage of bikes by customers and subscribers of Cyclistic, a bike sharing company in Chicago. People who purchase single-ride or full-day passes are referred to as customers whereas those who purchase annual memberships are Cyclistic subscribers. The results of this analysis will further be considered while taking data driven decisions of converting the customers into subscribers for optimizing the profits of the company.

Data Source

The data used for this case study is made available by Motivate International Inc. under this license. The data was downloaded from https://divvy-tripdata.s3.amazonaws.com/index.html - a link provided on Coursera's Data Analysis class webpage. Data was downloaded with R programming language. This public data is used to explore how different member types are using the Cyclistic bikes. (Note: The datasets have a different name as Cyclistic is a fictional company.)

Preparing the Environment

Various packages are loaded into RStudio in order to use the different functions associated with these packages to perform different tasks.

```
library("tidyverse") # For importing and wrangling data
library("rmarkdown") # For making reports in R markdown
library("lubridate") # For date functions
library("ggplot2") # For data visualization
library("knitr") # For tables in r markdown
```

Importing Data

Data is in .csv file format. It contains information on each trip from between 1st January 2016 to 30th April 2021 (Note : Data is available for the first four months of 2021)

```
January_march_2016_tripdata <- read.csv("~/January_march_2016_tripdata.csv")
April_2016_tripdata <- read.csv("~/April_2016_tripdata.csv")
May_2016_tripdata <- read.csv("~/May_2016_tripdata.csv")
June_2016_tripdata <- read.csv("~/June_2016_tripdata.csv")
July_september_2016_tripdata <- read.csv("~/July_september_2016_tripdata.csv"
)
October_december_2016_tripdata <- read.csv("~/October_december_2016_tripdata.
csv")

January_march_2017_tripdata <- read.csv("~/January_march_2017_tripdata.csv")
April_june_2017_tripdata <- read.csv("~/April_june_2017_tripdata.csv")
July_september_2017_tripdata <- read.csv("~/July_september_2017_tripdata.csv"
)
October_december_2017_tripdata <- read.csv("~/October_december_2017_tripdata.
csv")

January_march_2018_tripdata <- read.csv("~/January_march_2018_tripdata.csv")
April_june_2018_tripdata <- read.csv("~/April_june_2018_tripdata.csv")
July_september_2018_tripdata <- read.csv("~/July_september_2018_tripdata.csv"
)
October_december_2018_tripdata <- read.csv("~/October_december_2018_tripdata.
csv")

January_march_2019_tripdata <- read.csv("~/January_march_2019_tripdata.csv")
April_june_2019_tripdata <- read.csv("~/April_june_2019_tripdata.csv")
July_september_2019_tripdata <- read.csv("~/July_september_2019_tripdata.csv"
)
October_december_2019_tripdata <- read.csv("~/October_december_2019_tripdata.
csv")

January_march_2020_tripdata <- read.csv("~/January_march_2020_tripdata.csv")
April_2020_tripdata <- read.csv("~/April_2020_tripdata.csv")
May_2020_tripdata <- read.csv("~/May_2020_tripdata.csv")
June_2020_tripdata <- read.csv("~/June_2020_tripdata.csv")
July_2020_tripdata <- read.csv("~/July_2020_tripdata.csv")
August_2020_tripdata <- read.csv("~/August_2020_tripdata.csv")
September_2020_tripdata <- read.csv("~/September_2020_tripdata.csv")
October_2020_tripdata <- read.csv("~/October_2020_tripdata.csv")
November_2020_tripdata <- read.csv("~/November_2020_tripdata.csv")
December_2020_tripdata <- read.csv("~/December_2020_tripdata.csv")

January_2021_tripdata <- read.csv("~/January_2021_tripdata.csv")
Febraury_2021_tripdata <- read.csv("~/February_2021_tripdata.csv")
March_2021_tripdata <- read.csv("~/March_2021_tripdata.csv")
April_2021_tripdata <- read.csv("~/April_2021_tripdata.csv")
```

Data Cleaning and Manipulation

Documentation of the data cleaning process performed on the raw data that was imported into R.

Cleaning and wrangling of the datasets for the year 2016

```
# compare the column names of the datasets

colnames(January_march_2016_tripdata)
colnames(April_2016_tripdata)
colnames(May_2016_tripdata)
colnames(June_2016_tripdata)
colnames(July_september_2016_tripdata)
colnames(October_december_2016_tripdata)

# combine monthly datasets into one dataframe

tripdata_2016 <- rbind(January_march_2016_tripdata,April_2016_tripdata,
    May_2016_tripdata,June_2016_tripdata,July_september_2016_tripdata,
    October_december_2016_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2016)
head(tripdata_2016)
tail(tripdata_2016)
table(tripdata_2016$usertype)

# sort the dataframe according to trip id

tripdata_2016 <- tripdata_2016 %>%
  arrange(trip_id)

# rename the columns for consistency

tripdata_2016 <- tripdata_2016 %>%
  rename(start_time = starttime, end_time = stoptime)

# remove the columns from the dataframe that are not required

tripdata_2016$birthyear <- NULL
tripdata_2016$gender <- NULL
tripdata_2016$bikeid <- NULL
```

```r
# remove rows that contain 'Dependent' as the 'usertype'

tripdata_2016 <- tripdata_2016[!(tripdata_2016$usertype == "Dependent"),]

# change the data type of the columns for further calculations and analysis

tripdata_2016 <- mutate(tripdata_2016, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

tripdata_2016$start_time <- as.POSIXct(tripdata_2016$start_time,
                                       format = "%m/%d/%Y %H:%M")
tripdata_2016$end_time <- as.POSIXct(tripdata_2016$end_time,
                                     format = "%m/%d/%Y %H:%M")

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2016$date <- as.Date(as.POSIXct(tripdata_2016$start_time), tz = "")
tripdata_2016$year <- format(as.Date(tripdata_2016$date), "%Y")
tripdata_2016$month <- format(as.Date(tripdata_2016$date), "%m")
tripdata_2016$day <- format(as.Date(tripdata_2016$date), "%d")
tripdata_2016$day_of_week <- format(as.Date(tripdata_2016$date), "%A")
```

Cleaning and wrangling of the datasets for the year 2017

```r

# compare the column names of the datasets

colnames(January_march_2017_tripdata)
colnames(April_june_2017_tripdata)
colnames(July_september_2017_tripdata)
colnames(October_december_2017_tripdata)

# combine monthly datasets into one dataframe

January_september_2017_tripdata <- rbind(January_march_2017_tripdata,
              April_june_2017_tripdata, July_september_2017_tripdata)

# change the data type of the columns for further calculations and analysis

January_september_2017_tripdata$start_time <- as.POSIXct(January_september_20
17_tripdata$start_time,
            format = "%m/%d/%Y %H:%M:%S")
January_september_2017_tripdata$end_time <- as.POSIXct(January_september_2017
```

```
_tripdata$end_time,
           format = "%m/%d/%Y %H:%M:%S")

October_december_2017_tripdata$start_time <- as.POSIXct(October_december_2017
_tripdata$start_time,
           format = "%m/%d/%Y %H:%M")
October_december_2017_tripdata$end_time <- as.POSIXct(October_december_2017_t
ripdata$end_time,
           format = "%m/%d/%Y %H:%M")

# combine monthly datasets into one dataframe

tripdata_2017 <- rbind(January_september_2017_tripdata, October_december_2017
_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2017)
head(tripdata_2017)
tail(tripdata_2017)
table(tripdata_2017$usertype)

# sort the dataframe according to trip id

tripdata_2017 <- tripdata_2017 %>%
  arrange(trip_id)

# remove the columns from the dataframe that are not required

tripdata_2017$gender <- NULL
tripdata_2017$birthyear <- NULL
tripdata_2017$bikeid <- NULL

# remove rows that contain 'Dependent' as the 'usertype'

tripdata_2017 <- tripdata_2017[!(tripdata_2017$usertype == "Dependent"),]

# change the data type of the columns for further calculations and analysis

tripdata_2017 <- mutate(tripdata_2017, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2017$date <- as.Date(as.POSIXct(tripdata_2017$start_time), tz = "")
```

```
tripdata_2017$year <- format(as.Date(tripdata_2017$date), "%Y")
tripdata_2017$month <- format(as.Date(tripdata_2017$date), "%m")
tripdata_2017$day <- format(as.Date(tripdata_2017$date), "%d")
tripdata_2017$day_of_week <- format(as.Date(tripdata_20217$date), "%A")
```

Cleaning and wrangling of the datasets for the year 2018

```
# compare the column names of the datasets

colnames(January_march_2018_tripdata)
colnames(April_june_2018_tripdata)
colnames(July_september_2018_tripdata)
colnames(October_december_2018_tripdata)

# rename the columns for consistency

January_march_2018_tripdata <- January_march_2018_tripdata %>%
  rename(trip_id = X01...Rental.Details.Rental.ID, start_time = X01...Rental.
Details.Local.Start.Time, end_time = X01...Rental.Details.Local.End.Time, bik
eid = X01...Rental.Details.Bike.ID, tripduration = X01...Rental.Details.Durat
ion.In.Seconds.Uncapped, from_station_id = X03...Rental.Start.Station.ID, fro
m_station_name = X03...Rental.Start.Station.Name, to_station_id = X02...Renta
l.End.Station.ID, to_station_name = X02...Rental.End.Station.Name, usertype =
User.Type, gender = Member.Gender, birthyear = X05...Member.Details.Member.Bi
rthday.Year)

# combine monthly datasets into one dataframe

tripdata_2018 <- rbind(January_march_2018_tripdata, April_june_2018_tripdata,
July_september_2018_tripdata,October_december_2018_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2018)
head(tripdata_2018)
tail(tripdata_2018)
table(tripdata_2018$usertype)

# sort the dataframe according to trip id

tripdata_2018 <- tripdata_2018 %>%
  arrange(trip_id)

# remove the columns from the dataframe that are not required

tripdata_2018$gender <- NULL
```

```r
tripdata_2018$birthyear <- NULL
tripdata_2018$bikeid <- NULL

# change the data type of the columns for further calculations and analysis

tripdata_2018 <- mutate(tripdata_2018, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

tripdata_2018$start_time <- as.POSIXct(tripdata_2018$start_time,
                                       format = "%d-%m-%Y %H:%M")
tripdata_2018$end_time <- as.POSIXct(tripdata_2018$end_time,
                                     format = "%d-%m-%Y %H:%M")

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2018$date <- as.Date(as.POSIXct(tripdata_2018$start_time), tz = "")
tripdata_2018$year <- format(as.Date(tripdata_2018$date), "%Y")
tripdata_2018$month <- format(as.Date(tripdata_2018$date), "%m")
tripdata_2018$day <- format(as.Date(tripdata_2018$date), "%d")
tripdata_2018$day_of_week <- format(as.Date(tripdata_2018$date), "%A")
```

Cleaning and wrangling of the datasets for the year 2019

```r
# compare the column names of the datasets

colnames(January_march_2019_tripdata)
colnames(April_june_2019_tripdata)
colnames(July_september_2019_tripdata)
colnames(October_december_2019_tripdata)

# rename the columns for consistency

April_june_2019_tripdata <- April_june_2019_tripdata %>%
  rename(trip_id = X01...Rental.Details.Rental.ID, start_time = X01...Rental.
Details.Local.Start.Time, end_time = X01...Rental.Details.Local.End.Time, bik
eid = X01...Rental.Details.Bike.ID, tripduration = X01...Rental.Details.Durat
ion.In.Seconds.Uncapped, from_station_id = X03...Rental.Start.Station.ID, fro
m_station_name = X03...Rental.Start.Station.Name, to_station_id = X02...Renta
l.End.Station.ID, to_station_name =  X02...Rental.End.Station.Name, usertype
= User.Type, gender = Member.Gender, birthyear = X05...Member.Details.Member.
Birthday.Year)

# combine monthly datasets into one dataframe
```

```r
tripdata_2019 <- rbind(January_march_2019_tripdata, April_june_2019_tripdata,
July_september_2019_tripdata, October_december_2019_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2019)
head(tripdata_2019)
tail(tripdata_2019)
table(tripdata_2019$usertype)

# sort the dataframe according to trip id

tripdata_2019 <- tripdata_2019 %>%
  arrange(trip_id)

# remove the columns from the dataframe that are not required

tripdata_2019$gender <- NULL
tripdata_2019$birthyear <- NULL
tripdata_2019$bikeid <- NULL

# change the data type of the columns for further calculations and analysis

tripdata_2019 <- mutate(tripdata_2019, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

tripdata_2019$start_time <- as.POSIXct(tripdata_2019$start_time,
                                       format = "%d-%m-%Y %H:%M")
tripdata_2019$end_time <- as.POSIXct(tripdata_2019$end_time,
                                     format = "%d-%m-%Y %H:%M")

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2019$date <- as.Date(as.POSIXct(tripdata_2019$start_time), tz = "")
tripdata_2019$year <- format(as.Date(tripdata_2019$date), "%Y")
tripdata_2019$month <- format(as.Date(tripdata_2019$date), "%m")
tripdata_2019$day <- format(as.Date(tripdata_2019$date), "%d")
tripdata_2019$day_of_week <- format(as.Date(tripdata_2019$date), "%A")
```

Cleaning and wrangling of the datasets for the year 2020

```r
# compare the column names of the datasets

colnames(January_march_2020_tripdata)
colnames(April_2020_tripdata)
colnames(May_2020_tripdata)
colnames(June_2020_tripdata)
colnames(July_2020_tripdata)
colnames(August_2020_tripdata)
colnames(September_2020_tripdata)
colnames(October_2020_tripdata)
colnames(November_2020_tripdata)
colnames(December_2020_tripdata)

# combine monthly datasets into one dataframe

tripdata_2020 <- rbind(January_march_2020_tripdata,April_2020_tripdata,May_2020_tripdata, June_2020_tripdata, July_2020_tripdata, August_2020_tripdata, September_2020_tripdata, October_2020_tripdata, November_2020_tripdata, December_2020_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2020)
head(tripdata_2020)
tail(tripdata_2020)
table(tripdata_2020$usertype)

# sort the dataframe according to start time of the trip

tripdata_2020 <- tripdata_2020 %>%
  arrange(started_at)

# remove the columns from the dataframe that are not required

tripdata_2020$start_lat <- NULL
tripdata_2020$start_lng <- NULL
tripdata_2020$end_lat <- NULL
tripdata_2020$end_lng <- NULL

# create a new calculated field 'tripduration' to calculate the duration of each trip in seconds

tripdata_2020$tripduration <- difftime(tripdata_2020$ended_at,tripdata_2020$started_at)

# rearrange the columns in the desired consistent order

tripdata_2020 <- tripdata_2020[ ,c(1,3,4,2,10,6,5,8,7,9)]
```

```r
# rename the columns for consistency

tripdata_2020 <- tripdata_2020 %>%
  rename(trip_id = ride_id, start_time = started_at, end_time = ended_at, bik
eid = rideable_type, from_station_id = start_station_id,
         from_station_name = start_station_name, to_station_id = end_station_
id, to_station_name = end_station_name,
         usertype = member_casual)

# consolidate the labels of the 'usertype' column from 'casual' to 'Customer'
and 'member' to 'Subscriber'

tripdata_2020 <- tripdata_2020 %>%
  mutate(usertype=recode(usertype, "casual"="Customer", "member" = "Subscribe
r"))

# remove the columns from the dataframe that are not required

tripdata_2020$bikeid <- NULL

# change the data type of the columns for further calculations and analysis

tripdata_2020 <- mutate(tripdata_2020, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

tripdata_2020$start_time <- as.POSIXct(tripdata_2020$start_time,
                                       format = "%d-%m-%Y %H:%M")
tripdata_2020$end_time <- as.POSIXct(tripdata_2020$end_time,
                                     format = "%d-%m-%Y %H:%M")

tripdata_2020$tripduration <- as.numeric(gsub(",","",tripdata_2020$tripdurati
on))

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2020$date <- as.Date(as.POSIXct(tripdata_2020$start_time), tz = "")
tripdata_2020$year <- format(as.Date(tripdata_2020$date), "%Y")
tripdata_2020$month <- format(as.Date(tripdata_2020$date), "%m")
tripdata_2020$day <- format(as.Date(tripdata_2020$date), "%d")
tripdata_2020$day_of_week <- format(as.Date(tripdata_2020$date), "%A")

# filter out the bad data

tripdata_2020 <- tripdata_2020 %>%
  filter(!(tripduration <= 0 | from_station_name == "HQ QR"))
```

Cleaning and wrangling of the datasets for the year 2021

```
# compare the column names of the datasets

colnames(January_2021_tripdata)
colnames(February_2021_tripdata)
colnames(March_2021_tripdata)
colnames(April_2021_tripdata)

# combine monthly datasets into one dataframe

tripdata2021 <- rbind(January_2021_tripdata,February_2021_tripdata,March_2021
_tripdata,April_2021_tripdata)

# Inspect the dataframe for inconguencies

str(tripdata_2020)
head(tripdata_2020)
tail(tripdata_2020)
table(tripdata_2020$usertype)

# sort the dataframe according to start time of the trip

tripdata_2021 <- tripdata2021 %>%
  arrange(started_at)

# remove the columns from the dataframe that are not required

tripdata2021$start_lat <- NULL
tripdata2021$start_lng <- NULL
tripdata2021$end_lat <- NULL
tripdata2021$end_lng <- NULL

# create a new calculated field 'tripduration' to calculate the duration of e
ach trip in seconds

tripdata_2021$tripduration <- difftime(tripdata_2021$ended_at,tripdata_2021$s
tarted_at)

# rearrange the columns in the desired consistent order

tripdata_2021 <- tripdata_2021[ ,c(1,3,4,2,10,6,5,8,7,9)]

# rename the columns for consistency
```

```r
tripdata_2021 <- tripdata_2021 %>%
  rename(trip_id = ride_id, bikeid = rideable_type, start_time = started_at,
end_time = ended_at,
         from_station_id = start_station_id, from_station_name = start_statio
n_name, to_station_id = end_station_id,
         to_station_name = end_station_name, usertype = member_casual)

# consolidate the labels of the 'usertype' column from 'casual' to 'Customer'
and 'member' to 'Subscriber'

tripdata_2021 <- tripdata_2021 %>%
  mutate(usertype=recode(usertype, "casual"="Customer", "member" = "Subscribe
r"))

# remove the columns from the dataframe that are not required

tripdata2021$bikeid <- NULL

# change the data type of the columns for further calculations and analysis

tripdata_2021 <- mutate(tripdata_2021, trip_id = as.character(trip_id),
                        from_station_id = as.character(from_station_id),
                        to_station_id = as.character(to_station_id))

tripdata_2021$start_time <- as.POSIXct(tripdata_2021$start_time,
                                       format = "%d-%m-%Y %H:%M")
tripdata_2021$end_time <- as.POSIXct(tripdata_2021$end_time,
                                     format = "%d-%m-%Y %H:%M:")

tripdata_2021$tripduration <- as.numeric(gsub(",","",tripdata_2020$tripdurati
on))

# create columns for 'date', 'year', 'month', 'day' and 'day of week' to chan
ge the granularity
# data can be aggregated for each year, month, day instead of ride level

tripdata_2021$date <- as.Date(as.POSIXct(tripdata_2021$start_time), tz = "")
tripdata_2021$year <- format(as.Date(tripdata_2021$date), "%Y")
tripdata_2021$month <- format(as.Date(tripdata_2021$date), "%m")
tripdata_2021$day <- format(as.Date(tripdata_2021$date), "%d")
tripdata_2021$day_of_week <- format(as.Date(tripdata_2021$date), "%A")

# filter out the bad data

tripdata_2021 <- tripdata_2021 %>%
  filter(!(tripduration <= 0 | from_station_name == "HQ QR"))
```

Data Analysis and Visualization

A data frame for every year is created after the raw datasets are cleaned. Two calculated fields are introduced in these data frames. The average trip duration and number of trips is calculated for each day of the week for each year. The 'average_duration' column contains the average trip duration and 'number_of_rides' column contains the number of trips.

For the year 2016

```
tripdata_2016_final <- tripdata_2016 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

For the year 2017

```
tripdata_2017_final <- tripdata_2017 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

For the year 2018

```
tripdata_2018_final <- tripdata_2018 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

For the year 2019

```
tripdata_2019_final <- tripdata_2019 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

For the year 2020

```
tripdata_2020_final <- tripdata_2020 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

For the year 2021

```
tripdata_2021_final <- tripdata_2021 %>%
  group_by(usertype, day_of_week, year) %>%
  summarise(number_of_rides = n(), average_duration = mean(tripduration)) %>%
  arrange(usertype,day_of_week)
```

The rbind function is used to merge the final data frames of each year to create one data frame, 'tripdata'. This is the final data frame used for the analysis process.

```
tripdata <- rbind(tripdata_2016_final,tripdata_2017_final,tripdata_2018_final
, tripdata_2019_final, tripdata_2020_final, tripdata_2021_final)
head(tripdata) # shows first six rows of dataframe
```

```
## # A tibble: 6 x 5
## # Groups:   usertype, day_of_week [6]
##   usertype day_of_week year  number_of_rides average_duration
##   <chr>    <chr>       <chr>           <int>            <dbl>
## 1 Customer Sunday      2016           218279            1896.
## 2 Customer Monday      2016           118778            1846.
## 3 Customer Tuesday     2016            78180            1809.
## 4 Customer Wednesday   2016            59287            1838.
## 5 Customer Thursday    2016            67437            1798.
## 6 Customer Friday      2016           101940            1854.
```

```
tail(tripdata) # shows last six rows of dataframe
```

```
## # A tibble: 6 x 5
## # Groups:   usertype, day_of_week [6]
##   usertype   day_of_week year  number_of_rides average_duration
##   <chr>      <chr>       <chr>           <int>            <dbl>
## 1 Subscriber Monday      2021            65700             867.
## 2 Subscriber Tuesday     2021            71678             842.
## 3 Subscriber Wednesday   2021            65952             820.
## 4 Subscriber Thursday    2021            62055             779.
## 5 Subscriber Friday      2021            72905             829.
## 6 Subscriber Saturday    2021            68663             962.
```

The 'tripdata' dataset contains the average trip duration and the number of rides, the two main parameters that are considered while performing this analysis. The analysis is performed for two different user types that use the bikes: Customers and Subscribers. Four different cases are included in this analysis. The analysis is performed at different granularity levels of year and days of the week depending on each case. The tables below contain the results from the conducted analysis. Graphs are plotted for visualization of the data from the table to get a clear representation of the results.

Case 1:

The average trip duration is calculated for the days of the week. These results are further categorized on the basis of user type. The calculations include average of trip duration for all the years from 2016 to 2021.

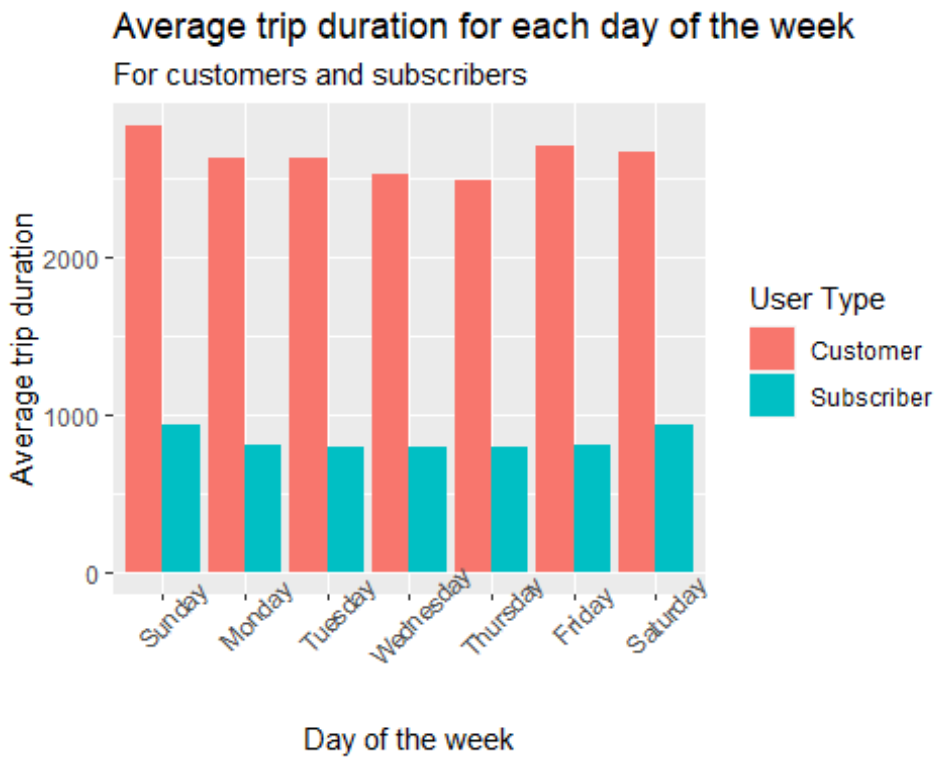
```
tripdata %>%
  group_by(day_of_week, usertype) %>%
  summarise(average_duration = mean(average_duration)) %>%
```

```
  kable(format = 'simple', caption = "Average trip duration for days of week"
)
```

Average trip duration for days of week

day_of_week	usertype	average_duration
Sunday	Customer	2829.0262
Sunday	Subscriber	928.2174
Monday	Customer	2628.3471
Monday	Subscriber	807.3644
Tuesday	Customer	2622.7833
Tuesday	Subscriber	796.2701
Wednesday	Customer	2527.3848
Wednesday	Subscriber	793.2232
Thursday	Customer	2492.1776
Thursday	Subscriber	789.4841
Friday	Customer	2709.5093
Friday	Subscriber	806.4464
Saturday	Customer	2668.5118
Saturday	Subscriber	928.6248

```
tripdata %>%
  group_by(day_of_week, usertype) %>%
  summarise(average_duration = mean(average_duration)) %>%
  ggplot(aes(x=day_of_week, y=average_duration, fill = usertype))+
  geom_col(position = "dodge")+
  labs(title = "Average trip duration for each day of the week",  fill = "Use
r Type", subtitle = "For customers and subscribers")+
  xlab("Day of the week")+
  ylab("Average trip duration")+
  theme(axis.text.x = element_text(angle = 45))
```



From the plot, it can be inferred that the average trip duration for the bike rides of the customers are much more than those of the subscribers. Customers are the people who purchase a single ride or full day pass. These users usually use their pass to the fullest in order to make it worth their spending.This results in Longer trip duration which in turn results in longer average trip duration. Subscribers on the other hand have an annual pass and they do not care much of the duration they use the bikes for. The duration won't affect their spending on the pass and therefore a shorter average trip distance can be seen for subscribers.

Case 2:

The number of rides are calculated in this case instead of the average trip duration. A sum of number of rides for each day of the week for all six years is calculated and categorized on the basis of user type.

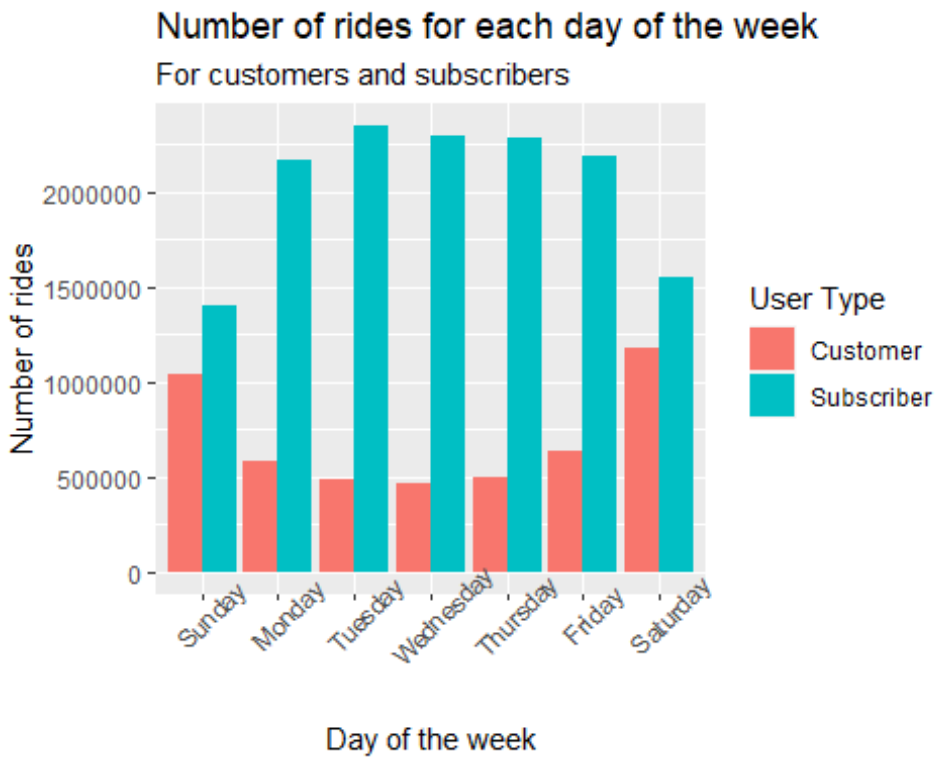
```
tripdata %>%
  group_by(day_of_week, usertype) %>%
  summarise(number_of_rides = sum(number_of_rides)) %>%
  kable(format = 'simple', caption = "Number of rides for days of week")
```

Number of rides for days of week

day_of_week	usertype	number_of_rides

Sunday	Customer	1037576
Sunday	Subscriber	1403982
Monday	Customer	578391
Monday	Subscriber	2169455
Tuesday	Customer	479386
Tuesday	Subscriber	2341563
Wednesday	Customer	457893
Wednesday	Subscriber	2295081
Thursday	Customer	490928
Thursday	Subscriber	2285060
Friday	Customer	638325
Friday	Subscriber	2183135
Saturday	Customer	1178903
Saturday	Subscriber	1544880

```
tripdata %>%
  group_by(day_of_week, usertype) %>%
  summarise(number_of_rides = sum(number_of_rides)) %>%
  ggplot(aes(x=day_of_week, y=number_of_rides, fill = usertype))+
  geom_col(position = "dodge")+
  labs(title = "Number of rides for each day of the week",  fill = "User Type
", subtitle = "For customers and subscribers")+
  xlab("Day of the week")+
  ylab("Number of rides")+
  theme(axis.text.x = element_text(angle = 45))
```



Even though the average trip duration for subscribers are less as compared to customers, the number of rides for subscribers are far more than the rides for the customers. This gives the use of the bikes by both the user types on the daily basis. On any day of the week the subscribers use the bikes more than the customers. This can be seen mainly because people with annual pass do not have to pay for every bike ride whereas customers with single ride or full day pass have to pay for every new ride or on each day they use the bike respectively.

Another trend that can be seen is individually the subscribers have a bike usage more on the weekdays than on the weekends. The customers use the bikes more on weekends as compared to on weekdays. This suggests that the subscribers use the bike for daily commute to their work on the weekdays which might be the reason they prefer an annual pass. The customers might be riding the bikes around the town in their free time on weekends making their preference tending towards the single ride or full day pass.

Case 3:

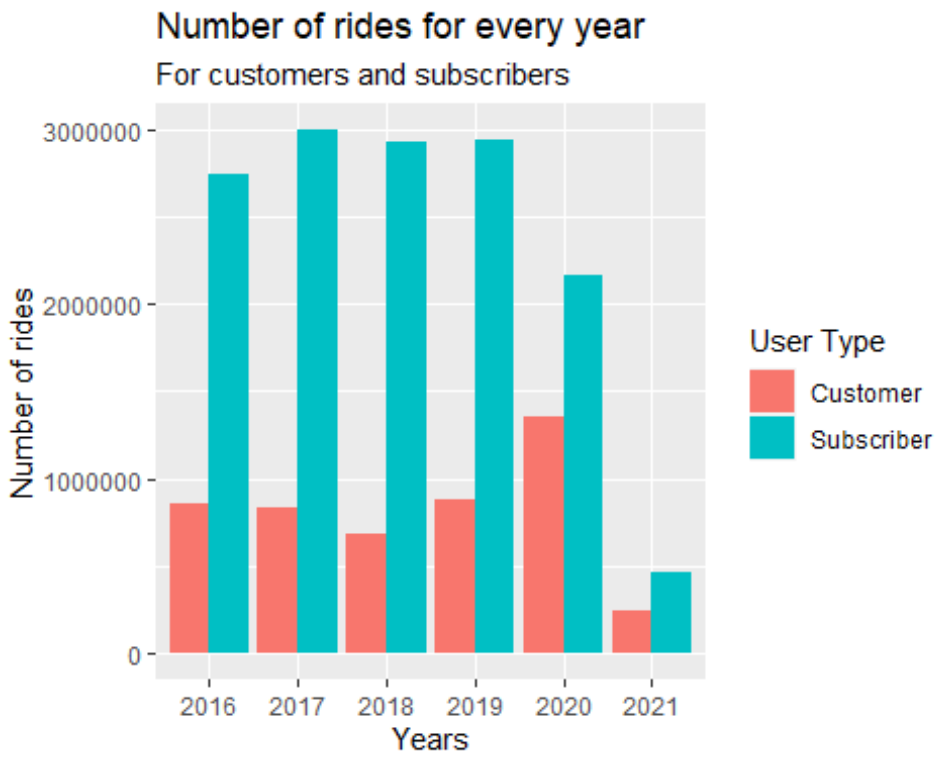
The number of rides for all the years from 2016 to 2021 are evaluated. The number of rides for the 2021 are low as the data for only the first four months is available. The results are further distinguished based on the user type that use the bikes of the company and represented in another table.

```
tripdata %>%
  group_by(year, usertype) %>%
  summarise(number_of_rides = sum(number_of_rides)) %>%
  kable(format = 'simple', caption = "Number of rides for years")
```

Number of rides for years

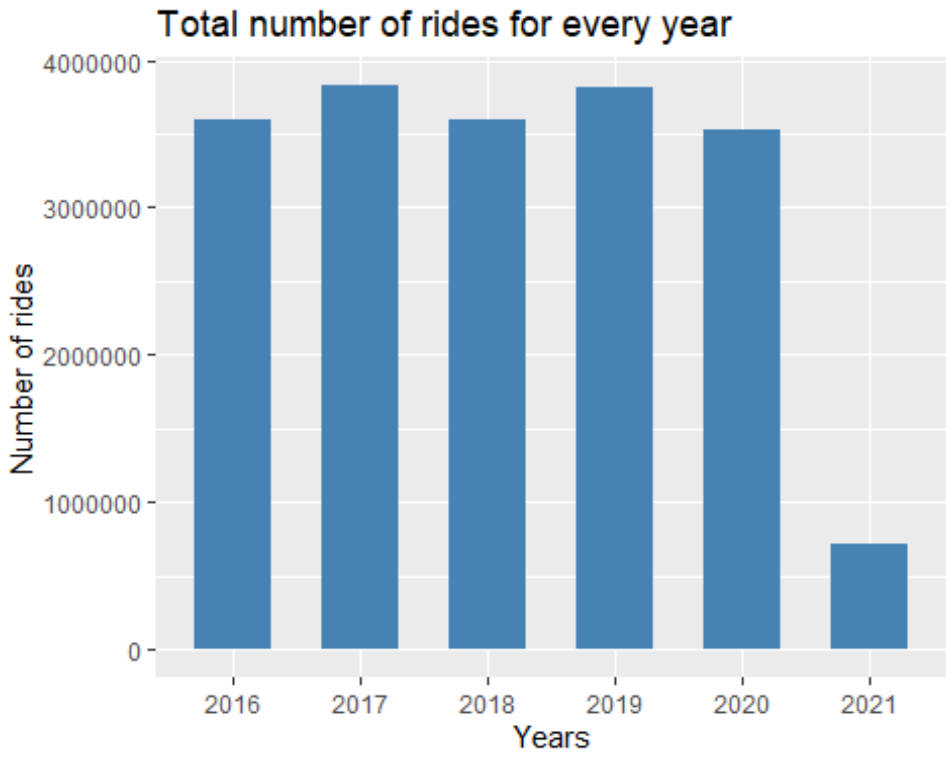
year	usertype	number_of_rides
2016	Customer	858474
2016	Subscriber	2736869
2017	Customer	836872
2017	Subscriber	2992135
2018	Customer	677156
2018	Subscriber	2925926
2019	Customer	880637
2019	Subscriber	2937367
2020	Customer	1359398
2020	Subscriber	2167602
2021	Customer	248865
2021	Subscriber	463257

```
tripdata %>%
  group_by(year, usertype) %>%
  summarise(number_of_rides = sum(number_of_rides)) %>%
  ggplot(aes(x=year, y=number_of_rides, fill = usertype))+
  geom_col(position = "dodge")+
  labs(title = "Number of rides for every year",  fill = "User Type", subtitl
e = "For customers and subscribers")+
  xlab("Years")+
  ylab("Number of rides")
```



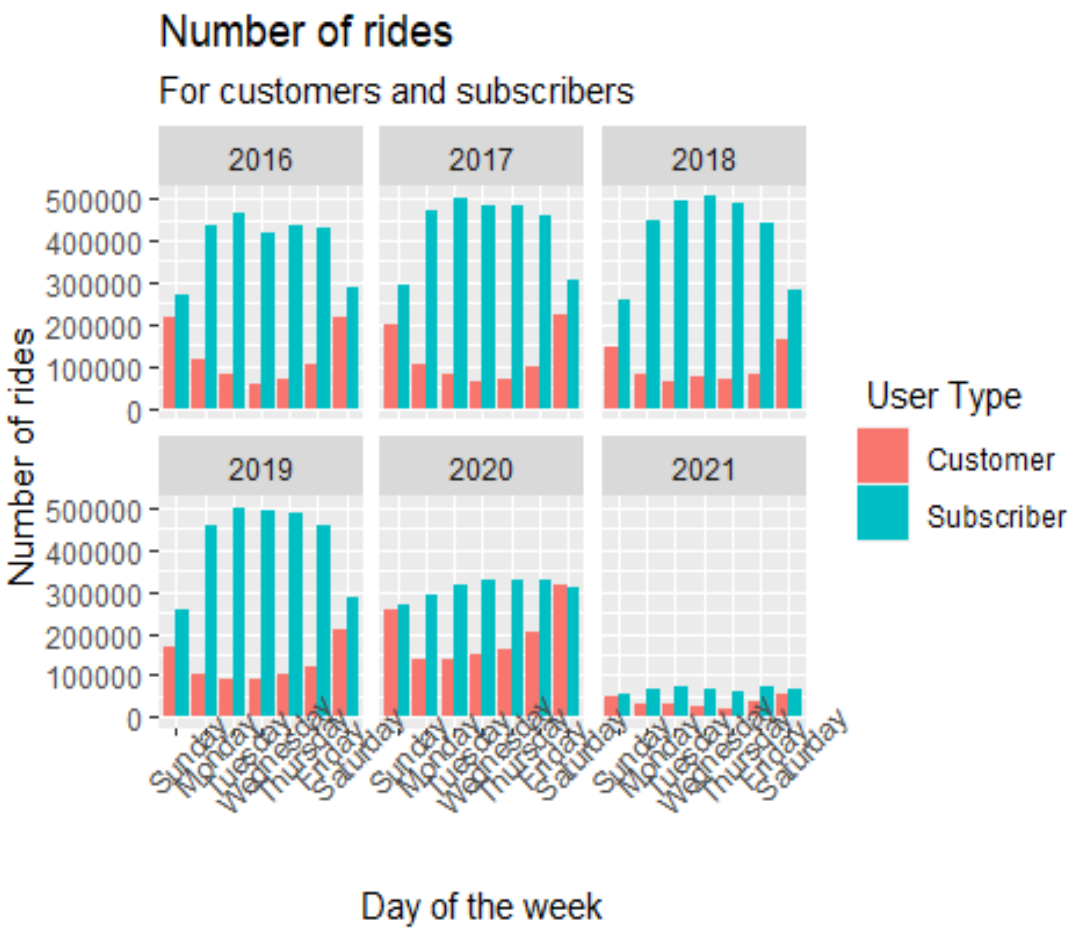
As inferred from the previous case, the number of rides for subscribers is more than for the customers. The plot in the previous case had a granular level of days of the week. This plot has a granularity level of years and same findings can be noticed for every year. The number of rides for the year 2021 are low as the data for only the first four months was available. Compared to other years, The year 2020 saw a decline in the subscribers and a rise in the number of customers. These changes can be the result of Covid-19 pandemic. Subscribers used the bikes as a way of commute to their work but due to the outburst of the pandemic a work from home policy was introduced in the most affected countries. Due to the uncertainty of the pandemic many subscribes might have unsubscribed from their annual pass and paid for the single ride or full day pass when necessary. This eventually raised the number of other user type in customers. This can be concluded as the total number of users of Cyclistic bikes did not change drastically as shown in the below plot.

```
tripdata %>%
  group_by(year) %>%
  summarise(number_of_rides = sum(number_of_rides)) %>%
  ggplot(aes(x=year, y=number_of_rides))+
  geom_col(position = "dodge", width = 0.6, fill = "Steelblue")+
  labs(title = "Total number of rides for every year")+
  xlab("Years")+
  ylab("Number of rides")
```



For better understanding, the results from the previous two cases, case 2 and case 3, are taken into consideration. A graph is plotted for the number of rides for days of the week. Another layer of all the years is added to the plot using the 'facet_wrap' function. This plot gives a clear representation of all the trips for customers and subscribers for every year and day of the week. The insights from the two cases can be visualized simultaneously from this plot. Similar findings can be concluded from this plot,

```
tripdata %>%
  ggplot(aes(x=day_of_week, y=number_of_rides, fill = usertype))+
           geom_col(position = "dodge")+
  labs(title = "Number of rides", subtitle = "For customers and subscribers",
fill = " User Type")+
  xlab("Day of the week")+
  ylab("Number of rides")+
  facet_wrap(~year)+
  theme(axis.text.x = element_text(angle = 45))
```



Conclusion and Recommendations

- The average trip duration of the bike rides of customers is much greater than subscribers. Most of the customers use the bikes until their single ride or full day pass expires to make it worth the money they spend. Whereas, the subscribers can use the bikes whenever they want and for any amount of time without being charged for each new ride. This results in the subscribers taking more short trips which drops their average trip duration. The subscribers use the bikes more efficiently which results in more availability of docked bikes for other riders benefiting the company. The subscribers also use the bikes at lower rates than customers. This information should be provided to the customers in a more compelling way to convert them into subscribers.
- The number of bike rides for customers is less than those for subscribers. However, customers use the bikes more on the weekends than on the weekdays. A non chargeable first weekend ride can be offered to the new subscribers. These rides will not be charged on their annual pass for certain number of minutes. Once these

minutes escalate, the subscribers will be charged according to the normal annual pass plan. This offer might convert the users from customers to subscribers as they will get a better deal for their weekend rides along with other benefits like using the bikes on weekdays at lower rates.

- The year 2020 saw drop in the number of subscribers because of the Covid-19 pandemic. Subscribers did not renew their annual pass. Instead there was a rise in the number of customers. However, the number of total users were approximately similar to the previous years. This suggests that the subscribers opted for a single ride or full day pass instead of the annual pass whenever necessary. To retain the existing subscribers, regain the subscribers that did not renew their annual pass in 2020 and to convert remaining customers to subscribers a special discount can be imposed on the annual pass. This discount can be made eligible to any individual who has been vaccinated through the vaccination drive in the year 2021. This offer will have two positive impacts. There will be a rise in the number of subscribers and the people riding the bikes would mostly be vaccinated, increasing the safety of all the bike users.