DivvyBikes Project

TeamDivvy - {Venkata Anirudh Thota | Swinidhi Manchiganti | Prasad Chandrakant Patil | Meghana Pamu}

4/30/2020

PROJECT : ANALYSIS OF DIVVY BIKE SHARE DATA AND INFLUENCE OF EXTERNAL FACTORS

Introduction:

Divvy is a bike share company that has smart bikes setup across various bikes stands in the City of Chicago. User will need to install an app and book bikes that are present at the stands to use them for commuting purposes in the City. The main reason for choosing this data set is due to the fact that this data can be combined with other datasets such as weather, pollution, census etc. This will help in providing exposure to different type of variables and enable us to explore various efficient data manipulations and analysis techniques due to the richness in the variables available.

About Data:

Divvy Bikes has stored data of these bookings and made it public for people across the globe to analyze. The data ranges from 2012 till 2019 in various files that are segregated by quarters (i.e., Q1 2019, Q2 2019,Q3 2019,Q4 2019) and also some in months (M7 2017 etc.). Apart from these files, there is another file that Divvy has posted that contains data about locations of their different bike stations along with the capacity of the station.

External Data:

We have in our analysis also planned to include external factors such as temperature of the city and other climatic conditions that play role how the bookings are made.

STEPS 1 TO 11: Data Wrangling

Describe the process we did to clean and organize data set for analysis

STEP 12: Data Inspection

Here we are inspecting the data formatting and presence of NA's in the data set

STEP 13: Deep Dive On Data And Dataset Boundaries

Now we are trying to understand data, the extremities and statistical figures

STEP 14: Data Visualizations

We are looking at charts to identify patterns and run models on them in the next phase

STEP 15: Data Analysis And Modeling

Here we are analysis for the significance of the model and making predictions based out of it.

STEP 1: LOAD REQUIRED LIBRARIES

In the first step here we are loading the required libraries

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------
------------------------------------ tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
library(sparklyr)
```

```
##
## Attaching package: 'sparklyr'
```

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

```
library(corrr)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa
```

```
library(cluster)
library(modelr)
library(broom)
```

```
##
## Attaching package: 'broom'

## The following object is masked from 'package:modelr':
##
##     bootstrap

library(timeDate)
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

library(sf)

## Linking to GEOS 3.6.1, GDAL 2.2.3, PROJ 4.9.3

library(rnaturalearth)
library(rnaturalearthdata)
library(rgeos)

## Loading required package: sp

## rgeos version: 0.5-2, (SVN revision 621)
##  GEOS runtime version: 3.6.1-CAPI-1.10.1
##  Linking to sp version: 1.4-1
##  Polygon checking: TRUE

library(maps)

##
## Attaching package: 'maps'

## The following object is masked from 'package:cluster':
##
##     votes.repub

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

library(lwgeom)

## Linking to liblwgeom 3.0.0beta1 r16016, GEOS 3.6.1, PROJ 4.9.3

library(kernlab)
```

```
##
## Attaching package: 'kernlab'

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

## The following object is masked from 'package:ggplot2':
##
##     alpha

library(e1071)

##
## Attaching package: 'e1071'

## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness

library(ISLR)
library(RColorBrewer)
library(ggplot2)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:kernlab':
##
##     alpha

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

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

#Using already backed up data to save load and time on machine

load(file = "Bkp\\TidyDivvy_For_Analysis_Final.rda")
load(file = "Bkp\\Divvy_Trips_Complete.rda")
Divvy_Complete_count <- count(Divvy_Complete)
rm(Divvy_Complete)
```

STEP 2: READ BOOKING FILES FOR LAST FIVE YEARS AND MERGE THEM

Since Divvy is offering data in separate csv files for each quarter with the following columns [trip_id,start_time,end_time,bikeid,tripduration,from_station_id,from_station_name,to_station_id,to_station_name,usertype,gender,birthyear]. We have imported the data and standardized them in to one single data type for each column so that we do not run into issues when doing union of the tibbles. Finally merged all the data and assined it to one variable.

```
#Eval set to false to save load and time on machine

# Collect 2019 Data
q1_19 <- read_csv("Data\\Divvy_Trips_2019_Q1.csv")
q2_19 <- read_csv("Data\\Divvy_Trips_2019_Q2.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M")))
read_csv("Data\\Divvy_Trips_2019_Q2.csv")
q3_19 <- read_csv("Data\\Divvy_Trips_2019_Q3.csv")
q4_19 <- read_csv("Data\\Divvy_Trips_2019_Q4.csv")

Divvy2019 <- q1_19 %>% union(q2_19) %>% union(q3_19) %>% union(q4_19)

# Collect 2018 Data
q1_18 <- read_csv("Data\\Divvy_Trips_2018_Q1.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M")))
q2_18 <- read_csv("Data\\Divvy_Trips_2018_Q2.csv")#,col_types =
cols(start_time = col_datetime(format = "%m/%d/%Y
%H:%M"),end_time=col_datetime(format = "%m/%d/%Y %H:%M")))
q3_18 <- read_csv("Data\\Divvy_Trips_2018_Q3.csv")
q4_18 <- read_csv("Data\\Divvy_Trips_2018_Q4.csv")

Divvy2018 <- q1_18 %>% union(q2_18) %>% union(q3_18) %>% union(q4_18)

# Collect 2017 Data
q1_17 <- read_csv("Data\\Divvy_Trips_2017_Q1.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M:%S"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M:%S")))
q2_17 <- read_csv("Data\\Divvy_Trips_2017_Q2.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M:%S"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M:%S")))
q3_17 <- read_csv("Data\\Divvy_Trips_2017_Q3.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M:%S"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M:%S")))
q4_17 <- read_csv("Data\\Divvy_Trips_2017_Q4.csv",col_types = cols(start_time
= col_datetime(format = "%m/%d/%Y %H:%M"),end_time=col_datetime(format =
"%m/%d/%Y %H:%M")))
```

```
Divvy2017 <- q1_17 %>% union(q2_17) %>% union(q3_17) %>% union(q4_17)

# Collect 2016 Data
q1_16 <- read_csv("Data\\Divvy_Trips_2016_Q1.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
q1_16 <- q1_16  %>% rename(start_time=starttime,end_time=stoptime)
m4_16 <- read_csv("Data\\Divvy_Trips_2016_04.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
m4_16 <- m4_16  %>% rename(start_time=starttime,end_time=stoptime)
m5_16 <- read_csv("Data\\Divvy_Trips_2016_05.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
m5_16 <- m5_16  %>% rename(start_time=starttime,end_time=stoptime)
m6_16 <- read_csv("Data\\Divvy_Trips_2016_06.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
m6_16 <- m6_16  %>% rename(start_time=starttime,end_time=stoptime)
q3_16 <- read_csv("Data\\Divvy_Trips_2016_Q3.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M:%S"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M:%S")))
q3_16 <- q3_16  %>% rename(start_time=starttime,end_time=stoptime)
q4_16 <- read_csv("Data\\Divvy_Trips_2016_Q4.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M:%S"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M:%S")))
q4_16 <- q4_16  %>% rename(start_time=starttime,end_time=stoptime)

Divvy2016 <- q1_16 %>% union(m4_16) %>% union(m5_16) %>% union(m6_16) %>%
union(q3_16) %>% union(q4_16)

# Collect 2015 Data
q1_15 <- read_csv("Data\\Divvy_Trips_2015-Q1.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
q1_15 <- q1_15  %>% rename(start_time=starttime,end_time=stoptime)
q2_15 <- read_csv("Data\\Divvy_Trips_2015-Q2.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
q2_15 <- q2_15  %>% rename(start_time=starttime,end_time=stoptime)
m7_15 <- read_csv("Data\\Divvy_Trips_2015_07.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
m7_15 <- m7_15  %>% rename(start_time=starttime,end_time=stoptime)
m8_15 <- read_csv("Data\\Divvy_Trips_2015_08.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
m8_15 <- m8_15  %>% rename(start_time=starttime,end_time=stoptime)
m9_15 <- read_csv("Data\\Divvy_Trips_2015_09.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
```

```
"%m/%d/%Y %H:%M")))
m9_15 <- m9_15  %>% rename(start_time=starttime,end_time=stoptime)
q4_15 <- read_csv("Data\\Divvy_Trips_2015_Q4.csv",col_types = cols(starttime
= col_datetime(format = "%m/%d/%Y %H:%M"),stoptime=col_datetime(format =
"%m/%d/%Y %H:%M")))
q4_15 <- q4_15  %>% rename(start_time=starttime,end_time=stoptime)

Divvy2015 <- q1_15 %>% union(q2_15) %>% union(m7_15) %>% union(m8_15) %>%
union(m9_15) %>% union(q4_15)

Divvy_Complete <- Divvy2015 %>% union(Divvy2016) %>% union(Divvy2017) %>%
union(Divvy2018) %>% union(Divvy2019)

Divvy_Complete %>% print(width=Inf)
```

STEP 3: BACKUP BOOKING DATA

We are taking a sample of 1 million records and working with the same. Also as a precautionary measure we are creating a backup for each of the data points.

```
Divvy_bkp_list =
list(Divvy2015,Divvy2016,Divvy2017,Divvy2018,Divvy2019,Divvy_Complete)
#save(Divvy_bkp_list,file="Bkp\\Divvy_Trips.rda")

#Divvy <- Divvy_Complete %>%  sample_n(1000000)
#save(Divvy,file="Bkp\\Divvy.rda")

#DO NOT CHANGE CODE IN THIS CHUNK
# SINCE WE HAVE NOT USED SEED DURING THE START OF THE PROJECT, LOADING THE
FILE FROM BACKUP TO MAINTAIN CONSISTENCY
load(file = "Bkp\\Divvy.rda")
```

STEP 4: CLEAR UNUSED OBJECTS

Removing unwanted variables to free memory for faster processing

```
Divvy_Complete_count <- count(Divvy_Complete)
rm(q1_19,q2_19,q3_19,q4_19,q1_18,q2_18,q3_18,q4_18,Divvy2018,Divvy2019,Divvy2
017,q1_17,q2_17,q3_17,q4_17,Divvy2016,q1_16,m4_16,m5_16,m6_16,q3_16,q4_16,Div
vy2015,q1_15,q2_15,m7_15,m8_15,m9_15,q4_15,Divvy_bkp_list,Divvy_Complete)
```

STEP 5: TIDYING/INSPECTING BOOKINGS DATA SET

In this step we are tidying the divvy bookings data:

1. We are splitting the date time column into Day,Month,Year,Time columns respectively.
2. Instead of having BirthDay column we are calculating the Age of the person

Also we are verifying the output.

```
TidyDivvy <- Divvy %>%
  mutate(Ride_Start_Time=
as.integer(format(start_time,"%H%M")),Ride_End_Time=as.integer(format(end_tim
e,"%H%M"))) %>%
  mutate(Ride_Start_Date=as.Date(start_time),Ride_End_Date=as.Date(end_time))
%>%
  mutate(Age = as.numeric(format(Sys.Date(),"%Y"))-birthyear) %>%
  separate(Ride_Start_Date,into =
c("Ride_Start_Year","Ride_Start_Month","Ride_Start_Day"),sep = "-") %>%
  separate(Ride_End_Date,into =
c("Ride_End_Year","Ride_End_Month","Ride_End_Day"),sep = "-") %>%

mutate(Ride_Start_Year=as.integer(Ride_Start_Year),Ride_Start_Month=as.intege
r(Ride_Start_Month),Ride_Start_Day=as.integer(Ride_Start_Day),Ride_End_Year=a
s.integer(Ride_End_Year),Ride_End_Month=as.integer(Ride_End_Month),Ride_End_D
ay=as.integer(Ride_End_Day),birthyear =as.factor(birthyear))

TidyDivvy %>% print(width=Inf)
```

STEP 6: READING STATIONS DATASET AND MERGING WITH BOOKINGS

In this step we are adding station details data to the original bookings dataset to get the GPS location of the station

```
Stations <- read_csv("Data\\Divvy_Bicycle_Stations.csv")

TidyDivvy_GPS_1 <-  TidyDivvy %>% left_join(Stations,by =
c("from_station_id"="ID")) %>% select(-`Station Name`,-`Total Docks`,-`Docks
in Service`,-Status,-Latitude,-Longitude) %>% rename(From_Loc_GPS=Location)

TidyDivvy_GPS <-  TidyDivvy_GPS_1 %>% left_join(Stations,by =
c("to_station_id"="ID")) %>% select(-`Station Name`,-`Total Docks`,-`Docks in
Service`,-Status,-Latitude,-Longitude) %>% rename(To_Loc_GPS=Location)

Stations %>% print(width=Inf)
```

STEP 7: INSPECT DIVVY BIKES COMBINED DATA SET

Inspect GPS combined dataset cleaning unwanted objects

```
TidyDivvy_GPS %>% print(width=Inf)
```

STEP 8: READ TEMPARTURE DATA FOR LAST TWO YEARS

Reading temperature data

```
temp_18 <- read_csv("Data\\weather_2018_chicago.csv",col_types = cols(Date =
col_date(format = "%m/%d/%Y")))
temp_19 <- read_csv("Data\\weather_2019_chicago.csv",col_types = cols(Date =
col_date(format = "%m/%d/%Y")))
temp_15_16_17 <- read_csv("Data\\weather151617chicago.csv",col_types =
cols(Date = col_date(format = "%m/%d/%Y")))
```

STEP 9: TIDY TEMPARTURE DATA AND MERGE IT

Tidying temperature data and combining the five year data into one tibble variable. Note: There are few columns in temperature data such as (Precipitation and New Snow and Snow Depth), that have value as T. T according to NOAA is <0.001 inches, so we are converting that to 0 (assumption here is 0.001 inches of rain/snow is negligible amount). Link: https://www.weather.gov/climateservices/nowdatafaq

```
tidy_temp_18 <- temp_18 %>%
  mutate(Precip = if_else(Precipitation == "T",0,as.double(Precipitation)))
%>%
  mutate(New_snow = if_else(`New Snow` == "T",0,as.double(`New Snow`))) %>%
  rename(Snow_depth = `Snow Depth`) %>%
  select(-Precipitation,-`New Snow`)

tidy_temp_19 <- temp_19 %>%
  mutate(Precip = if_else(Precipitation == "T",0,as.double(Precipitation)))
%>%
  mutate(New_snow = if_else(`New Snow` == "T",0,as.double(`New Snow`))) %>%
  mutate(Snow_depth = if_else(`Snow Depth` == "T",0,as.double(`Snow Depth`)))
%>%
  select(-Precipitation,-`New Snow`,-`Snow Depth`)
```

```
tidy_temp_15_16_17 <- temp_15_16_17 %>%
  mutate(Precip = if_else(Precipitation == "T",0,as.double(Precipitation)))
%>%
  mutate(New_snow = if_else(`New Snow` == "T",0,as.double(`New Snow`))) %>%
  mutate(Snow_depth = if_else(`Snow Depth` == "T",0,as.double(`Snow Depth`)))
%>%
  select(-Precipitation,-`New Snow`,-`Snow Depth`)

temparature <- tidy_temp_19 %>% union(tidy_temp_18) %>%
union(tidy_temp_15_16_17)

tidy_temparature <-  temparature %>%
  separate(Date,into =c("Year","Month","Day"),sep = "-",convert = T) %>%
  rename(Temp_Max=`Temperature Max`,Temp_Min=`Temperature
Min`,Temp_Avg=`Temperature Avg`,Temp_Dep=`Temperature Dep`) %>%
  mutate(Year=as.integer(Year),Month=as.integer(Month),Day=as.integer(Day))
%>%
  select(-HDD,-CDD)

tidy_temparature %>% print(width=Inf)
```

STEP 10: MERGE DIVVY BIKES AND TEMPERATURE DATA SET

Combing the bikes resevation and temperature data using day as the combining factor.

```
TidyDivvy_GPS_Temp <-  TidyDivvy_GPS %>% inner_join(tidy_temparature,by =
c("Ride_Start_Year"="Year","Ride_Start_Month"="Month","Ride_Start_Day"="Day")
)

TidyDivvy_GPS_Temp %>% print(width=Inf)
```

STEP 11: BACKUP TIDY DATA CLEAN ENVIRONMENT

Backing up all tidy data and cleaning unused objects for improving performance

```
TidyDivvy_list =
list(TidyDivvy,TidyDivvy_GPS,TidyDivvy_GPS_Temp,tidy_temparature)
#save(TidyDivvy_list,file="Bkp\\TidyDivvy.rda")

rm(Divvy,TidyDivvy,TidyDivvy_GPS_1,temp_18,temp_19,temp_15_16_17,tidy_temp_18
,tidy_temp_19,tidy_temp_15_16_17,temparature,TidyDivvy_GPS)
```

STEP 12: INSPECT DATA

Inspect if all columns are of the right data type for analysis. We are checking if there are any records in the sample data set that are not of the correct data type and displaying the output

```
#Create a variable specific for the analysis purpose, modifying trip duration
from seconds to minutes
TidyDivvy_For_Analysis <- TidyDivvy_GPS_Temp %>%
mutate(Ride_Start_Year=as.factor(Ride_Start_Year),Ride_Start_Month=as.factor(
Ride_Start_Month),Ride_Start_Day=as.factor(Ride_Start_Day),Ride_End_Year=as.f
actor(Ride_End_Year),Ride_End_Month=as.factor(Ride_End_Month),Ride_End_Day=as
.factor(Ride_End_Day),tripduration=tripduration/60)

#Inspect if there are any formatting issues in the dataset
TidyDivvy_For_Analysis %>% filter(
!is.double(trip_id) |
!is.double(bikeid) |
!is.double(tripduration) |
!is.double(from_station_id) |
!is.double(to_station_id)|
!is.double(Age ) |
!is.double(Temp_Max ) |
!is.double(Temp_Min ) |
!is.double(Temp_Avg ) |
!is.double(Temp_Dep ) |
!is.double(Precip ) |
!is.double(New_snow ) |
!is.double(Snow_depth)  |
!is.character(from_station_name) |
!is.character(to_station_name) |
!is.character(usertype) |
!is.character(gender) |
!is.character(From_Loc_GPS) |
!is.character(To_Loc_GPS) |
!is.integer(Ride_Start_Time) |
!is.integer(Ride_End_Time) |
!is.factor(Ride_Start_Year) |
!is.factor(Ride_Start_Month) |
!is.factor(Ride_Start_Day) |
!is.factor(Ride_End_Year) |
!is.factor(Ride_End_Month) |
!is.factor(Ride_End_Day) |
!is.factor(birthyear)) %>% print(width=Inf)

#Verify if there are any NA's in any of the columns, and also see how many
```

```
are present to see their impact on analysis
TidyDivvy_For_Analysis %>%
  select(everything()) %>%
  summarise_all(funs(sum(is.na(.)))) %>% print(width=Inf)

#purposely commented to prevent changes in result due to resaving the file
#save(TidyDivvy_For_Analysis,file="Bkp\\TidyDivvy_For_Analysis.rda")
```

STEP 13: DEEP DIVE ON DATA AND DATASET BOUNDARIES

We are trying to understand the extremes of the data and dataset, by knowing highest, lowest, mean, median and count of a particular variables.

---

1 > The number of rows in the original dataset : 17969273

```
rm(Divvy_Complete_count)
```

---

2 > Subset number of rows used for the sake of Analysis : 999524

---

3 > All the columns present in the dataset for analysis (Total: 32 columns)

```
colnames(TidyDivvy_For_Analysis)
```

```
##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"
## [13] "Ride_Start_Time"   "Ride_End_Time"     "Ride_Start_Year"
## [16] "Ride_Start_Month"  "Ride_Start_Day"    "Ride_End_Year"
## [19] "Ride_End_Month"    "Ride_End_Day"      "Age"
## [22] "From_Loc_GPS"      "To_Loc_GPS"        "Temp_Max"
## [25] "Temp_Min"          "Temp_Avg"          "Temp_Dep"
## [28] "Precip"            "New_snow"          "Snow_depth"
```

---

4 > The years and months groups considered for analysis and respective count :

This is helpful in knowing across how many months and years is our data distributed and how is it distributed

---

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year,Ride_Start_Month) %>%
                           summarise(ride_count=n()) %>%
                         arrange(Ride_Start_Year,Ride_Start_Month) %>%
print(n=60)
```

```
## # A tibble: 60 x 3
## # Groups:   Ride_Start_Year [5]
##    Ride_Start_Year Ride_Start_Month ride_count
##    <fct>           <fct>                <int>
##  1 2015            1                     3279
##  2 2015            2                     1908
##  3 2015            3                     5985
##  4 2015            4                     9814
##  5 2015            5                    16610
##  6 2015            6                    23410
##  7 2015            7                    29765
##  8 2015            8                    27662
##  9 2015            9                    23801
## 10 2015            10                   17625
## 11 2015            11                   10810
## 12 2015            12                    6711
## 13 2016            1                     5148
## 14 2016            2                     6507
## 15 2016            3                    10179
## 16 2016            4                    12953
## 17 2016            5                    20015
## 18 2016            6                    26647
## 19 2016            7                    28835
## 20 2016            8                    26893
## 21 2016            9                    24366
## 22 2016            10                   19457
## 23 2016            11                   13425
## 24 2016            12                    5068
## 25 2017            1                     6383
## 26 2017            2                     9373
## 27 2017            3                     8552
## 28 2017            4                    14961
## 29 2017            5                    19158
## 30 2017            6                    28090
## 31 2017            7                    31395
## 32 2017            8                    30898
## 33 2017            9                    26658
## 34 2017            10                   19870
## 35 2017            11                   10517
## 36 2017            12                    7014
## 37 2018            1                     6140
## 38 2018            2                     5736
## 39 2018            3                     9643
## 40 2018            4                    11066
## 41 2018            5                    22371
```

```
## 42 2018            6                    25140
## 43 2018            7                    30580
## 44 2018            8                    29594
## 45 2018            9                    24181
## 46 2018            10                   19180
## 47 2018            11                    9354
## 48 2018            12                    7206
## 49 2019            1                     5704
## 50 2019            2                     5373
## 51 2019            3                     9116
## 52 2019            4                    14705
## 53 2019            5                    20749
## 54 2019            6                    23092
## 55 2019            7                    31055
## 56 2019            8                    32916
## 57 2019            9                    27562
## 58 2019            10                   20883
## 59 2019            11                    9796
## 60 2019            12                    8640
```

---

5 > The mean, maximum, minimum, median and standard deviation of the amount of time that a bike has been rented for all the years (In minutes) :

Being a core bike rental business, it is essential for us to understand how do the booking look like, what is the maximum, minumum, average time a bike is being rented. This will help us understand if users are really interested in our business. We see that on average our user books a ride between 17 to 22 minutes. A minimum time of 1 minutes that may suggest, user ended the booking in 1 minute as he might have had other last minute plans. The maximum time from 2015 to 2017 suggest it is almost close to 24 hours, but if we look at 2018 and 2019, the result is off the charts (over 138 days) that might suggest that user forgot to end a booking or a technical fault, unfortunately there is not sufficient data to conclude the reason for such outlier and influencer. The standard deviation for 2015,2016,2017 informs us that the deviation in booking is about 25 to 33 minutes above or below the mean, but if we looking at 2018 and 2019, those high values are being possibly caused by the influencer records.

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year) %>%
summarise(mean_rent_time=mean(tripduration),max_rent_time=max(tripduration),m
in_rent_time=min(tripduration),std_dev_rent_time=sd(tripduration),median_rent
_time=median(tripduration)) %>%
  print(width=Inf)
```

```
## # A tibble: 5 x 6
##   Ride_Start_Year mean_rent_time max_rent_time min_rent_time
std_dev_rent_time
##   <fct>                   <dbl>         <dbl>         <dbl>
<dbl>
## 1 2015                     17.1         1436.          1
```

```
33.5
## 2 2016                      16.6         1434.          1
32.1
## 3 2017                      15.9         1434.          1
26.3
## 4 2018                      18.1         1436.          1.02
32.5
## 5 2019                      18.9         1435.          1.02
35.0
##   median_rent_time
##              <dbl>
## 1             12.1
## 2             11.7
## 3             11.4
## 4             11.2
## 5             11.8
```

In continuation to the above result tyring our best to know if the spike in max_rent_time was due to user error or system glitch and weather or not to include these results in the analysis. Assuming 24 hours as the maximum average booking from the trend seen in 2015,2016,2017. Let us see how many records exceed 24 hours in 2018 and 2019 and compare them to the total count of records in 2018 and 2019. We see that it is just 0.0004% of records the exceed 24 hours. Now this low percent for sure suggest something wrong going on in the system. So for the purpose of this analysis we are exlcuding these records to obtain accurate records. Ofcourse the median,avg,max and standard deviation will change after removing the rows, all of the values will relative fall for 2018 and 2019 as compared to before. Considering these bookings to be a cause of system defect, we are removing these outlier to create better models.

```
#Total count of bookings in 2018 2019
TidyDivvy_For_Analysis %>% filter(Ride_Start_Year %in% c('2018','2019')) %>%
group_by(Ride_Start_Year) %>% summarise(total_count=n())
```

```
## # A tibble: 2 x 2
##   Ride_Start_Year total_count
##   <fct>                 <int>
## 1 2018                  200191
## 2 2019                  209591
```

```
#Total count of bookings in 2018 2019 exceed 24 hours trip duration
TidyDivvy_For_Analysis %>% filter(Ride_Start_Year %in% c('2018','2019')) %>%
group_by(Ride_Start_Year) %>%  filter(tripduration>1440) %>%
summarise(Bookings_greaterthan_24hrs=n())
```

```
## # A tibble: 1 x 2
##   Ride_Start_Year Bookings_greaterthan_24hrs
##   <fct>                                <int>
## 1 <NA>                                     0
```

```r
#Removing the outliers
TidyDivvy_For_Analysis <-  TidyDivvy_For_Analysis %>%
filter(tripduration<=1440)

#Rechecking the results now
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year) %>%
summarise(mean_rent_time=mean(tripduration),max_rent_time=max(tripduration),m
in_rent_time=min(tripduration),std_dev_rent_time=sd(tripduration),median_rent
_time=median(tripduration)) %>%
  print(width=Inf)
```

```
## # A tibble: 5 x 6
##   Ride_Start_Year mean_rent_time max_rent_time min_rent_time
std_dev_rent_time
##   <fct>                   <dbl>         <dbl>         <dbl>
<dbl>
## 1 2015                     17.1         1436.          1
33.5
## 2 2016                     16.6         1434.          1
32.1
## 3 2017                     15.9         1434.          1
26.3
## 4 2018                     18.1         1436.          1.02
32.5
## 5 2019                     18.9         1435.          1.02
35.0
##   median_rent_time
##              <dbl>
## 1             12.1
## 2             11.7
## 3             11.4
## 4             11.2
## 5             11.8
```

---

6 > Number of different bikes operated by Divvy Bikes over the past 5 years:

Now that we got an understanding of our booking durations, we want to see how many bikes do we have in service to see if we need to increase or reduce the fleet size

```r
TidyDivvy_For_Analysis %>% summarise(n_distinct(bikeid))
```

```
## # A tibble: 1 x 1
##   `n_distinct(bikeid)`
##                  <int>
## 1                 6481
```

---

7 > Average, maximum, minimum, median and standard deviation of age of people (In years) renting bikes:

Here we are trying to understand the age of our customer renting bikes. We see that mean is about 38 with deviation being 10, so maximum focus should be between 28 and 48 years group range. The minimum age is a 3 year old. The maximum age is 121, which seems outlandish.

```
TidyDivvy_For_Analysis %>%
  filter(!is.na(Age)) %>%

summarise(Mean_Age=mean(Age),Max_Age=max(Age),Min_Age=min(Age),Median_Age=med
ian(Age),Std_Dev_Age=sd(Age))

## # A tibble: 1 x 5
##   Mean_Age Max_Age Min_Age Median_Age Std_Dev_Age
##      <dbl>   <dbl>   <dbl>      <dbl>       <dbl>
## 1     38.1      99       4         35        10.8
```

From the above result, let us inspect the maximum age that seems to be out of order. Let us see, how many of our customers are above the age of 100. We have about 269 records, that could be because of customers incorrectly providing their details.Though it cannot be confirmed for sure. Coming to the analysis Since 269 is a much smaller number compared to 793625 records that possibily have the correct age, we can eliminate these records and still have rich amount data to perform analysis on. On again checking the results now, we see a change in median,mean,max and standard deviation in the new result.

```
#Count of user having age greater than 100 and grouped
TidyDivvy_For_Analysis %>%
  filter(!is.na(Age)&Age>100) %>%
  group_by(Age) %>% summarise(n())

## # A tibble: 0 x 2
## # ... with 2 variables: Age <dbl>, `n()` <int>

#Count of user having age less than equal to 100
TidyDivvy_For_Analysis %>%
  filter(!is.na(Age)&Age<=100) %>% summarise(n())

## # A tibble: 1 x 1
##    `n()`
##    <int>
## 1 793779

#Removing the outliers
TidyDivvy_For_Analysis <- TidyDivvy_For_Analysis %>%
filter(is.na(Age)|Age<100)

#Rechecking the results now
```

```
TidyDivvy_For_Analysis %>%
  filter(!is.na(Age)) %>%

summarise(Mean_Age=mean(Age),Max_Age=max(Age),Min_Age=min(Age),Median_Age=med
ian(Age),Std_Dev_Age=sd(Age))
```

```
## # A tibble: 1 x 5
##   Mean_Age Max_Age Min_Age Median_Age Std_Dev_Age
##      <dbl>   <dbl>   <dbl>      <dbl>       <dbl>
## 1     38.1      99       4         35        10.8
```

---

8 > Top and bottom 5 stations from where most and least bookings were made, along with their count across the 5 year span:

The key takeaway from this analysis is to know where to strategically place more or less bikes based on the usage. Ofcourse we are going to have more bikes on stands on the Top 5 list and relatively reduces or close bike stands that not being properly utilized. One of the peculiar thing that we found was, more bikes were being rented at the Navy Pier,Lake Shore and Willis Tower locations likely indicating renting of bikes by liesure and tourist crowd.

```
TidyDivvy_For_Analysis %>% count(from_station_name) %>%

rename(Station_Name=from_station_name,Booking_count=n) %>%
                          arrange(desc(Booking_count)) %>% head(5) %>%
                          mutate(Category="Top 5")  %>%
                          union (TidyDivvy_For_Analysis %>%
                                   count(from_station_name) %>%

rename(Station_Name=from_station_name,Booking_count=n) %>%
                                   arrange(Booking_count) %>% head(5) %>%
                                   mutate(Category="Bottom 5"))
```

```
## [[1]]
## [1] "Streeter Dr & Grand Ave"      "Lake Shore Dr & Monroe St"
## [3] "Canal St & Adams St"          "Clinton St & Washington Blvd"
## [5] "Clinton St & Madison St"
##
## [[2]]
## [1] 18087 13400 13153 13000 11749
##
## [[3]]
## [1] "Top 5" "Top 5" "Top 5" "Top 5" "Top 5"
##
## [[4]]
## [1] "California Ave & Roosevelt Rd" "Carpenter St & 63rd St"
## [3] "Damen Ave & 61st St"           "Damen Ave & Garfield Blvd"
## [5] "LBS - BBB La Magie"
```

```
##
## [[5]]
## [1] 1 1 1 1 1
##
## [[6]]
## [1] "Bottom 5" "Bottom 5" "Bottom 5" "Bottom 5" "Bottom 5"
```

---

9 > Top and bottom 5 destination stations to where most and least bookings were made, along with their count across the 5 year span:

The key takeaway from this analysis is to know where to strategically place more or less bikes based on the usage. Ofcourse we are going to have more bikes on stands on the Top 5 list and relatively reduces or close bike stands that not being properly utilized.

```
TidyDivvy_For_Analysis %>% count(to_station_name) %>%

rename(Station_Name=to_station_name,Booking_count=n) %>%
                          arrange(desc(Booking_count)) %>% head(5) %>%
                          mutate(Category="Top 5")  %>%
                          union (TidyDivvy_For_Analysis %>%
                                   count(to_station_name) %>%

rename(Station_Name=to_station_name,Booking_count=n) %>%
                                   arrange(Booking_count) %>% head(5) %>%
                                   mutate(Category="Bottom 5"))
```

```
## [[1]]
## [1] "Streeter Dr & Grand Ave"      "Canal St & Adams St"
## [3] "Lake Shore Dr & North Blvd"   "Clinton St & Washington Blvd"
## [5] "Theater on the Lake"
##
## [[2]]
## [1] 20295 12638 12581 12517 12015
##
## [[3]]
## [1] "Top 5" "Top 5" "Top 5" "Top 5" "Top 5"
##
## [[4]]
## [1] "Carpenter St & 63rd St"   "Damen Ave & 61st St"
## [3] "Elizabeth St & 59th St"   "LBS - BBB La Magie"
## [5] "TS ~ DIVVY PARTS TESTING"
##
## [[5]]
## [1] 1 1 1 1 1
##
## [[6]]
## [1] "Bottom 5" "Bottom 5" "Bottom 5" "Bottom 5" "Bottom 5"
```

---

10 > Lets figure out top 10 stations between which most people use Divvy bikes for

Now that we know which are the most and least utilized station, let us now figure out where our crowd is heading from and to where are they going most. In short between which two stations are people most travelling. We see a lot of traffic between Lake Shore Dr & Monroe St and Streeter Dr & Grand Ave. Also there is a lot of picks and drop within these streets. Other than that, in general, we see alot of activity on Lake Shore, Michigan and Streerter street. Definitely something interest going on these streets.

```
TidyDivvy_For_Analysis %>%
                  group_by(from_station_name,to_station_name) %>%
                  count() %>%
                  arrange(desc(n)) %>%
                  head(10)
```

```
## # A tibble: 10 x 3
## # Groups:   from_station_name, to_station_name [10]
##    from_station_name          to_station_name               n
##    <chr>                      <chr>                     <int>
##  1 Lake Shore Dr & Monroe St  Streeter Dr & Grand Ave     2293
##  2 Streeter Dr & Grand Ave    Streeter Dr & Grand Ave     2033
##  3 Lake Shore Dr & Monroe St  Lake Shore Dr & Monroe St   1755
##  4 Streeter Dr & Grand Ave    Theater on the Lake         1254
##  5 Michigan Ave & Oak St      Michigan Ave & Oak St       1172
##  6 Streeter Dr & Grand Ave    Lake Shore Dr & North Blvd  1140
##  7 Streeter Dr & Grand Ave    Lake Shore Dr & Monroe St   1108
##  8 Theater on the Lake        Streeter Dr & Grand Ave     1065
##  9 Lake Shore Dr & North Blvd Streeter Dr & Grand Ave     1009
## 10 Streeter Dr & Grand Ave    Millennium Park              892
```

---

11 > Different membership types and their meaning, and the booking counts for each user type across years:

Note:

1.Customers: Renters

2.Subscribers: Membership Holders

3.Dependent: Someone under 16 whose parent purchased them a membership

We are trying to figure out how bookings vary by different type of users across various years. We see a trend of subscribers being the major part of our customer base, followed by customer and then dependent.

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year,usertype) %>%
summarise(count=n())
```

```
## # A tibble: 12 x 3
## # Groups:   Ride_Start_Year [5]
##    Ride_Start_Year usertype    count
##    <fct>           <chr>       <int>
##  1 2015            Customer    51760
##  2 2015            Dependent       9
##  3 2015            Subscriber 125611
##  4 2016            Customer    47728
##  5 2016            Dependent       5
##  6 2016            Subscriber 151760
##  7 2017            Customer    46475
##  8 2017            Subscriber 166394
##  9 2018            Customer    37611
## 10 2018            Subscriber 162580
## 11 2019            Customer    47667
## 12 2019            Subscriber 161924
```

---

12 > Ratio of bookings made by subscribers (including dependents) to that of customers respectively:

```
Temp_Ratio <- TidyDivvy_For_Analysis %>%
  mutate(Subscriber_Flag=if_else(usertype == "Subscriber" | usertype ==
"Dependent","Subscriber","Customer")) %>%
  group_by(Subscriber_Flag) %>%
  summarise(Count = n()) %>%
  pivot_wider(names_from = Subscriber_Flag,values_from = Count) %>%
  summarise(Ratio=Subscriber/Customer)
```

We see from the data of the past five years that Subscribers account to 3.3 times bookings to that of Customer

```
rm(Temp_Ratio)
```

---

13 > Number of time a bike has been used repeatedly, to plan maintainance and service activites on the bikes

We want to observe which bikes are being used more so that a maintainance can be planned on them before they breakdown

```
TidyDivvy_For_Analysis %>%
                        count(bikeid) %>%
                        rename(Bike_Used_Count = n) %>%
                        arrange(desc(Bike_Used_Count))
```

```
## # A tibble: 6,481 x 2
##    bikeid Bike_Used_Count
##     <dbl>           <int>
##  1   3387             278
```

```
##  2   3481             273
##  3   3888             271
##  4   3216             270
##  5   2155             261
##  6   3171             261
##  7   4372             261
##  8    577             260
##  9   3729             259
## 10   2241             258
## # ... with 6,471 more rows
```

---

14 > Day with the most and least booking and corresponding average temparature for the day, per each year:

We see that the generally days with the most bookings is during july and august when the tempature is between 67F to 77F, on the contrast the least booking days were generally observed in winter mainly December,January and February when temparature are between -10 and 25. One interesting find is that, Christmas which is a public holiday where least bookings are not to be expected exist in the least booking list.

```
TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_End_Month,Ride_End_Day) %>%

summarise(Booking_Count=n(),Avg_Temp=mean(Temp_Avg)) %>%
                            arrange(desc(Booking_Count)) %>% head(5) %>%
                            mutate(Day_Type="Most Booking") %>%
union(TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_End_Month,Ride_End_Day) %>%

summarise(Booking_Count=n(),Avg_Temp=mean(Temp_Avg)) %>%
                            arrange((Booking_Count)) %>% head(5) %>%
                            mutate(Day_Type="Least Booking"))
```

```
## [[1]]
## [1] 2015 2019 2016 2019 2015
## Levels: 2015 2016 2017 2018 2019
##
## [[2]]
## [1] 7 8 7 8 6
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12
##
## [[3]]
## [1] 4  3  16 9  27
## 31 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ...
31
##
## [[4]]
## [1] 1359 1324 1314 1250 1219
```

```
##
## [[5]]
## [1] 69.97057 75.96073 67.00114 72.51080 65.99426
##
## [[6]]
## [1] "Most Booking" "Most Booking" "Most Booking" "Most Booking" "Most
Booking"
##
## [[7]]
## [1] 2015 2019 2016 2015 2017
## Levels: 2015 2016 2017 2018 2019
##
## [[8]]
## [1] 2  1  12 2  12
## Levels: 1 2 3 4 5 6 7 8 9 10 11 12
##
## [[9]]
## [1] 1  30 25 15 25
## 31 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ...
31
##
## [[10]]
## [1] 10 11 12 15 16
##
## [[11]]
## [1]  25.000000 -15.000000  35.500000   6.533333  11.500000
##
## [[12]]
## [1] "Least Booking" "Least Booking" "Least Booking" "Least Booking"
## [5] "Least Booking"
```

Resaving the Analysis files, as after our analysis we have determined and removed the outliers

```
#purposely commented to prevent changes in results due to file resave
#save(TidyDivvy_For_Analysis,file="Bkp\\TidyDivvy_For_Analysis_Final.rda")
```

STEP 14: DATA VISUALIZATION

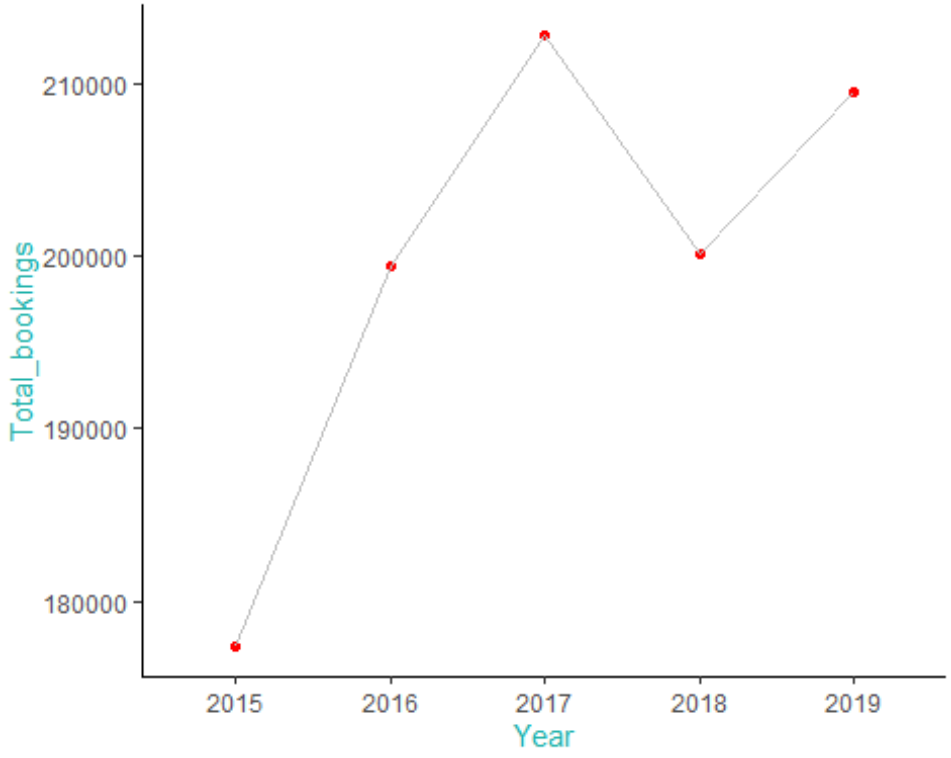
We are trying to visualize data and understand the business.

---

15 > The trend of bookings and business over the years

From the chart it can be observed that there has been a sharp raise in the total number of bookings from 2015 to 2017, until a dip that happened in 2018, post which the bookings increased in 2019.

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year) %>%
                           summarise(Total_bookings = n()) %>%
                           rename(Year=Ride_Start_Year) %>%
                           ggplot(aes(x=Year,y = Total_bookings)) +
                           geom_point(color="red")+
                           geom_line(color="gray",group=1)+
                           theme_classic()+
                           theme(text = element_text(color =
"lightseagreen"),strip.text=element_text( colour="navy"))
```



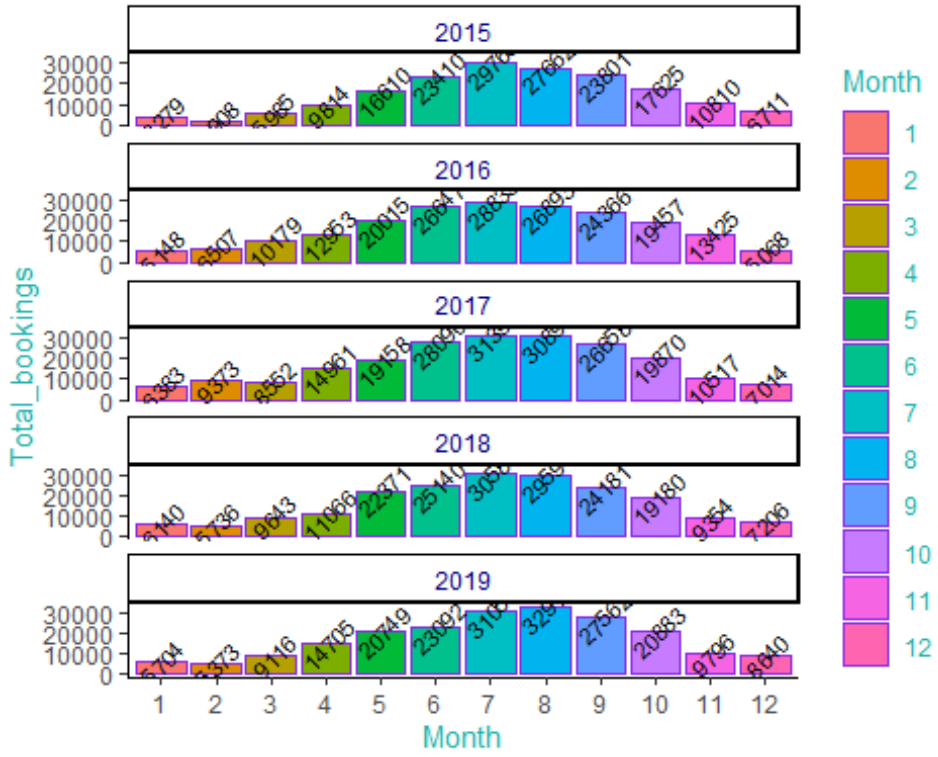
---

16 > Trend of bookings across various months for different years in seperate graphs

This graph was to illustrate which months of an year, since the past five years was our business busy. Divvy bikes being a predictable one it may seem obvious, but for data like sales data from Amazon, it is hard to guess a pattern. This graph was to help us practice and learn how to see patterns across months and years of any given data

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year,Ride_Start_Month) %>%
                           summarise(Total_bookings = n()) %>%
```

```
                            rename(Month=Ride_Start_Month) %>%
                            ggplot() +
                            geom_bar(mapping = aes(x=Month,y =
Total_bookings,fill=Month),stat = "identity",color="blueviolet")+
                            # geom_text(mapping = aes(x=Month,y =
Total_bookings,label=Total_bookings)) +
                            geom_text(mapping = aes(x=Month,y =
Total_bookings,label=Total_bookings),angle=45,size=3)+
                            theme_classic()+
                            theme(text = element_text(color =
"lightseagreen"),strip.text=element_text( colour="navy"))+
                            facet_wrap(~ Ride_Start_Year,nrow=5)
```



---

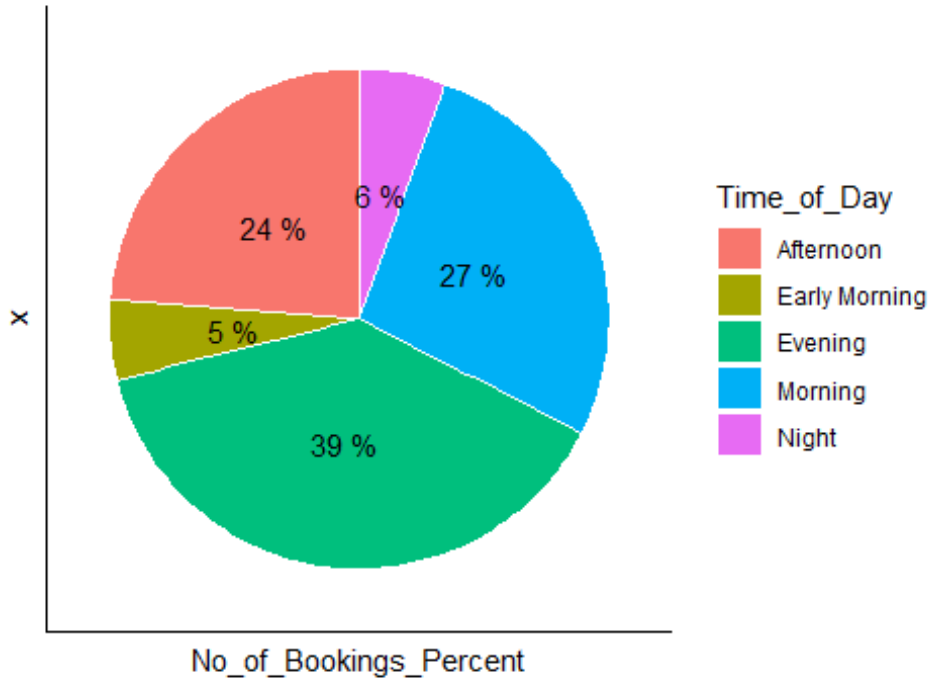
17 > How does booking vary over different times of the day (Early Morning: 12am to 6am , Morning: 6am to 12pm, Afternoon: 12pm to 4pm, Evening: 4pm to 9pm, Night: 9pm to 12am). Which time of the day do we see more bookings overall

Note: We observe the most bookings during the evenings. It is a bit of a surprise to see evening, we were expecting morning as bikes can help commute to work. It appears more people are interested in riding bike for leisure time rather than work.

Per review, we have also inlcuded the new bar graph that seggrates monthwise count.

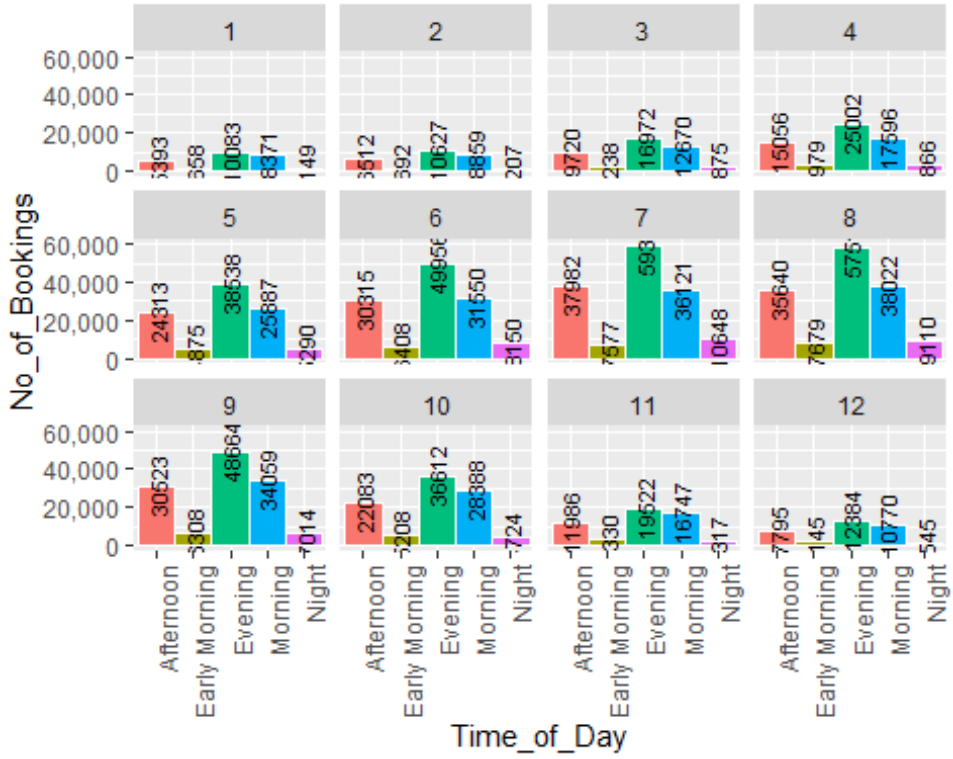
```
TidyDivvy_For_Analysis %>%
mutate(Hour_of_Day=as.numeric(format(start_time,"%H"))) %>%
```

```
                            mutate(Time_of_Day=if_else(Hour_of_Day<=6,"Early
Morning",if_else(Hour_of_Day>6 &
Hour_of_Day<12,"Morning",if_else(Hour_of_Day>=12&Hour_of_Day<16,"Afternoon",i
f_else(Hour_of_Day>=16&Hour_of_Day<21,"Evening",if_else(Hour_of_Day>=21,"Nigh
t","Unknown")))))) %>%
                           group_by(Time_of_Day) %>%
                           summarise(No_of_Bookings = n()) %>%
                           mutate(No_of_Bookings_Percent =
No_of_Bookings/sum(No_of_Bookings)*100 ) %>%

ggplot(aes(x="",y=No_of_Bookings_Percent,fill=Time_of_Day)) +
                           geom_bar(width = 1,stat = "identity",color="white")
+
                           coord_polar("y",start = 0) +
                           geom_text(aes(label =
paste(round(No_of_Bookings_Percent), "%")),
                                     position = position_stack(vjust =
0.5))+
                           theme_classic()+
                           theme(axis.text = element_blank(),axis.ticks =
element_blank())
```




```
TidyDivvy_For_Analysis %>%
mutate(Hour_of_Day=as.numeric(format(start_time,"%H"))) %>%
                            mutate(Time_of_Day=if_else(Hour_of_Day<=6,"Early
Morning",if_else(Hour_of_Day>6 &
Hour_of_Day<12,"Morning",if_else(Hour_of_Day>=12&Hour_of_Day<16,"Afternoon",i
```

```
f_else(Hour_of_Day>=16&Hour_of_Day<21,"Evening",if_else(Hour_of_Day>=21,"Nigh
t","Unknown")))))) %>%
                          group_by(Ride_Start_Month,Time_of_Day) %>%
                          summarise(No_of_Bookings = n()) %>%

ggplot(aes(y=No_of_Bookings,x=Time_of_Day,fill=Time_of_Day)) +
                          geom_bar(width = 1,stat =
"identity",color="white",show.legend = F) +
                          geom_text(aes(label =
No_of_Bookings),angle=90,size=3)+
                          scale_y_continuous(labels = scales::comma) +
                          facet_wrap(~Ride_Start_Month)+
                          theme(axis.text.x = element_text(angle = 90, hjust
= 1))
```



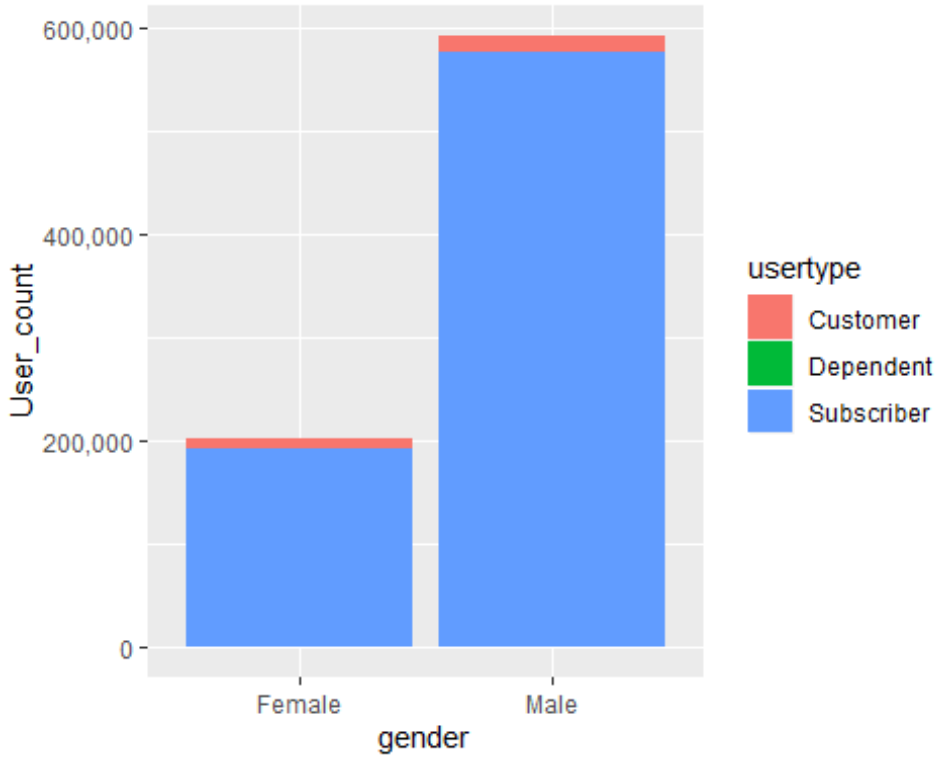
---

18 > To understand how customers are distributed across different genders, let us visualize the same using a graph.

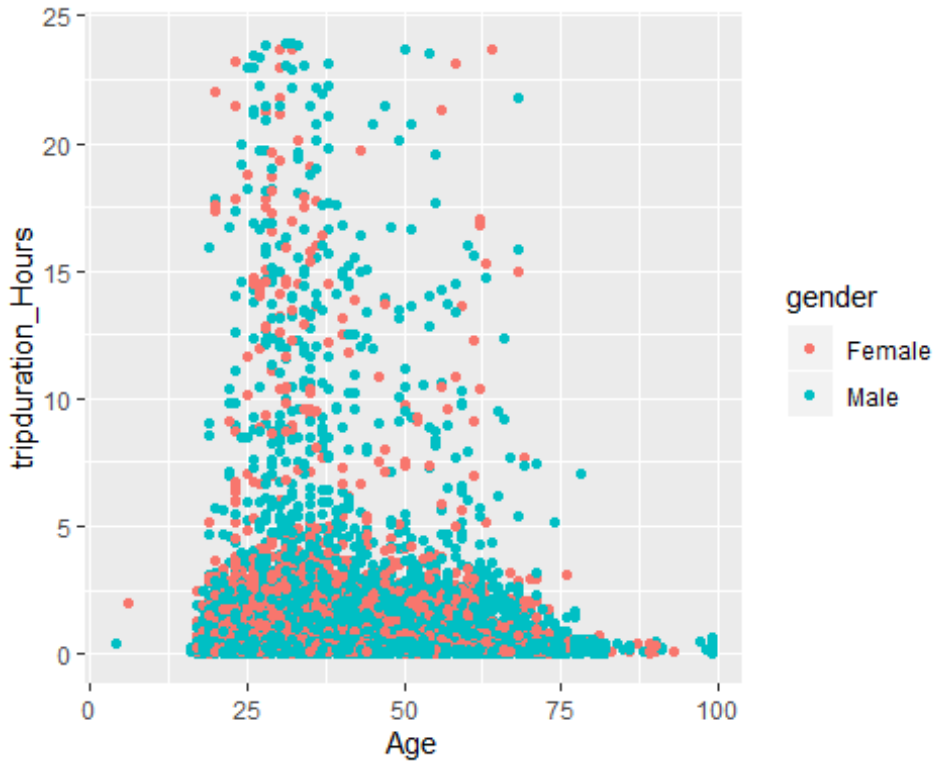
We see that overall we have more number of subscribers. The number of male user are more than twice as compared to female users.

Per review, we also want to see a trend on how booking time varies. There is no suprise in here though, trip duration for male and female at various age groups remain equally distributed

```
TidyDivvy_For_Analysis %>% filter(!is.na(gender)) %>%
                           group_by(usertype,gender) %>%
                           summarise(User_count=n()) %>%
                           ggplot(aes(x=gender,y=User_count,fill=usertype))
+
                           geom_bar(stat = "Identity")+
                           scale_y_continuous(labels = scales::comma)
```




```
TidyDivvy_For_Analysis %>% filter( !is.na(Age) & !is.na(gender)) %>%
mutate(tripduration_Hours=tripduration/60) %>%
select(gender,Age,tripduration_Hours) %>% ggplot() +
                          geom_point(mapping =
aes(x=Age,y=tripduration_Hours,color=gender))
```

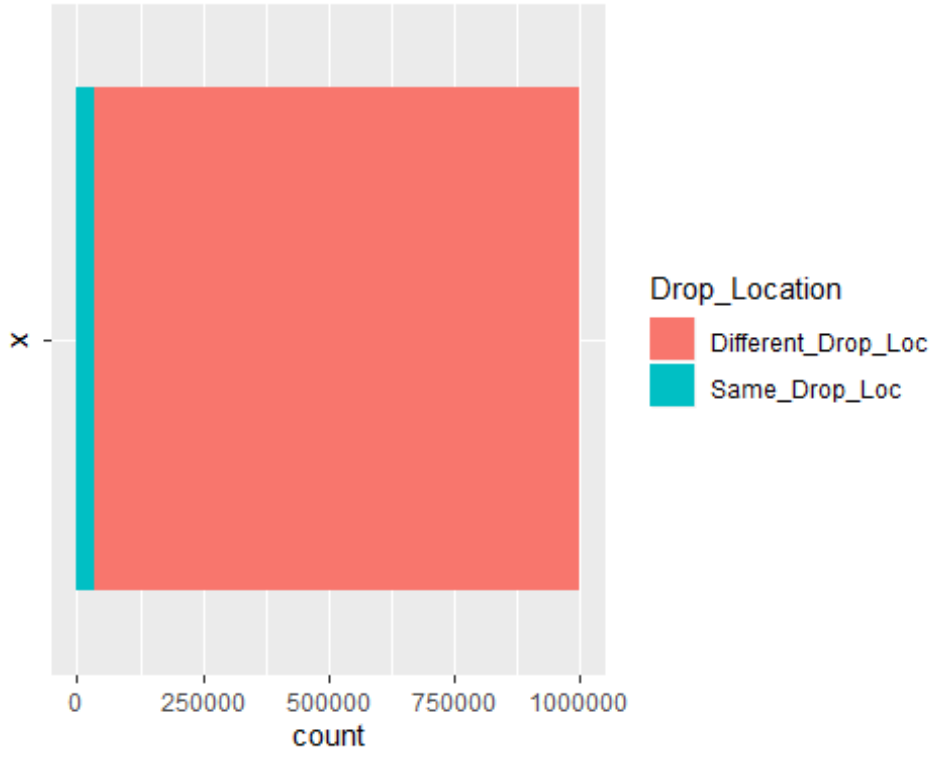


---

19 > In a horizontal bar graph let us see if most people return bikes to the same location that they booked the ride from or they drop the bike at another station.

Here we see that round trips are very rare, but that is not much of a surprise as it suggest that more people use bikes to commute from Point A to Point B in one trip, once their work at Point B is completed they make another booking to return to Point A. As that will save money instead of keeping the booking active all the way through the round trip. It also highlights one other thing that our bikes are reliably present for users to book later, User would generally not do this if they feel there won't be a bike with them on their return journey.

```
TidyDivvy_For_Analysis %>%

transmute(Drop_Location=if_else(from_station_id==to_station_id,"Same_Drop_Loc
","Different_Drop_Loc")) %>%
  count(Drop_Location) %>%
  rename(count=n) %>%
  ggplot()+
  geom_bar(mapping = aes(x="",fill=Drop_Location,y=count),stat = "identity")+
  coord_flip()
```



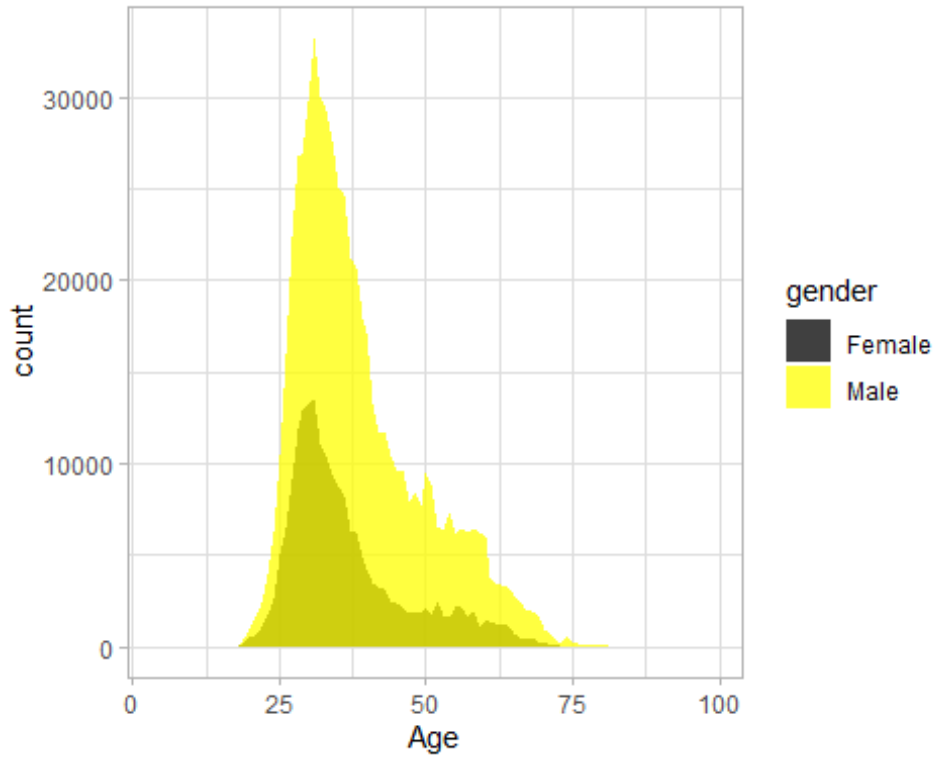
---

20 > Through a graph let us see how the bookings vary over different age groups and genders

We see that a majorly of trips between the age of 26 and 70 are made by males, almost twice or greater. While in the other ends of the spectrum it is more or less the same.

As per review, we have also included the scatter plot of the same trend

```
TidyDivvy_For_Analysis %>%
                    filter(!is.na(Age) & !is.na(gender))  %>%
                    ggplot(aes(x= Age,fill=gender)) +
                    geom_area(position = "identity",stat = "count")+
                    scale_fill_manual(values =
alpha(c("black","yellow"),0.75))+
                    theme_light()
```



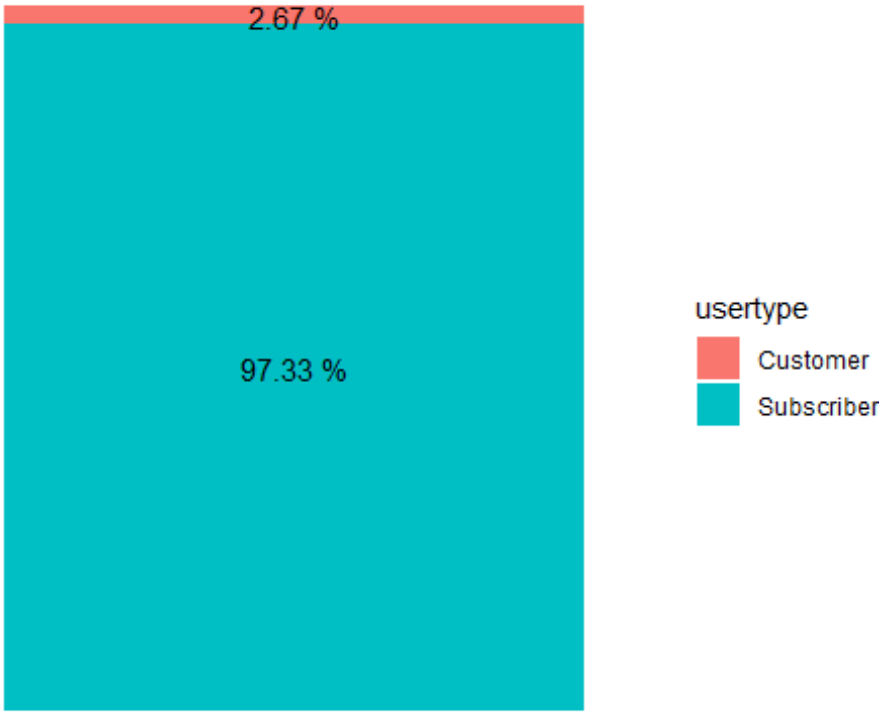

```
TidyDivvy_For_Analysis %>%
                       filter(!is.na(Age) & !is.na(gender))  %>%
                       group_by(gender,Age) %>%
                       summarise(count=n()) %>%
                       ggplot(aes(x= Age,y=count,color=gender)) +
                       geom_point()
```

---

21 > Since we know males use Divvy bikes more compared to Females, let us see what percent of males are subscribers, dependents and customers. This is to understand who is our biggest customer. We are ignoring Dependents as we know they are very minute.

From this we come to know that males subscribers are the biggest customer type among the rest in our business

```
TidyDivvy_For_Analysis %>% filter(!is.na(gender) & gender=="Male" &
usertype!='Dependent') %>%
                           group_by(usertype) %>%
                           summarise(User_count=n()) %>%

mutate(User_Percentage=User_count/sum(User_count)*100) %>%
                           ggplot(aes(x="",fill=usertype,y=User_Percentage))
+
                           geom_bar(stat = "identity")+
                           geom_text(mapping = aes(label=
paste(round(User_Percentage,2),"%")),position = position_stack(vjust = 0.5))
+
                           theme_void()
```



---

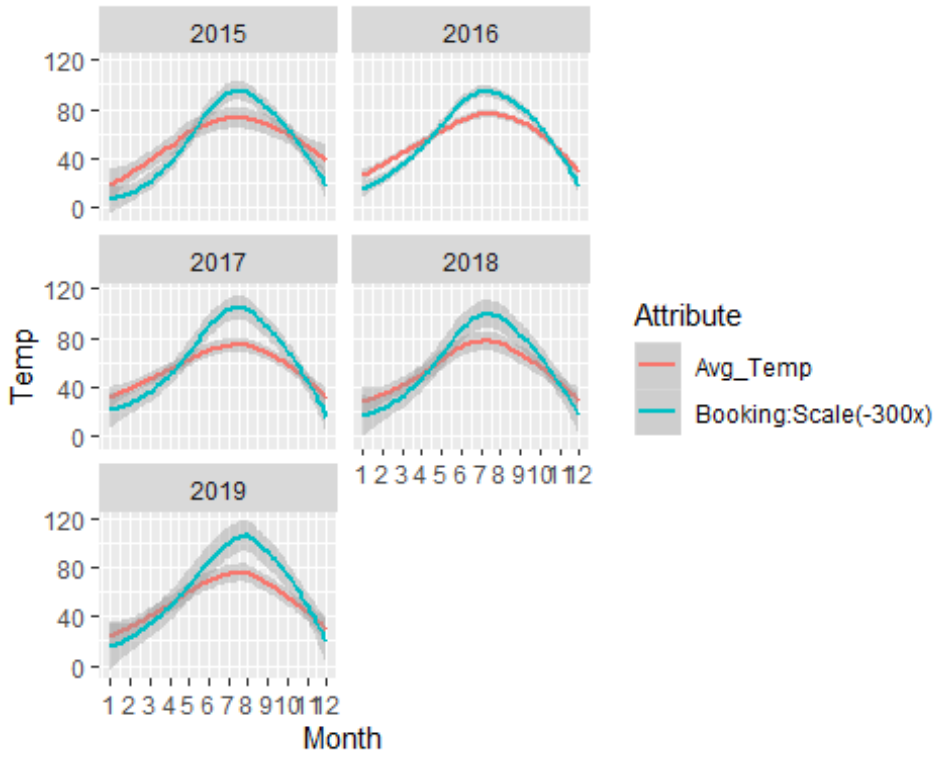
22 > Through a graph lets us understand if the monthly average temperature, precipitation and snow plays any role in affecting the number of bookings for that month, across different years.

Note: The temperature is in Fahrenheit The booking count has been scaled to 300 times less for aesthetics purpose.

Note: We can observe that temperature is playing a role in bookings. From the graphs we can see that 75F is when we see a peak in bookings count

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year,Ride_Start_Month) %>%

summarise(Avg_Temp=mean(Temp_Avg),Bookings_Count=n()) %>%

mutate(Ride_Start_Month=as.integer(Ride_Start_Month),Bookings_Count=Bookings_
Count/300) %>% #To bring it to scale
                          rename(Year=Ride_Start_Year,
Month=Ride_Start_Month,`Booking:Scale(-300x)`=Bookings_Count) %>%
                          pivot_longer(c(Avg_Temp,`Booking:Scale(-
300x)`),names_to="Attribute",values_to ="Temp") %>%
                          ggplot() +
                          geom_smooth(aes(x=Month,y=Temp,color=Attribute))+
                          scale_x_continuous(breaks = seq(1,12)) +
                          facet_wrap(~Year,nrow = 3)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

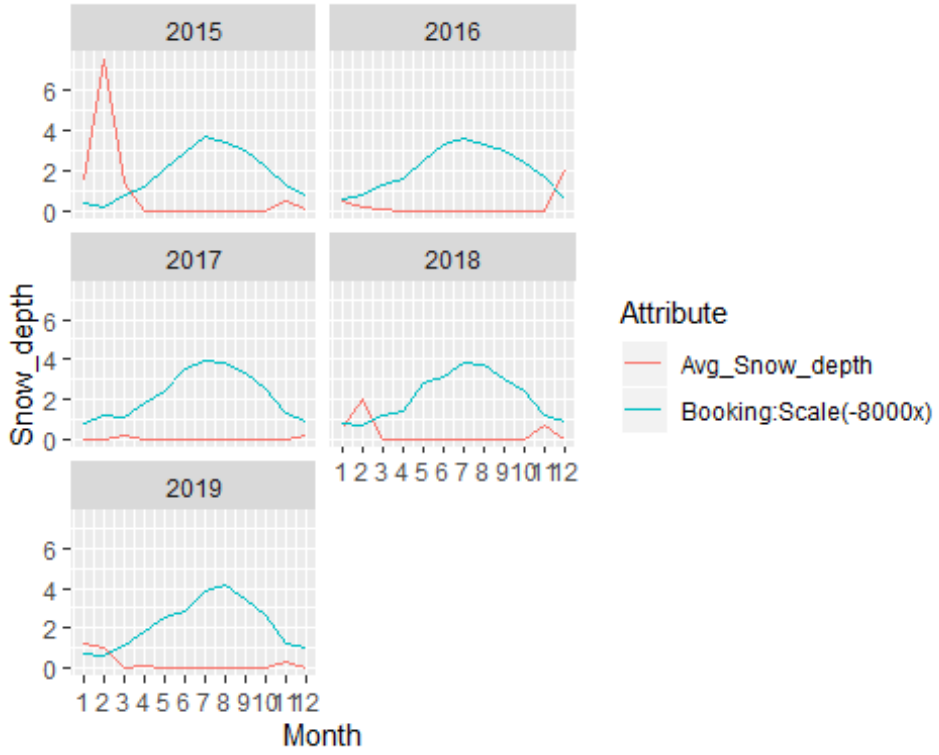


---

23 > Similarly lets us understand if there is any relation between snow depth affecting the number of bookings for particular month, across different years.

Note: The snow depth is in inches. The booking count has been scaled to 8000 times less for asthetics purpose. Snow depth has an opposite affect on bookings. As we see more snow depth, there is fall in bookings made.

```
TidyDivvy_For_Analysis %>% group_by(Ride_Start_Year,Ride_Start_Month) %>%

summarise(Avg_Snow_depth=mean(Snow_depth),Bookings_Count=n()) %>%

mutate(Ride_Start_Month=as.integer(Ride_Start_Month),Bookings_Count=Bookings_
Count/8000) %>% #To bring it to scale
                            rename(Year=Ride_Start_Year,
Month=Ride_Start_Month,`Booking:Scale(-8000x)`=Bookings_Count) %>%
                            pivot_longer(c(Avg_Snow_depth,`Booking:Scale(-
8000x)`),names_to="Attribute",values_to ="Snow_depth") %>%
                            ggplot() +

geom_line(aes(x=Month,y=Snow_depth,color=Attribute))+
                            scale_x_continuous(breaks = seq(1,12)) +
                            facet_wrap(~Year,nrow = 3)
```



STEP 15: DATE ANALYSIS AND MODELING

We are trying to visualize data and understand the business.

LOAD DATA

Loading previously saved data

```
load(file = "Bkp\\TidyDivvy_For_Analysis_Final.rda")
```

DISCOVERY

After analyzing data for individual bookings, there is one common finding that we came across:

Individual bookings are certainly influenced by factors such as weather conditions, month of year, time of the day, Age, Gender and other factors. But the amount of variance explained by these factors is small. Pondering over it, the picture becomes much clearer, if we as individuals want to book a bike and go for a ride, things like temperature, gender, age

are not really influencing rather much, though may be to a certain extent. What is more influential are personal characteristics and the environment an individual is surrounded by, like:

1. I am feeling energetic today?
2. How far is the nearest divvy bikes stand to my house?
3. Are divvy bikes available near my house?
4. Is there any event in downtown that I want to attend, but have parking problem with my car?
5. Do I have an important event to attend and my car broke?

so on..

Added the fact that most people living in Chicago have vehicles and public transits, and considering bike riding as a leisure sport.

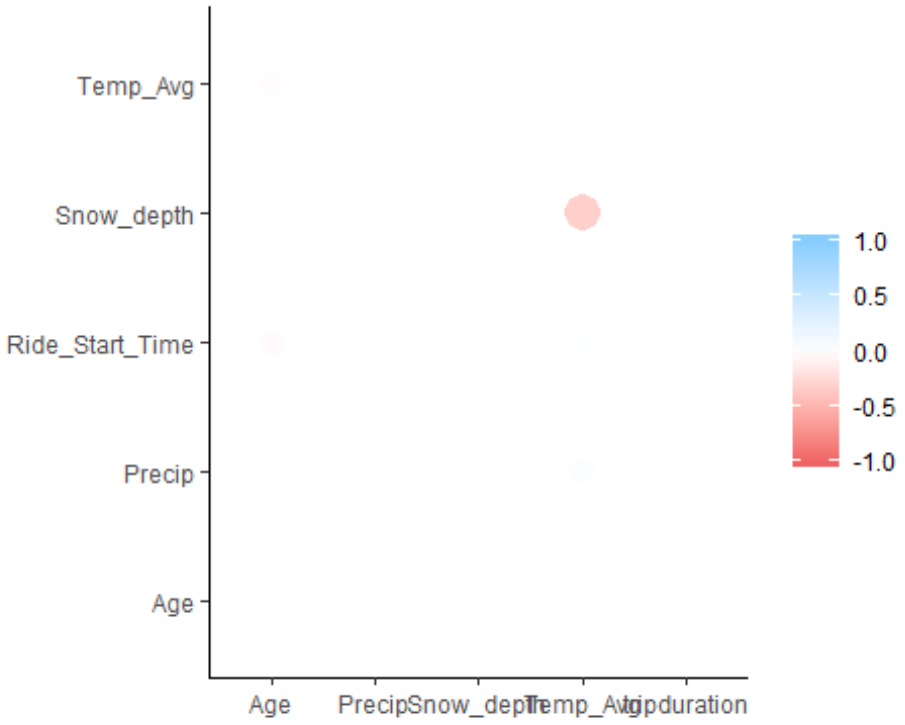
This is a complete contrast of what was originally thought and hypothesized. When we first picked this dataset, we were in the notion that factors like weather, age of a person and other traits play a very significant role in how bookings are made. This becomes more apparent when we take a look at the below results that show that though the factors play an important role, there are other variables and data points missing in our dataset which are more significantly affecting trip duration and booking rate.

In the below analysis, we have run various correlation checks and linear as well as logistic models to see which variables are showing close relations.

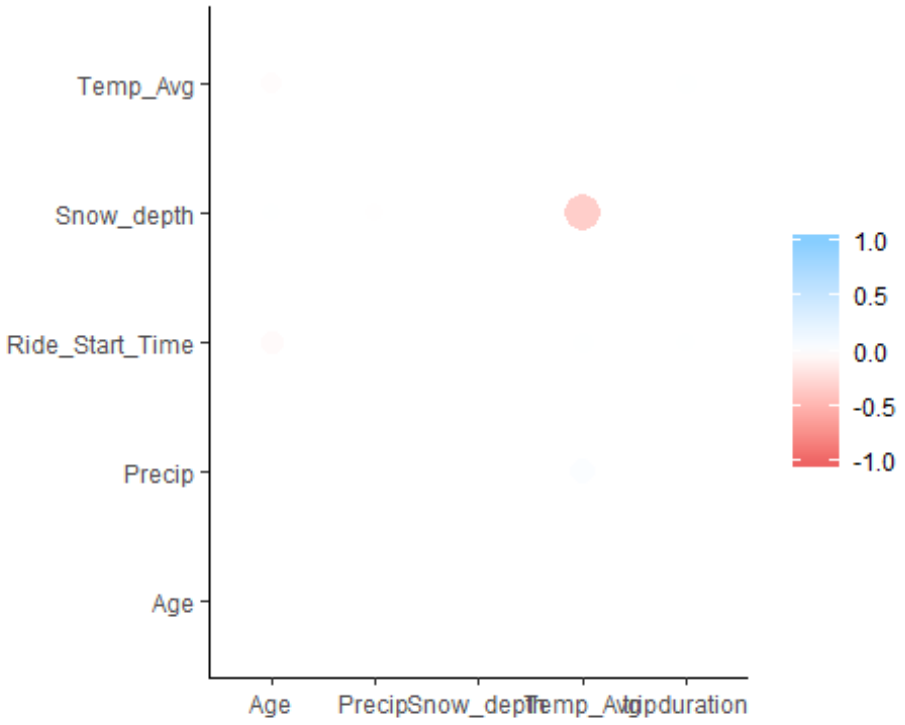
Different variables effect on number of bookings and its correlation

---

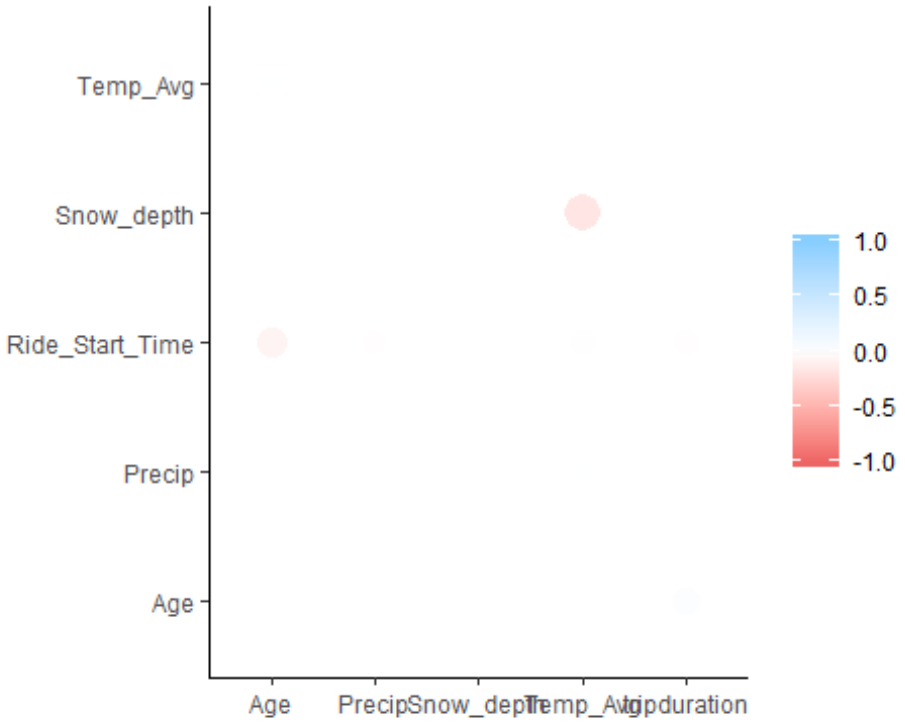
```
#Correlation check on all possible numeric variables
TidyDivvy_For_Analysis %>% filter(!is.na(Age)) %>%
select(tripduration,Age,Temp_Avg,Precip,Snow_depth,Ride_Start_Time) %>%
correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave()
%>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'

## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```



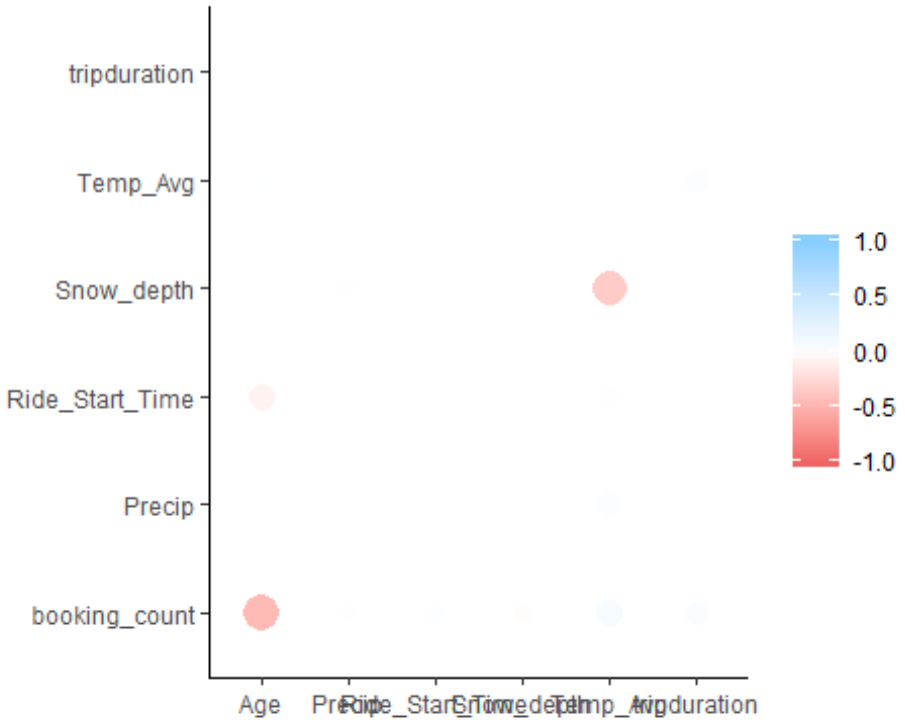

```
#Correlation check on all possible numeric variables when user is Subscriber
TidyDivvy_For_Analysis %>% filter(!is.na(Age),usertype=="Subscriber") %>%
select(tripduration,Age,Temp_Avg,Precip,Snow_depth,Ride_Start_Time) %>%
correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave()
%>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```



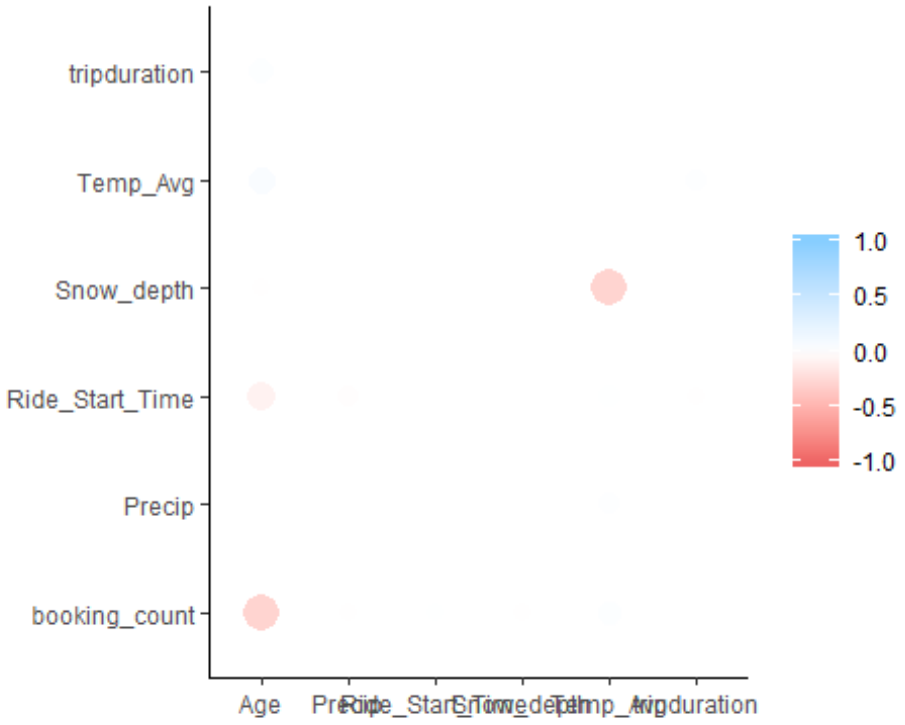

```
#Correlation check on all possible numeric variables when user is Customer
TidyDivvy_For_Analysis %>% filter(!is.na(Age),usertype=="Customer") %>%
select(tripduration,Age,Temp_Avg,Precip,Snow_depth,Ride_Start_Time) %>%
correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave()
%>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```



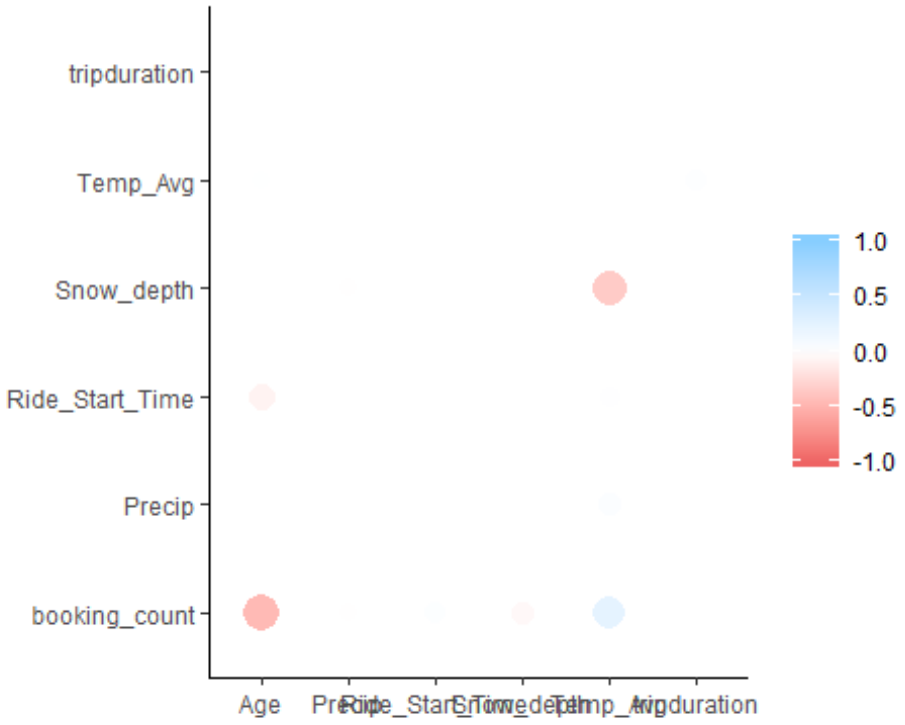

```
#Correlation check to include number of bookings per day as well as all other numeric variables
TidyDivvy_For_Analysis %>% group_by(as.Date(start_time),Age) %>% summarise(tripduration=mean(tripduration),Temp_Avg=mean(Temp_Avg),Precip=mean(Precip),Snow_depth=mean(Snow_depth),Ride_Start_Time=mean(Ride_Start_Time),booking_count=n()) %>% ungroup() %>% select(-`as.Date(start_time)`) %>% correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave() %>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```




```
#Correlation check to include number of bookings per day as well as all other
numeric variables for Customer type users
TidyDivvy_For_Analysis %>% filter(usertype=="Customer") %>%
group_by(as.Date(start_time),Age) %>%
summarise(tripduration=mean(tripduration),Temp_Avg=mean(Temp_Avg),Precip=mean
(Precip),Snow_depth=mean(Snow_depth),Ride_Start_Time=mean(Ride_Start_Time),bo
oking_count=n()) %>% ungroup() %>% select(-`as.Date(start_time)`) %>%
correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave()
%>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```




```
#Correlation check to include number of bookings per day as well as all other
numeric variables for Subscriber type   users
TidyDivvy_For_Analysis %>% filter(usertype=="Subscriber") %>%
group_by(as.Date(start_time),Age) %>%
summarise(tripduration=mean(tripduration),Temp_Avg=mean(Temp_Avg),Precip=mean
(Precip),Snow_depth=mean(Snow_depth),Ride_Start_Time=mean(Ride_Start_Time),bo
oking_count=n()) %>% ungroup() %>% select(-`as.Date(start_time)`) %>%
correlate( use = "pairwise.complete.obs", method = "pearson") %>%  shave()
%>%  rplot()

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote.
Defaulting to continuous.
```



Different variables effect on trip duration

```
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>% filter(!is.na(Age)) %>%
mutate(weekend=ifelse(weekdays(as.Date(start_time)) %in%
c("Saturday","Sunday"),"Yes","No")) #code added to included effect of weekend

#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#SET FORMULAS
model_1 <- tripduration ~ Temp_Avg*usertype
model_2 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender +
usertype
model_3 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender +
Ride_Start_Time + Ride_Start_Year + Ride_Start_Month + Ride_Start_Day+
weekend
model_4 <- tripduration ~ Temp_Avg*gender*Ride_Start_Month
```

```r
model_5 <- tripduration ~ usertype*weekend*Ride_Start_Month


#MODEL
model1 <- lm(formula = model_1, data = train)
model2 <- lm(formula = model_2, data = train)
model3 <- lm(formula = model_3, data = train)
model4 <- lm(formula = model_4, data = train)
model5 <- lm(formula = model_5, data = train)

#SUMMARY
rsquare(model1, data = train)
```

```
## [1] 0.04059248
```

```r
tidy(model1)
```

```
## # A tibble: 6 x 5
##   term                          estimate std.error statistic   p.value
##   <chr>                            <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)                     28.8       1.01      28.4   9.57e-177
## 2 Temp_Avg                         0.119     0.0146     8.10  5.29e- 16
## 3 usertypeDependent              -32.9      30.8       -1.07  2.86e-  1
## 4 usertypeSubscriber             -20.0       1.02     -19.6   2.47e- 85
## 5 Temp_Avg:usertypeDependent       0.149     0.541      0.275 7.83e-  1
## 6 Temp_Avg:usertypeSubscriber     -0.0606    0.0148    -4.11  4.01e-  5
```

```r
rsquare(model2, data = train)
```

```
## [1] 0.04207957
```

```r
tidy(model2)
```

```
## # A tibble: 8 x 5
##   term                  estimate std.error statistic   p.value
##   <chr>                    <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)             33.5      0.251      133.   0.
## 2 Temp_Avg                 0.0615   0.00204     30.1  1.75e-198
## 3 Precip                  -0.751    0.114       -6.61 3.80e- 11
## 4 Snow_depth               0.252    0.0422       5.98 2.28e-  9
## 5 Age                      0.0149   0.00295      5.04 4.70e-  7
## 6 genderMale              -1.94     0.0734     -26.4  4.74e-154
## 7 usertypeDependent      -24.8      7.74        -3.21 1.33e-  3
## 8 usertypeSubscriber     -24.0      0.180     -133.   0.
```

```r
rsquare(model3, data = train)
```

```
## [1] 0.01325407
```

```r
tidy(model3)
```

```
## # A tibble: 53 x 5
##    term                estimate std.error statistic   p.value
##    <chr>                  <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)          7.25    0.346        21.0   1.37e- 97
##  2 Temp_Avg             0.0736  0.00425      17.3   3.62e- 67
##  3 Precip              -1.02    0.117        -8.70  3.41e- 18
##  4 Snow_depth           0.277   0.0444        6.24  4.28e- 10
##  5 Age                  0.0101  0.00304       3.31  9.25e-  4
##  6 genderMale          -2.24    0.0746      -30.1   2.35e-198
##  7 Ride_Start_Time      0.00107 0.0000671    15.9   7.83e- 57
##  8 Ride_Start_Year2016  0.0258  0.110         0.235 8.15e-  1
##  9 Ride_Start_Year2017 -0.174   0.108        -1.60  1.09e-  1
## 10 Ride_Start_Year2018  1.72    0.108        16.0   2.36e- 57
## # ... with 43 more rows

rsquare(model4, data = train)

## [1] 0.006749771

tidy(model4)

## # A tibble: 48 x 5
##    term               estimate std.error statistic  p.value
##    <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)          8.87      1.29       6.88  5.86e-12
##  2 Temp_Avg             0.0784    0.0413     1.90  5.73e- 2
##  3 genderMale           1.08      1.42       0.762 4.46e- 1
##  4 Ride_Start_Month2    0.376     1.75       0.215 8.30e- 1
##  5 Ride_Start_Month3    2.12      1.93       1.10  2.72e- 1
##  6 Ride_Start_Month4   -0.788     1.96      -0.403 6.87e- 1
##  7 Ride_Start_Month5   -1.32      1.93      -0.684 4.94e- 1
##  8 Ride_Start_Month6    7.79      2.36       3.30  9.76e- 4
##  9 Ride_Start_Month7   -1.62      2.81      -0.575 5.65e- 1
## 10 Ride_Start_Month8   -1.76      2.90      -0.604 5.46e- 1
## # ... with 38 more rows

rsquare(model5, data = train)

## [1] 0.04206476

tidy(model5)

## # A tibble: 54 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)            69.5       4.38     15.9  7.97e-57
##  2 usertypeDependent     -24.8      15.5      -1.60 1.10e- 1
##  3 usertypeSubscriber    -58.9       4.38    -13.4  3.44e-41
##  4 weekendYes            -42.9       6.66     -6.45 1.12e-10
##  5 Ride_Start_Month2     -54.5       6.66     -8.19 2.66e-16
##  6 Ride_Start_Month3     -36.4       4.91     -7.40 1.34e-13
##  7 Ride_Start_Month4     -33.4       4.58     -7.30 2.84e-13
```

```
##  8 Ride_Start_Month5     -33.0      4.47     -7.39 1.50e-13
##  9 Ride_Start_Month6     -32.4      4.43     -7.31 2.63e-13
## 10 Ride_Start_Month7     -32.5      4.40     -7.39 1.46e-13
## # ... with 44 more rows
```

Different variables effect on trip duration when user type is Subscriber

---

```r
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
filter(usertype=="Subscriber",!is.na(Age)) %>%
mutate(weekend=ifelse(weekdays(as.Date(start_time)) %in%
c("Saturday","Sunday"),"Yes","No")) #code added to include the effect of
weekend

#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#CHECK DATA AND SET FORMULA
model_1 <- tripduration ~ Temp_Avg + Precip + Snow_depth
model_2 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender
model_3 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender +
Ride_Start_Time + Ride_Start_Year + Ride_Start_Month + Ride_Start_Day +
weekend
model_4 <- tripduration ~ Temp_Avg*gender*Ride_Start_Month
model_5 <- tripduration ~ Temp_Avg*gender*Ride_Start_Month*Age


#MODEL
model1 <- lm(formula = model_1, data = train)
model2 <- lm(formula = model_2, data = train)
model3 <- lm(formula = model_3, data = train)
model4 <- lm(formula = model_4, data = train)
model5 <- lm(formula = model_5, data = train)

#SUMMARY
rsquare(model1, data = train)

## [1] 0.002695028

tidy(model1)
```

```
## # A tibble: 4 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   8.48     0.114       74.3  0.
## 2 Temp_Avg      0.0636   0.00182     35.0  1.80e-267
## 3 Precip       -0.774    0.102       -7.58 3.48e- 14
## 4 Snow_depth    0.337    0.0379       8.89 6.35e- 19
```

```
rsquare(model2, data = train)
```

```
## [1] 0.003739547
```

```
tidy(model2)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  9.58      0.164       58.3  0.
## 2 Temp_Avg     0.0610    0.00183     33.4  7.84e-245
## 3 Precip      -0.742     0.102       -7.26 3.78e- 13
## 4 Snow_depth   0.341     0.0379       9.00 2.29e- 19
## 5 Age          0.00918   0.00265      3.47 5.20e-  4
## 6 genderMale  -1.74      0.0664     -26.2  2.34e-151
```

```
rsquare(model3, data = train)
```

```
## [1] 0.006207144
```

```
tidy(model3)
```

```
## # A tibble: 53 x 5
##    term                 estimate std.error statistic   p.value
##    <chr>                   <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)          7.01     0.304        23.1   1.19e-117
##  2 Temp_Avg             0.0517   0.00375      13.8   3.03e- 43
##  3 Precip              -0.787    0.103        -7.60  2.88e- 14
##  4 Snow_depth           0.332    0.0393        8.47  2.50e- 17
##  5 Age                  0.0225   0.00269       8.36  6.20e- 17
##  6 genderMale          -1.68     0.0664      -25.3   6.78e-141
##  7 Ride_Start_Time      0.000989 0.0000592    16.7   1.10e- 62
##  8 Ride_Start_Year2016  0.136    0.0959        1.42  1.56e-  1
##  9 Ride_Start_Year2017 -0.0538   0.0943       -0.570 5.68e-  1
## 10 Ride_Start_Year2018  0.828    0.0946        8.75  2.11e- 18
## # ... with 43 more rows
```

```
rsquare(model4, data = train)
```

```
## [1] 0.003939689
```

```
tidy(model4)
```

```
## # A tibble: 48 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)         8.98      1.13      7.95   1.88e-15
##  2 Temp_Avg            0.0584    0.0360    1.63   1.04e- 1
##  3 genderMale          0.100     1.24      0.0809 9.36e- 1
##  4 Ride_Start_Month2   0.435     1.54      0.283  7.77e- 1
##  5 Ride_Start_Month3   2.27      1.68      1.35   1.76e- 1
##  6 Ride_Start_Month4   3.36      1.73      1.94   5.21e- 2
##  7 Ride_Start_Month5   0.153     1.70      0.0898 9.28e- 1
##  8 Ride_Start_Month6   7.17      2.11      3.40   6.75e- 4
##  9 Ride_Start_Month7   7.30      2.53      2.88   3.95e- 3
## 10 Ride_Start_Month8  -1.48      2.61     -0.570  5.69e- 1
## # ... with 38 more rows
```

```
rsquare(model5, data = train)
```

```
## [1] 0.004216575
```

```
tidy(model5)
```

```
## # A tibble: 96 x 5
##    term              estimate std.error statistic p.value
##    <chr>                <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)         6.69      4.29     1.56     0.119
##  2 Temp_Avg            0.121     0.137    0.878    0.380
##  3 genderMale          0.0183    4.70     0.00390  0.997
##  4 Ride_Start_Month2 -14.3       5.79    -2.48     0.0132
##  5 Ride_Start_Month3   7.44      6.30     1.18     0.237
##  6 Ride_Start_Month4  -2.80      6.44    -0.435    0.663
##  7 Ride_Start_Month5  -6.89      6.36    -1.08     0.278
##  8 Ride_Start_Month6  12.9       7.87     1.65     0.0998
##  9 Ride_Start_Month7  -5.42      9.72    -0.558    0.577
## 10 Ride_Start_Month8 -13.0       9.80    -1.32     0.185
## # ... with 86 more rows
```

Different variables effect on trip duration when user type is Customer

---

```
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
filter(usertype=="Customer",!is.na(Age)) %>%
mutate(weekend=ifelse(weekdays(as.Date(start_time)) %in%
c("Saturday","Sunday"),"Yes","No")) #code added to include the effect of
weekend

#SAMPLE SPLIT
```

```r
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#SET FORMULA
model_1 <- tripduration ~ Temp_Avg + Precip + Snow_depth
model_2 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender
model_3 <- tripduration ~ Temp_Avg + Precip + Snow_depth + Age + gender +
Ride_Start_Time + Ride_Start_Year + Ride_Start_Month + Ride_Start_Day +
weekend
model_4 <- tripduration ~ Temp_Avg*gender*Ride_Start_Month
model_5 <- tripduration ~ Temp_Avg*gender*Ride_Start_Month*Age

#MODEL
model1 <- lm(formula = model_1, data = train)
model2 <- lm(formula = model_2, data = train)
model3 <- lm(formula = model_3, data = train)
model4 <- lm(formula = model_4, data = train)
model5 <- lm(formula = model_5, data = train)

#SUMMARY
rsquare(model1, data = train)

## [1] 0.0006052531

tidy(model1)

## # A tibble: 4 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  30.4       2.73       11.1  9.26e-29
## 2 Temp_Avg      0.0994    0.0393      2.53 1.14e- 2
## 3 Precip       -2.23      1.64       -1.36 1.74e- 1
## 4 Snow_depth    4.24      2.77        1.53 1.27e- 1

rsquare(model2, data = train)

## [1] 0.0008633837

tidy(model2)

## # A tibble: 6 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  22.3       3.21        6.95 3.84e-12
## 2 Temp_Avg      0.0725    0.0397      1.83 6.74e- 2
## 3 Precip       -2.00      1.65       -1.21 2.27e- 1
## 4 Snow_depth    4.42      2.77        1.59 1.11e- 1
```

```
## 5 Age           0.413     0.0489      8.45 3.11e-17
## 6 genderMale   -5.78      0.972      -5.95 2.76e- 9
```

```
rsquare(model3, data = train)
```

```
## [1] 0.008832469
```

```
tidy(model3)
```

```
## # A tibble: 53 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)         21.7      42.4        0.511  6.09e- 1
##  2 Temp_Avg             0.203     0.0776     2.62   8.90e- 3
##  3 Precip              -2.07      1.72      -1.20   2.31e- 1
##  4 Snow_depth           4.79      2.86       1.68   9.31e- 2
##  5 Age                  0.395     0.0492     8.02   1.14e-15
##  6 genderMale          -5.25      0.974     -5.40   6.93e- 8
##  7 Ride_Start_Time     -0.00338   0.00113   -2.99   2.81e- 3
##  8 Ride_Start_Year2016  0.963    46.1        0.0209 9.83e- 1
##  9 Ride_Start_Year2017  2.33     42.5        0.0548 9.56e- 1
## 10 Ride_Start_Year2018 26.9      41.2        0.653  5.14e- 1
## # ... with 43 more rows
```

```
rsquare(model4, data = train)
```

```
## [1] 0.004683779
```

```
tidy(model4)
```

```
## # A tibble: 48 x 5
##    term               estimate std.error statistic  p.value
##    <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)          599.      88.8       6.75 1.51e-11
##  2 Temp_Avg             -12.4      2.35     -5.29 1.25e- 7
##  3 genderMale          -594.      98.5      -6.04 1.62e- 9
##  4 Ride_Start_Month2   -577.     106.       -5.43 5.78e- 8
##  5 Ride_Start_Month3   -647.     105.       -6.19 6.13e-10
##  6 Ride_Start_Month4   -598.      93.4      -6.41 1.53e-10
##  7 Ride_Start_Month5   -580.      90.5      -6.41 1.50e-10
##  8 Ride_Start_Month6   -558.      91.5      -6.10 1.11e- 9
##  9 Ride_Start_Month7   -571.      91.9      -6.21 5.43e-10
## 10 Ride_Start_Month8   -527.      92.5      -5.69 1.26e- 8
## # ... with 38 more rows
```

```
rsquare(model5, data = train)
```

```
## [1] 0.01604832
```

```
tidy(model5)
```

```
## # A tibble: 96 x 5
##    term              estimate std.error statistic      p.value
##    <chr>                <dbl>     <dbl>     <dbl>        <dbl>
##  1 (Intercept)         2277.      431.       5.28 0.000000131
##  2 Temp_Avg             -43.3      13.9     -3.13 0.00177
##  3 genderMale         -2208.      512.      -4.31 0.0000161
##  4 Ride_Start_Month2  -2379.      677.      -3.52 0.000441
##  5 Ride_Start_Month3  -2028.      467.      -4.34 0.0000141
##  6 Ride_Start_Month4  -2151.      441.      -4.87 0.00000111
##  7 Ride_Start_Month5  -2342.      436.      -5.37 0.0000000795
##  8 Ride_Start_Month6  -2212.      438.      -5.05 0.000000437
##  9 Ride_Start_Month7  -2229.      440.      -5.06 0.000000417
## 10 Ride_Start_Month8  -2188.      441.      -4.96 0.000000718
## # ... with 86 more rows
```

Different independent variables effect on prediction of User type

---

```
#WRANGLE DATA
TidyDivvy_For_Analysis1 <-  TidyDivvy_For_Analysis %>%
filter(!is.na(Age),usertype!="Dependent") %>%
mutate(Is_subscriber=ifelse(usertype=="Subscriber",1,0),weekend=ifelse(weekdays(as.Date(start_time)) %in% c("Saturday","Sunday"),"Yes","No")) #code added to include the effect of weekend

#SET FORMULA
model_1 <- Is_subscriber ~ tripduration  + gender + Age + weekend

#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T, prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#RUN MODEL
model1 <- glm(formula = model_1, family = "binomial", data = train)

#SUMMARY
list(model1 = pscl::pR2(model1)["McFadden"])

## fitting null model for pseudo-r2

## $model1
##  McFadden
## 0.1219096
```

```
summary(model1)

##
## Call:
## glm(formula = model_1, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5853   0.1469   0.2085   0.2698   8.4904
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.4154395  0.0414762   34.13   <2e-16 ***
## tripduration -0.0290306  0.0004005  -72.49   <2e-16 ***
## genderMale    0.4010423  0.0176821   22.68   <2e-16 ***
## Age           0.0699780  0.0011764   59.48   <2e-16 ***
## weekendYes   -0.8699211  0.0174728  -49.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 136525  on 475522  degrees of freedom
## Residual deviance: 119881  on 475518  degrees of freedom
##   (937 observations deleted due to missingness)
## AIC: 119891
##
## Number of Fisher Scoring iterations: 8
```

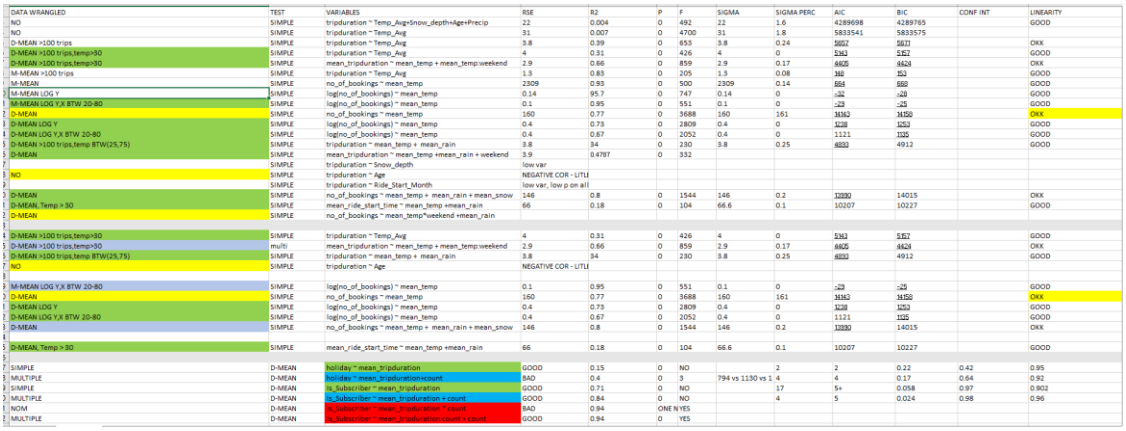
EXTRACTION OF POSSIBLE INSIGHTS

From the above results it can be noticed that though many factors are significant in predicting the output, the amount of variance explained by them in quite low. So, having a level of analysis at individual booking it becoming a challenge, as most of our data point are not able to explain good amount of variance.

For us, we feel this to be a real-world scenario where data collected by companies may not produce sufficient result in prediction. That being said, we did not want the data collected by the company to be wasted and wanted to scavenge what was possible.

We have worked around the problem, by group bookings by the day and other aggregations levels, to see if there is a bigger picture hidden in the data. This eagle eye view helps us better understand what consensus the data can offer with a good prediction and how can it be useful to the company in designing its marketing and other strategies.

A point to note here is that, keeping the valuable reader's time value in mind, we have skipped presentation of analysis that explain low variance in the response variable even though they have extremely low P-value. Nevertheless, surprising results are still reported in the report. Below are some of the analysis that we performed and reported the best models based on comparing the results in an excel sheet. As well, a quick point to add here is, since we are now dealing with an aggregated data level analysis, current desktops and laptops are able to run them using native R without requiring the use of spark.



DATA WRANGLED	TEST	VARIABLES	RSE	R2	P	F	SIGMA	SIGMA PERC	AIC	BIC	CONF INT	LINEARITY
NO	SIMPLE	tripduration ~ Temp_Avg+Snow_depth+Age+Precip	22	0.004	0	492	22	1.6	4289698	4289765		GOOD
NO	SIMPLE	tripduration ~ Temp_Avg	31	0.007	0	4700	31	1.8	5833541	5833575		
D-MEAN >100 trips	SIMPLE	tripduration ~ Temp_Avg	3.8	0.39	0	653	3.8	0.24	5657	5671		OKK
D-MEAN >100 trips,temp>30	SIMPLE	tripduration ~ Temp_Avg	4	0.31	0	426	4	0	5743	5757		GOOD
D-MEAN >100 trips,temp>30	SIMPLE	mean_tripduration ~ mean_temp + mean_temp:weekend	2.9	0.66	0	859	2.9	0.17	4405	4424		OKK
M-MEAN >100 trips	SIMPLE	tripduration ~ Temp_Avg	1.3	0.83	0	205	1.3	0.08	340	353		GOOD
M-MEAN	SIMPLE	no_of_bookings ~ mean_temp	2309	0.93	0	500	2309	0.14	694	688		GOOD
M-MEAN LOG Y	SIMPLE	log(no_of_bookings) ~ mean_temp	0.14	95.7	0	747	0.14	0	-32	-28		GOOD
M-MEAN LOG Y,X BTW 20-80	SIMPLE	log(no_of_bookings) ~ mean_temp	0.1	0.95	0	551	0.1	0	-29	-25		GOOD
D-MEAN	SIMPLE	no_of_bookings ~ mean_temp	160	0.77	0	3688	160	161	14163	14150		OKK
D-MEAN LOG Y	SIMPLE	log(no_of_bookings) ~ mean_temp	0.4	0.73	0	2809	0.4	0	1238	1253		GOOD
D-MEAN LOG Y,X BTW 20-80	SIMPLE	log(no_of_bookings) ~ mean_temp	0.4	0.67	0	2052	0.4	0	1121	1126		GOOD
D-MEAN >100 trips,temp BTW(25,75)	SIMPLE	tripduration ~ mean_temp + mean_rain	3.8	34	0	230	3.8	0.25	4859	4912		GOOD
D-MEAN	SIMPLE	mean_tripduration ~ mean_temp +mean_rain + weekend	3.9	0.4787	0	332						
	SIMPLE	tripduration ~ Snow_depth	low var									
NO	SIMPLE	tripduration ~ Age	NEGATIVE COR - LITLE									
	SIMPLE	tripduration ~ Ride_Start_Month	low var, low p on all									
D-MEAN	SIMPLE	no_of_bookings ~ mean_temp + mean_rain + mean_snow	146	0.8	0	1544	146	0.2	13990	14015		OKK
D-MEAN, Temp > 30	SIMPLE	mean_ride_start_time ~ mean_temp +mean_rain	66	0.18	0	104	66.6	0.1	10207	10227		GOOD
D-MEAN	SIMPLE	no_of_bookings ~ mean_temp*weekend +mean_rain										
D-MEAN >100 trips,temp>30	SIMPLE	tripduration ~ Temp_Avg	4	0.31	0	426	4	0	5743	5757		GOOD
D-MEAN >100 trips,temp>30	multi	mean_tripduration ~ mean_temp + mean_temp:weekend	2.9	0.66	0	859	2.9	0.17	4405	4424		OKK
D-MEAN >100 trips,temp BTW(25,75)	SIMPLE	tripduration ~ mean_temp + mean_rain	3.8	34	0	230	3.8	0.25	4859	4912		GOOD
NO	SIMPLE	tripduration ~ Age	NEGATIVE COR - LITLE									
M-MEAN LOG Y,X BTW 20-80	SIMPLE	log(no_of_bookings) ~ mean_temp	0.1	0.95	0	551	0.1	0	-29	-25		GOOD
D-MEAN	SIMPLE	no_of_bookings ~ mean_temp	160	0.77	0	3688	160	161	14163	14150		OKK
D-MEAN LOG Y	SIMPLE	log(no_of_bookings) ~ mean_temp	0.4	0.73	0	2809	0.4	0	1238	1253		GOOD
D-MEAN LOG Y,X BTW 20-80	SIMPLE	log(no_of_bookings) ~ mean_temp	0.4	0.67	0	2052	0.4	0	1121	1126		GOOD
D-MEAN	SIMPLE	no_of_bookings ~ mean_temp + mean_rain + mean_snow	146	0.8	0	1544	146	0.2	13990	14015		OKK
D-MEAN, Temp > 30	SIMPLE	mean_ride_start_time ~ mean_temp +mean_rain	66	0.18	0	104	66.6	0.1	10207	10227		GOOD
SIMPLE	D-MEAN	holiday ~ mean_tripduration	GOOD	0.15	0	NO		2	2	0.22	0.42	0.95
MULTIPLE	D-MEAN	holiday ~ mean_tripduration+count	BAD	0.4	0	3	794 vs 1130 vs 1	4	4	0.17	0.64	0.92
SIMPLE	D-MEAN	is_Subscriber ~ mean_tripduration	GOOD	0.71	0	NO		17	5+	0.058	0.97	0.902
MULTIPLE	D-MEAN	is_Subscriber ~ mean_tripduration + count	GOOD	0.84	0	NO		4	5	0.024	0.98	0.96
NOM	D-MEAN	is_Subscriber ~ mean_tripduration * count	BAD	0.94	ONE N	YES						
MULTIPLE	D-MEAN	is_Subscriber ~ mean_tripduration:count + count	GOOD	0.94	0	YES						

Various Models Ran

ANALYSIS 1.1

Predict average trip duration per day, given average temperature of the day and interaction affect between mean temperature and if the day is weekday or weekend

Note: For creating the model we are using only those days that have greater than 100 bookings to improve accuracy. Let's say if we have less than 100 bookings, and there are some influential bookings in it that are drastically increasing or decreasing the mean trip duration, the model will become in-accurate. This model prediction works only when the temperature is above 30 degrees. The reason is that, under 30 degrees the relationship between trip duration and temperature is not linear, and modeling including those temperatures cause a lot of error in prediction when temperature falls below 30. So, this model is only accurate to predict mean trip duration when temperatures are above 30.

Overview for Data Scientist:

Temperature, and the interaction affect between temperature and knowing if the day is weekday or weekend explains 66% of variance that is occurring for the mean trip duration per day. From the result following can be concluded:

1. The P-value for the model and independent variables is less than 0.5 and F-Statistic is 859.1, indicating they are very good predictors and model it-self is significant

2. The residual error on the response variable is 2.9 minutes

3. From the co-efficient we can say that, there is a positive correlation between temperature for the day and mean trip duration. It says that for every 1 degree raise in temperature, the mean trip duration for the bookings on that day raise by 0.16 * 60 = 9.6 seconds. Also knowing that it is a weekend or not also has a positive correlation. It states, for a given temperature, the average trip duration for the bookings on weekend is .10 * 60 = 6 seconds more. The intercept can not be interpreted, as 0 degrees is not a practical case.

4. From the confidence interval we can say that one unit increase in temperature causes between .14 to .17 times increase in mean trip duration in minutes. Also given weekend, a unit increase in temperature causes between 0.09 to 0.1 times increase in mean trip duration

5. Plotting the graph between predictor and response variable we can see that relationship between them is linear

6. Looking at the Residual vs Fitted, Scale-Location are looking good with even distribution of residuals. The Q-Q plot looks good for the most part except towards the end, which shows slight deviation from residual being normal. The Residual vs Leverage shows that there are couple of outliers

7. From checking the collinearity, we can see that it is close to 1 showing very less collinearity

8. The MSE value for Test is 8.17 compared to 8.42 on the Train sample set, that shows the model is doing a good job.

Overview for Business:

With every degree increase in temperature we see a raise in mean trip duration, by this we can say that having Divvy bikes expand to warmer regions like California or Arizona can be beneficial, though other considerations have to be kept in mind, such as moving to areas with extreme hot climates will again drop the number of bookings. Also, we see an overall increase in trip duration during the weekends. So. having marketing plans such as Bikers Weekend etc. that focus on engaging more users to ride bikes on weekends is beneficial for increasing business revenue. It is also to be noted, the variance explained is 66% which shows that there are other factors currently unavailable in the data that influence mean trip duration that need to be considered.

MODEL

```
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
mutate(weekend=ifelse(weekdays(as.Date(start_time)) %in%
```

```
c("Saturday","Sunday"),1,0)) %>%
group_by(Ride_Start_Year,Ride_Start_Month,Ride_Start_Day,weekend) %>%
summarise(mean_tripduration=mean(tripduration),mean_temp=mean(Temp_Avg),no_of
_bookings=n()) %>% ungroup() %>% filter(no_of_bookings > 100 & mean_temp>30)

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)
```

```
## # A tibble: 3 x 7
##   Ride_Start_Year Ride_Start_Month Ride_Start_Day weekend mean_tripdurati~
##   <fct>           <fct>            <fct>            <dbl>            <dbl>
## 1 2015            1                17                   1            12.5
## 2 2015            1                19                   0             9.55
## 3 2015            1                20                   0            11.4
## # ... with 2 more variables: mean_temp <dbl>, no_of_bookings <int>
```

```
model_for <- mean_tripduration ~ mean_temp + mean_temp:weekend

#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#MODEL
model1 <- lm(formula = model_for, data = train)

#SUMMARY
summary(model1)
```

```
##
## Call:
## lm(formula = model_for, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6529 -1.7326 -0.4456  1.3954 14.8081
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.335865   0.385223   13.85   <2e-16 ***
## mean_temp         0.162345   0.006484   25.04   <2e-16 ***
## mean_temp:weekend 0.101712   0.003585   28.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.906 on 882 degrees of freedom
## Multiple R-squared:  0.6608, Adjusted R-squared:   0.66
## F-statistic: 859.1 on 2 and 882 DF,  p-value: < 2.2e-16
```

```r
list(model1 = broom::glance(model1))
```

```
## $model1
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <int>  <dbl> <dbl>
<dbl>
## 1     0.661         0.660  2.91      859. 8.69e-208     3 -2198. 4405.
4424.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```r
sigma(model1)/mean(train$mean_tripduration)
```

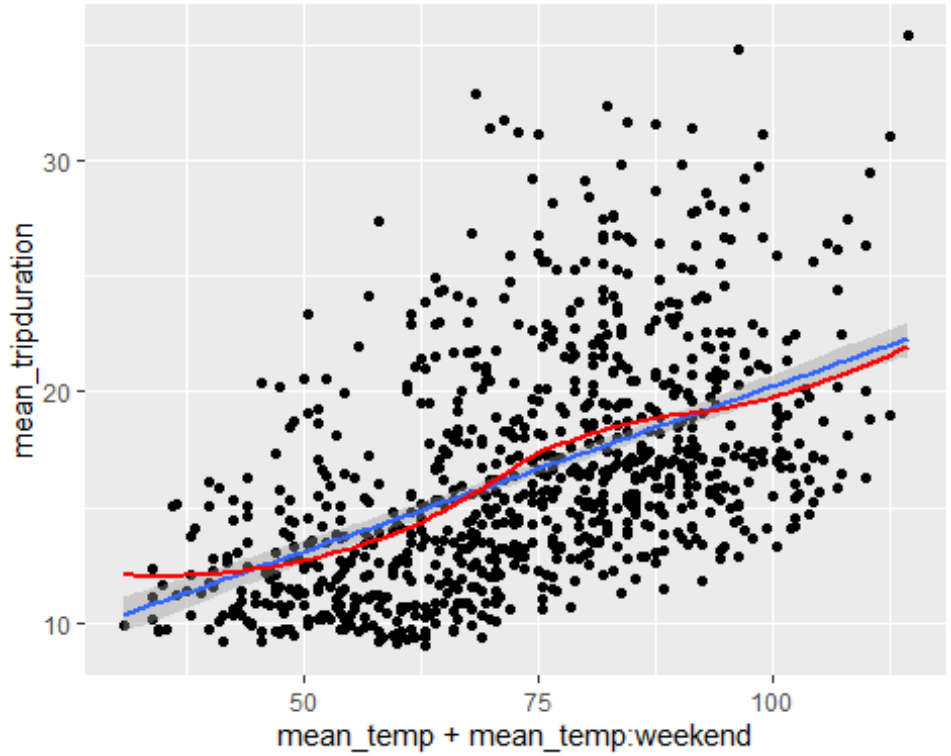
```
## [1] 0.1762212
```

```r
confint(model1)
```

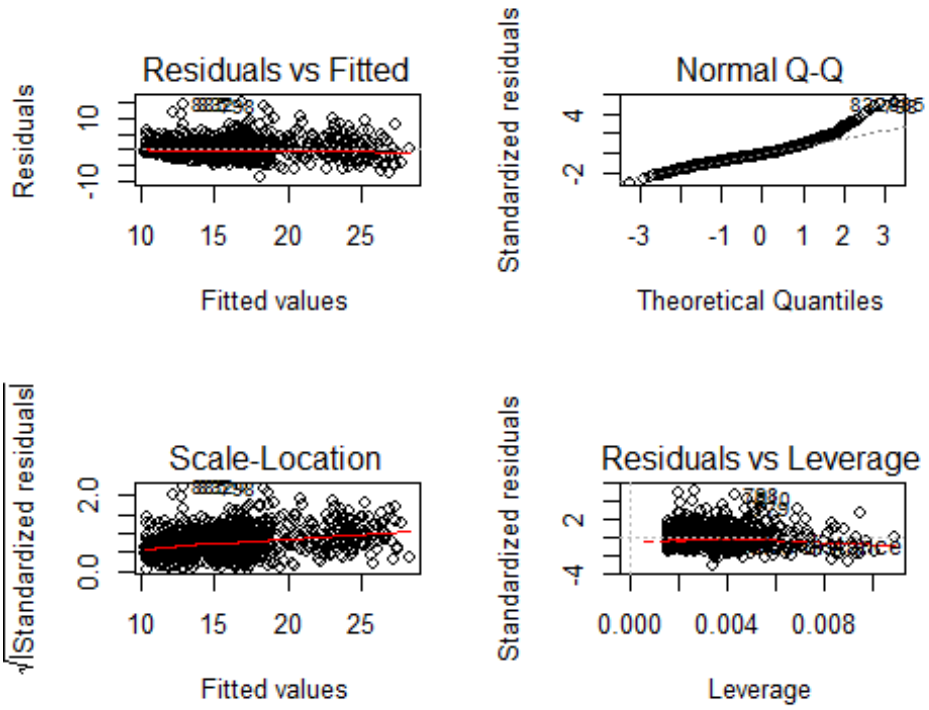
```
##                         2.5 %    97.5 %
## (Intercept)       4.57980535 6.0919251
## mean_temp         0.14962005 0.1750708
## mean_temp:weekend 0.09467565 0.1087476
```

```r
#RSE CHECK
ggplot(train, aes(mean_temp + mean_temp:weekend , mean_tripduration)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
#ALL ASSUMPTIONS - PLOT
par(mfrow=c(2, 2))
plot(model1)
```



```r
par(mfrow=c(1, 1))

#COLLINEARITY CHECK
car::vif(model1)
```

```
##         mean_temp mean_temp:weekend
##          1.028869          1.028869
```

```r
#MSE CHECK
 test %>%
  add_predictions(model1) %>%
  summarise(Test_MSE = mean((mean_tripduration - pred)^2))
```

```
## # A tibble: 1 x 1
##   Test_MSE
##      <dbl>
## 1     8.17
```

```r
train %>%
  add_predictions(model1) %>%
  summarise(Train_MSE = mean((mean_tripduration - pred)^2))
```

```
## # A tibble: 1 x 1
##   Train_MSE
##       <dbl>
## 1      8.42
```

```r
#PREDICT
test %>% add_predictions(model1)
```

```
## # A tibble: 598 x 8
##    Ride_Start_Year Ride_Start_Month Ride_Start_Day weekend
mean_tripdurati~
##    <fct>           <fct>            <fct>            <dbl>
<dbl>
##  1 2015            1                17                   1           12.5
##  2 2015            1                19                   0
9.55
##  3 2015            1                20                   0           11.4
##  4 2015            1                24                   1           11.7
##  5 2015            1                29                   0           13.4
##  6 2015            3                9                    0           12.0
##  7 2015            3                10                   0           11.9
##  8 2015            3                11                   0           11.3
##  9 2015            3                12                   0           14.4
## 10 2015            3                15                   1           21.0
## # ... with 588 more rows, and 3 more variables: mean_temp <dbl>,
## #   no_of_bookings <int>, pred <dbl>
```

ANALYSIS 1.2

Predict average bookings per month given the average temperature for that month

Note

This model prediction works only when the temperature is between 25 and 80 degrees, as that is the range between which the graph is linear. In order to make sure we are preserving good volume of data for analysis, we found that over 90% of booking data we have are between 25 and 80 degrees.

Overview for Data Scientist:

Average monthly Temperature explains 93% variance in the total number of bookings for a particular month

1. The P-value for the model and independent variable is less than 0.5 and F-Statistic is 449.3, indicating average temperature as a very good predictor and model it-self is significant

2. The residual error on the response variable is 2550 booking count

3. From the co-efficient we can say that, there is a positive correlation between average temperature and booking count for that month. It says that for every 1 degree raise in average temperature, the average booking count for the month raise by 520. Again, the intercept can not be interpreted, because 0 degrees is not practical.

4. From the confidence interval we can say that one unit increase in temperature causes between 470 to 570 times increase in booking count.

5. Plotting the graph between predictor and response variable we can see that relationship between them is linear with a slight curve

6. Q-Q plot and Scale-Location are looking good with even distribution of residuals. The Residual vs Fitted plot looks good without funneling. The Residual vs Leverage shows that there are couple of outliers

7. The MSE percentage value for Test is 11% compared to 15% on the Train sample set, that shows the model is doing a good job.

Overview for Business:

With every degree increase in average temperature we see a raise in average bookings per month, this is only true when temperatures are between 20 and 80 degrees. By this we can say that, having Divvy bikes expand to warmer regions can be beneficial, though other considerations have to be kept in mind, such as moving to areas with extreme hot climates will again drop the number of bookings.

MODEL

```r
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_Start_Month) %>%
summarise(mean_temp=mean(Temp_Avg),no_of_bookings=n()) %>% ungroup() %>%
filter(between(mean_temp,20,80))

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)
```

```
## # A tibble: 3 x 4
##   Ride_Start_Year Ride_Start_Month mean_temp no_of_bookings
##   <fct>           <fct>                <dbl>          <int>
## 1 2015            1                     25.5           3279
## 2 2015            3                     40.0           5985
## 3 2015            4                     50.2           9814
```

```r
model_for <- no_of_bookings ~ mean_temp

#SAMPLE SPLIT
set.seed(999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#MODEL
model1 <- lm(formula = model_for, data = train)

#SUMMARY
summary(model1)
```

```
##
## Call:
## lm(formula = model_for, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5694.1 -1553.1    85.1  1681.8  5572.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10591.38    1350.36  -7.843 3.94e-09 ***
## mean_temp      520.18      24.54  21.198  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2550 on 34 degrees of freedom
## Multiple R-squared:  0.9297, Adjusted R-squared:  0.9276
## F-statistic: 449.3 on 1 and 34 DF,  p-value: < 2.2e-16
```

```r
list(model1 = broom::glance(model1))
```

```
## $model1
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl>
<dbl>
## 1     0.930         0.928 2550.      449. 3.55e-21     2  -332.  671.
676.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```r
sigma(model1)/mean(train$no_of_bookings)
```

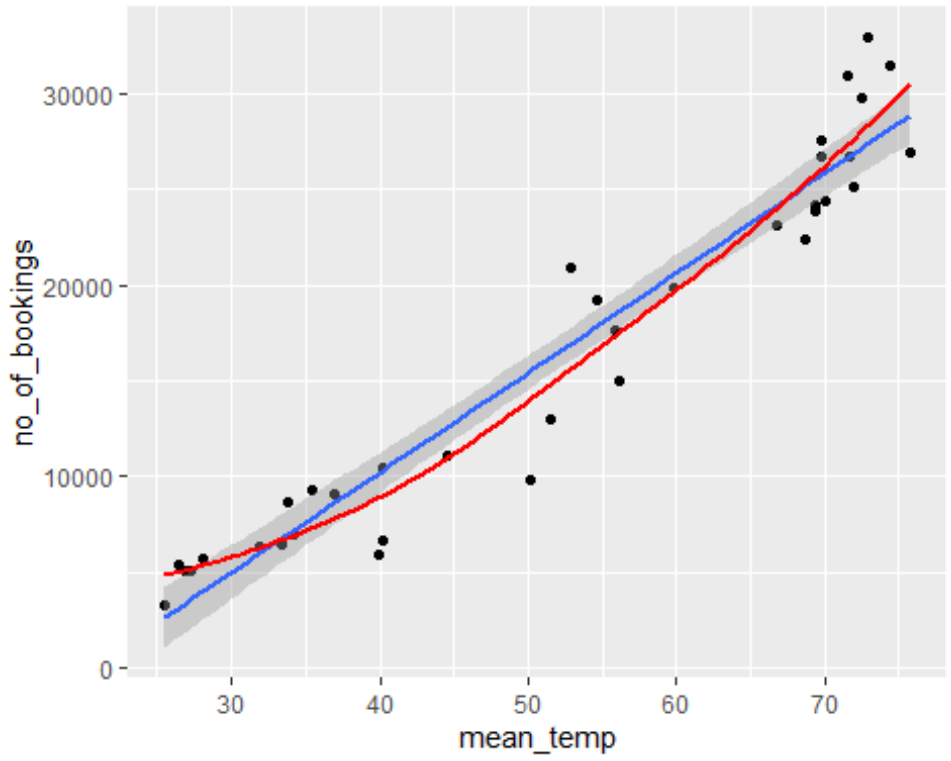
```
## [1] 0.1537958
```

```r
confint(model1)
```

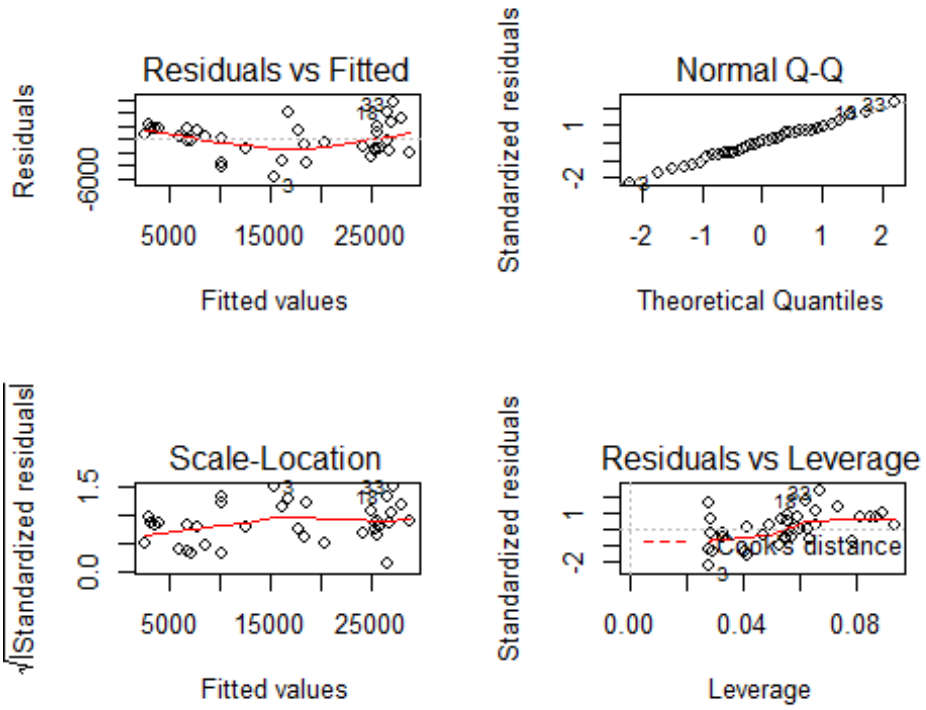
```
##                    2.5 %     97.5 %
## (Intercept) -13335.6320 -7847.1230
## mean_temp      470.3113   570.0517
```

```r
#RSE CHECK
ggplot(train, aes(mean_temp, no_of_bookings)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
#ALL ASSUMPTIONS - PLOT
par(mfrow=c(2, 2))
plot(model1)
```



```r
par(mfrow=c(1, 1))

#MSE CHECK
 test %>%
  add_predictions(model1) %>%
  summarise(Test_MSE = mean((no_of_bookings -
pred)^2),MSE_Perc=sqrt(Test_MSE)/mean(no_of_bookings))# test MSE
```

```
## # A tibble: 1 x 2
##   Test_MSE MSE_Perc
##      <dbl>    <dbl>
## 1 3819210.    0.112
```

```r
train %>%
  add_predictions(model1) %>%
  summarise(Train_MSE = mean((no_of_bookings -
pred)^2),MSE_Perc=sqrt(Train_MSE)/mean(no_of_bookings))# training MSE
```

```
## # A tibble: 1 x 2
##   Train_MSE MSE_Perc
##       <dbl>    <dbl>
## 1  6140032.    0.149
```

```r
#PREDICT
test %>% add_predictions(model1)
```

```
## # A tibble: 23 x 5
##    Ride_Start_Year Ride_Start_Month mean_temp no_of_bookings   pred
##    <fct>           <fct>                <dbl>          <int>  <dbl>
##  1 2015            5                     62.2          16610 21766.
##  2 2015            6                     67.6          23410 24586.
##  3 2015            8                     72.0          27662 26870.
##  4 2015            11                    48.3          10810 14524.
##  5 2016            3                     45.7          10179 13168.
##  6 2016            5                     62.0          20015 21678.
##  7 2016            7                     75.1          28835 28472.
##  8 2016            10                    57.4          19457 19290.
##  9 2016            11                    50.1          13425 15489.
## 10 2017            2                     42.2           9373 11353.
## # ... with 13 more rows
```

Note:

Since the graph between number of bookings per month and mean temperature is curved and not very linear, to improve the accuracy we have used log transformation.

It can be observed that now we have a more linear curve and a better Residual vs Fitted graph as compared to the previous result. Also, we were able to reduce the Test MSE error percentage by doing the transformation.

TRANSFORMED MODEL

```
#DATA WRANGLE
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_Start_Month) %>%
summarise(mean_temp=mean(Temp_Avg),no_of_bookings=n()) %>% ungroup() %>%
filter(between(mean_temp,20,80))

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)
```

```
## # A tibble: 3 x 4
##   Ride_Start_Year Ride_Start_Month mean_temp no_of_bookings
##   <fct>           <fct>                <dbl>          <int>
## 1 2015            1                     25.5           3279
## 2 2015            3                     40.0           5985
## 3 2015            4                     50.2           9814
```

```
model_for <- log(no_of_bookings) ~ mean_temp

#SAMPLE SPLIT
set.seed(999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#MODEL
model1 <- lm(formula = model_for, data = train)

#SUMMARY
summary(model1)
```

```
##
## Call:
## lm(formula = model_for, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42612 -0.06900  0.00489  0.09830  0.40677
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.567856   0.091424   82.78   <2e-16 ***
## mean_temp   0.037331   0.001661   22.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1726 on 34 degrees of freedom
## Multiple R-squared:  0.9369, Adjusted R-squared:  0.935
## F-statistic: 504.9 on 1 and 34 DF,  p-value: < 2.2e-16
```

```
list(model1 = broom::glance(model1))
```

```
## $model1
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl>
<dbl>
## 1     0.937         0.935 0.173      505. 5.57e-22     2   13.2 -20.4 -
15.6
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
sigma(model1)/mean(train$no_of_bookings)
```

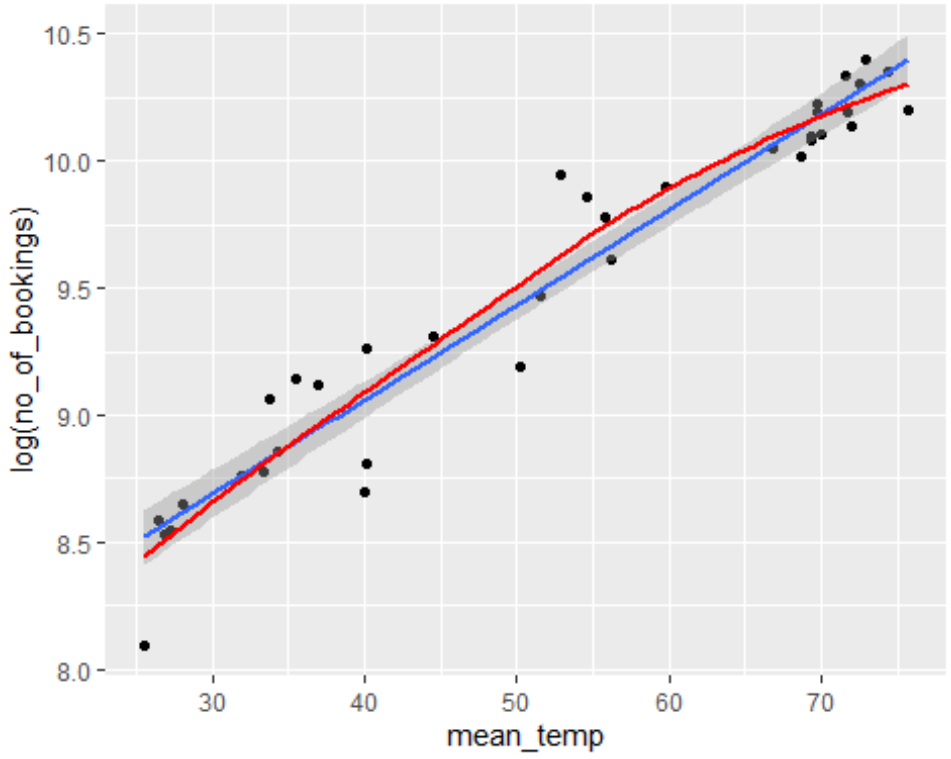
```
## [1] 1.041253e-05
```

```
confint(model1)
```

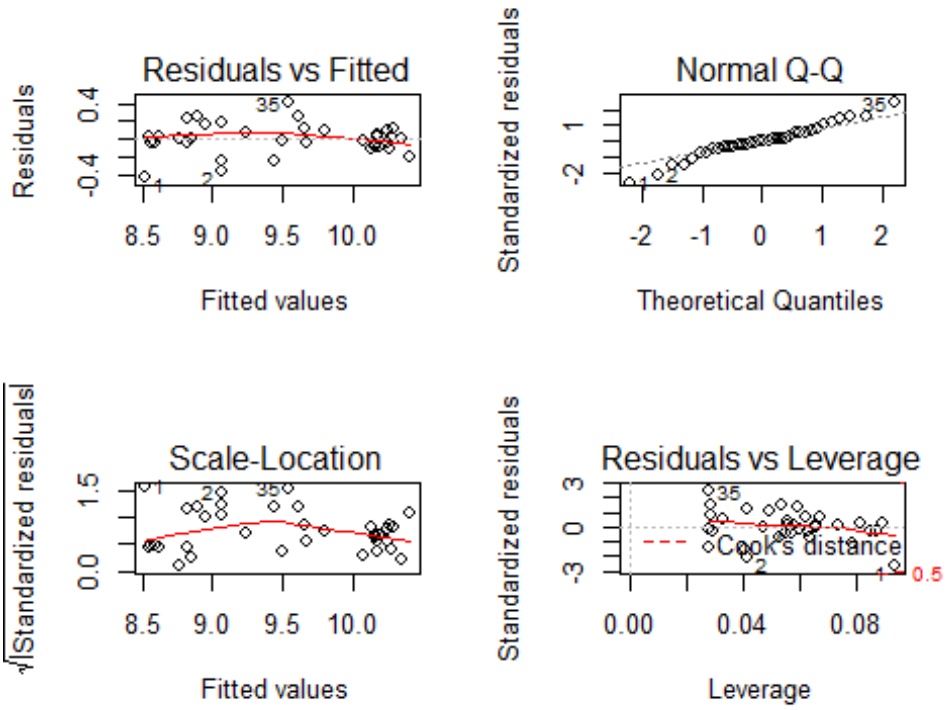
```
##                  2.5 %     97.5 %
## (Intercept) 7.38206064 7.75365235
## mean_temp   0.03395453 0.04070731
```

```
#*VARIABLES* Residual standard error (RSE)
ggplot(train, aes(mean_temp, log(no_of_bookings))) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
#ALL ASSUMPTIONS - PLOT
par(mfrow=c(2, 2))
plot(model1)
```



```r
par(mfrow=c(1, 1))

#MSE CHECK
#Note: Since we are doing a log transformation, we have to perform an
exponential to unwrap the logged result
test %>%
  add_predictions(model1) %>% mutate(pred =exp(pred))  %>%
  summarise(Test_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Test_MSE),mean_response=mean(no_of_bookings),MSE_Perc=A
ct_Dev/mean_response)# test MSE
```

```
## # A tibble: 1 x 4
##   Test_MSE Act_Dev mean_response MSE_Perc
##      <dbl>   <dbl>         <dbl>    <dbl>
## 1 3563036.   1888.        17425.    0.108
```

```r
train %>%
  add_predictions(model1) %>% mutate(pred =exp(pred))  %>%
  summarise(Train_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Train_MSE),mean_response=mean(no_of_bookings),MSE_Perc=
Act_Dev/mean_response)# training MSE
```

```
## # A tibble: 1 x 4
##   Train_MSE Act_Dev mean_response MSE_Perc
##       <dbl>   <dbl>         <dbl>    <dbl>
## 1  5551171.   2356.        16579.    0.142
```

```r
#PREDICT
#Note: Since we are doing a log transformation, we have to perform an
exponential to unwrap the logged result
test %>% add_predictions(model1) %>% mutate(pred =exp(pred))
```

```
## # A tibble: 23 x 5
##    Ride_Start_Year Ride_Start_Month mean_temp no_of_bookings   pred
##    <fct>           <fct>                <dbl>          <int>  <dbl>
##  1 2015            5                     62.2          16610 19732.
##  2 2015            6                     67.6          23410 24158.
##  3 2015            8                     72.0          27662 28460.
##  4 2015            11                    48.3          10810 11734.
##  5 2016            3                     45.7          10179 10647.
##  6 2016            5                     62.0          20015 19608.
##  7 2016            7                     75.1          28835 31928.
##  8 2016            10                    57.4          19457 16520.
##  9 2016            11                    50.1          13425 12576.
## 10 2017            2                     42.2           9373  9346.
## # ... with 13 more rows
```

ANALYSIS 1.3

Predict average bookings per day given the average temperature, snow depth and rain for that day

Overview for Data Scientist

Average Temperature, rain and snow depth explains 80% variance in average bookings per day

1. The P-value for the model and independent variables are less than 0.5 and F-Statistic is 1544, indicating average temperature, snow depth and rain as a good predictor and model it-self is significant

2. The residual error on the response variable is 146 booking count

3. From the co-efficient we can say that, there is a positive correlation between average temperature and number of bookings for that day. It says, for every 1 degree raise in average temperature, an average increase of 15 bookings is observed. A strange discovery shows a positive correlation with snow depth, it informs that every 1 inch raise in snow depth is increasing the booking count by 10. Coming to rain effect, it is as expected, mean rain in inches has a negative correlation, that shows that one inch of rain causes the booking count to fall by 229 times for that day.

4. Looking at the confidence interval we can see the range between the co-efficient that explain the amount of variation in number of bookings

5. Plotting the graph between predictor and response variable we can see that relationship is not linear, suggesting a transformation

6. Q-Q plot and Scale-Location are looking good with even distribution of residuals. The Residual vs Fitted plot shows funneling effect, highlighting the issue with residuals. The Residual vs Leverage shows that there are couple of outliers

7. The MSE percentage value for Test is 26.8% compared to 26.1% on the Train sample set, that shows the model is not very impressive.

Though the variance explained and P-value show that the model is doing a good job, it is breaking the residual assumptions and following non-linear curve, as well having a high MSE error rate. So, we shall be doing a polynomial transformation.

Overview for Business:

There are variables outside the available data, that are influencing the number of bookings per day, our model though to certain extent is able to explain the variation, is not doing justification to the outcome. We will need to include other elements to significantly make a prediction.

Though the model was insignificant, a key takeaway here is that snow depth has a positive correlation, meaning that as snow depth increases the number of bookings are increasing,

which is an interest finding. Possible explanation, during the winter times there is more traffic on road due to snow accumulation and also people are having hard time finding parking spots on roadside due to heavy snow. So, people travelling short distances may be using bikes more often in snow.

MODEL

```
#WRANGLE DATA
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_Start_Month,Ride_Start_Day)  %>%
summarise(mean_temp=mean(Temp_Avg),no_of_bookings=n(),mean_rain=mean(Precip),
mean_snow=mean(Snow_depth)) %>% ungroup()

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)

## # A tibble: 3 x 7
##   Ride_Start_Year Ride_Start_Month Ride_Start_Day mean_temp no_of_bookings
##   <fct>           <fct>            <fct>              <dbl>          <int>
## 1 2015            1                1                   23               53
## 2 2015            1                2                   24.5            132
## 3 2015            1                3                   31.5             41
## # ... with 2 more variables: mean_rain <dbl>, mean_snow <dbl>

model_for <- no_of_bookings ~ mean_temp +  mean_rain + mean_snow

#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#MODEL
model1 <- lm(formula = model_for, data = train)

#SUMMARY
summary(model1)

##
## Call:
## lm(formula = model_for, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -469.14  -99.00    4.02   97.59  493.84
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -224.5373    14.0485 -15.983  < 2e-16 ***
```

```
## mean_temp     15.6243     0.2487  62.836  < 2e-16 ***
## mean_rain   -229.0100    13.4372 -17.043  < 2e-16 ***
## mean_snow     10.8070     2.9325   3.685  0.00024 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 146.9 on 1087 degrees of freedom
## Multiple R-squared:  0.8099, Adjusted R-squared:  0.8094
## F-statistic:  1544 on 3 and 1087 DF,  p-value: < 2.2e-16
```

```
list(model1 = broom::glance(model1))
```

```
## $model1
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl>  <dbl>
<dbl>
## 1     0.810         0.809  147.     1544.       0     4 -6990. 13990.
14015.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
sigma(model1)/mean(train$no_of_bookings)
```

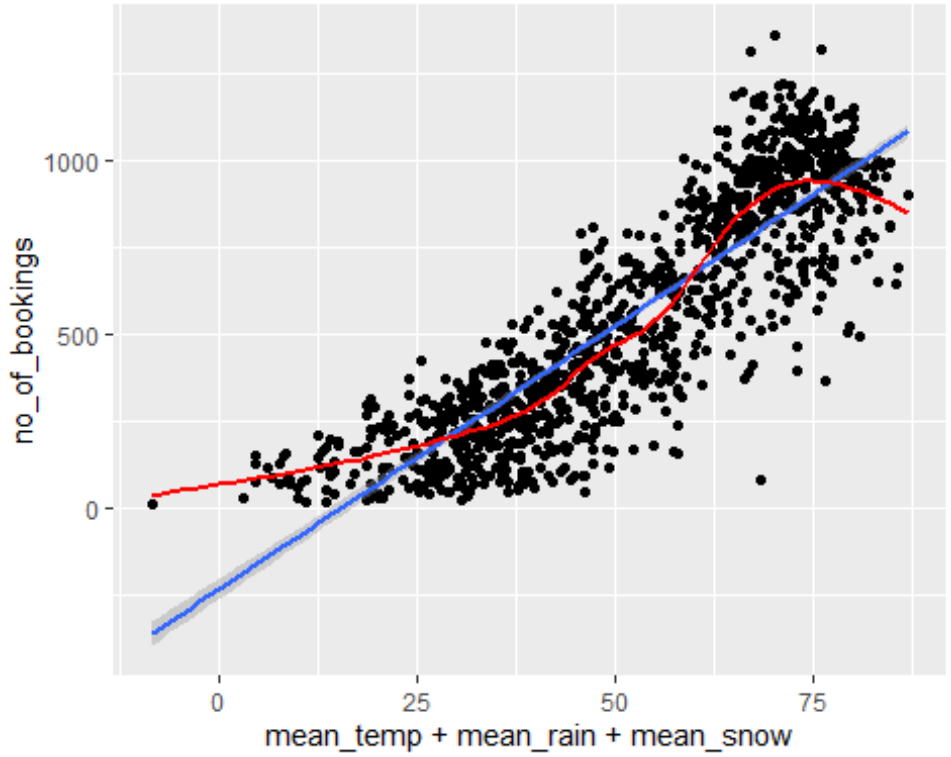
```
## [1] 0.2616626
```

```
confint(model1)
```

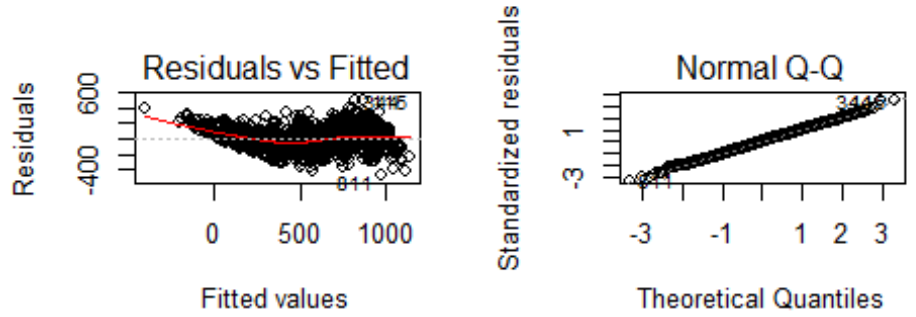
```
##                   2.5 %     97.5 %
## (Intercept) -252.102560 -196.97197
## mean_temp     15.136390   16.11217
## mean_rain   -255.375793 -202.64417
## mean_snow      5.052888   16.56106
```

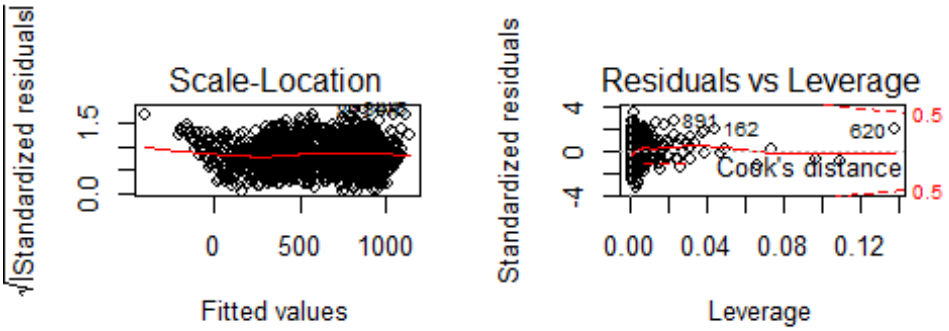
```
#RSE CHECK
ggplot(train, aes(mean_temp +  mean_rain + mean_snow  , no_of_bookings)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_smooth(se = FALSE, color = "red")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#ASSUMPTIONS CHECK - PLOT
par(mfrow=c(2, 2))
plot(model1)
```





```r
par(mfrow=c(1, 1))

#COLLINEARITY CHECK
car::vif(model1)

## mean_temp mean_rain mean_snow
##  1.222052  1.013858  1.206945

#MSE CHECK
 test %>%
  add_predictions(model1) %>%
  summarise(Test_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Test_MSE),mean_response=mean(no_of_bookings),MSE_Perc=A
ct_Dev/mean_response)# test MSE

## # A tibble: 1 x 4
##   Test_MSE Act_Dev mean_response MSE_Perc
##      <dbl>   <dbl>         <dbl>    <dbl>
## 1   20116.    142.          529.    0.268

train %>%
  add_predictions(model1) %>%
  summarise(Train_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Train_MSE),mean_response=mean(no_of_bookings),MSE_Perc=
Act_Dev/mean_response)# training MSE

## # A tibble: 1 x 4
##   Train_MSE Act_Dev mean_response MSE_Perc
##       <dbl>   <dbl>         <dbl>    <dbl>
## 1    21500.    147.          561.    0.261

#PREDICT
 test %>% add_predictions(model1)

## # A tibble: 732 x 8
##    Ride_Start_Year Ride_Start_Month Ride_Start_Day mean_temp
no_of_bookings
##    <fct>           <fct>            <fct>              <dbl>
<int>
##  1 2015            1                1                   23
53
##  2 2015            1                2                   24.5
132
##  3 2015            1                3                   31.5
41
##  4 2015            1                5                    1.5
66
##  5 2015            1                6                    2.5
45
##  6 2015            1                8                    5
28
```

```
##  7 2015            1                9                    5.5
69
##  8 2015            1                10                  11
32
##  9 2015            1                11                  27
38
## 10 2015            1                14                  10
101
## # ... with 722 more rows, and 3 more variables: mean_rain <dbl>,
## #   mean_snow <dbl>, pred <dbl>
```

Note:

Since there is an issue with residuals having a funneling affect and non-linearity of the model. We are applying a polynomial transformation.

On inspecting the results, though the graph has turned to be better, the overall improvement is still not significant. We can say that by looking at the MSE percentage values which are still relatively high. There seems to be a lack of an important predictor variable that better explains the deviations.

TRANSFORMED MODEL

```
#WRANGLE DATA
TidyDivvy_For_Analysis1 <-TidyDivvy_For_Analysis %>%
group_by(Ride_Start_Year,Ride_Start_Month,Ride_Start_Day)  %>%
summarise(mean_temp=mean(Temp_Avg),no_of_bookings=n(),mean_rain=mean(Precip),
mean_snow=mean(Snow_depth)) %>% ungroup()


#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)

## # A tibble: 3 x 7
##   Ride_Start_Year Ride_Start_Month Ride_Start_Day mean_temp no_of_bookings
##   <fct>           <fct>            <fct>              <dbl>          <int>
## 1 2015            1                1                   23               53
## 2 2015            1                2                   24.5            132
## 3 2015            1                3                   31.5             41
## # ... with 2 more variables: mean_rain <dbl>, mean_snow <dbl>

model_for <- no_of_bookings ~ mean_temp + I(mean_temp^2) +  mean_rain +
I(mean_rain^2) #+ mean_snow + I(mean_snow^2)
# snow depth has been commented on purpose as it has become a non-significant
predictor after applying polynomial transformation
```

```r
#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#MODEL
model1 <- lm(formula = model_for, data = train)

#SUMMARY
summary(model1)
```

```
##
## Call:
## lm(formula = model_for, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -512.15  -95.13    4.64   90.00  493.42
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -14.74429   22.94586  -0.643    0.521
## mean_temp         6.12822    1.00976   6.069 1.78e-09 ***
## I(mean_temp^2)    0.09506    0.01022   9.303  < 2e-16 ***
## mean_rain      -342.64105   24.28510 -14.109  < 2e-16 ***
## I(mean_rain^2)   63.83906   11.04325   5.781 9.71e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 140.2 on 1086 degrees of freedom
## Multiple R-squared:  0.8269, Adjusted R-squared:  0.8263
## F-statistic:  1297 on 4 and 1086 DF,  p-value: < 2.2e-16
```

```r
list(model1 = broom::glance(model1))
```

```
## $model1
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik    AIC
BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl>  <dbl>
<dbl>
## 1     0.827         0.826  140.     1297.       0     5 -6939. 13890.
13920.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

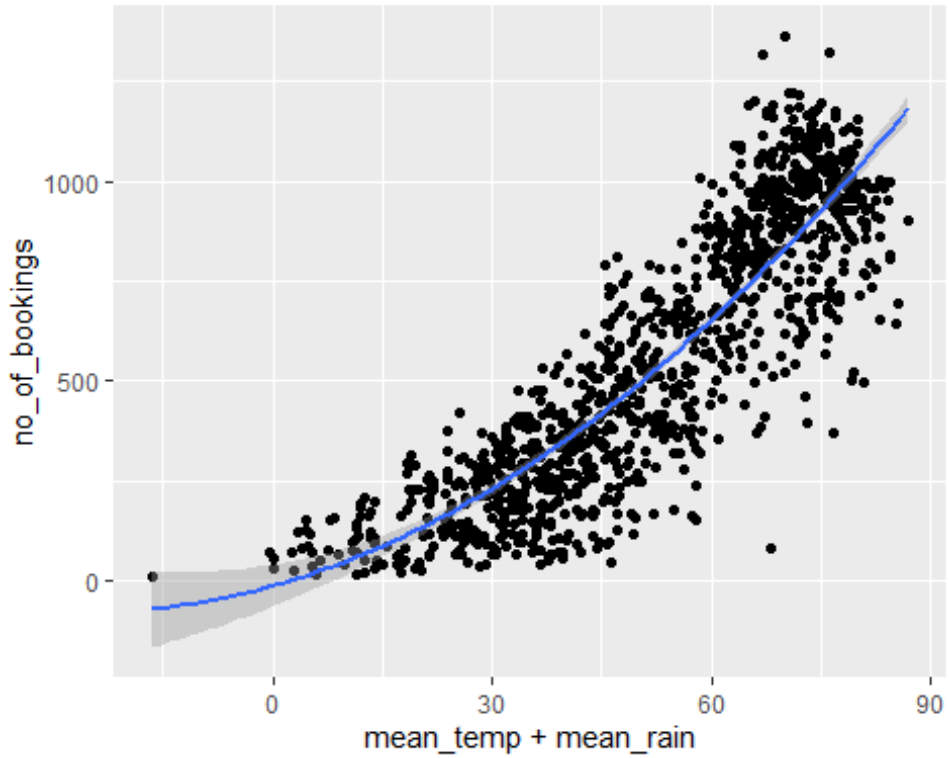
```r
sigma(model1)/mean(train$no_of_bookings)
```

```
## [1] 0.2498168
```

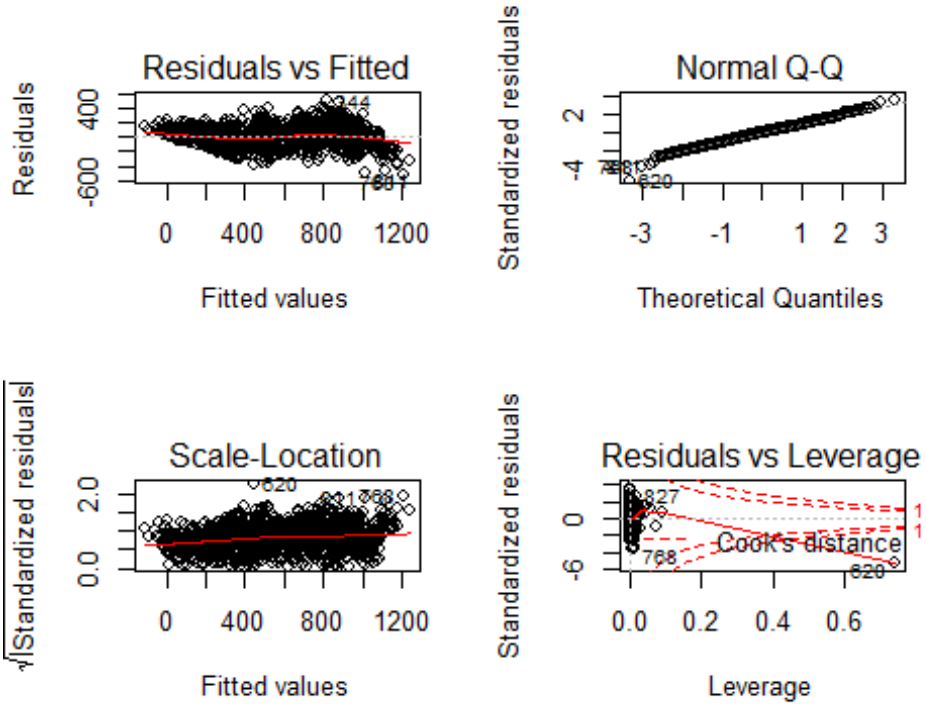
```r
confint(model1)
```

```
##                         2.5 %       97.5 %
## (Intercept)     -59.76752993   30.2789581
## mean_temp         4.14690827    8.1095221
## I(mean_temp^2)    0.07501299    0.1151121
## mean_rain      -390.29206962 -294.9900291
## I(mean_rain^2)   42.17053099   85.5075881
```

```r
#RSE
ggplot(train, aes(mean_temp+mean_rain, no_of_bookings)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x + I(x^2))
```




```r
#ASSUMPTIONS CHECK - PLOT
par(mfrow=c(2, 2))
plot(model1)
```




```
par(mfrow=c(1, 1))

#COLLINEARITY CHECK
car::vif(model1)

##      mean_temp I(mean_temp^2)      mean_rain I(mean_rain^2)
##      22.110010      22.049027       3.633107       3.601473

#MSE CHECK
#Note: Since we are doing a log transformation, we have to perform an
exponential to unwrap the logged result
 test %>%
  add_predictions(model1) %>% mutate(pred =(pred))  %>%
  summarise(Test_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Test_MSE),mean_response=mean(no_of_bookings),MSE_Perc=A
ct_Dev/mean_response)# test MSE

## # A tibble: 1 x 4
##   Test_MSE Act_Dev mean_response MSE_Perc
##      <dbl>   <dbl>         <dbl>    <dbl>
## 1   17154.    131.          529.    0.248

train %>%
  add_predictions(model1) %>% mutate(pred =(pred))  %>%
  summarise(Train_MSE = mean((no_of_bookings -
pred)^2),Act_Dev=sqrt(Train_MSE),mean_response=mean(no_of_bookings),MSE_Perc=
Act_Dev/mean_response)# training MSE
```

```
## # A tibble: 1 x 4
##   Train_MSE Act_Dev mean_response MSE_Perc
##       <dbl>   <dbl>         <dbl>    <dbl>
## 1    19579.    140.          561.    0.249
```

```
#PREDICTION
#Note: Since we are doing a log transformation, we have to perform an
exponential to unwrap the logged result
 test %>% add_predictions(model1)
```

```
## # A tibble: 732 x 8
##    Ride_Start_Year Ride_Start_Month Ride_Start_Day mean_temp
no_of_bookings
##    <fct>           <fct>            <fct>              <dbl>
<int>
##  1 2015            1                1                   23
53
##  2 2015            1                2                   24.5
132
##  3 2015            1                3                   31.5
41
##  4 2015            1                5                    1.5
66
##  5 2015            1                6                    2.5
45
##  6 2015            1                8                    5
28
##  7 2015            1                9                    5.5
69
##  8 2015            1                10                  11
32
##  9 2015            1                11                  27
38
## 10 2015            1                14                  10
101
## # ... with 722 more rows, and 3 more variables: mean_rain <dbl>,
## #   mean_snow <dbl>, pred <dbl>
```

ANALYSIS 2.1

Predicting if user type group is Subscriber or Customer based upon average trip duration and number of bookings on a particular day

Overview for Data Scientist:

Given average trip duration per day and the count of number of bookings of each group, we can predict the User type group (Subscriber or Customer group) for that day with a variance of 85%.

1. The P-value for the model and independent variable is less than 0.5 and residual deviation between null and our model is very large, indicating average trip duration and count as a very good predictor of which user type group it could be, and model it-self is significant

2. Subscriber are denoted by 1, also dependents are considered subscriber as essentially, they are under-aged subscribers

3. Looking at the co-efficient, average trip duration has a negative correlation with Subscriber and count has a positive correlation. Which means subscribers have an average trip duration of 0.64 times less minutes than customers, but the number of bookings per day of subscribers is 206 times more than customers.

4. The confidence interval provides the co-efficient range between which the estimates lie

5. From the variable importance function, we see that both predictor variables are equally important in prediction, with a slight edge for average trip duration being the better one

6. There is no collinearity among the predictors

7. There are couple of residual and outliers that are beyond the threshold

8. The percentage of error from this model is 2% on running it with test data set

9. From the confusion matrix and running the prediction on test, we see that our model has a precision of 98% and accuracy of 96%

10. The ROC curve and AUC value of 0.99 shows that the model is doing a great job.

Overall, we can conclude that this is a very significant model.

Overview for Business

Let us say given daily average trip duration and number of bookings by a certain user group, we are able to determine which type of users are they, either subscribers or customers.

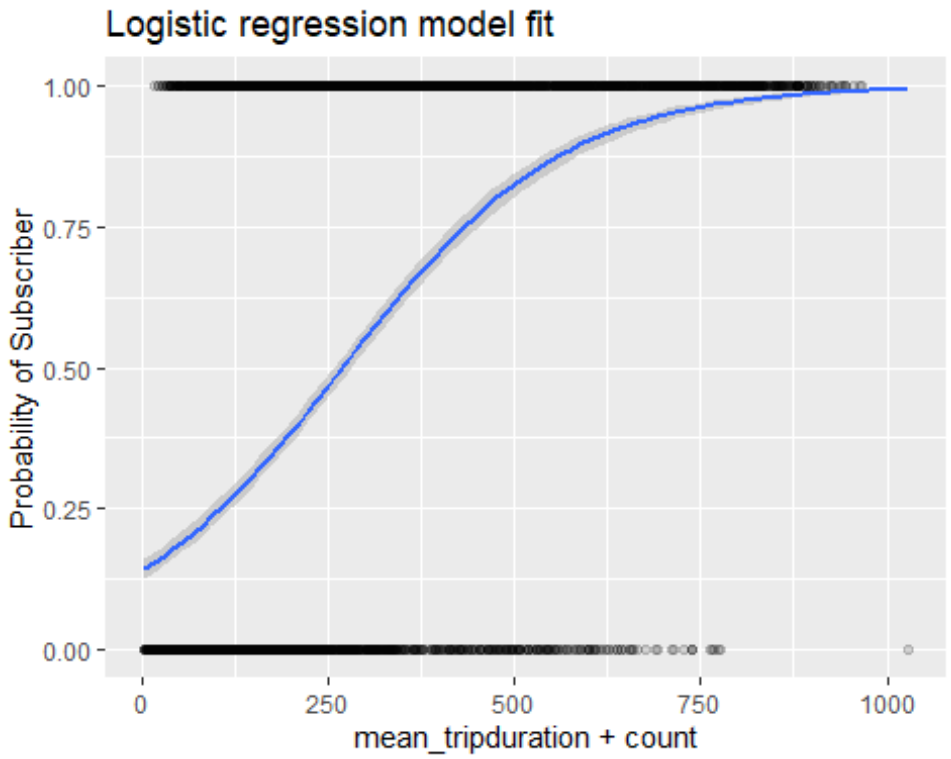
If we flip the coin on the other side, we can also see that a customers in general rides for longer duration compared to subscribers, but on the other hand we have more number of bookings being made per day by subscribers. Marketing programs designed to target customer user group to bring more bookings will boost the company's financial growth, as they are the ones who book bikes for a longer duration.

MODEL

```r
#WRANGLE DATA
TidyDivvy_For_Analysis1 <- TidyDivvy_For_Analysis %>% mutate(Is_Subscriber =
ifelse(usertype=="Customer",0,1)) %>%
group_by(Ride_Start_Year,Ride_Start_Month,Ride_End_Day,Is_Subscriber) %>%
summarise(mean_tripduration=mean(tripduration),count=n(),mean_ride_start_time
=mean(Ride_Start_Time)) %>% ungroup()

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)
```

```
## # A tibble: 3 x 7
##   Ride_Start_Year Ride_Start_Month Ride_End_Day Is_Subscriber
mean_tripdurati~
##   <fct>           <fct>            <fct>                <dbl>
<dbl>
## 1 2015            1                1                        0
23.0
## 2 2015            1                1                        1
9.06
## 3 2015            1                2                        0
21.2
## # ... with 2 more variables: count <int>, mean_ride_start_time <dbl>
```

```r
model_for <- Is_Subscriber ~ mean_tripduration + count

#GRAPH
 TidyDivvy_For_Analysis1 %>%  ggplot(aes(mean_tripduration + count,
Is_Subscriber)) +
  geom_point(alpha = .15) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  ggtitle("Logistic regression model fit") +
  xlab("mean_tripduration + count") +
  ylab("Probability of Subscriber")
```




```
#SAMPLE SPLIT
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#RUN MODEL
model1 <- glm(formula = model_for, family = "binomial", data = train)

#SUMMARY
summary(model1)

##
## Call:
## glm(formula = model_for, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8631  -0.0752   0.0070   0.1283   7.0426
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         5.3319371  0.3360221   15.87   <2e-16 ***
## mean_tripduration -0.4453797  0.0241921  -18.41   <2e-16 ***
## count               0.0119003  0.0008606   13.83   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2981.88  on 2150  degrees of freedom
## Residual deviance:  450.11  on 2148  degrees of freedom
## AIC: 456.11
##
## Number of Fisher Scoring iterations: 8
```

```
list(model1 = pscl::pR2(model1)["McFadden"])
```

```
## fitting null model for pseudo-r2
```

```
## $model1
##  McFadden
## 0.8490512
```

```
exp(coef(model1))
```

```
##       (Intercept) mean_tripduration             count
##         206.838254           0.640581           1.011971
```

```
confint(model1)
```

```
## Waiting for profiling to be done...
```

```
##                          2.5 %      97.5 %
## (Intercept)         4.69994413  6.02001832
## mean_tripduration -0.49553129 -0.40048243
## count              0.01028797  0.01366857
```

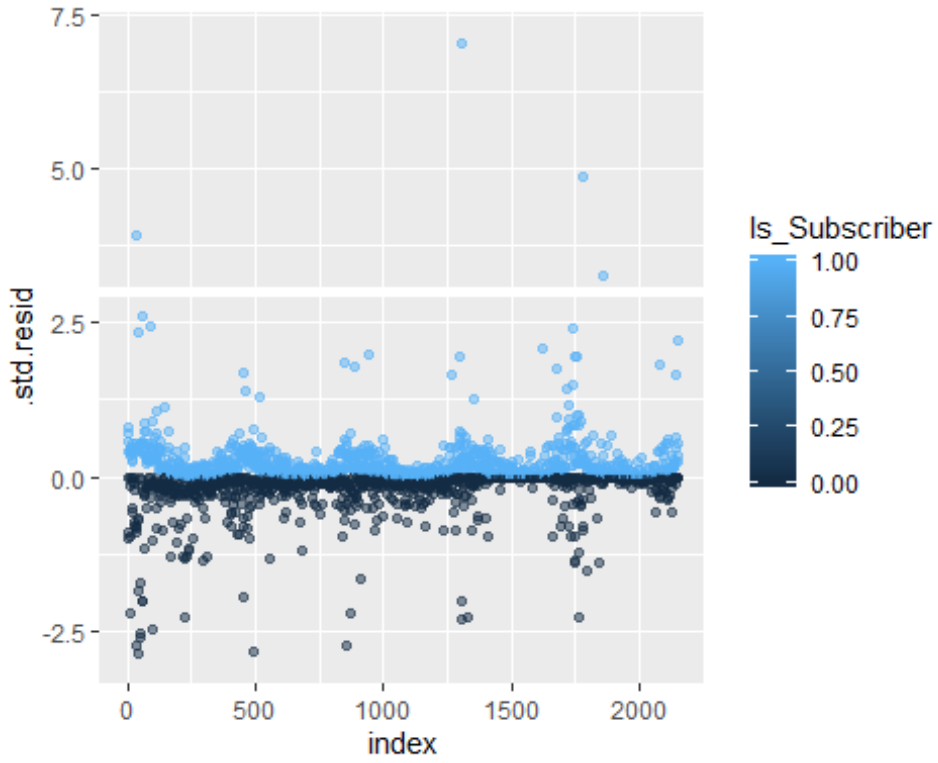
```
#Importance of Predictors
caret::varImp(model1)
```

```
##                    Overall
## mean_tripduration 18.41012
## count             13.82756
```

```
#COLLINIEARITY
car::vif(model1)
```

```
## mean_tripduration             count
##            1.6134            1.6134
```

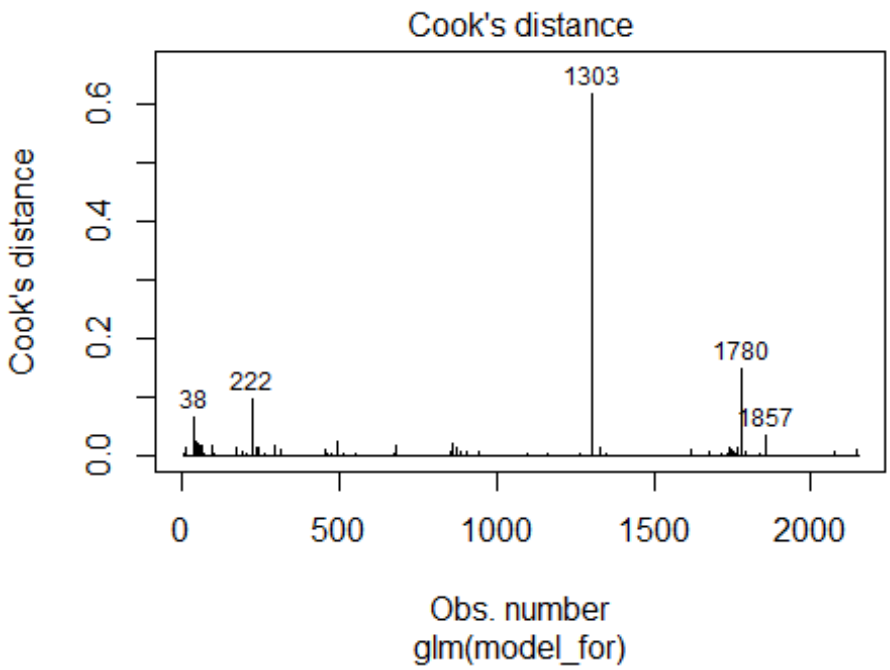
```
#Residual Assessment - OUTLIER CHECK
model1_data <- augment(model1) %>%
  mutate(index = 1:n())
ggplot(model1_data, aes(index, .std.resid, color = Is_Subscriber)) +
  geom_point(alpha = .5) +
  geom_ref_line(h = 3)
```




```
model1_data %>%
  filter(abs(.std.resid) > 3)
```

```
## # A tibble: 4 x 11
##   Is_Subscriber mean_tripdurati~ count .fitted .se.fit .resid     .hat
.sigma
##           <dbl>            <dbl> <int>   <dbl>   <dbl>  <dbl>    <dbl>
<dbl>
## 1             1             30.4    50   -7.63   0.447   3.91 9.68e- 5
0.450
## 2             1             68.3    26  -24.8    1.36    7.04 3.14e-11
0.432
## 3             1             39.9    53  -11.8    0.666   4.86 3.24e- 6
0.446
## 4             1             25.9    76   -5.29   0.329   3.25 5.40e- 4
0.452
## # ... with 3 more variables: .cooksd <dbl>, .std.resid <dbl>, index <int>
```

```
#Cooks Distance
 plot(model1, which = 4, id.n = 5)
```




```
  model1_data %>%   top_n(5, .cooksd)
```

```
## # A tibble: 5 x 11
##   Is_Subscriber mean_tripdurati~ count .fitted .se.fit .resid     .hat
.sigma
##           <dbl>            <dbl> <int>   <dbl>   <dbl>  <dbl>    <dbl>
<dbl>
## 1             1             30.4    50   -7.63   0.447   3.91 9.68e- 5
0.450
## 2             0             33.2   996    2.41   0.575  -2.24 2.50e- 2
0.455
## 3             1             68.3    26  -24.8    1.36    7.04 3.14e-11
0.432
## 4             1             39.9    53  -11.8    0.666   4.86 3.24e- 6
0.446
## 5             1             25.9    76   -5.29   0.329   3.25 5.40e- 4
0.452
## # ... with 3 more variables: .cooksd <dbl>, .std.resid <dbl>, index <int>
```

```
#MISSCLASSIFICATION ERROR
test.predicted.m1 <- predict(model1, newdata = test, type = "response")
list(  model1 = table(test$Is_Subscriber, test.predicted.m1 > 0.5) %>%
prop.table() %>% round(3))
```

```
## $model1
##
##     FALSE  TRUE
```

```
##   0 0.472 0.015
##   1 0.009 0.503
```

```
test %>%
  mutate(m1.pred = ifelse(test.predicted.m1 > 0.5, 1, 0)) %>%
  summarise(m1.error = mean(Is_Subscriber != m1.pred))
```

```
## # A tibble: 1 x 1
##   m1.error
##      <dbl>
## 1   0.0241
```

```
#SENSITIVITY AND SPECIFICITY
(confusion_matrix <- table(test$Is_Subscriber, test.predicted.m1 > 0.5))
```

```
##
##     FALSE TRUE
##   0   685   22
##   1    13  730
```

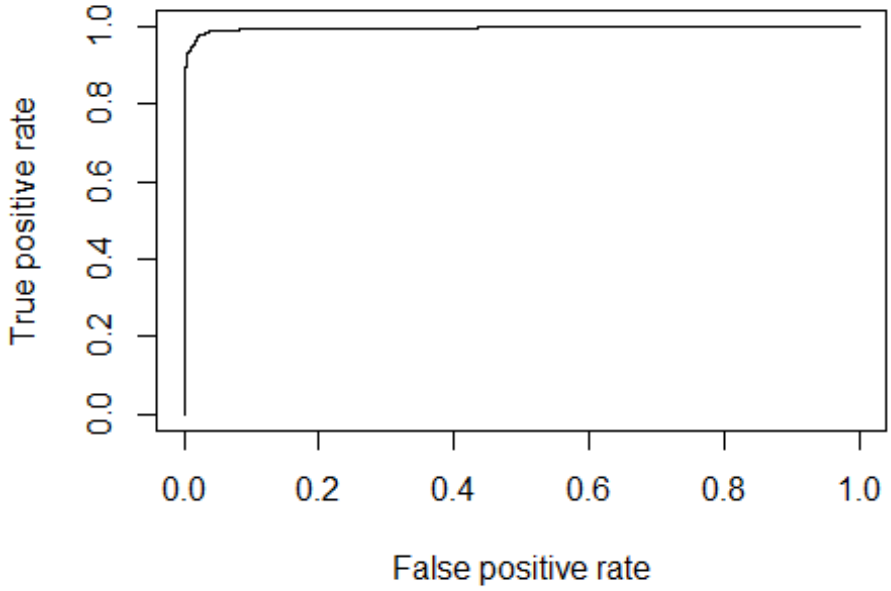
```
(sensitivity_precision  <-
as.numeric(confusion_matrix[2,2])/(as.numeric(confusion_matrix[2,1]+as.numeri
c(confusion_matrix[2,2]))))
```

```
## [1] 0.9825034
```

```
(specificity_accuracy  <-
as.numeric(confusion_matrix[1,1])/(as.numeric(confusion_matrix[1,1]+as.numeri
c(confusion_matrix[1,2]))))
```

```
## [1] 0.9688826
```

```
#ROC AND AUC CURVE
prediction(test.predicted.m1, test$Is_Subscriber) %>%
  performance(measure = "tpr", x.measure = "fpr") %>%
  plot()
```




```
prediction(test.predicted.m1, test$Is_Subscriber) %>%
  performance(measure = "auc") %>%
  .@y.values

## [[1]]
## [1] 0.9946621
```

ANALYSIS 2.2

Predicting if the day is public holiday or not based upon average trip duration and count of bookings made that day.

Overview for Data Scientist:

Given average trip duration per day and the count of number of bookings on that day we can determine if it was a holiday or not with variance of 40%.

1. The P-value for the model and independent variable is less than 0.5 and residual deviation between null and our model is very large, indicating average trip duration and count as a good predictor of public holiday, and model it-self is significant

2. The public holiday is denoted by 1

3. Looking at the co-efficient, average trip duration has a positive correlation with holiday and count has a negative correlation. As trip duration increase by 1 minute, the odd of the day being a holiday are increasing by 1.7 times, but as number of bookings per day increase by 1 count, the chances of it being a holiday fall by 0.99 times. For better understanding, it can also be interpreted as holidays having an average trip duration of 1.7 times more minutes than normal days, but the number of bookings per day on holiday is 0.99 times less which is an interesting discovery.

4. The confidence interval provides the co-efficient range between which the estimates lie

5. From the variable importance function, we see that both predictor variables are equally important in prediction

6. There seems to be a bit of collinearity among the variables but within the threshold

7. There are couple of residual and outliers that are beyond the threshold

8. The percentage of error from running model on test data is 17.7%

9. From the confusion matrix and running the prediction on test, we see that our model has a precision of 64% and accuracy of 92%

10. The ROC curve and AUC value of 0.9 shows that the model is doing a good job.

Overall, there seems to be a bit of problem with precision, nevertheless in applications related to accuracy the model is doing a good job.

Overview for Business

We are able to see that trip duration and count are able to determine if a day is a public holiday or not. Which means that public holidays have an association with trip duration and count.

It can be interpreted that during holidays in general, rides are for longer duration, but as compared working days the number of bookings are low. The same is concluded from our first analysis with weekends (Analysis 1.1). Having promotional plan to engage more users to ride bikes on public holidays can be beneficial for the company.

MODEL

```
#GETTING LIST OF ALL CHICAGO PUBLIC HOLIDAYS
publicholidays <-  as.tibble(USElectionDay(2015:2019)) %>%
union_all(as.tibble(USNewYearsDay(2015:2019))) %>%
union_all(as.tibble(USMLKingsBirthday(2015:2019))) %>%
union_all(as.tibble(USLincolnsBirthday(2015:2019))) %>%
union_all(as.tibble(USWashingtonsBirthday(2015:2019))) %>%
union_all(as.tibble(USCPulaskisBirthday(2015:2019))) %>%
union_all(as.tibble(USMemorialDay(2015:2019))) %>%
union_all(as.tibble(USIndependenceDay(2015:2019))) %>%
union_all(as.tibble(USLaborDay(2015:2019))) %>%
```

```r
union_all(as.tibble(USColumbusDay(2015:2019))) %>%
union_all(as.tibble(USVeteransDay(2015:2019))) %>%
union_all(as.tibble(USThanksgivingDay(2015:2019))) %>%
union_all(as.tibble(USChristmasDay(2015:2019))) %>% rename(Day=`GMT:value`)

#DATA WRANGLE
TidyDivvy_For_Analysis1 <- TidyDivvy_For_Analysis %>%
mutate(ride_day=as.Date(start_time)) %>%
group_by(ride_day,Ride_Start_Year,Ride_Start_Month,Ride_Start_Day) %>%
summarise(count=n(),mean_tripduration=mean(tripduration)) %>% ungroup()  %>%
mutate(day=weekdays(ride_day),weekend=ifelse(day %in%
c("Saturday","Sunday"),"Yes","No")) %>%  mutate(holiday=ifelse(ride_day %in%
as.Date(publicholidays$Day),"Yes",ifelse(weekend =="Yes","Yes","No"))) %>%
mutate(holiday = ifelse(holiday== "Yes",1,0))

#CHECK DATA AND SET FORMULA
TidyDivvy_For_Analysis1 %>% head(3)
```

```
## # A tibble: 3 x 9
##   ride_day   Ride_Start_Year Ride_Start_Month Ride_Start_Day count
##   <date>     <fct>           <fct>            <fct>          <int>
## 1 2015-01-01 2015            1                1                 53
## 2 2015-01-02 2015            1                2                132
## 3 2015-01-03 2015            1                3                 41
## # ... with 4 more variables: mean_tripduration <dbl>, day <chr>, weekend
<chr>,
## #   holiday <dbl>
```

```r
model_for <- holiday ~ mean_tripduration+count

#SAMPLE
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(TidyDivvy_For_Analysis1), replace = T,
prob = c(0.6,0.4))
train <- TidyDivvy_For_Analysis1[sample, ]
test <- TidyDivvy_For_Analysis1[!sample, ]

#RUN THE MODEL
model1 <- glm(formula = model_for, family = "binomial", data = train)

#SUMMARY
summary(model1)
```

```
##
## Call:
## glm(formula = model_for, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.8933  -0.5155  -0.3344   0.2842   2.6700
```

```
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -6.0835626  0.3853720  -15.79   <2e-16 ***
## mean_tripduration  0.5633240  0.0357859   15.74   <2e-16 ***
## count             -0.0067203  0.0004755  -14.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1337.45  on 1090  degrees of freedom
## Residual deviance:  794.35  on 1088  degrees of freedom
## AIC: 800.35
##
## Number of Fisher Scoring iterations: 6
```

```
list(model1 = pscl::pR2(model1)["McFadden"])
```

```
## fitting null model for pseudo-r2
```

```
## $model1
##  McFadden
## 0.4060708
```

```
exp(coef(model1))
```

```
##       (Intercept) mean_tripduration             count
##       0.002280039       1.756501342       0.993302257
```

```
confint(model1)
```

```
## Waiting for profiling to be done...
```

```
##                          2.5 %       97.5 %
## (Intercept)       -6.865563257 -5.353251152
## mean_tripduration  0.495802626  0.636228169
## count             -0.007685261 -0.005819203
```

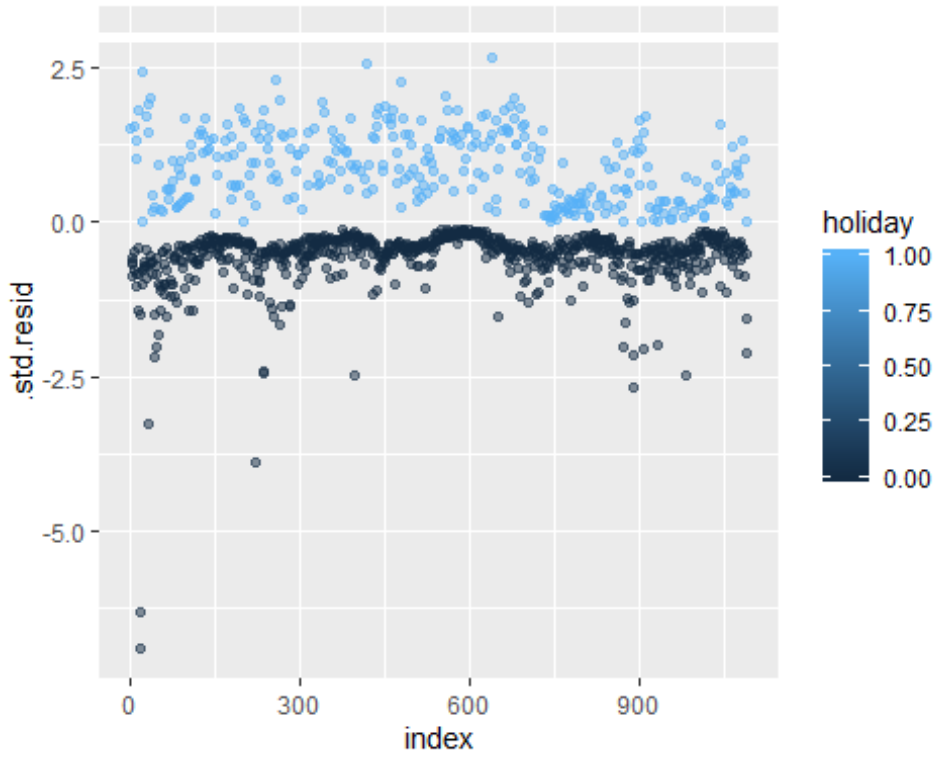
```
#Most influential in predictor
caret::varImp(model1)
```

```
##                   Overall
## mean_tripduration 15.74151
## count             14.13253
```

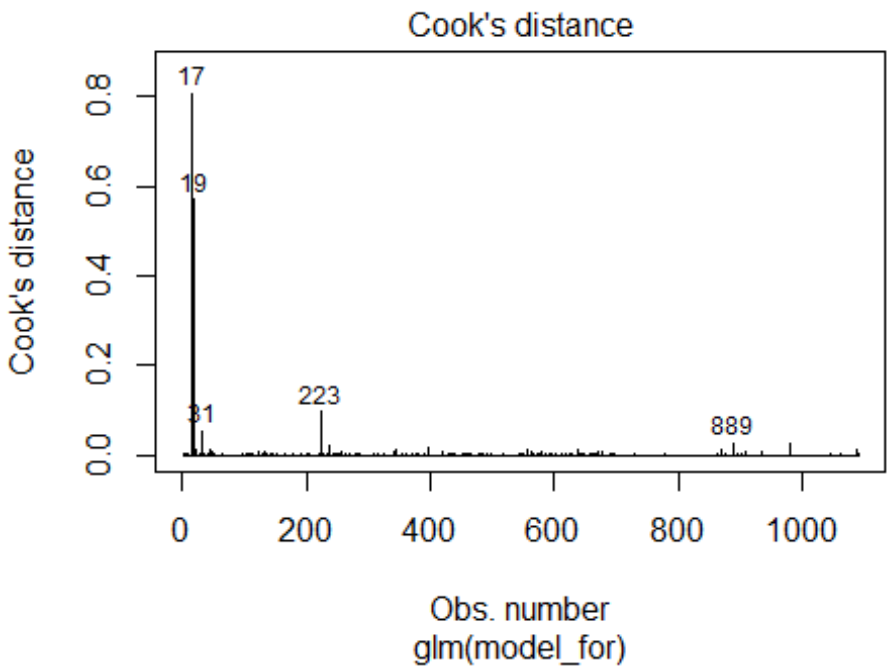
```
#COLLINIEARITY
car::vif(model1)
```

```
## mean_tripduration             count
##          3.231551          3.231551
```

```
#Residual Assessment - OUTLIER CHECK
model1_data <- augment(model1) %>%
  mutate(index = 1:n())
ggplot(model1_data, aes(index, .std.resid, color = holiday)) +
  geom_point(alpha = .5) +
  geom_ref_line(h = 3)
```




```
 model1_data %>%
  filter(abs(.std.resid) > 3)

## # A tibble: 4 x 11
##   holiday mean_tripdurati~ count .fitted .se.fit .resid     .hat .sigma
.cooksd
##     <dbl>            <dbl> <int>   <dbl>   <dbl>  <dbl>    <dbl>  <dbl>
<dbl>
## 1       0             53.2    23   23.8    1.55   -6.89 1.16e-10  0.829
0.803
## 2       0             46.7    52   19.9    1.31   -6.30 4.02e- 9  0.833
0.569
## 3       0             20.8    44    5.32   0.402  -3.26 7.87e- 4  0.849
0.0535
## 4       0             24.8    43    7.58   0.538  -3.89 1.48e- 4  0.847
0.0966
## # ... with 2 more variables: .std.resid <dbl>, index <int>

#Cooks Distance
 plot(model1, which = 4, id.n = 5)
```




```
  model1_data %>%   top_n(5, .cooksd)
```

```
## # A tibble: 5 x 11
##   holiday mean_tripdurati~ count .fitted .se.fit .resid     .hat .sigma
.cooksd
##     <dbl>            <dbl> <int>   <dbl>   <dbl>  <dbl>    <dbl>  <dbl>
<dbl>
## 1       0             53.2    23   23.8    1.55   -6.89 1.16e-10  0.829
0.803
## 2       0             46.7    52   19.9    1.31   -6.30 4.02e- 9  0.833
0.569
## 3       0             20.8    44    5.32   0.402  -3.26 7.87e- 4  0.849
0.0535
## 4       0             24.8    43    7.58   0.538  -3.89 1.48e- 4  0.847
0.0966
## 5       0             17.8    63    3.54   0.299  -2.67 2.45e- 3  0.851
0.0283
## # ... with 2 more variables: .std.resid <dbl>, index <int>
```

```
#MISSCLASSIFICATION ERROR
test.predicted.m1 <- predict(model1, newdata = test, type = "response")
list(  model1 = table(test$holiday, test.predicted.m1 > 0.5) %>% prop.table()
%>% round(3))
```

```
## $model1
##
##     FALSE  TRUE
```

```
##   0 0.611 0.051
##   1 0.122 0.217
```

```
test %>%
  mutate(m1.pred = ifelse(test.predicted.m1 > 0.5, 1, 0)) %>%
  summarise(m1.error = mean(holiday != m1.pred))
```

```
## # A tibble: 1 x 1
##   m1.error
##      <dbl>
## 1    0.172
```

```
#SENSITIVITY AND SPECIFICITY
(confusion_matrix <- table(test$holiday, test.predicted.m1 > 0.5))
```

```
##
##     FALSE TRUE
##   0   447   37
##   1    89  159
```

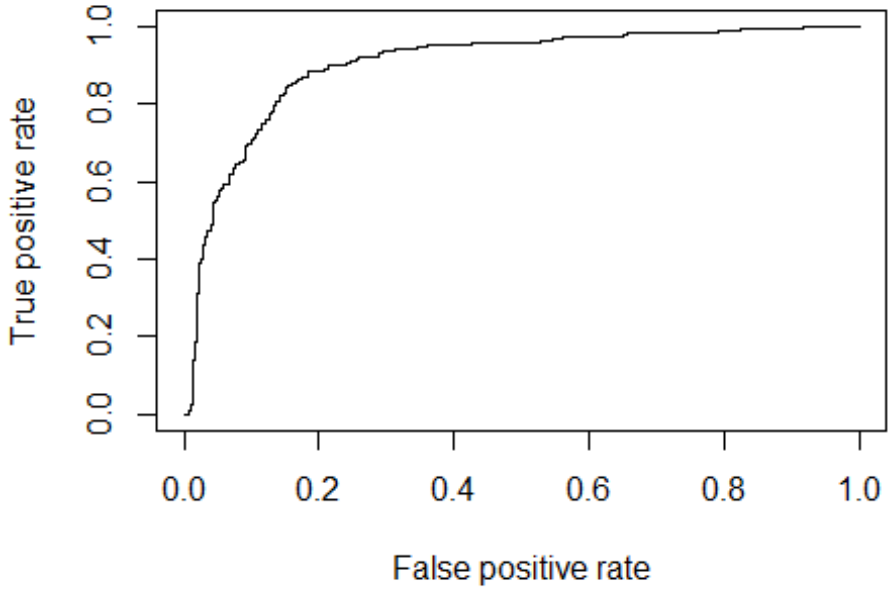
```
(sensitivity_precision  <-
as.numeric(confusion_matrix[2,2])/(as.numeric(confusion_matrix[2,1]+as.numeri
c(confusion_matrix[2,2]))))
```

```
## [1] 0.641129
```

```
(specificity_accuracy  <-
as.numeric(confusion_matrix[1,1])/(as.numeric(confusion_matrix[1,1]+as.numeri
c(confusion_matrix[1,2]))))
```

```
## [1] 0.9235537
```

```
#ROC AND AUC CURVE
prediction(test.predicted.m1, test$holiday) %>%
  performance(measure = "tpr", x.measure = "fpr") %>%
  plot()
```




```
prediction(test.predicted.m1, test$holiday) %>%
  performance(measure = "auc") %>%
  .@y.values

## [[1]]
## [1] 0.9014846
```

ANALYSIS 3.1

CLUSTER AGE GROUPS BASED UPON BOOKING COUNT AND TRIP DURATION

Overview

Here we have 3 clusters that has shown to be an optimum number from ELBOW and SILOUETTE graphs.

1.The first cluster age groups those who contribute to more number of bookings, and have an average trip booking time of 13 minutes

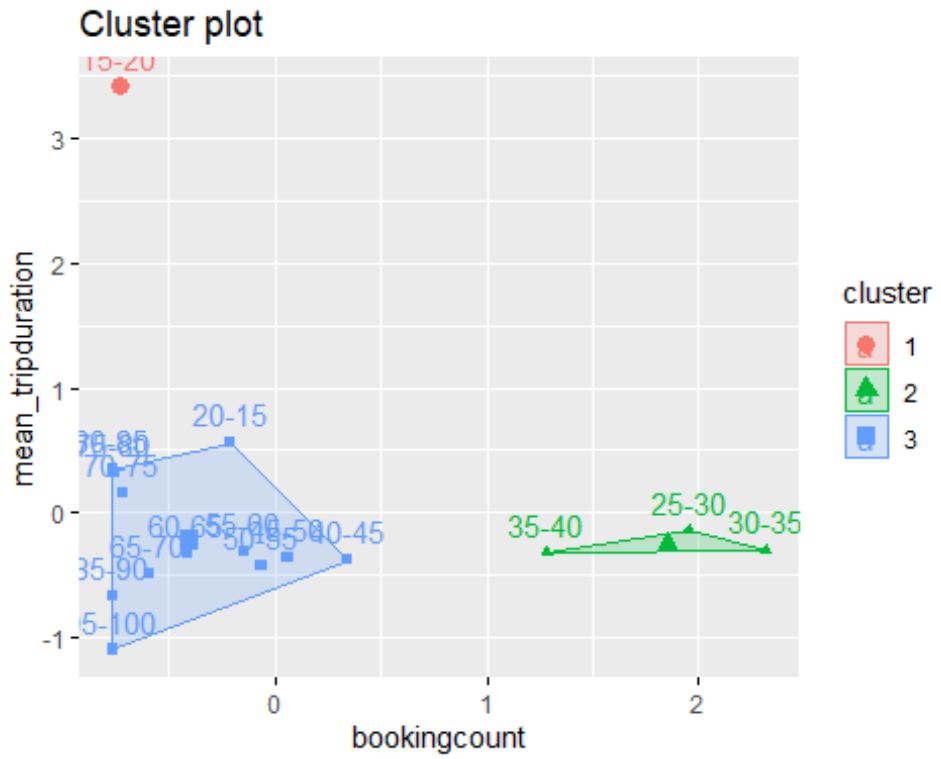
2.The second cluster are users in the age groups that book less often, but generally for more duration (around 22 minutes)

3.The third cluster are users in the age group that account to average number of bookings with a mean duration almost as that of first cluster, at about 13 minutes

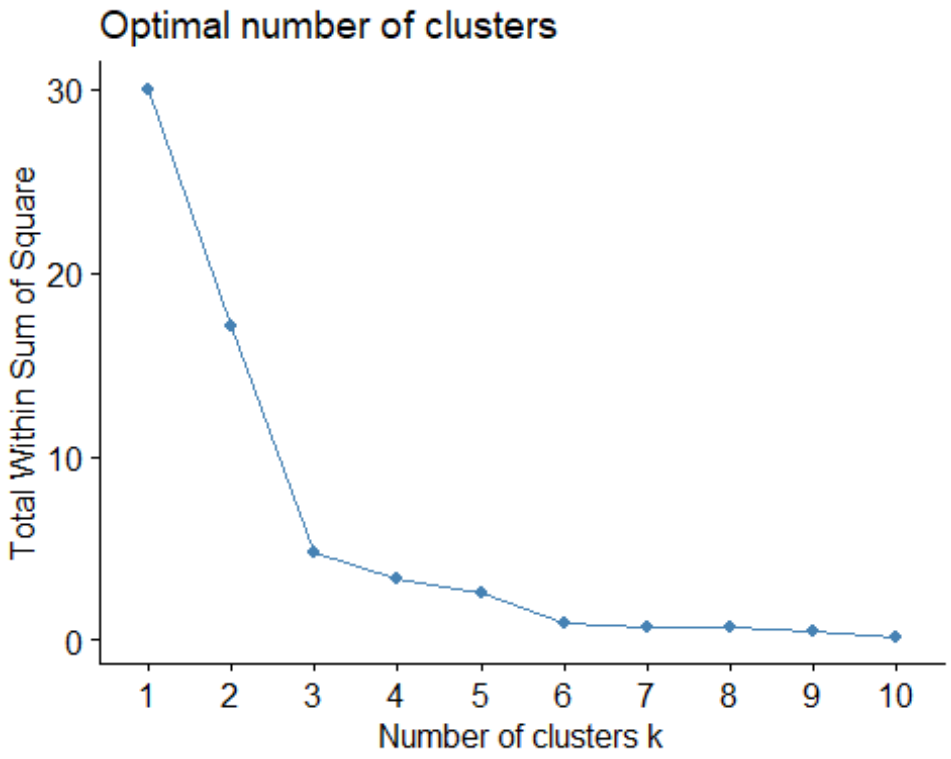
From the above details, targeted advertising can be made to improve booking rate and duration based on Age groups. Also, we can see that the most important age group are between 25-40 who contribute to the majority of bookings.

MODEL

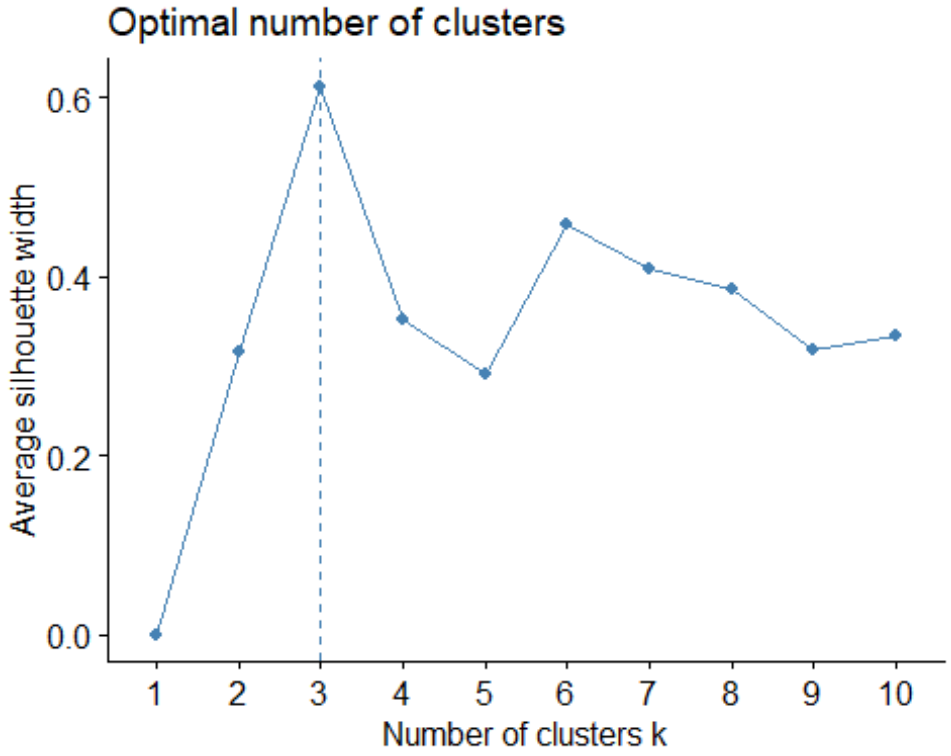
```
#WRANGLE DATA
 TidyDivvy_For_Analysis1 <- TidyDivvy_For_Analysis  %>%  filter(!is.na(Age))
%>% mutate(Age_grp =case_when(
    .$Age  < 5 ~ "1-5",
    .$Age  > 5 & .$Age <= 10 ~ "5-10",
    .$Age  > 10 & .$Age <= 15 ~ "10-15",
    .$Age  > 15 & .$Age <= 20 ~ "15-20",
    .$Age  > 20 & .$Age <= 25 ~ "20-15",
    .$Age  > 25 & .$Age <= 30 ~ "25-30",
    .$Age  > 30 & .$Age <= 35 ~ "30-35",
    .$Age  > 35 & .$Age <= 40 ~ "35-40",
    .$Age  > 40 & .$Age <= 45 ~ "40-45",
    .$Age  > 45 & .$Age <= 50 ~ "45-50",
    .$Age  > 50 & .$Age <= 55 ~ "50-55",
    .$Age  > 55 & .$Age <= 60 ~ "55-60",
    .$Age  > 60 & .$Age <= 65 ~ "60-65",
    .$Age  > 65 & .$Age <= 70 ~ "65-70",
    .$Age  > 70 & .$Age <= 75 ~ "70-75",
    .$Age  > 75 & .$Age <= 80 ~ "75-80",
    .$Age  > 80 & .$Age <= 85 ~ "80-85",
    .$Age  > 85 & .$Age <= 90 ~ "85-90",
    .$Age  > 90 & .$Age <= 95 ~ "90-95",
    .$Age  > 95 & .$Age <= 100 ~ "95-100",
    .$Age  > 100 & .$Age <= 105 ~ "100-105",
    .$Age  > 105 & .$Age <= 110 ~ "105-110",
    TRUE ~ "other"
  ) ) %>%   group_by(Age_grp) %>%
summarise(bookingcount=n(),mean_tripduration=mean(tripduration))  %>%
filter(bookingcount>10) #Setting booking count filter to eleminate clusternig
of age groups based on few records only, as they may contain outliers casuing
them to be designated to in-correct class.

#NORMALIZE
  temp <- na.omit(TidyDivvy_For_Analysis1)
 temp <- temp %>% tibble::column_to_rownames(var="Age_grp")
 temp <- scale(temp)

#CLUSTER
 k2 <- kmeans(temp, centers = 3, nstart = 50)
#VISUALIZE CLUSTER
 fviz_cluster(k2, data = temp)
```




```
#SUMMARISE CLUSTER
 str(k2)

## List of 9
##  $ cluster     : Named int [1:16] 1 3 2 2 2 3 3 3 3 3 ...
##   ..- attr(*, "names")= chr [1:16] "15-20" "20-15" "25-30" "30-35" ...
##  $ centers     : num [1:3, 1:2] -0.734 1.854 -0.402 3.419 -0.255 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : chr [1:2] "bookingcount" "mean_tripduration"
##  $ totss       : num 30
##  $ withinss    : num [1:3] 0 0.57 4.17
##  $ tot.withinss: num 4.74
##  $ betweenss   : num 25.3
##  $ size        : int [1:3] 1 3 12
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

#ELBOW
 fviz_nbclust(temp, kmeans, method = "wss")
```



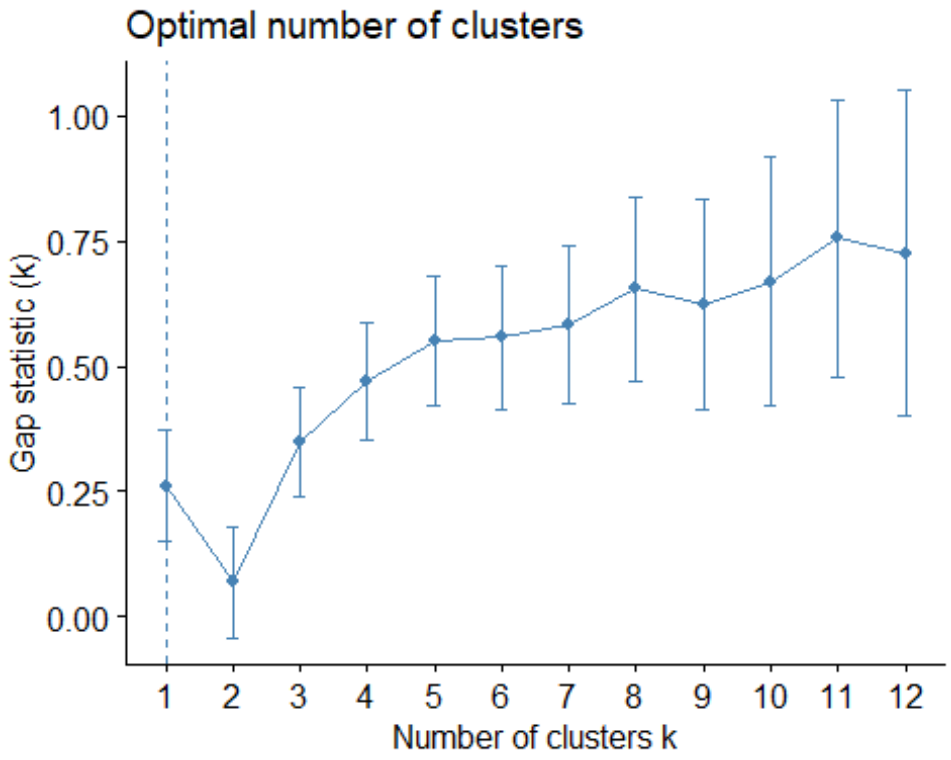

```r
#SILOUETTE
 fviz_nbclust(temp, FUNcluster=kmeans, method = "silhouette")
```



```r
#GAP
 gap_stat <- clusGap(temp, FUN = kmeans, nstart = 25,K.max = 12, B = 50)
  print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = temp, FUNcluster = kmeans, K.max = 12, B = 50, nstart = 25)
## B=50 simulated reference sets, k = 1..12; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 1
##              logW      E.logW        gap    SE.sim
##  [1,]  1.80508918  2.06615281 0.26106363 0.1105970
##  [2,]  1.51850933  1.58710395 0.06859462 0.1109899
##  [3,]  0.92436764  1.27201687 0.34764924 0.1090214
##  [4,]  0.52904808  0.99755089 0.46850281 0.1168620
##  [5,]  0.19133450  0.74300340 0.55166889 0.1281536
##  [6,] -0.04600147  0.51233011 0.55833157 0.1430013
##  [7,] -0.30041079  0.28323465 0.58364544 0.1576372
##  [8,] -0.61735281  0.03780196 0.65515476 0.1829948
##  [9,] -0.84771012 -0.22376798 0.62394214 0.2085500
## [10,] -1.17698821 -0.50712133 0.66986688 0.2468862
## [11,] -1.55974082 -0.80331523 0.75642559 0.2764879
## [12,] -1.89617624 -1.17003912 0.72613711 0.3260198
```

```r
  fviz_gap_stat(gap_stat)
```




```r
#SUMMARISE
TidyDivvy_For_Analysis1 %>%
  mutate(Cluster = k2$cluster) %>%
```

```
  group_by(Cluster) %>%
  summarise(count=n(),mean_tripduration = mean(mean_tripduration))
```

```
## # A tibble: 3 x 3
##   Cluster count mean_tripduration
##     <int> <int>             <dbl>
## 1       1     1              22.5
## 2       2     3              13.0
## 3       3    12              13.1
```

```
#VIEW
TidyDivvy_For_Analysis1 %>%
  mutate(Cluster = k2$cluster)
```

```
## # A tibble: 16 x 4
##    Age_grp bookingcount mean_tripduration Cluster
##    <chr>          <int>             <dbl>   <int>
##  1 15-20           2323              22.5       1
##  2 20-15          36039              15.1       3
##  3 25-30         175679              13.3       2
##  4 30-35         198968              12.9       2
##  5 35-40         132290              12.9       2
##  6 40-45          71761              12.7       3
##  7 45-50          53383              12.8       3
##  8 50-55          45440              12.6       3
##  9 55-60          40044              12.9       3
## 10 60-65          22786              12.8       3
## 11 65-70          11107              12.4       3
## 12 70-75           3150              14.1       3
## 13 75-80            563              14.5       3
## 14 80-85            115              14.6       3
## 15 85-90             66              12.0       3
## 16 95-100            56              10.9       3
```

ANALYSIS 3.2

CLUSTER STATION IDS BASED UPON BOOKING COUNT AND TRIP DURATION

Here we have 3 clusters that has shown to be an optimum number from ELBOW and SILOUETTE graphs.

1. The first cluster are the stations from where most bookings are made, and they generally tend to be around 17 minutes duration

2. The second cluster are stations from where less bookings are made, but they generally tend to be for longer duration around 3 hours.

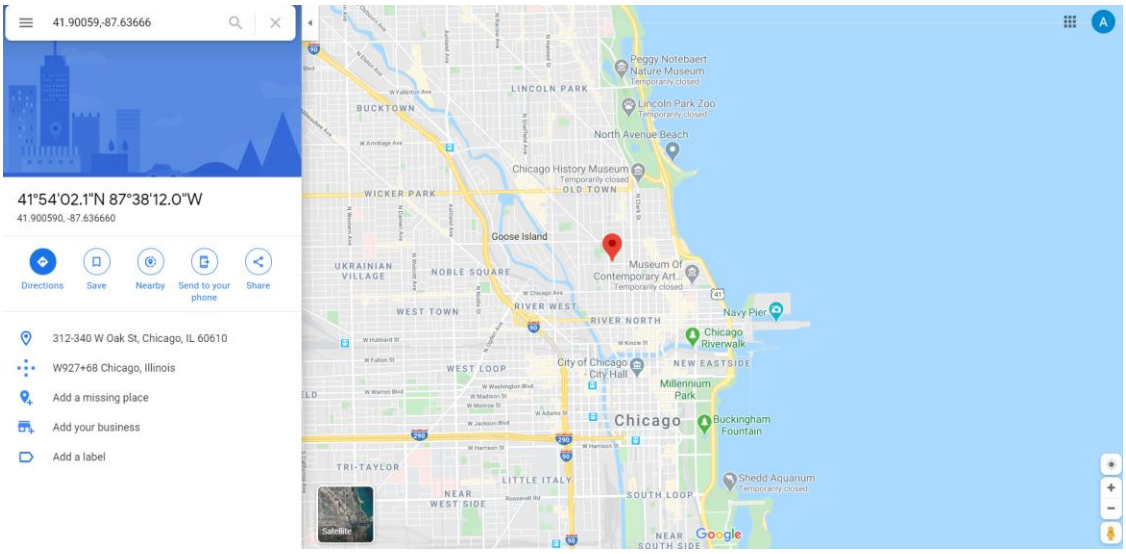
3. Cluster three describes stations that have average rate of bookings with mean trip duration that is about 20 minutes

From the above details, strategic planning can be made to improve booking rate and duration based on stations IDs.

We can use this data to prioritize which stations should have more number of bike stands. The location which are high in demand (more bookings made - cluster 1) require more number of stations. Cluster 1, to us should be the most important stations. They must have sufficient number of bikes stands as there are more number of bookings from there.

Included in this analysis, is how these clusters are distributed on geographic scale. These cluster have been placed on the Chicago city map to look for any possible patterns.

On mapping these clusters, it can be clearly seen that, cluster 1 which are the highest used stations are situated mostly in the coast area and near key tourism infrastructure such as the Millennium Park, Willis Tower, Navy Pier, Aquarium and zoo. This indicates bikes being used for leisure and tourism activities.



Cluster 1 Centroid

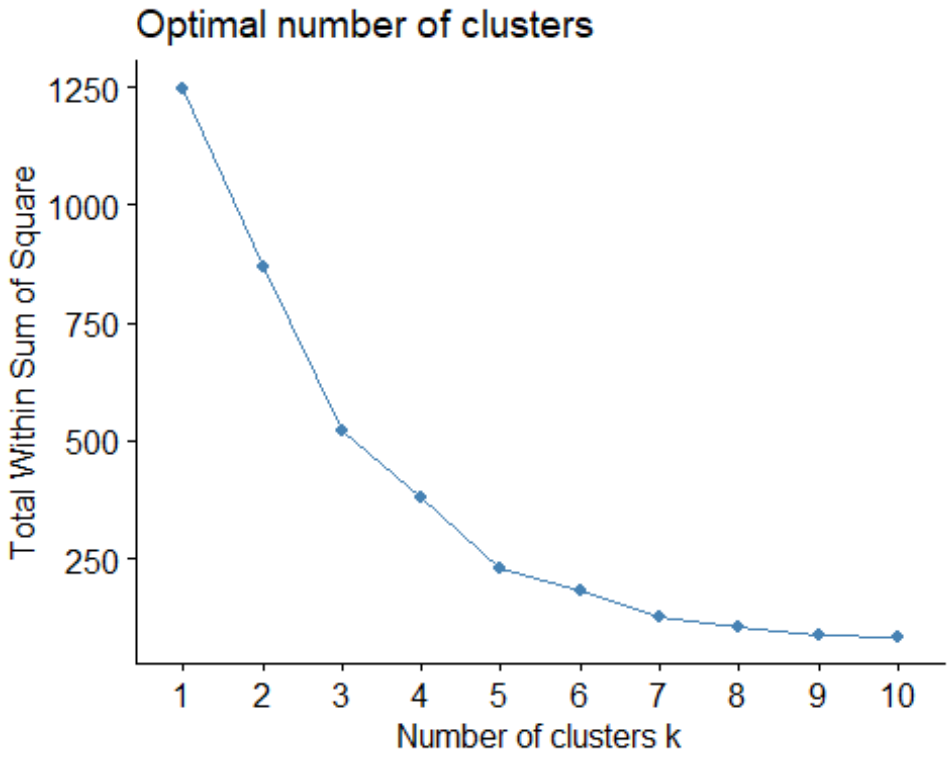
MODEL

```
#DATA WRANGLE
 TidyDivvy_For_Analysis1 <- TidyDivvy_For_Analysis %>%
filter(!is.na(tripduration)) %>% group_by(from_station_id) %>%
summarise(bookingcount=n(),mean_tripduration=mean(tripduration)) %>%
filter(bookingcount>1) #booking filter added to filter stations remove
stations that are being plotted with just 1 record and not mean of trip
duration. There is one station with one booking and very high trip duration
```

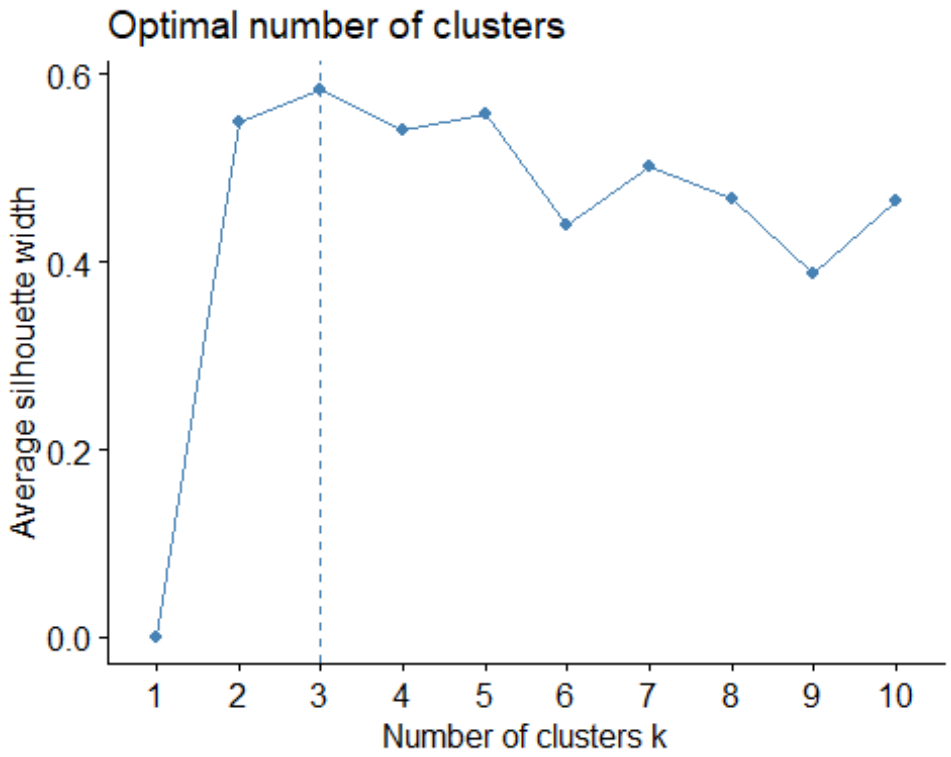
```
that does not appear fair enough to be added to this cluster.

#NORMALIZE
 temp <- na.omit(TidyDivvy_For_Analysis1)
 temp <- temp %>% tibble::column_to_rownames(var="from_station_id")
 temp <- scale(temp)

#CLUSTER
 k2 <- kmeans(temp, centers = 3, nstart = 50)
#VISUALIZE CLUSTER
 fviz_cluster(k2, data = temp,geom="point")
```


```
#SUMMARISE CLUSTER
  str(k2)

## List of 9
##  $ cluster     : Named int [1:624] 1 1 1 3 1 3 3 3 3 1 ...
##   ..- attr(*, "names")= chr [1:624] "2" "3" "4" "5" ...
##  $ centers     : num [1:3, 1:2] 1.526 -0.714 -0.385 -0.225 7.043 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:3] "1" "2" "3"
##   .. ..$ : chr [1:2] "bookingcount" "mean_tripduration"
##  $ totss       : num 1246
##  $ withinss    : num [1:3] 198 137 185
##  $ tot.withinss: num 520
##  $ betweenss   : num 726
##  $ size        : int [1:3] 127 7 490
```

```
##  $ iter        : int 2
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

#ELBOW
 fviz_nbclust(temp, kmeans, method = "wss")
```



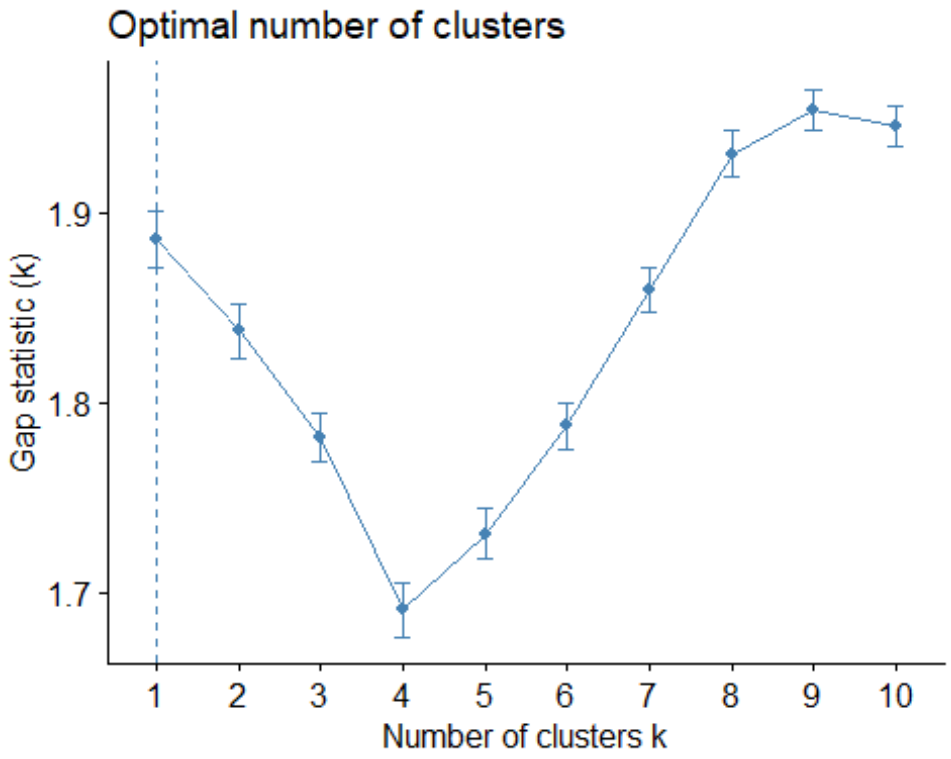

```
#SILOUETTE
 fviz_nbclust(temp, FUNcluster=kmeans, method = "silhouette")
```




```
#GAP
 gap_stat <- clusGap(temp, FUN = kmeans, nstart = 25,K.max = 10, B = 50)
  print(gap_stat, method = "firstmax")
```

```
## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = temp, FUNcluster = kmeans, K.max = 10, B = 50, nstart = 25)
## B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
##  --> Number of clusters (method 'firstmax'): 1
##          logW   E.logW      gap     SE.sim
##  [1,] 5.289347 7.175068 1.885721 0.01452056
##  [2,] 4.974507 6.812006 1.837499 0.01418343
##  [3,] 4.855003 6.636135 1.781132 0.01280421
##  [4,] 4.778404 6.468898 1.690494 0.01423916
##  [5,] 4.607281 6.338042 1.730760 0.01298193
##  [6,] 4.444184 6.231627 1.787444 0.01246843
##  [7,] 4.298466 6.157385 1.858919 0.01173629
##  [8,] 4.158722 6.089446 1.930724 0.01196464
##  [9,] 4.070908 6.024720 1.953812 0.01063354
## [10,] 4.020420 5.965594 1.945174 0.01042507
```

```
  fviz_gap_stat(gap_stat)
```




```
#SUMMARISE
TidyDivvy_For_Analysis1 %>%
  mutate(Cluster = k2$cluster) %>%
  group_by(Cluster) %>%

summarise(stations=n(),mean_tripduration=mean(mean_tripduration),mean_booking
_count=mean(bookingcount))

## # A tibble: 3 x 4
##   Cluster stations mean_tripduration mean_booking_count
##     <int>    <int>             <dbl>              <dbl>
## 1       1      127              16.9              5000.
## 2       2        7             173.                11
## 3       3      490              20.8               744.

#VIEW CLUSTER AND STATION DATA
TidyDivvy_For_Analysis1 %>%  mutate(Cluster = k2$cluster) %>%
left_join(TidyDivvy_For_Analysis,by=c("from_station_id"="from_station_id"))
%>% distinct(from_station_name,from_station_id,From_Loc_GPS,Cluster)

## # A tibble: 717 x 4
##    from_station_name       from_station_id From_Loc_GPS
Cluster
##    <chr>                             <dbl> <chr>
<int>
##  1 Buckingham Fountain                   2 (41.87651122881695, -87.6205~
1
```

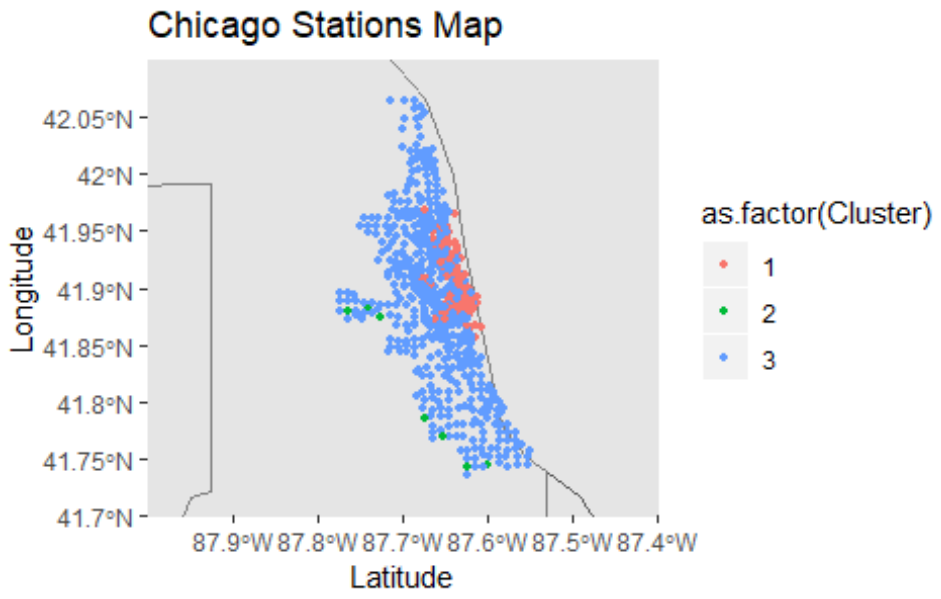
```
##  2 Michigan Ave & Balbo A~                2 (41.87651122881695, -87.6205~
1
##  3 Buckingham Fountain (T~                2 (41.87651122881695, -87.6205~
1
##  4 Shedd Aquarium                         3 (41.86722595682, -87.6153553~
1
##  5 Burnham Harbor                         4 (41.856268, -87.613348)
1
##  6 State St & Harrison St                 5 (41.874053, -87.627716)
3
##  7 Dusable Harbor                         6 (41.886976, -87.612813)
1
##  8 Field Blvd & South Wat~                7 (41.88634906269, -87.6175165~
3
##  9 Leavitt St & Archer Ave                9 (41.82879201994, -87.6806044~
3
## 10 Jeffery Blvd & 71st St                11 (41.76663823695, -87.5764501~
3
## # ... with 707 more rows
```

```
#SHOW GPS DATA NO MAP TO SEE IF THESE CLUSTER HAVE ANY SIGNIFICANT
RELATIONSHIP WITH CHICAGO'S INFRASTRUCTURE

#GET FULL MAP
worldmap <- ne_countries(scale = "medium", returnclass = "sf")

#GET STATIONID BY CLUSTER
clustered_station <- TidyDivvy_For_Analysis1 %>%  mutate(Cluster =
k2$cluster,as.factor(Cluster)) %>% select(from_station_id,Cluster)

#LOAD STATION DATA AND JOIN WITH CLUSTERS
Stations <- read_csv("Data\\Divvy_Bicycle_Stations.csv")
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   `Station Name` = col_character(),
##   `Total Docks` = col_double(),
##   `Docks in Service` = col_double(),
##   Status = col_character(),
##   Latitude = col_double(),
##   Longitude = col_double(),
##   Location = col_character()
## )
```

```
Stations <- Stations %>% select(ID,Latitude,Longitude)
plot_station <- clustered_station %>% left_join(Stations,by=
c("from_station_id"="ID"))
plot_station <-  plot_station %>% na.omit()
```

```r
#GET STATE DATA
states <- st_as_sf(map("state", plot = FALSE, fill = TRUE))
states <- cbind(states, st_coordinates(st_centroid(states)))

#GET COUNTY DATA
counties <- st_as_sf(map("county", plot = FALSE, fill = TRUE))
counties <- subset(counties, grepl("illinois", counties$ID))
counties$area <- as.numeric(st_area(counties))

#PLOT MAP
ggplot(data = worldmap) +
    geom_sf() +
    geom_sf(data = states, fill = NA) +
    geom_text(data = states, aes(X, Y, label = ID), size = 5) +
    geom_sf(data = counties, fill = NA, color = gray(.5)) +
    geom_point(data = plot_station, aes(x = Longitude, y=
Latitude,color=as.factor(Cluster)), size = 1) +
    coord_sf(xlim = c(-88, -87.4), ylim = c(42.1, 41.7), expand = FALSE)+
    ggtitle("Chicago Stations Map")+
    xlab("Latitude") +
    ylab("Longitude")
```




```r
#Cluster Epicenter
plot_station %>% group_by(Cluster) %>%
summarise(Latitude=mean(Latitude),Longitude=mean(Longitude))
```

```
## # A tibble: 3 x 3
##   Cluster Latitude Longitude
##     <int>    <dbl>     <dbl>
## 1       1     41.9     -87.6
## 2       2     41.8     -87.7
## 3       3     41.9     -87.7
```

ANALYSIS 4.1

CLASSIFY STATIONS BASED ON GPS, CONSIDERING CLUSTER 1 FROM ABOVE ANALYSIS AS CORE STATIONS FOR OUR BUSINESS AND CLUSTER 2,3 AS SUPPORT STATIONS(NON-CORE). THIS WILL HELP CLASSIFY ANY FUTURE STATIONS THAT MAY BE PLANNED INTO EITHER CORE OR SUPPORT STATION.

Note

This is a continued exploration from the previous analysis. We are classifying stations falling under cluster 2 and 3 from the previous analysis as one group. Cluster 1 will be another class, the class that is more important to our business, let us call it core stations class.

In this analysis we are trying to create a model to classify stations into either Core stations or Support stations class. This model will help us decide to which class will a future station be assigned based on it's GPS co-ordinates.

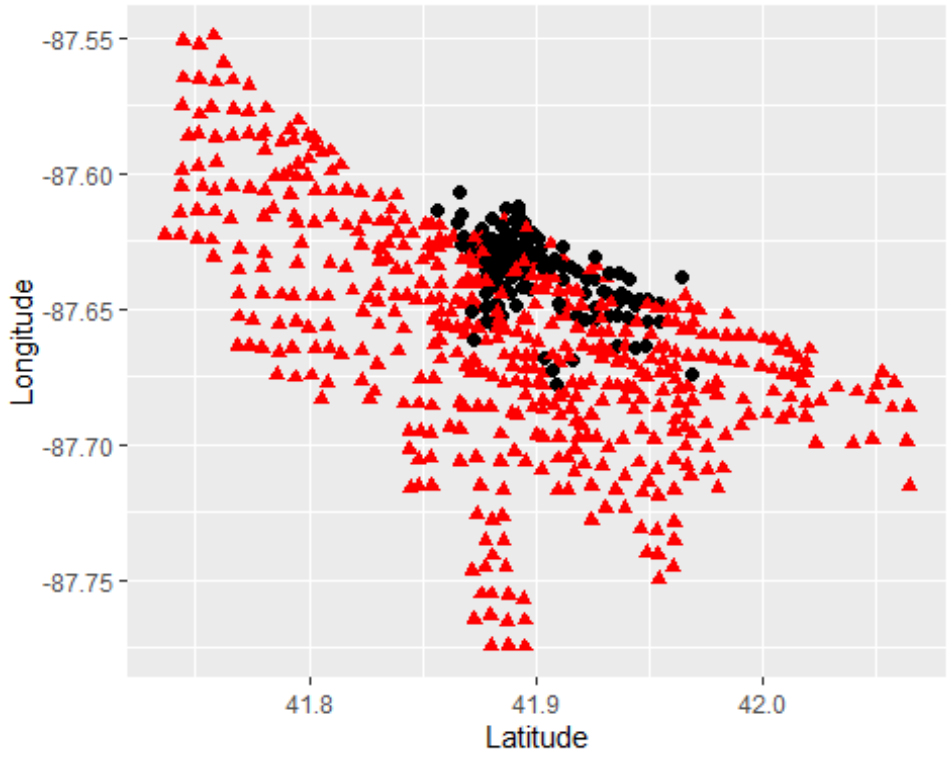
This will help us decide things such as, Imagine a scenario that a new station setup is being planned by the CEO or other Executive and it;s GPS is falling in Core stations class, then we need to make sure that station has more number of bike stands, to ensure the customer demand.

From the results we can observe that our model is able to classify the station into Core (1) and Support (2) classes with an error percentage of 6%, which shows that our model is doing a good job. So, in the future if Divvy bikes is planning to expand and increase it bike stations in Chicago, they can use this model to make appropriate decisions.

MODEL

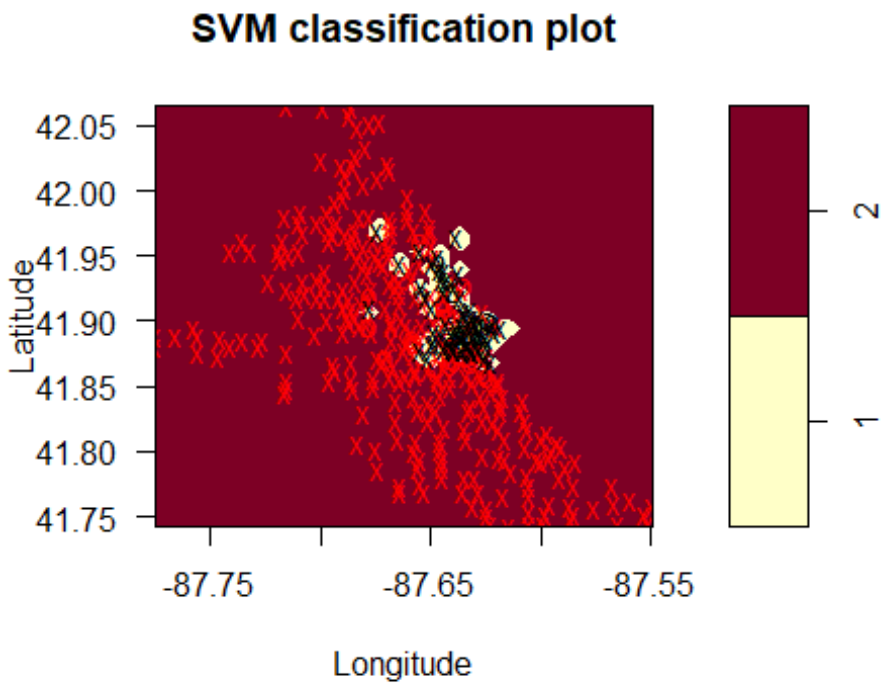
```
#DATA WRANGLE
stations_classify <- plot_station %>% select(Cluster,Latitude,Longitude) %>%
mutate(Cluster=ifelse(Cluster==3,2,Cluster)) %>%
mutate(Cluster=as.factor(Cluster))

#PLOT TO SEE CLUSTERS
ggplot(data = stations_classify, aes(x = Latitude, y = Longitude, color =
```

```
Cluster, shape = Cluster)) +
  geom_point(size = 2) +
  scale_color_manual(values=c("#000000", "#FF0000","#0000FF")) +
  theme(legend.position = "none")
```




```
#SAMPLE
set.seed(9999)
sample <- sample(c(TRUE, FALSE), nrow(stations_classify), replace = T, prob =
c(0.6,0.4))
train <- stations_classify[sample, ]
test <- stations_classify[!sample, ]

#FIND THE BEST COST
tune.out <- tune(svm, Cluster~., data = train, kernel = "radial",
                 ranges = list(cost = c(0.1,1,10,100,1000),
                 gamma = c(0.5,1,2,3,4)))
tune.out$best.model

##
## Call:
## best.tune(method = svm, train.x = Cluster ~ ., data = train, ranges =
list(cost = c(0.1,
##     1, 10, 100, 1000), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  106

#RUN MODEL
svmfit <- svm(Cluster~., data = train, kernel = "radial", gamma =140 , cost =
10)

#PLOT MODEL
plot(svmfit, train)
```




```
#TEST MODEL PRECTION
ypred <- predict(svmfit, test)
(misclass <- table(predict = ypred, truth = test$Cluster))

##        truth
## predict   1   2
##       1  27   9
##       2  27 172

#ERROR PERCENT
(overall_error <- (misclass[1,2]+misclass[2,1]) / ( misclass[1,1] +
misclass[2,2] ) )

## [1] 0.1809045
```

Conclusion

When we first started off with this dataset, we had a vague idea of what our results will look like. We had some predefined thoughts as to how the outcome of the analysis would look like, such as thinking temperature would have a serious effect on each individual booking and also expected to see that the booking patterns varied for different user types when we joined the external data. Post analysis we understood, not all that we assumed came true, we are shocked to see some results such as snow depth having a positive correlation with booking count, temperature not being a major factor in influencing individual bookings. There were some interesting results that changed our mindset on how important it is to run analysis to completely understand the scenario based out of true facts in the data.

Overall, it has been more of challenge than expected to work on this dataset, which at first glance gives a false impression that external factors such as temperature obviously effect individual bookings. The journey of molding data to understand it's true meaning has exposed us to depths that wouldn't have been possible without selection of such a challenging dataset. It has not only allowed us to learn modeling from an academic standpoint, but took us a level higher and put us in the shoes of real world data scientist to work our way through the entire analysis process. In the end, we are glad that we were able to explore and explain some very interest finding from the dataset and create models that predict the outcome with a good accuracy and precision.

References:

Divvy Data:

https://divvy-tripdata.s3.amazonaws.com/index.html

Weather Data:

https://w2.weather.gov/climate/xmacis.php?wfo=cle

Other Sources:

https://rstudio.com/ https://rstudio.com/

Thank you!

DIVVY TEAM