Cyclistic Coursera Case Study

Rintaro Inoue

2023-11-27

Cyclistic

1 Preguntar / Ask

About the Company

A bike-sharing program that includes 5,800 bicycles and 600 stations. Cyclistic stands out for also offering recumbent bikes, hand tricycles, and cargo bikes, providing a more inclusive use of shared bikes for people with disabilities and cyclists who cannot use a standard two-wheeled bike.

Buisness Strategy

Design marketing strategies aimed at converting occasional cyclists (hereinafter referred to as customers) into annual members (hereinafter referred to as subscribers), achieving a better understanding of how to differentiate between annual members and occasional cyclists, why occasional cyclists would purchase a membership, and how digital media could impact their marketing tactics. We need to analyze historical bike trip data from Cyclistic to identify trends.

Stakeholders

Lily Moreno: The Marketing Director and your manager. Moreno is responsible for developing campaigns and initiatives to promote the bike-sharing program. Campaigns may include email, social media, and other channels.

Cyclistic Marketing Data Computational Analysis Team: A team of data analysts responsible for collecting, analyzing, and reporting data that helps drive Cyclistic's marketing strategy. You joined this team six months ago and have not only familiarized yourself with Cyclistic's mission and business goals but also explored how you can contribute to Cyclistic's success from your position as a junior data analyst.

Cyclistic Executive Team: The highly detail-oriented executive team will decide whether to approve the recommended marketing program.

2 - Prepare

Source Description

The data comes from the first quarter of 2019. Additional data could not be loaded due to technical limitations of the computer used for the task.

Dataset Description:

1 CSV file containing 1 table with 13 columns named "Divvy_Trips_2019_Q1.csv".

Credibility and Data Integrity:

(Note: The datasets have different names because Cyclistic is a fictional company. For the purposes of this case study, the datasets are appropriate and will enable you to answer the company's questions. The data

has been provided by Motivate International Inc. under this license.) These are public data that you can use to explore how different types of customers use Cyclistic bikes. However, for data privacy reasons, you are prohibited from using personally identifiable information of the cyclists. This means you cannot link pass purchases to credit card numbers to determine if occasional cyclists live in the Cyclistic service area or if they purchased multiple single-ride passes.

Loading Libraries library(tidyverse)
library(lubridate)
library(janitor)
library(dplyr)
library(tidyr)
library(skimr)
library(ggplot2)
library(patchwork)

Importing data Divvy_Trips_2019_Q1 <- read_csv("Divvy_Trips_2019_Q1.csv")
Divvy_Trips_2019_Q2 <- read_csv("Divvy_Trips_2019_Q2.csv")
Divvy_Trips_2019_Q3 <- read_csv("Divvy_Trips_2019_Q3.csv")
Divvy_Trips_2019_Q4 <- read_csv("Divvy_Trips_2019_Q4.csv")

```
library(readr)
library(dplyr)
library(lubridate)
Divvy_Trips_2019_Q1 <- read_csv("Divvy_Trips_2019_Q1.csv")
```

We verify the proper loading of data

```
## Rows: 365069 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): from_station_name, to_station_name, usertype, gender
## dbl  (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num  (1): tripduration
## dttm (2): start_time, end_time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

3- Data Cleaning

We proceed with cleaning the data

```
library(readr)
 colSums(is.na(Divvy_Trips_2019_Q1))
```

We verify which columns has a null value

```
##           trip_id        start_time          end_time            bikeid
##                 0                 0                 0                 0
##      tripduration   from_station_id from_station_name     to_station_id
##                 0                 0                 0                 0
##   to_station_name          usertype            gender         birthyear
##                 0                 0             19711             18023
```

We verify that the columns with null values are the gender and year columns. Therefore, we can proceed

We standardize the column names to lowercase. Divvy_Trips_2019_Q1 <- rename_with(Divvy_Trips_2019_Q1,to

4- Analize

```
library(reader)
```

Creation of a table with average start times for each day of the week for subscribers.

```
## Loading required package: NCmisc
```

```
##
## Attaching package: 'reader'
```

```
## The following objects are masked from 'package:NCmisc':
##
##     cat.path, get.ext, rmv.ext
```

```
mean_time_Subscriber <- subset(Divvy_Trips_2019_Q1, usertype=='Subscriber')
# Assuming your DataFrame is mean_time_Subscriber and start_time is in datetime format
mean_time_Subscriber$start_time <- ymd_hms(mean_time_Subscriber$start_time)

# Extract weekday, hour, and minute information
mean_time_Subscriber <- mean_time_Subscriber %>%
  mutate(
    weekday = wday(start_time, label = TRUE),
    hour = hour(start_time),
    minute = minute(start_time)
  )

# Group by weekday and calculate mean hour and minute
mean_time_by_weekday <- mean_time_Subscriber %>%
  group_by(weekday) %>%
  summarize(
    mean_hour = round(mean(hour)),
    mean_minute = round(mean(minute))
  )

# Combine mean_hour and mean_minute into a single column
mean_time_by_weekday$mean_time_combined <- sprintf("%02d:%02d", mean_time_by_weekday$mean_hour, mean_ti

# Display the result

mean_time_Subs <-mean_time_by_weekday[, c("weekday", "mean_time_combined")]
```

```
library(reader)

mean_time_Customer <- subset(Divvy_Trips_2019_Q1, usertype=='Customer')
# Assuming your DataFrame is mean_time_Customer and start_time is in datetime format
mean_time_Customer$start_time <- ymd_hms(mean_time_Customer$start_time)
```

```r
# Extract weekday, hour, and minute information
mean_time_Customer <- mean_time_Customer %>%
  mutate(
    weekday = wday(start_time, label = TRUE),
    hour = hour(start_time),
    minute = minute(start_time)
  )

# Group by weekday and calculate mean hour and minute
mean_time_by_weekday <- mean_time_Customer %>%
  group_by(weekday) %>%
  summarize(
    mean_hour = round(mean(hour)),
    mean_minute = round(mean(minute))
  )

# Combine mean_hour and mean_minute into a single column
mean_time_by_weekday$mean_time_combined <- sprintf("%02d:%02d", mean_time_by_weekday$mean_hour, mean_ti

# Display the result

mean_time_Cust <-mean_time_by_weekday[, c("weekday", "mean_time_combined")]
```

Creation of a table with average start times for each day of the week for customers

Comparison of the average start time between customers and subscribers

```r
library(reader)
print(mean_time_Cust)
```

```
## # A tibble: 7 x 2
##   weekday mean_time_combined
##   <ord>   <chr>
## 1 Sun     14:29
## 2 Mon     14:29
## 3 Tue     14:29
## 4 Wed     15:29
## 5 Thu     14:30
## 6 Fri     14:30
## 7 Sat     14:29
```

```r
print(mean_time_Subs)
```

```
## # A tibble: 7 x 2
##   weekday mean_time_combined
##   <ord>   <chr>
## 1 Sun     13:29
## 2 Mon     13:29
## 3 Tue     13:30
## 4 Wed     14:30
## 5 Thu     13:29
## 6 Fri     13:29
## 7 Sat     13:30
```

Including Plots

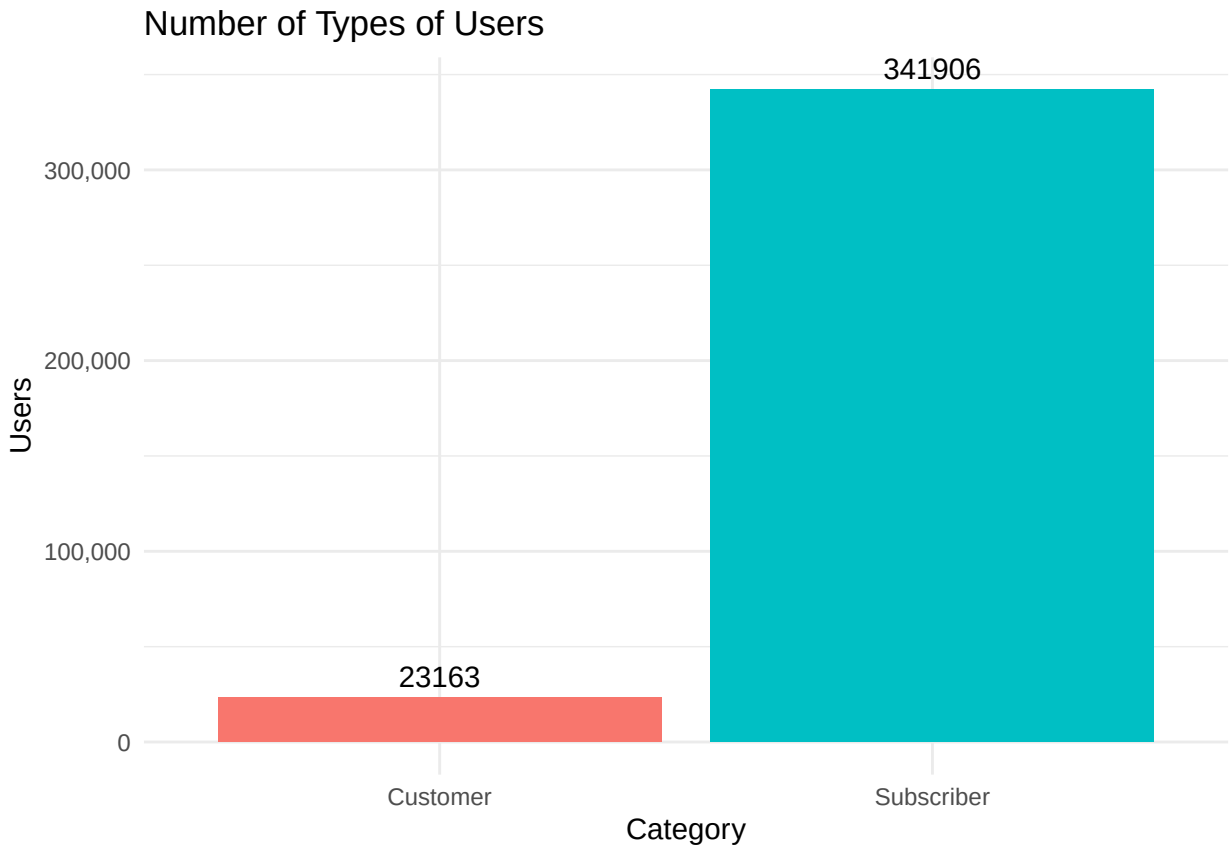
Let's examine the number of subscribers versus the number of customers

```
# Install and load required packages
install.packages(c("ggplot2", "dplyr"))
```

```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
# Load the packages
library(ggplot2)
library(dplyr)

# Example data frame (replace this with your own data frame)
df <- Divvy_Trips_2019_Q1

# Create a bar plot with exact numbers on top of each bar
ggplot(df, aes(x = usertype, fill = usertype)) +
  geom_bar(stat = "count", show.legend = FALSE) +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) +  # Display exact numbers on top
  theme_minimal() +
  labs(title = "Number of Types of Users", x = "Category", y = "Users")+
  scale_y_continuous(labels = scales::comma)  # Format y-axis labels as comma-separated numbers)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



We can observe a clear difference between the number of customer and subscriber users. It represents a 0.06% difference

Visualization of the peak bike usage hour

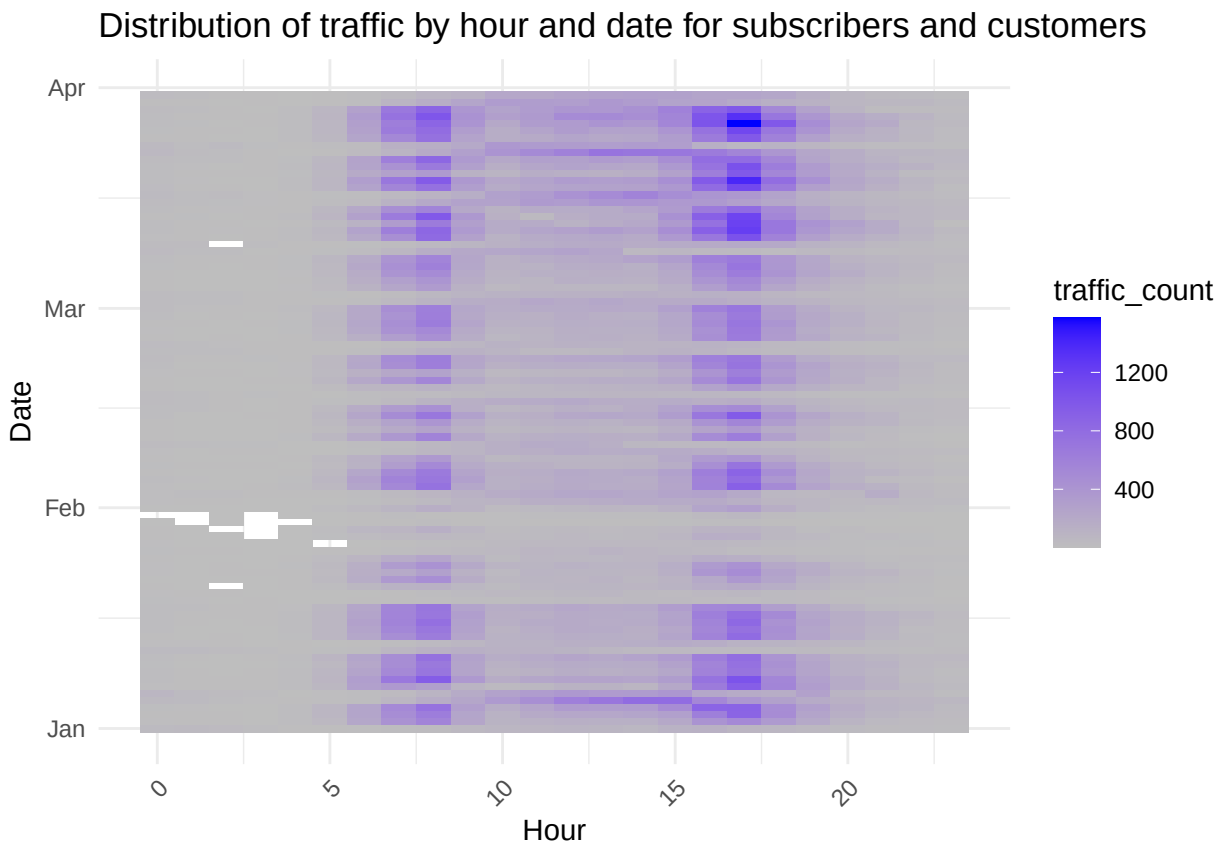
```
# Install and load required packages
install.packages(c("ggplot2", "dplyr", "lubridate"))
```

```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
# Load packages
library(ggplot2)
library(ggplot2)
library(dplyr)

# Sample data frame (replace this with your actual data frame)
mean_time <- Divvy_Trips_2019_Q1
mean_time$start_time <- ymd_hms(mean_time$start_time)

# Extract date and hour from start_time and count occurrences
traffic_counts <- mean_time %>%
  mutate(date = as.Date(start_time),
         hour = hour(start_time)) %>%
  group_by(date, hour) %>%
  summarise(traffic_count = n())
```

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

```
# Create a heatmap
ggplot(traffic_counts, aes(x = hour, y = date, fill = traffic_count)) +
  geom_tile() +
  scale_fill_gradient(low = "grey", high = "blue") +
  theme_minimal() +
  labs(title = "Distribution of traffic by hour and date for subscribers and customers", x = "Hour", y =
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Install and load required packages
install.packages(c("ggplot2", "dplyr", "lubridate"))
```

We can see that the peak hours are in the morning at 8 am and in the afternoon at 5 pm. Let's check if the same pattern holds for subscribers and customers

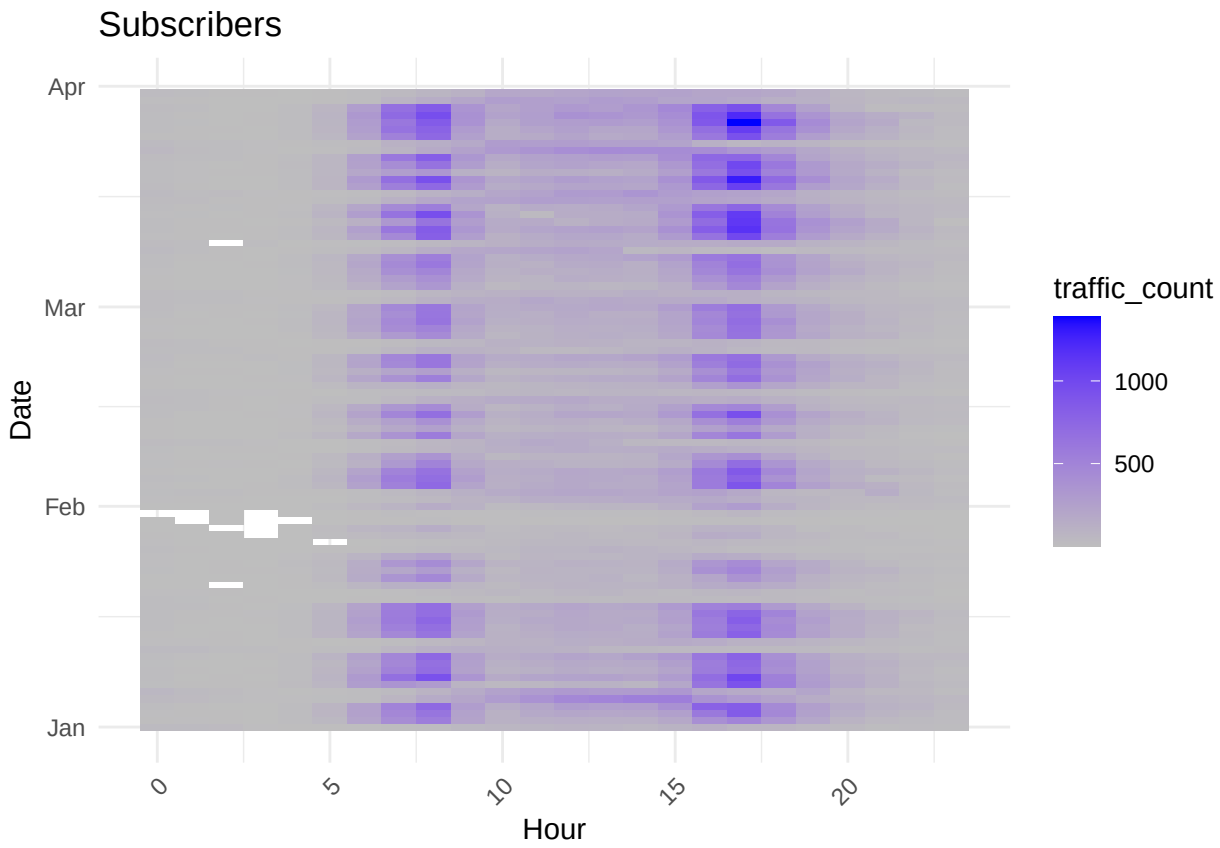
```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```
# Load packages
library(ggplot2)
library(ggplot2)
library(dplyr)

# Sample data frame (replace this with your actual data frame)

mean_time_Subscriber$start_time <- ymd_hms(mean_time_Subscriber$start_time)
```

```r
# Extract date and hour from start_time and count occurrences
traffic_counts <- mean_time_Subscriber %>%
  mutate(date = as.Date(start_time),
         hour = hour(start_time)) %>%
  group_by(date, hour) %>%
  summarise(traffic_count = n())
```

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

```r
# Create a heatmap
ggplot(traffic_counts, aes(x = hour, y = date, fill = traffic_count)) +
  geom_tile() +
  scale_fill_gradient(low = "grey", high = "blue") +
  theme_minimal() +
  labs(title = "Subscribers", x = "Hour", y = "Date") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

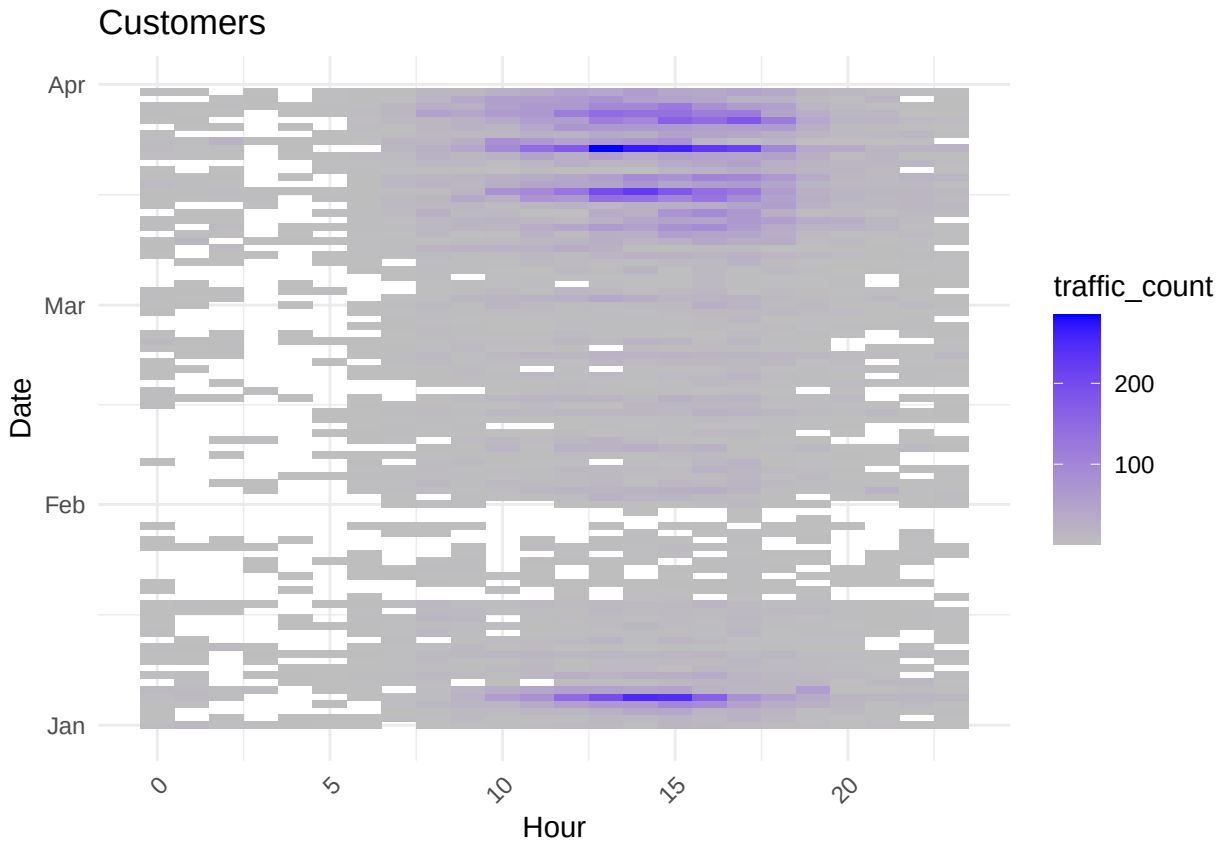



```r
# Sample data frame (replace this with your actual data frame)

mean_time_Customer$start_time <- ymd_hms(mean_time_Customer$start_time)

# Extract date and hour from start_time and count occurrences
traffic_counts <- mean_time_Customer %>%
  mutate(date = as.Date(start_time),
         hour = hour(start_time)) %>%
  group_by(date, hour) %>%
```

```
  summarise(traffic_count = n())
```

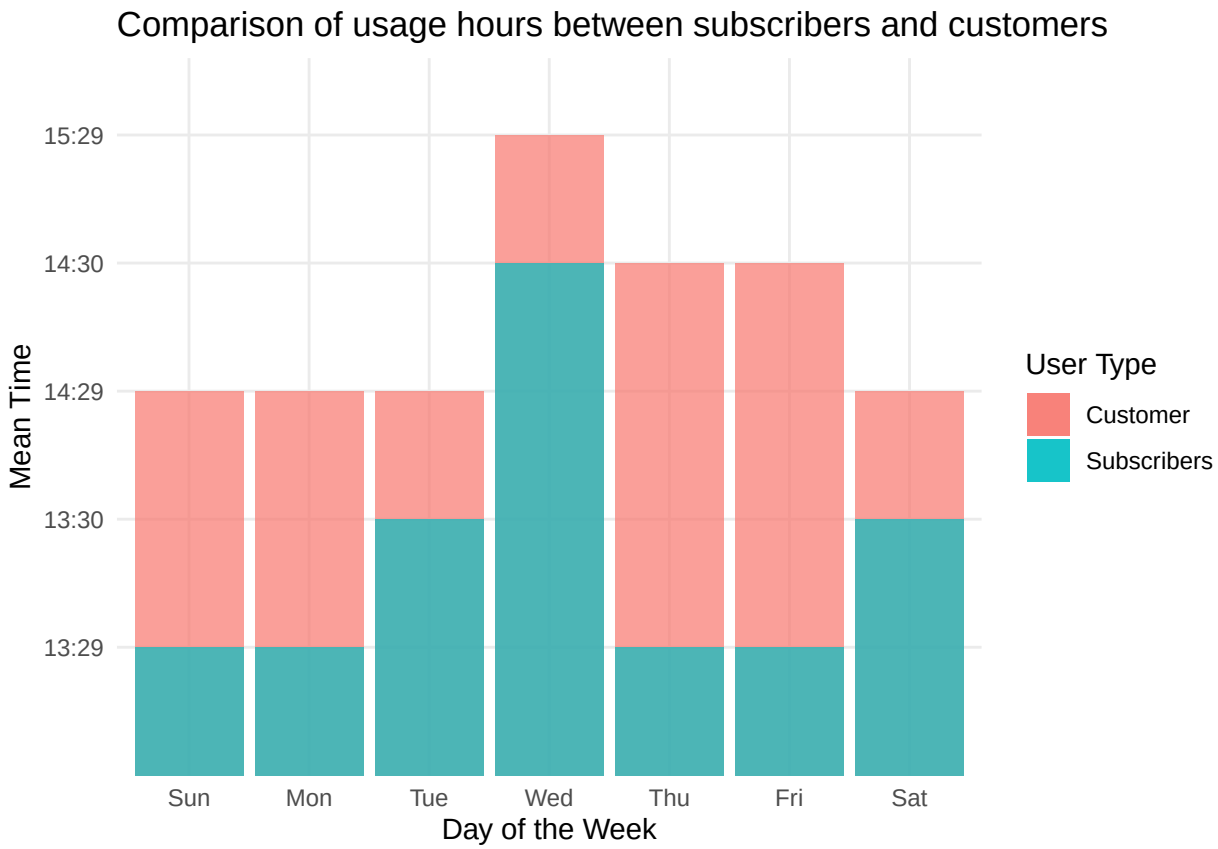
```
## `summarise()` has grouped output by 'date'. You can override using the
## `.groups` argument.
```

```
# Create a heatmap
ggplot(traffic_counts, aes(x = hour, y = date, fill = traffic_count)) +
  geom_tile() +
  scale_fill_gradient(low = "grey", high = "blue") +
  theme_minimal() +
  labs(title = "Customers", x = "Hour", y = "Date") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



We can observe that both for customers and subscribers, the peak traffic hours coincide in the afternoon

Comparison of usage hours between subscribers and customers



It can be observed that for both subscribers and customers, the average usage hours are close

Now we examine the comparison between the number of customers and subscribers using the bike for each day of the week

```
daily_count_subscriber <- mean_time_Subscriber %>%
  mutate(day_of_week = weekdays(start_time),
         day_of_week = factor(day_of_week, levels = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thurs
  count(day_of_week, sort = FALSE)

daily_count_customer <- mean_time_Customer %>%
  mutate(day_of_week = weekdays(start_time),
         day_of_week = factor(day_of_week, levels = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thurs
  count(day_of_week, sort = FALSE)

# Display the resulting table
print(daily_count_subscriber)
```

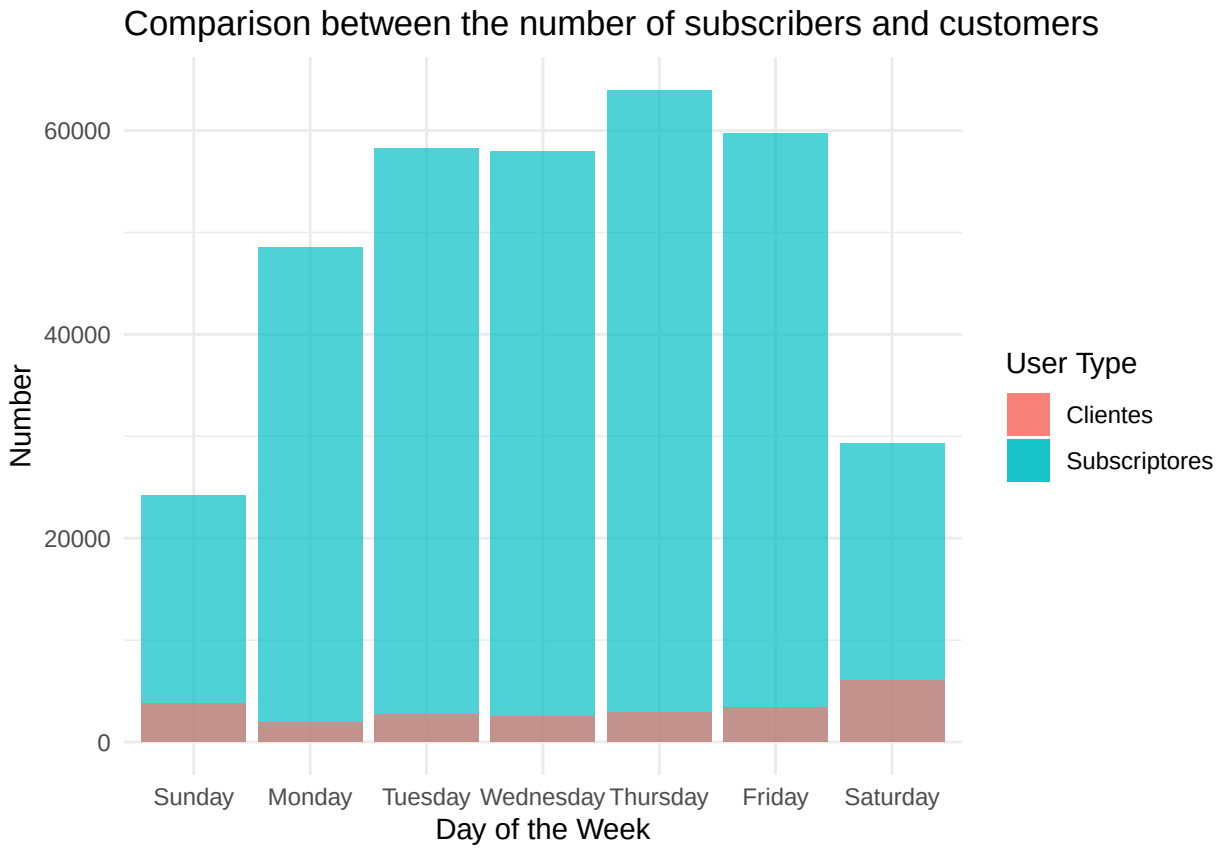
```
## # A tibble: 7 x 2
##   day_of_week     n
##   <fct>       <int>
## 1 Sunday      24233
## 2 Monday      48507
## 3 Tuesday     58277
## 4 Wednesday   57925
```

```
## 5 Thursday    63983
## 6 Friday      59672
## 7 Saturday    29309
```

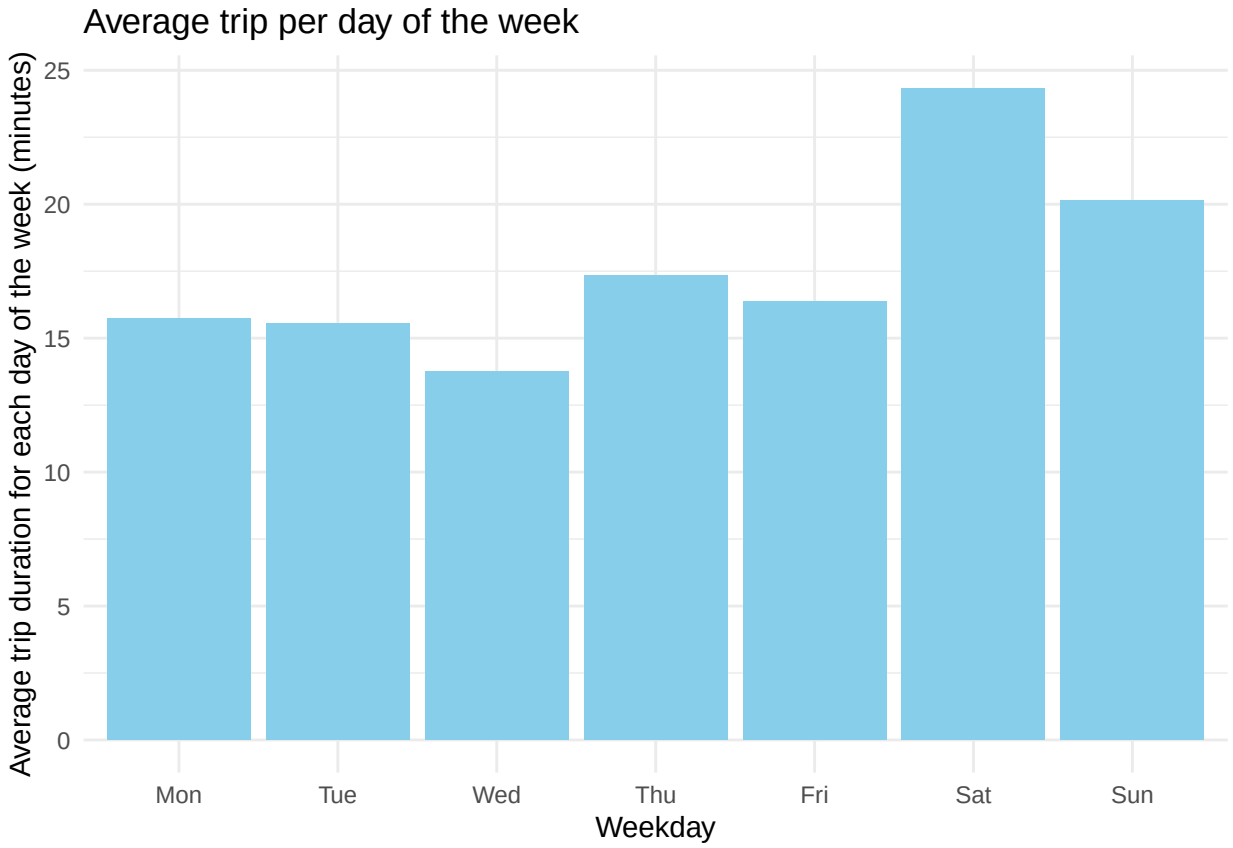
```
print(daily_count_customer)
```

```
## # A tibble: 7 x 2
##   day_of_week     n
##   <fct>       <int>
## 1 Sunday       3766
## 2 Monday       1892
## 3 Tuesday      2728
## 4 Wednesday    2489
## 5 Thursday     2920
## 6 Friday       3375
## 7 Saturday     5993
```

```
library(ggplot2)
plot <- ggplot() +
   geom_bar(data = daily_count_subscriber, aes(x = day_of_week, y = n, fill = "Subscriptores"),
            stat = "identity", position = "dodge", alpha = 0.7) +
  geom_bar(data = daily_count_customer, aes(x = day_of_week, y = n, fill = "Clientes"),
            stat = "identity", position = "dodge", alpha = 0.7) +

  labs(title = "Comparison between the number of subscribers and customers",
       x = "Day of the Week",
       y = "Number",
       fill = "User Type") +
  theme_minimal()

# Print the plot
print(plot)
```




```
# Convert the 'start_time' column to a date and time object.
df$start_time <- ymd_hms(df$start_time)

# Extract the day of the week and calculate the average trip duration
df_avg <- df %>%
  mutate(weekday = wday(start_time, label = TRUE, week_start = 1)) %>%
  group_by(weekday) %>%
  summarise(avg_duration = mean(tripduration/60))

# Create a bar chart.
ggplot(df_avg, aes(x = weekday, y = avg_duration)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme_minimal() +
  labs(title = "Average trip per day of the week", x = "Weekday", y = "Average trip duration for each da
```



6- Conclusions

- Due to machine capacity issues, only the first quarter of 2019 could be analyzed. All observations and conclusions are based on this period.
- Despite the fact that the number of customers represents only 0.06% of subscriber users, it is still crucial to attract more subscribers to the platform.
- As expected, users use bikes during the morning and afternoon, coinciding with the times of entering and leaving work.
- On average, users (both customers and subscribers) spend between 10-25 minutes per trip.
- There is a slight increase in usage time on weekends.

Recommendations:

- It is recommended to limit bike usage during peak hours for customers.
- Consider limiting bike usage during the weekend for customer users.
- This approach encourages appreciation and value for being subscribers to the platform, offering access whenever needed and for the required duration.
- Discounts can be offered to both users during days and hours with lower usage (Wednesday and during hours from 9 pm to 6 am).