Caso practico Cyclistic

Rintaro Inoue

2023-11-27

Cyclistic

1 Preguntar / Ask

Sobre la compaña

Un programa de bicicletas compartidas que incluye 5,800 bicicletas y 600 estaciones. Cyclistic se destaca por ofrecer también bicicletas reclinadas, triciclos manuales y bicicletas de carga que ofrecen un uso más inclusivo de las bicicletas compartidas para las personas con discapacidad y los ciclistas que no pueden utilizar una bicicleta estándar de dos ruedas

Tarea empresarial

Diseñar estrategias de marketing orientadas a convertir a los ciclistas ocasionales (de ahora en mas lo llamaremos clientes) en miembros anuales (de ahora en mas lo llamaremos subscriptores), logrando mejor cómo difrenciar a los miembros anuales y los ciclistas ocasionales, por qué los ciclistas ocasionales comprarían una membresía y cómo los medios digitales podrían afectar sus tácticas de marketing. Necesitamos analizar los datos históricos de viajes en bicicleta de Cyclistic para identificar tendencia.

Intersados

Lily Moreno: La directora de marketing y tu gerente. Moreno es responsable del desarrollo de campañas e iniciativas para promover el programa de bicicletas compartidas. Las campañas pueden incluir correo electrónico, redes sociales y otros canales.

Equipo de análisis computacional de datos de marketing de Cyclistic: Un equipo de analistas de datos que se encargan de recopilar, analizar e informar datos que ayudan a conducir la estrategia de marketing de Cyclistic. Te incorporaste a este equipo hace seis meses y te has dedicado no solo a conocer la misión y las metas de negocios de Cyclistic, sino también a ver cómo puedes ayudar a Cyclistic a lograrlo, desde tu posición de analista de datos júnior.

Equipo ejecutivo de Cyclistic: El equipo ejecutivo, sumamente detallista, decidirá si aprueba el programa de marketing recomendado.

2 - Preparar

Descripcion de la fuente

Los datos provienen del primer cuatrimestre de 2019. No se pudo cargar mas datos por la limitacion tecnica de la computadora con la que se trabajo.

Descripcion del conjunto de datos

1 archivos csv con 1 tabla de 13 columnas Divvy_Trips_2019_Q1.csv

Credibilidad e integridad de los datos:

(Nota: Los conjuntos de datos tienen un nombre diferente porque Cyclistic es una empresa ficticia. A los fines de este caso práctico, los conjuntos de datos son apropiados y te permitirán responder las preguntas de la empresa. Los datos han sido proporcionados por Motivate International Inc. bajo esta licencia.) Estos son datos públicos que puedes usar para explorar cómo difieren los tipos de clientes que usan las bicicletas Cyclistic. Sin embargo, ten en cuenta que, por cuestiones de privacidad de los datos, se te prohíbe usar información de identificación personal de los ciclistas. Esto significa que no podrás conectar las compras de pases con los números de tarjetas de crédito para determinar si los ciclistas ocasionales viven en el área de servicio de Cyclistic o si compraron varios pases de un solo viaje.

Cargamos Librerias library(tidyverse)
library(lubridate)
library(janitor)
library(dplyr)
library(tidyr)
library(skimr)
library(ggplot2)
library(patchwork)

Importamos los datos 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")
```

Comprobamos la correcta carga de datos

```
## 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- Limpieza de datos

Procedemos con la limpieza de datos

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

Verificamos que columnas contiene nulos

```
##           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
```

Comprobamos que las columnas que tienen null son las columnas de genero y ano. Por lo tanto podemos proceder

Estandarizamos el nombre de las columnas a minusculas Divvy_Trips_2019_Q1 <- rename_with(Divvy_Trips_2019_Q1,tolower)

4- Analizar

```
library(reader)
```

Creacion de una tabla con promedios de hora de inicio por cada dia de la semana de los subsciptores

```
## 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")]
```

```r
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)

# 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")]
```

Creacion de una tabla con promedios de hora de inicio por cada dia de la semana de los clientes
Comparacion de la media de hora de inicio entre clinetes y subsciptores

```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

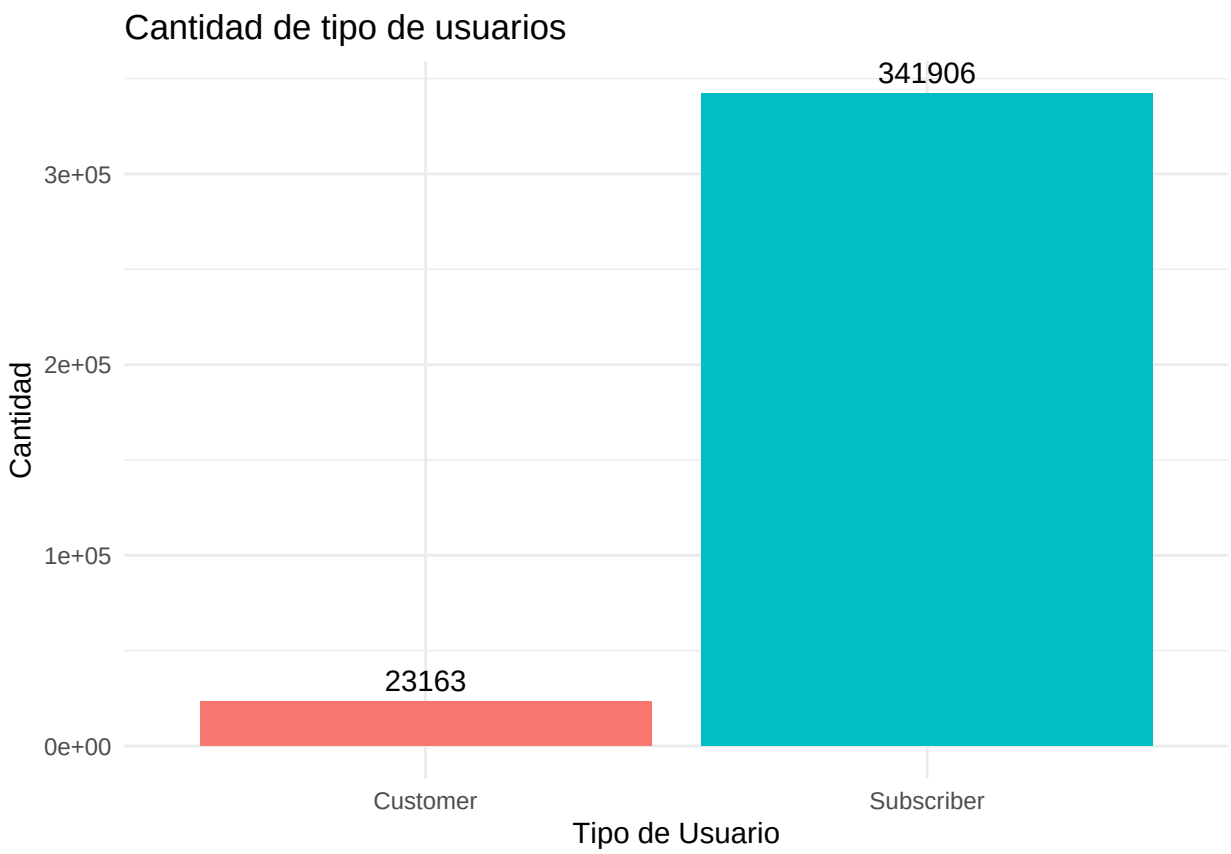
Veamos la cantidad de subscriptores vs la cantidad de clientes

```
# 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 = "Cantidad de tipo de usuarios", x = "Tipo de Usuario", y = "Cantidad")
```

```
## 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.
```



Podemos ver una clara diferencia entre la cantidad de usuarios clientes y subscriptores. Representa un 0.06%.

Visualizacion de la hora con mas uso de bicicletas

```
# 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 = "Distribucion de traficos por hora y fecha para subscriptores y clientes", x = "Hour", y
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Vemos que la hora pico se da a la mañana a las 8am y a la tarde a las 5pm. Vamos a ver si se da el mismo patron para los suscriptores y clientes.

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

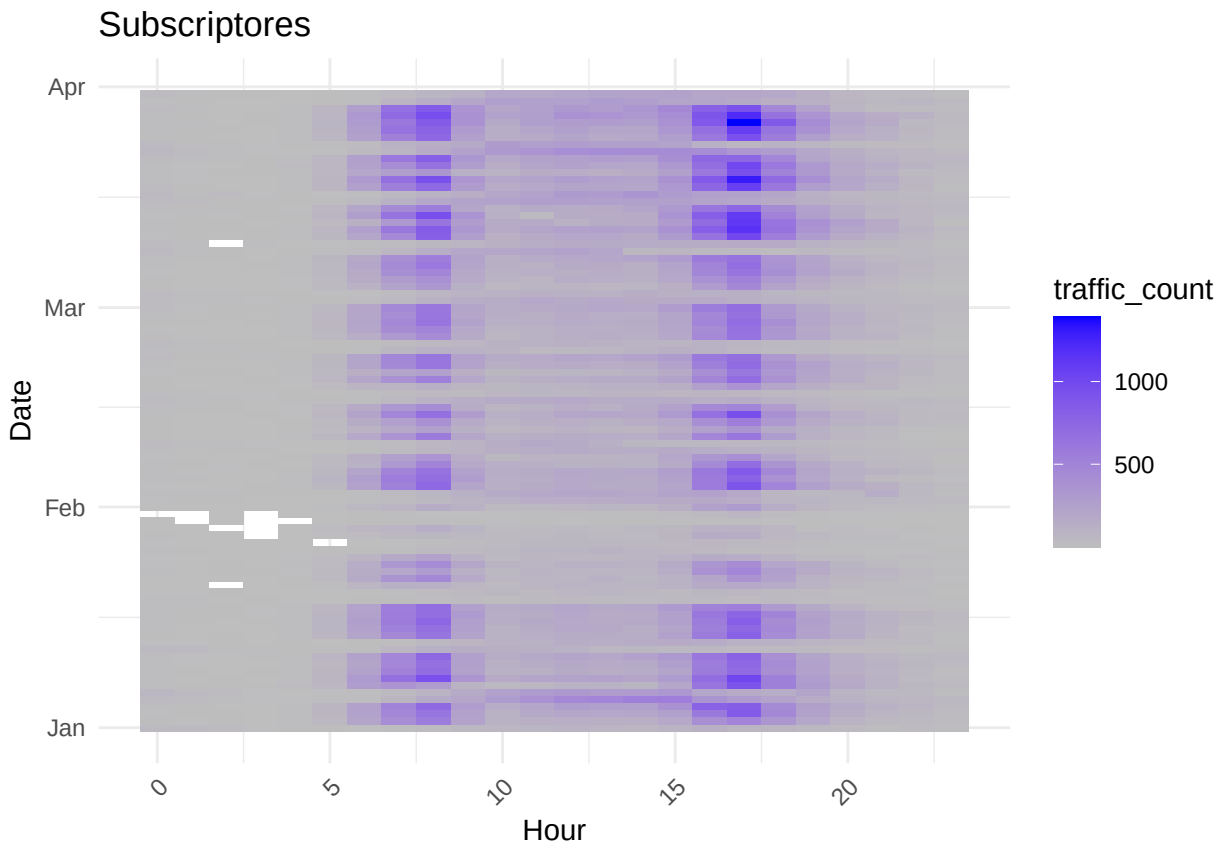
```
## Installing packages into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

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

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

mean_time_Subscriber$start_time <- ymd_hms(mean_time_Subscriber$start_time)

# Extract date and hour from start_time and count occurrences
```

```r
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.
```

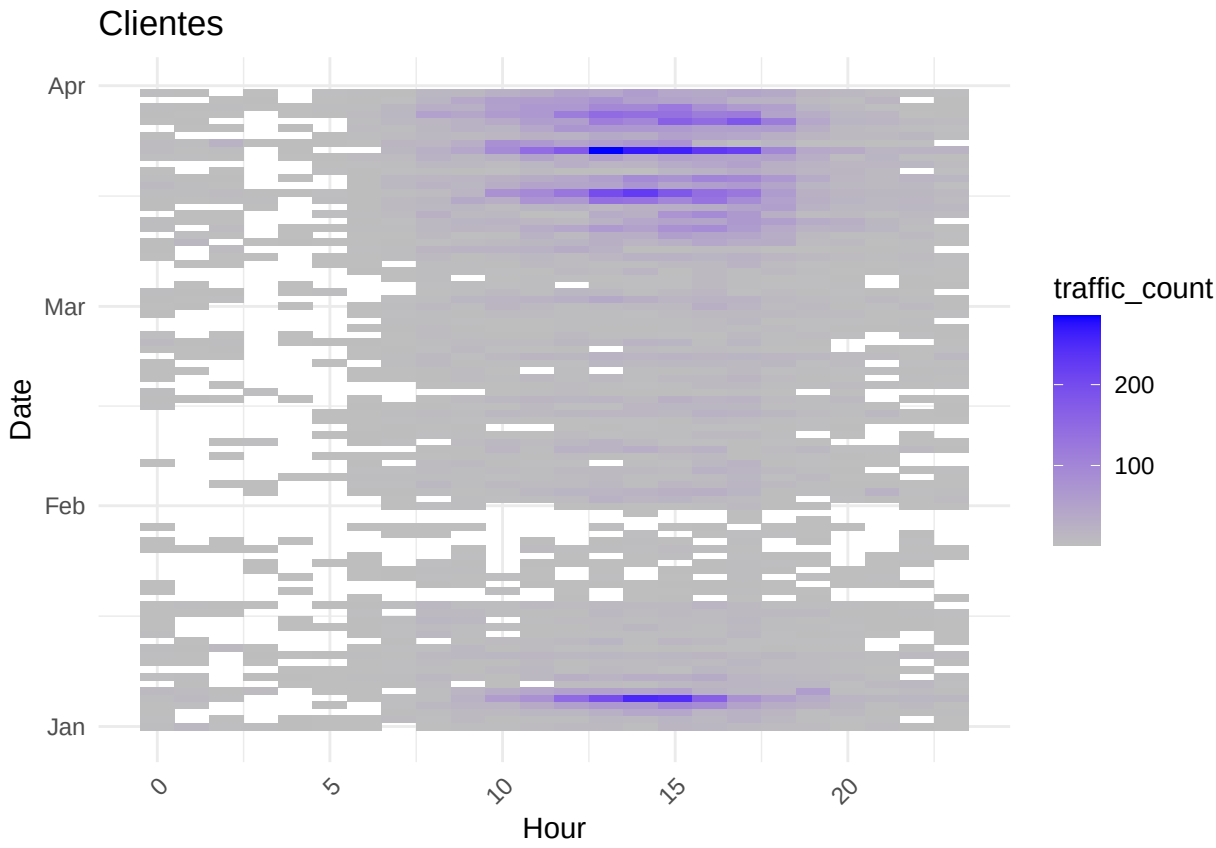
```r
# Create a heatmap
ggplot(traffic_counts, aes(x = hour, y = date, fill = traffic_count)) +
  geom_tile() +
  scale_fill_gradient(low = "grey", high = "blue") +
  theme_minimal() +
  labs(title = "Subscriptores", x = "Hour", y = "Date") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))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




```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())
```

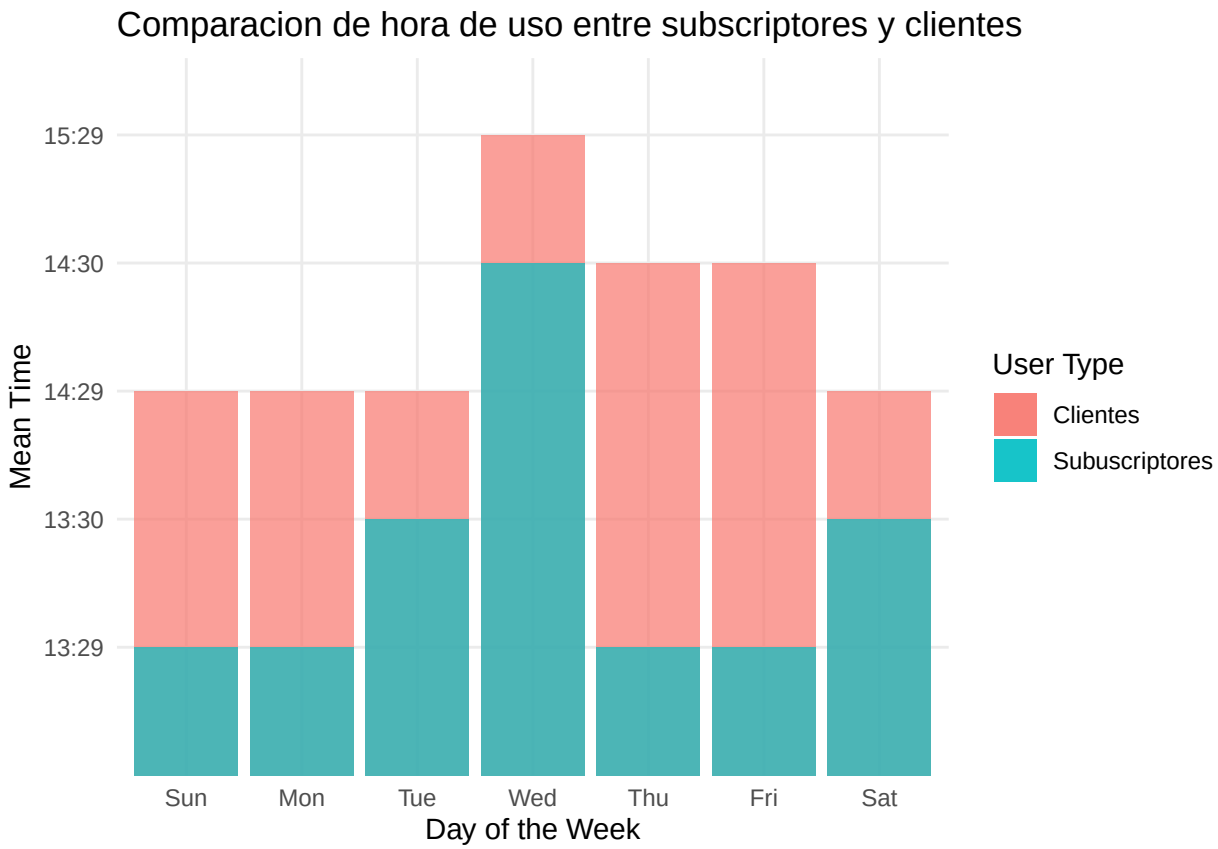
```
## `summarise()` has grouped output by 'date'. You can override using the
## `.groups` argument.
```

```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 = "Clientes", x = "Hour", y = "Date") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Podemos ver que tanto para clientes y subscriptores la hora de trafico coinciden para la tarde.

Comparacion de hora de uso entre subscriptores y clientes



Se puede observar que tanto como para subscriptores y para clientes, el horario promedio de uso son cercanos.

Ahora vemos la comparacion entre la cantidad de clientes y subscriptores que usan la bicicleta por cada dia de la semana.

```
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)
```

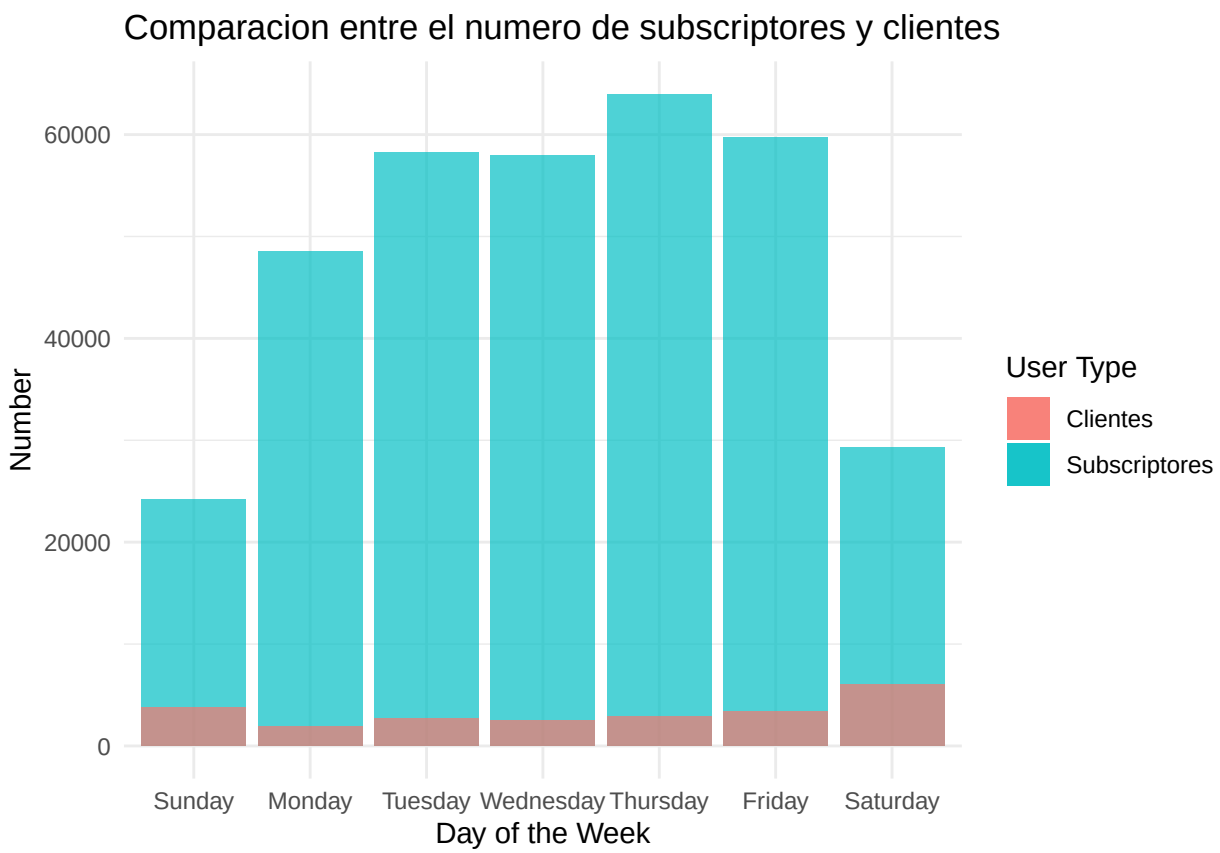
```
## # A tibble: 7 x 2
##   day_of_week     n
##   <fct>       <int>
## 1 Sunday      24233
## 2 Monday      48507
## 3 Tuesday     58277
## 4 Wednesday   57925
```

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

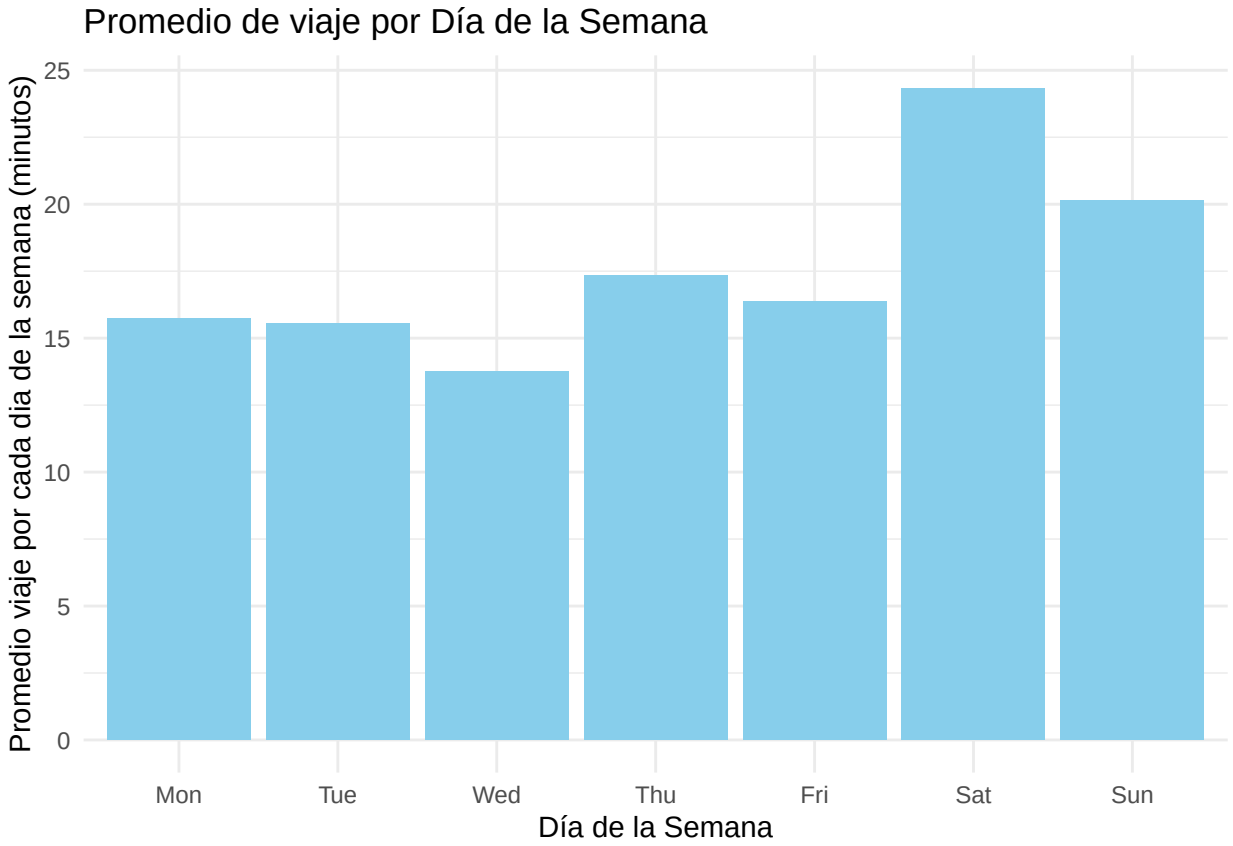
```
print(daily_count_customer)
```

```
## # 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 = "Comparacion entre el numero de subscriptores y clientes",
       x = "Day of the Week",
       y = "Number",
       fill = "User Type") +
  theme_minimal()

# Print the plot
print(plot)
```




```r
# Convertir la columna start_time a un objeto de fecha y hora
df$start_time <- ymd_hms(df$start_time)

# Extraer el día de la semana y calcular el promedio de tripduration
df_avg <- df %>%
  mutate(weekday = wday(start_time, label = TRUE, week_start = 1)) %>%
  group_by(weekday) %>%
  summarise(avg_duration = mean(tripduration/60))

# Crear un gráfico de barras
ggplot(df_avg, aes(x = weekday, y = avg_duration)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  theme_minimal() +
  labs(title = "Promedio de viaje por Día de la Semana", x = "Día de la Semana", y = "Promedio viaje po
```



6- Conclusiones

- Por cuestiones de la capacidad de la maquina solo se pudo analizar el primero cuatrimestre de 2019. Toda obvservacion y conclusion se basaron en este periodo.
- Mas alla de que la cantidad de clientes representa el 0.06% de los usuarios subscriptores sigue siendo fundamental tratar de atraer a mayor cantidad de subuscriptores a la plataforma
- Como fue de esperar, lo usuarios usan las bicicletas durante la mañana y a la tarde que coinicide con los horarios de entrada y salida de trabajo.
- El promedio de usuarios (clientes y subscriptores) usan entre 10 -25 min por viaje.
- Hay un leve aumento del tiempo de uso en los fines de semana

Recomendaciones

- Se recomienda limitar el uso de las bicicletas durante las horas pico para los clientes
- Tambien se puede optar por limitar el uso de las bicicletas durante el fin de semana para aquellos usuarios que son clientes
- De esta forma se consigue una apreciacion y valorizacion por ser subscriptores de la plataforma para acceder cuando quiere y por el tiempo que precisa.
- Se puede ofrecer descuentos para ambos usuarios en dias y horarios con menor uso (miercoles y en horarios despues de las 9pm hasta las 6am)