Case Study 1

2023-06-04

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

Scenario

You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

Characters and teams

- Cyclistic: A bike-share program that features more than 5,800 bicycles and 600 docking stations. Cyclistic sets itself apart by also offering reclining bikes, hand tricycles, and cargo bikes, making bike-share more inclusive to people with disabilities and riders who can't use a standard two-wheeled bike. The majority of riders opt for traditional bikes; about 8% of riders use the assistive options. Cyclistic users are more likely to ride for leisure, but about 30% use them to commute to work each day.
- Lily Moreno: The director of marketing and your manager. Moreno is responsible for the development of campaigns and initiatives to promote the bike-share program. These may include email, social media, and other channels.
- Cyclistic marketing analytics team: A team of data analysts who are responsible for collecting, analyzing, and reporting data that helps guide Cyclistic marketing strategy. You joined this team six months ago and have been busy learning about Cyclistic's mission and business goals — as well as how you, as a junior data analyst, can help Cyclistic achieve them.
- Cyclistic executive team: The notoriously detail-oriented executive team will decide whether to approve the recommended marketing program.

About the company

In 2016, Cyclistic launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system anytime

Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members.

Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance to convert casual riders into members.

She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs.

Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Ask

Under stand the difference between casual and member usage

- How do annual members and casual riders use Cyclistic bikes differently?
- Why would casual riders buy Cyclistic annual memberships?
- How can Cyclistic use digital media to influence casual rider to become members?

Prepare data

Data sources used

- The dataset using in this case study is Cyclistic trip data under the license by Motivate International Inc.

The datasets used in this case study are divided into four files, group by the quarter of the specific year. so it's needed to be combine into a single file

Process

In this case study, I'll will be utilizing the R programming language to conduct the analysis. R is chosen for its proficiency in handling large datasets efficiently and providing extensive control over data manipulation.

To ensure the cleanliness and analysis readiness of the data, it's necessary to perform two key steps. Firstly, the data sets need to be merged or combined. This consolidation process brings together relevant data from multiple sources into a unified data set. Secondly, it may be essential to adjust the data types of specific values to enable accurate calculations. By appropriately converting these values, they can be processed effectively for further analysis.

The following steps is done by Rstudio

Step 1: Get data set

Install requiremnet packages

Install these packages for convenience in data manipulation.

```
options(repos = structure(c(CRAN = "https://cran.rstudio.com/")))
install.packages("tidyverse")
```

```
##
## The downloaded binary packages are in
##  /var/folders/19/c4kdq8fs3374pjnv1kx5vmzr0000gn/T//RtmpvoAqJG/downloaded_packages
```

```
install.packages("lubridate")
```

```
##
## The downloaded binary packages are in
##  /var/folders/19/c4kdq8fs3374pjnv1kx5vmzr0000gn/T//RtmpvoAqJG/downloaded_packages
```

```
install.packages("ggplot2")
```

```
##
## The downloaded binary packages are in
##  /var/folders/19/c4kdq8fs3374pjnv1kx5vmzr0000gn/T//RtmpvoAqJG/downloaded_packages
```

To load and make packages available for extend functionality using the following commands.

```
library(tidyverse)  #helps wrangle data
library(lubridate)  #helps wrangle date attributes
library(ggplot2)  #helps visualize data
```

Set the working directory and store the datasets

To load data sets from external storage. In this case, the data sets are stored in my personal computer, To check and set a working directory use the following functions.

```
getwd() #to check the working directory
```

```
## [1] "/Users/skk/Desktop"
```

```
setwd("/Users/skk/Desktop/Data Analyst/Cyclists data/csv") #to set the working directory

# Store datasets in variables to join them later
q1_2019 <- read_csv("Divvy_Trips_2019_Q1.csv")
q2_2019 <- read_csv("Divvy_Trips_2019_Q2.csv")
q3_2019 <- read_csv("Divvy_Trips_2019_Q3.csv")
q4_2019 <- read_csv("Divvy_Trips_2019_Q4.csv")
```

Check column names for each file

```
colnames(q1_2019)
```

```
##  [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"
```

```
colnames(q2_2019)
```

```
##  [1] "01 - Rental Details Rental ID"
##  [2] "01 - Rental Details Local Start Time"
##  [3] "01 - Rental Details Local End Time"
##  [4] "01 - Rental Details Bike ID"
##  [5] "01 - Rental Details Duration In Seconds Uncapped"
##  [6] "03 - Rental Start Station ID"
##  [7] "03 - Rental Start Station Name"
##  [8] "02 - Rental End Station ID"
##  [9] "02 - Rental End Station Name"
## [10] "User Type"
## [11] "Member Gender"
## [12] "05 - Member Details Member Birthday Year"
```

```
colnames(q3_2019)
```

```
##  [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"
```

```
colnames(q4_2019)
```

```
##  [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"
```

The 'q2_2019' file contains column names that differ from the other datasets. To facilitate the merging of these datasets, it's essential to rename the column to match across all files.

rename q2 to match the rest of the files

```
q2_2019 <- rename(q2_2019
                  ,trip_id = "01 - Rental Details Rental ID"
                  ,start_time = "01 - Rental Details Local Start Time"
                  ,end_time = "01 - Rental Details Local End Time"
                  ,bikeid = "01 - Rental Details Bike ID"
                  ,tripduration = "01 - Rental Details Duration In Seconds Uncapped"
                  ,from_station_id = "03 - Rental Start Station ID"
                  ,from_station_name = "03 - Rental Start Station Name"
                  ,to_station_id = "02 - Rental End Station ID"
                  ,to_station_name = "02 - Rental End Station Name"
                  ,usertype = "User Type"
                  ,gender = "Member Gender"
                  ,birthyear = "05 - Member Details Member Birthday Year")
```

Concatenate rows into a single dataframe

Merging the data sets into a unified data set

```
all_trips <- bind_rows(q1_2019, q2_2019, q3_2019, q4_2019)
```

Step 2: Clean up and prepare data for analysis

Inspect new table

```
colnames(all_trips) # List of column names
```

```
##  [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"
```

```
nrow(all_trips) # Number of rows in dataframe
```

```
## [1] 3818004
```

```
dim(all_trips) # Dimension of the dataframe
```

```
## [1] 3818004      12
```

```
head(all_trips) # Inspect the first six rows
```

```
## # A tibble: 6 x 12
##    trip_id start_time          end_time            bikeid tripduration
##      <dbl> <dttm>              <dttm>               <dbl>        <dbl>
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07   2167          390
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34   4386          441
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12   1524          829
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28    252         1783
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56   1170          364
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09   2437          216
## # i 7 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender <chr>,
## #   birthyear <dbl>
```

```
summary(all_trips) # Statistical summary of data
```

```
##     trip_id           start_time
##  Min.   :21742443   Min.   :2019-01-01 00:04:37.00
##  1st Qu.:22873787   1st Qu.:2019-05-29 15:49:26.50
##  Median :23962320   Median :2019-07-25 17:50:54.00
##  Mean   :23915629   Mean   :2019-07-19 21:47:37.11
```

```
##  3rd Qu.:24963703   3rd Qu.:2019-09-15 06:48:05.75
##  Max.   :25962904   Max.   :2019-12-31 23:57:17.00
##
##     end_time                          bikeid      tripduration
##  Min.   :2019-01-01 00:11:07.00   Min.   :   1   Min.   :      61
##  1st Qu.:2019-05-29 16:09:28.25   1st Qu.:1727   1st Qu.:     411
##  Median :2019-07-25 18:12:23.00   Median :3451   Median :     709
##  Mean   :2019-07-19 22:11:47.56   Mean   :3380   Mean   :    1450
##  3rd Qu.:2019-09-15 08:30:13.25   3rd Qu.:5046   3rd Qu.:    1283
##  Max.   :2020-01-21 13:54:35.00   Max.   :6946   Max.   :10628400
##
##  from_station_id from_station_name  to_station_id   to_station_name
##  Min.   :  1.0   Length:3818004     Min.   :  1.0   Length:3818004
##  1st Qu.: 77.0   Class :character   1st Qu.: 77.0   Class :character
##  Median :174.0   Mode  :character   Median :174.0   Mode  :character
##  Mean   :201.7                      Mean   :202.6
##  3rd Qu.:289.0                      3rd Qu.:291.0
##  Max.   :673.0                      Max.   :673.0
##
##    usertype            gender            birthyear
##  Length:3818004     Length:3818004     Min.   :1759
##  Class :character   Class :character   1st Qu.:1979
##  Mode  :character   Mode  :character   Median :1987
##                                        Mean   :1984
##                                        3rd Qu.:1992
##                                        Max.   :2014
##                                        NA's   :538751
```

Rename Subscriber to member and Customer to casual for readiness of the analysis

```
all_trips <- all_trips %>%
  mutate(usertype = recode(usertype
                           ,"Subscriber" = "member"
                           ,"Customer" = "casual"))
```

Check new table after modified

```
table(all_trips$usertype)
```

```
##
##  casual  member
##  880637 2937367
```

Add columns that list a date, month, day, year of each ride for further calculations.

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

To calculate the duration of a bike ride by adding a 'ride_length' column, perform the following steps

```
all_trips$ride_length <- difftime(all_trips$end_time, all_trips$start_time) #mins

# Convert "ride_length" from Factor to numeric so we can run calculations on the data
is.factor(all_trips$ride_length) # return FALSE value
```

```
## [1] FALSE
```

```
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length)) #convert diftime num -> chr ->
```

Remove "bad" data The dataframe includes a few hundred entries when bikes were taken out of docks and checked for quality by Divvy or ride_length was negative Create a new version of the dataframe (v2) since data is being removed.

```
all_trips_v2 <- all_trips[!(all_trips$from_station_name == "HQ QR" | all_trips$ride_length<0),]
```

STEP 3: Conduct descriptive analysis

Descriptive in ride_length (seconds)

```
mean(all_trips_v2$ride_length) #average duration
```

```
## [1] 24.17443
```

```
median(all_trips_v2$ride_length) #midpoint value in the ascending array values
```

```
## [1] 11.81667
```

```
max(all_trips_v2$ride_length) #longest duration
```

```
## [1] 177200.4
```

```
min(all_trips_v2$ride_length) #shortest duration
```

```
## [1] 1.016667
```

Compare member and casual users

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$usertype, FUN = mean)
```

```
##   all_trips_v2$usertype all_trips_v2$ride_length
## 1                casual                 57.01802
## 2                member                 14.32780
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$usertype, FUN = median)
```

```
##   all_trips_v2$usertype all_trips_v2$ride_length
## 1                casual                 25.83333
## 2                member                  9.80000
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$usertype, FUN = max)
```

```
##   all_trips_v2$usertype all_trips_v2$ride_length
## 1                casual                 177200.4
## 2                member                 150943.9
```

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$usertype, FUN = min)
```

```
##   all_trips_v2$usertype all_trips_v2$ride_length
## 1                casual                 1.016667
## 2                member                 1.016667
```

Arrange and Compare ride_time

```
# Arrange the days of the week in order
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "
# Compare average ride_time for each day
aggregate(all_trips_v2$ride_length ~ all_trips_v2$usertype + all_trips_v2$day_of_week, FUN = mean) # th
```

```
##    all_trips_v2$usertype all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                 casual                   Sunday                 56.18519
## 2                 member                   Sunday                 15.40290
## 3                 casual                   Monday                 54.49989
## 4                 member                   Monday                 14.24928
## 5                 casual                  Tuesday                 57.41328
## 6                 member                  Tuesday                 14.15259
## 7                 casual                Wednesday                 60.33407
## 8                 member                Wednesday                 13.80984
## 9                 casual                 Thursday                 59.95112
## 10                member                 Thursday                 13.77979
## 11                casual                   Friday                 60.17561
## 12                member                   Friday                 13.89748
## 13                casual                 Saturday                 54.06111
## 14                member                 Saturday                 16.30271
```

Analyze ridership data by type and weekday

```
all_trips_v2 %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%  #creates weekday field using wday()
  group_by(usertype, weekday) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n()                           #calculates the number of rides and average
            ,average_duration = mean(ride_length)) %>%      # calculates the average duration
  arrange(usertype, weekday)
```

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

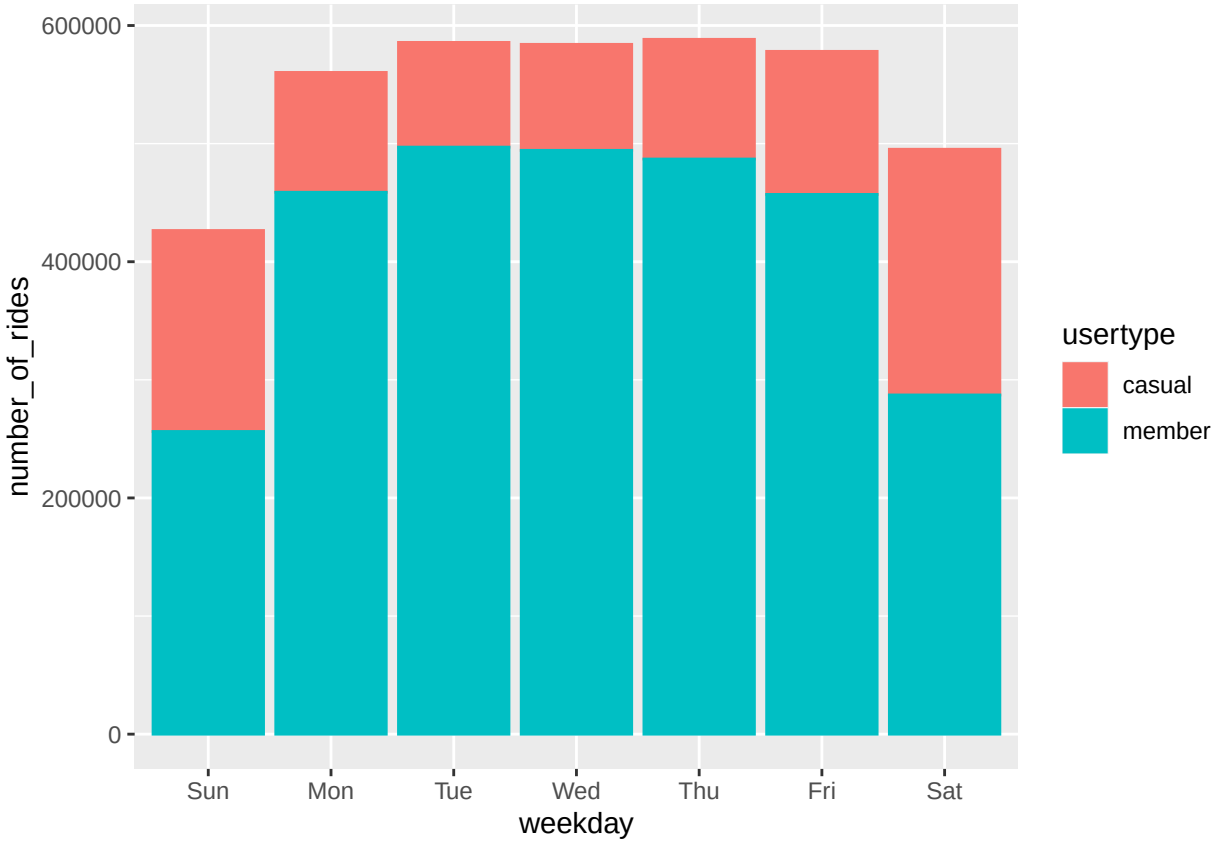
```
## # A tibble: 14 x 4
## # Groups:   usertype [2]
##    usertype weekday number_of_rides average_duration
```

```
##    <chr>    <ord>            <int>           <dbl>
##  1 casual   Sun             170173            56.2
##  2 casual   Mon             101489            54.5
##  3 casual   Tue              88655            57.4
##  4 casual   Wed              89745            60.3
##  5 casual   Thu             101372            60.0
##  6 casual   Fri             121141            60.2
##  7 casual   Sat             208056            54.1
##  8 member   Sun             256234            15.4
##  9 member   Mon             458780            14.2
## 10 member   Tue             497025            14.2
## 11 member   Wed             494277            13.8
## 12 member   Thu             486915            13.8
## 13 member   Fri             456966            13.9
## 14 member   Sat             287163            16.3
```

Visualize the number of rides by usertype

```
all_trips_v2 %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(usertype, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = usertype, color = usertype)) +
  geom_col() +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
```

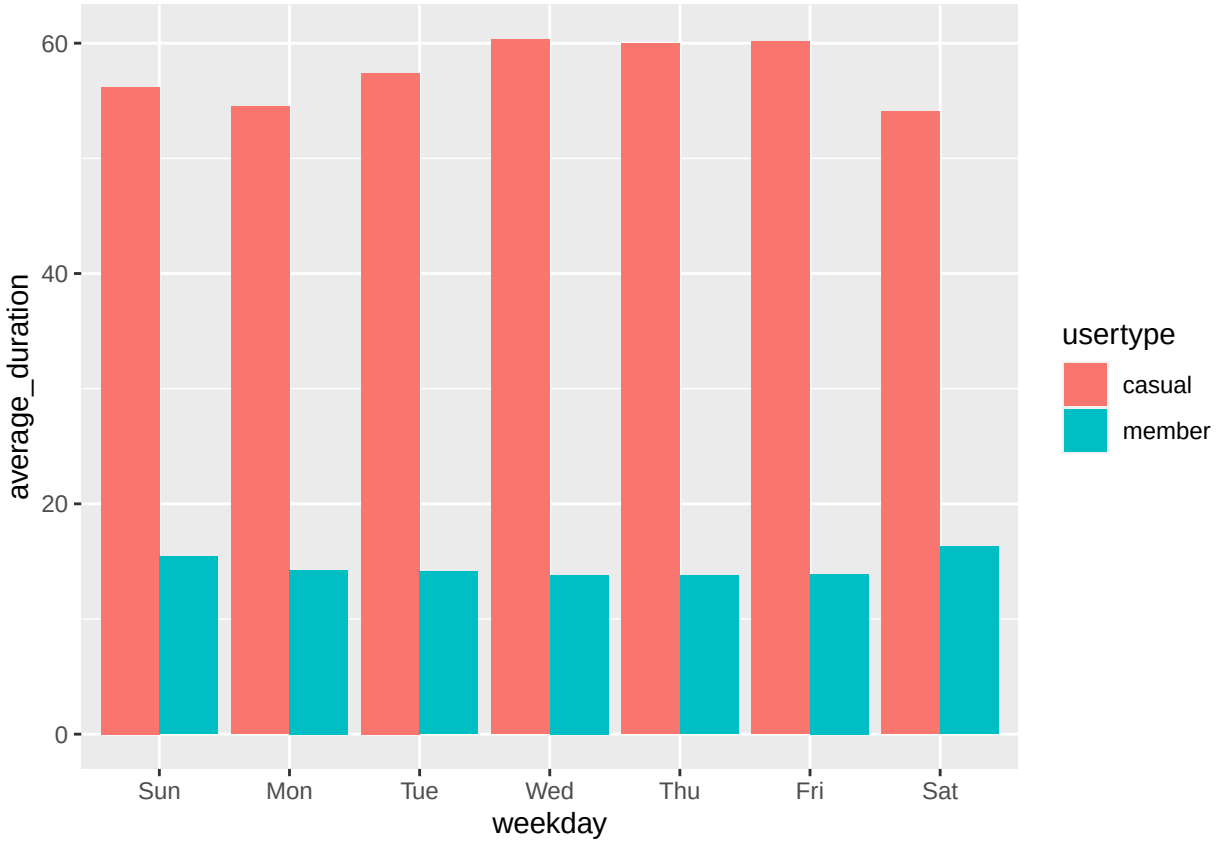
```
## `summarise()` has grouped output by 'usertype'. You can override using the
## `.groups` argument.
```



Visualize average duration

```
all_trips_v2 %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(usertype, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = usertype)) +
  geom_col(position = "dodge")
```

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



Export files to further visualization

```
write.csv(all_trips_v2, file = 'all_trips_dataset.csv', row.names = FALSE)
```