Case Study 1

Vishal Patel

2023-05-24

Which membership tier will be more profitable?

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. However, 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.

Business 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.

Buisness Tasks:

- 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 riders to become members?
- Why would casual riders buy Cyclistic annual memberships?

About the Dataset:

- The dataset consists of 12 files which conisits ride history data of each month
- The data has been made available by Motivate International Inc. under this license
- The Divvy system data is owned by the City ("Data") available to the public, thus it is trustable and has integrity
- Data has been improved every time a data analyst makes recommendations after using it to run analysis

Processing Data:

1. Installing and loading the required libraries

```r
# Installing packages
install.packages("tidyverse")
```

```
##
## The downloaded binary packages are in
##  /var/folders/xx/6sjp9g5j07x1hjzv7g3szwwh0000gp/T//Rtmp5sHr2f/downloaded_packages
```

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

```
##
## The downloaded binary packages are in
##  /var/folders/xx/6sjp9g5j07x1hjzv7g3szwwh0000gp/T//Rtmp5sHr2f/downloaded_packages
```

```r
install.packages("dplyr")
```

```
##
## The downloaded binary packages are in
##  /var/folders/xx/6sjp9g5j07x1hjzv7g3szwwh0000gp/T//Rtmp5sHr2f/downloaded_packages
```

```r
install.packages("readr")
```

```
##
## The downloaded binary packages are in
##  /var/folders/xx/6sjp9g5j07x1hjzv7g3szwwh0000gp/T//Rtmp5sHr2f/downloaded_packages
```

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

```
##
## The downloaded binary packages are in
##  /var/folders/xx/6sjp9g5j07x1hjzv7g3szwwh0000gp/T//Rtmp5sHr2f/downloaded_packages
```

```r
#Loading packages
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.2     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
library(lubridate)
library(ggplot2)
library(dplyr)
library(readr)
```

2. Loading all the data

```
# Creating a list of filenames so that they can be loaded all at once
file_names <- c("202205-divvy-tripdata.csv", "202206-divvy-tripdata.csv", "202207-div
vy-tripdata.csv", "202208-divvy-tripdata.csv",
                "202209-divvy-tripdata.csv", "202210-divvy-tripdata.csv", "202211-div
vy-tripdata.csv", "202212-divvy-tripdata.csv",
                "202301-divvy-tripdata.csv", "202302-divvy-tripdata.csv", "202303-div
vy-tripdata.csv", "202304-divvy-tripdata.csv")
```

```
#Setting working directory
setwd("/Users/MeowMac/Downloads/Case Study/")
```

```
# Create an empty list to store the data frames for each month
fileNames_list <- list()

# Initialize the df_names vector
df_names <- character(length(file_names))

# Initialize the vector to store filenames if they have rows with missing values in e
very column
filenames_with_missing_rows <- character(0)
```

```r
# Loop through each file and load the .csv files

for (i in seq_along(file_names)) {

  # Read the CSV file and assign it to a data frame
  df_names[i] <- paste0("m", sprintf("%02d", 13 - i), "_", gsub("-", "_", substr(file
_names[i], 3, 8)))
  fileNames_list[[df_names[i]]] <- read_csv(file_names[i])

  # Count the number of rows with missing values in all columns
  missing_rows_count <- sum(apply(is.na(fileNames_list[[df_names[i]]]), 1, all))

  # Print the result
  cat("Rows with missing values in all columns in", file_names[i], ":", missing_rows_
count, "\n")

  # Check if missing_rows_count is greater than 0
  if (missing_rows_count > 0) {
    filenames_with_missing_rows <- c(filenames_with_missing_rows, file_names[i])
  }
}
```

```
## Rows: 634858 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202205-divvy-tripdata.csv : 0
```

```
## Rows: 769204 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202206-divvy-tripdata.csv : 0
## Rows: 823488 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202207-divvy-tripdata.csv : 0
```

```
## Rows: 785932 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202208-divvy-tripdata.csv : 0
```

```
## Rows: 701339 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202209-divvy-tripdata.csv : 0
```

```
## Rows: 558685 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
## Rows with missing values in all columns in 202210-divvy-tripdata.csv : 0
```

```
## Rows: 337735 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202211-divvy-tripdata.csv : 0
```

```
## Rows: 181806 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202212-divvy-tripdata.csv : 0
```

```
## Rows: 190301 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202301-divvy-tripdata.csv : 0
## Rows: 190445 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202302-divvy-tripdata.csv : 0
```

```
## Rows: 258678 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202303-divvy-tripdata.csv : 0
```

```
## Rows: 426590 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## dttm (2): started_at, ended_at
##
## 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.
```

```
## Rows with missing values in all columns in 202304-divvy-tripdata.csv : 0
```

3. Combining all data into one data frame

- Checking if column names are same in all files *

```r
# Checking if column names are same in all files before combining the data

for (i in seq_along(fileNames_list)) {

  # Get the dataframe name
  df_name <- names(fileNames_list)[i]

  # Print the dataframe name
  cat("Dataframe:", df_name, "\n")

  # Print the column names
  cat("Column Names:", colnames(fileNames_list[[df_name]]), "\n")

  # Check if column names match with the first dataframe
  if (i > 1) {
    if (!identical(colnames(fileNames_list[[df_name]]), colnames(fileNames_list[[1]])
)) {
      cat("Column names do not match with the first dataframe.\n")
    } else{
      cat("\n \n Column names do match with the first dataframe.\n\n\n")
    }
  }
}
```

```
## Dataframe: m12_2205_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
## Dataframe: m11_2206_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m10_2207_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m09_2208_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
```

```
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m08_2209_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m07_2210_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m06_2211_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m05_2212_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m04_2301_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
```

```
##
## Dataframe: m03_2302_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m02_2303_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
##
##
## Dataframe: m01_2304_d
## Column Names: ride_id rideable_type started_at ended_at start_station_name start_s
tation_id end_station_name end_station_id start_lat start_lng end_lat end_lng member_
casual
##
##
##  Column names do match with the first dataframe.
```

```
# Check was successful, now combining all files into one

all_trips_last_year <- bind_rows(fileNames_list)
```

```
# Calculate the number of null values in each column
null_counts <- colSums(is.na(all_trips_last_year))

# Calculate the percentage of null values in each column
null_percentages <- (null_counts / nrow(all_trips_last_year)) * 100

# Print the number and percentage of null values for each column
for (i in seq_along(null_counts)) {
  cat("Column:", names(null_counts)[i], "\n")
  cat("Number of null values:", null_counts[i], "\n")
  cat("Percentage of null values:", null_percentages[i], "%\n\n")
}
```

```
## Column: ride_id
## Number of null values: 0
## Percentage of null values: 0 %
##
```

```
## Column: rideable_type
## Number of null values: 0
## Percentage of null values: 0 %
##
## Column: started_at
## Number of null values: 0
## Percentage of null values: 0 %
##
## Column: ended_at
## Number of null values: 0
## Percentage of null values: 0 %
##
## Column: start_station_name
## Number of null values: 832009
## Percentage of null values: 14.20038 %
##
## Column: start_station_id
## Number of null values: 832141
## Percentage of null values: 14.20263 %
##
## Column: end_station_name
## Number of null values: 889661
## Percentage of null values: 15.18436 %
##
## Column: end_station_id
## Number of null values: 889802
## Percentage of null values: 15.18677 %
##
## Column: start_lat
## Number of null values: 0
## Percentage of null values: 0 %
##
## Column: start_lng
## Number of null values: 0
## Percentage of null values: 0 %
##
## Column: end_lat
## Number of null values: 5973
## Percentage of null values: 0.1019447 %
##
## Column: end_lng
## Number of null values: 5973
## Percentage of null values: 0.1019447 %
##
## Column: member_casual
## Number of null values: 0
## Percentage of null values: 0 %
```

Errors and Inconsistencies in the Data

Station Name and id columns have a lot of null values which accounts for almost 15% of total data Trip end lattitude and longitude data is also missing in 5953 roows which is very negligible as it accounts for 0.1% of total data Lattitude and longitude data is unnecessary and will be removed from dataset

Data Cleaning

```
#Checking the structure of dataset
str(all_trips_last_year)
```

```
## spc_tbl_ [5,859,061 × 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5859061] "EC2DE40644C6B0F4" "1C31AD03897EE385" "1542
FBEC830415CF" "6FF59852924528F8" ...
##  $ rideable_type     : chr [1:5859061] "classic_bike" "classic_bike" "classic_bike
" "classic_bike" ...
##  $ started_at        : POSIXct[1:5859061], format: "2022-05-23 23:06:58" "2022-05-
11 08:53:28" ...
##  $ ended_at          : POSIXct[1:5859061], format: "2022-05-23 23:40:19" "2022-05-
11 09:31:22" ...
##  $ start_station_name: chr [1:5859061] "Wabash Ave & Grand Ave" "DuSable Lake Shor
e Dr & Monroe St" "Clinton St & Madison St" "Clinton St & Madison St" ...
##  $ start_station_id  : chr [1:5859061] "TA1307000117" "13300" "TA1305000032" "TA13
05000032" ...
##  $ end_station_name  : chr [1:5859061] "Halsted St & Roscoe St" "Field Blvd & Sout
h Water St" "Wood St & Milwaukee Ave" "Clark St & Randolph St" ...
##  $ end_station_id    : chr [1:5859061] "TA1309000025" "15534" "13221" "TA130500003
0" ...
##  $ start_lat         : num [1:5859061] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:5859061] -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ end_lat           : num [1:5859061] 41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:5859061] -87.6 -87.6 -87.7 -87.6 -87.7 ...
##  $ member_casual     : chr [1:5859061] "member" "member" "member" "member" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Data clean up plan:

- New columns will be added foe better analysis
  - Extracting date column from started_at
  - Extracting day, month and year from the date column
- Adding Day of the week name to identify weekdays vs weekend trend
- Checking the range of data and find out if they are not within the selcted 12 month range
- Calculate ride_length to identify trends in how long riders use the bikes

```
#Adding new columns
all_trips_last_year$date <- as.Date(all_trips_last_year$started_at)
all_trips_last_year$month <- format(as.Date(all_trips_last_year$date), "%m")
all_trips_last_year$monthName <- month(as.Date(all_trips_last_year$date), label = TRU
E, abbr = FALSE)
all_trips_last_year$day <- format(as.Date(all_trips_last_year$date), "%d")
all_trips_last_year$year <- format(as.Date(all_trips_last_year$date), "%Y")
all_trips_last_year$day_of_week <- format(as.Date(all_trips_last_year$date), "%A")
```

```
# Check if the trip times are within the range
cat("Earliest Trip Start: ", as.character(min(all_trips_last_year$started_at)))
```

```
## Earliest Trip Start:  2022-05-01 00:00:06
```

```
cat("\nLatest Trip Start: :",as.character(max(all_trips_last_year$started_at)))
```

```
##
## Latest Trip Start: : 2023-04-30 23:59:05
```

```
cat("\n\nEarliest Trip End: ",as.character(min(all_trips_last_year$ended_at)))
```

```
##
##
## Earliest Trip End:  2022-05-01 00:05:17
```

```
cat("\nLatest Trip End: ",as.character(max(all_trips_last_year$ended_at)))
```

```
##
## Latest Trip End:  2023-05-03 10:37:12
```

```
# Add a "ride_length" calculation to all_trips (in seconds)
all_trips_last_year$ride_length <- difftime(all_trips_last_year$ended_at,all_trips_la
st_year$started_at)

# Convert "ride_length" to numeric so we can run calculations on the data
all_trips_last_year$ride_length <- as.numeric(as.character(all_trips_last_year$ride_l
ength))
is.numeric(all_trips_last_year$ride_length)
```

```
## [1] TRUE
```

```
#Checking the range of ride_length
print(min(all_trips_last_year$ride_length))
```

```
## [1] -621201
```

```
print(max(all_trips_last_year$ride_length))
```

```
## [1] 2483235
```

```
cat("Count of Negative values: ", sum(all_trips_last_year$ride_length < 0))
```

```
## Count of Negative values:  103
```

- It is found that ride_length values are negative, which should be zero or more
- Only 103 entires are found negative, which is very negligible

```
# Columns with "HQ QR" values in start_station_name is for QA purposes which needs to
be removed along wiht negative ride_length values
# Creating a new version of the dataframe (v2) since data is being removed
all_trips_last_year_v2 <- all_trips_last_year[!(all_trips_last_year$start_station_nam
e == "HQ QR" | all_trips_last_year$ride_length < 0), ]
```

```
# Count rows where every column has missing values
print(sum(rowSums(is.na(all_trips_last_year_v2)) == ncol(all_trips_last_year_v2)))
```

```
## [1] 831984
```

```
# Remove rows with missing values in every column
all_trips_last_year_v2 <- all_trips_last_year_v2[!rowSums(is.na(all_trips_last_year_v
2)) == ncol(all_trips_last_year_v2), ]

# Count rows where every column has missing values
print(sum(rowSums(is.na(all_trips_last_year_v2)) == ncol(all_trips_last_year_v2)))
```

```
## [1] 0
```

Data Analysis Stage:

- ride_length attribute will be a main focus for this analysis
- We will aggregate the data to find out mean, max and min of ride_length by member type (Casual or Member)
- We will identify how many times bike services are used by month or by day of the week
- We will also identify the nature of the ride_length based on month, day of the week and membership type

```
# Notice that the days of the week are out of order. Let's fix that.
all_trips_last_year_v2$day_of_week <- ordered(all_trips_last_year_v2$day_of_week, lev
els=c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))

all_trips_last_year_v2$monthName <- ordered(all_trips_last_year_v2$monthName, levels=
c("May", "June", "July", "August", "September", "October", "November", "December", "J
anuary", "February", "March", "April"))

all_trips_last_year_v2$monthName <- ordered(all_trips_last_year_v2$monthName, levels=
c("January", "February", "March", "April", "May", "June", "July", "August", "Septembe
r", "October", "November", "December"))
```

```
#Summarizing data for ride_length columnm
summary(all_trips_last_year_v2$ride_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0     347     611    1196    1099 2483235
```

```
# Compare members vs casual users for mean ride_length

# analyze ridership data by type and weekday
all_trips_last_year_v2 %>%
  group_by(member_casual) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n()                           #calculates the number of
rides and average duration
            ,average_duration = mean(ride_length), total_duration = sum(ride_length))
%>%       # calculates the average duration
  arrange(average_duration)
```

```
## # A tibble: 2 × 4
##   member_casual number_of_rides average_duration total_duration
##   <chr>                   <int>            <dbl>          <dbl>
## 1 member                3015036             755.     2275366275
## 2 casual                2011938            1858.     3738293609
```

```
# Compare members vs casual users for max ride_length

print(setNames(aggregate(all_trips_last_year_v2$ride_length ~ all_trips_last_year_v2$
member_casual, FUN = max),
               c("Member Type", "Max Ride Length")))
```

```
##   Member Type Max Ride Length
## 1      casual         2483235
## 2      member           93580
```

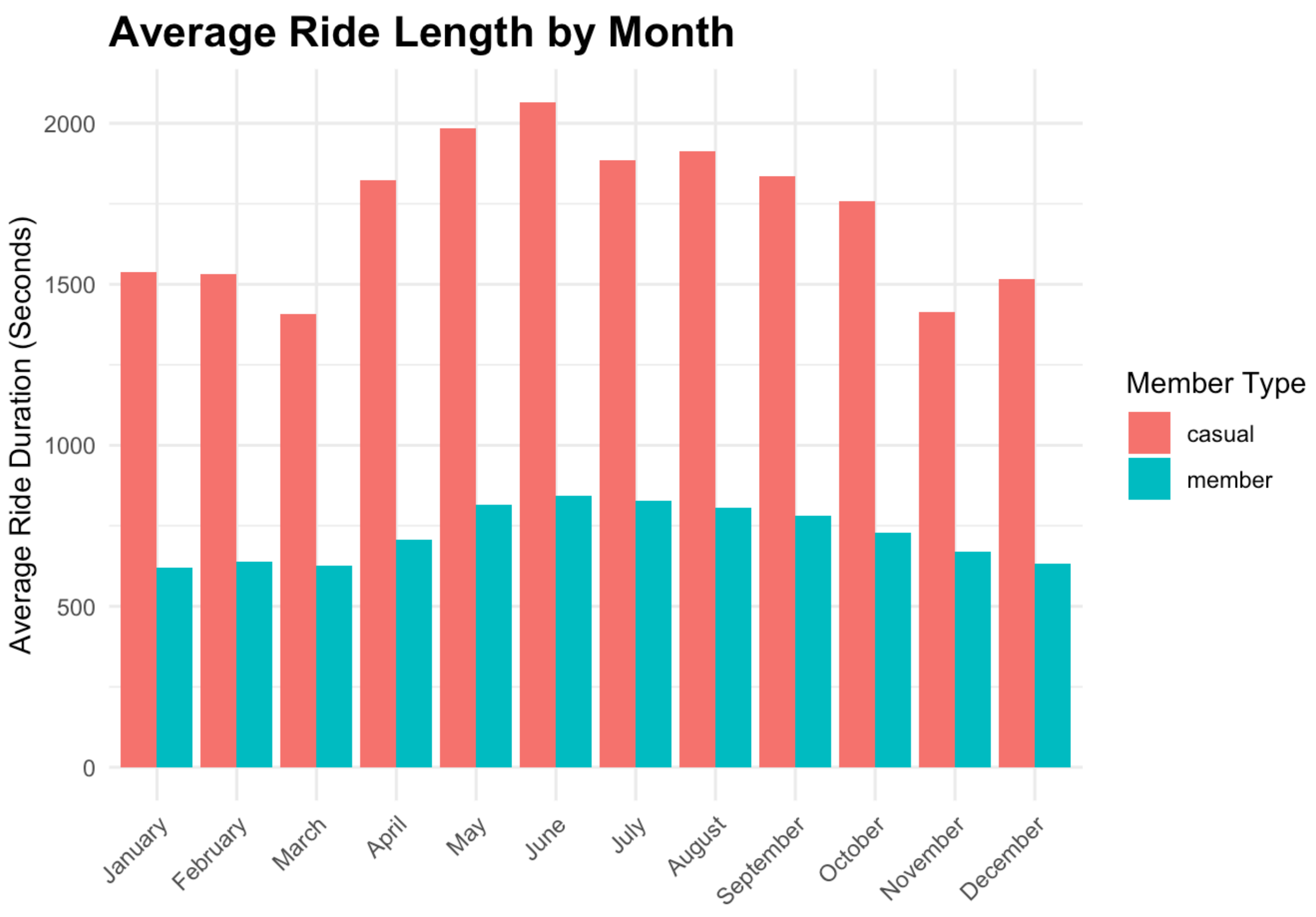
```
# Compare members vs casual users for min ride_length

print(setNames(aggregate(all_trips_last_year_v2$ride_length ~ all_trips_last_year_v2$
member_casual, FUN = min),
               c("Member Type", "Min Ride Length")))
```

```
##   Member Type Min Ride Length
## 1      casual               0
## 2      member               0
```

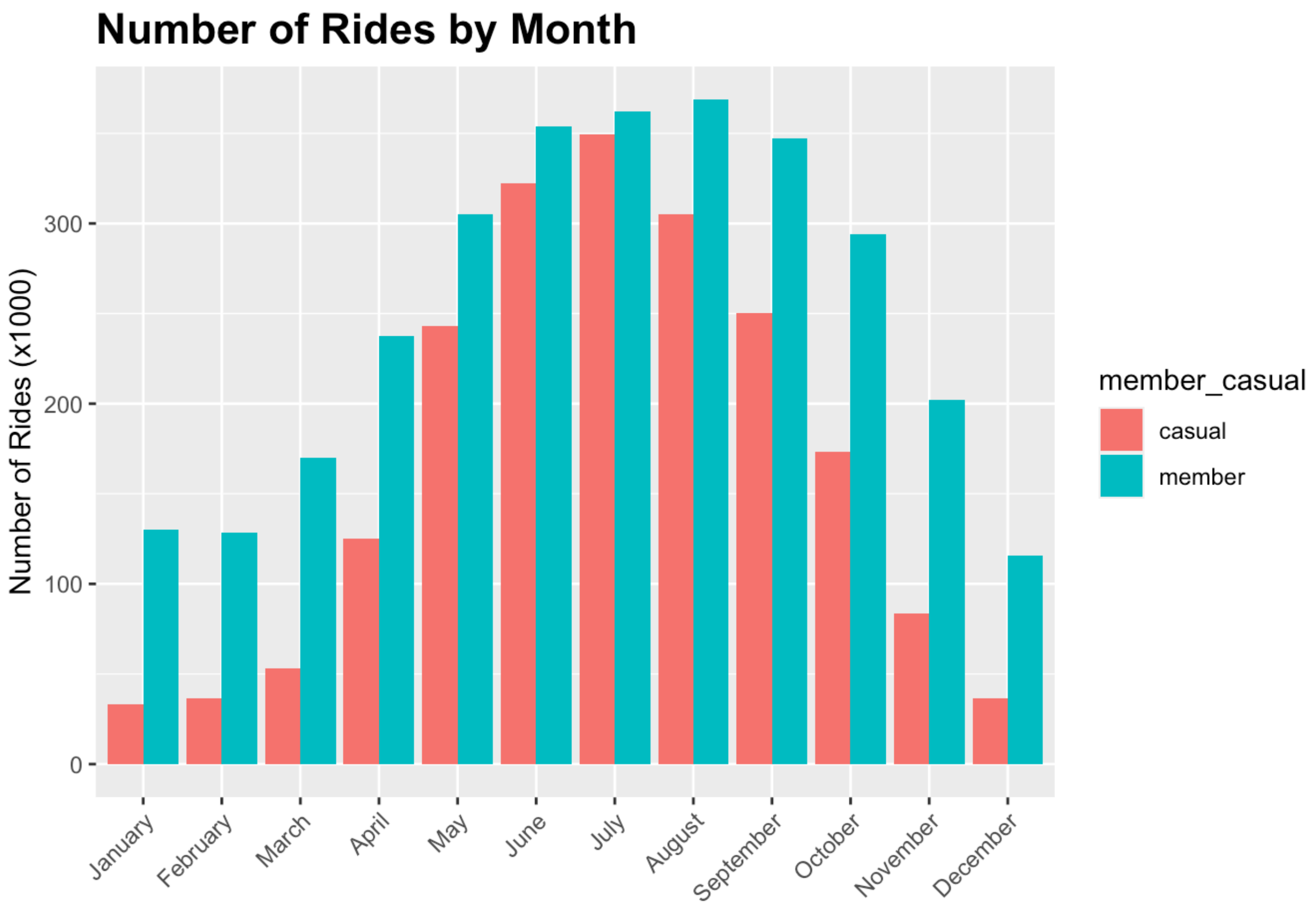
```
# Create the plot
plot <- ggplot(all_trips_last_year_v2, aes(x = monthName, y = ride_length, fill = mem
ber_casual)) +
  geom_bar(stat = "summary", fun = "mean", position = "dodge") +
  labs(title = "Average Ride Length by Month",
       x = "",
       y = "Average Ride Duration (Seconds)",
       fill = "Member Type") +
  theme_minimal()

# Adjust the appearance of the plot
plot +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))
```



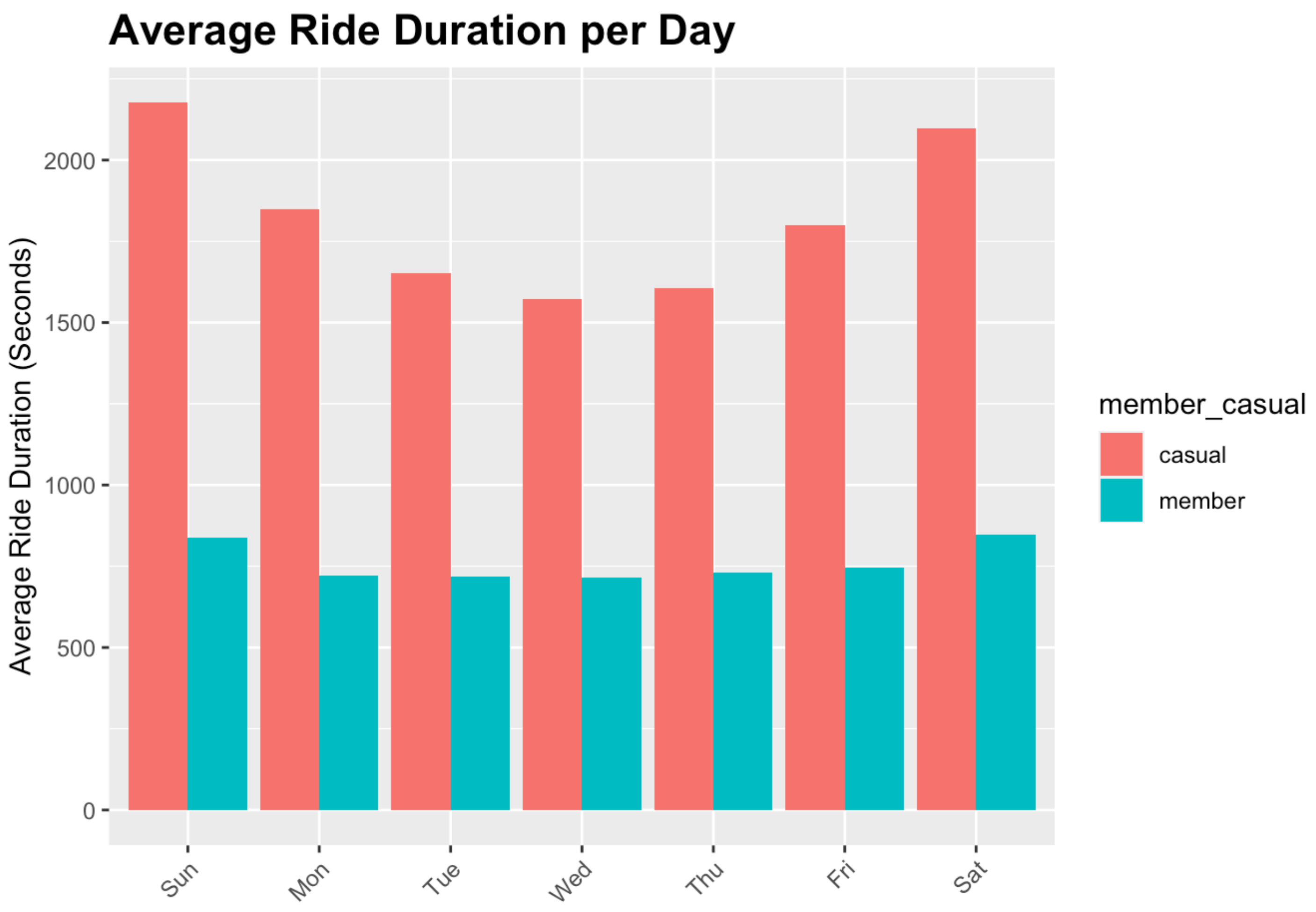
```
# Let's visualize the number of rides by rider type
all_trips_last_year_v2 %>%
  group_by(member_casual, monthName) %>%
  summarise(number_of_rides = n()/1000, average_duration = mean(ride_length)) %>%
  arrange(member_casual, monthName)  %>%
  ggplot(aes(x = monthName, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") + labs(title = "Number of Rides by Month",
       x = "",
       y = "Number of Rides (x1000)") + theme(plot.title = element_text(size = 16, fa
ce = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))
```

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



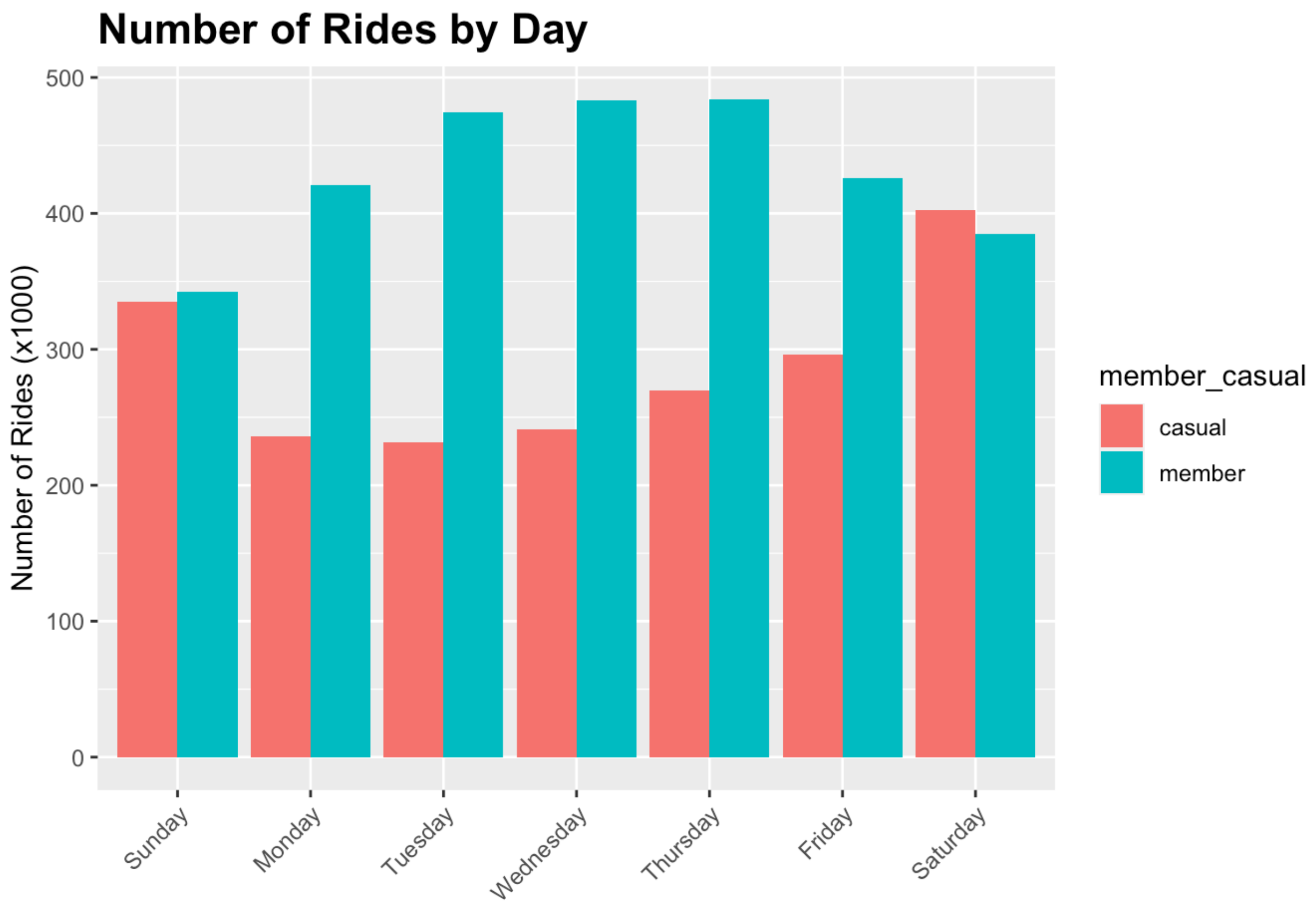
```
# Let's create a visualization for average duration
all_trips_last_year_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") + labs(title = "Average Ride Duration per Day",
       x = "",
       y = "Average Ride Duration (Seconds)") + theme(plot.title = element_text(size
= 16, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))
```

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



```
# Let's visualize the number of rides by rider type
all_trips_last_year_v2 %>%
  group_by(member_casual, day_of_week) %>%
  summarise(number_of_rides = n()/1000, average_duration = mean(ride_length)) %>%
  arrange(member_casual, day_of_week)  %>%
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") + labs(title = "Number of Rides by Day",
       x = "",
       y = "Number of Rides (x1000)") + theme(plot.title = element_text(size = 16, fa
ce = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))
```

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



Recommendations to Buinsess:

How do annual members and casual riders use Cyclistic bikes differently?

- It appears that casual riders are most active in the summer season and their ride length are twice as long than the members
- Members use the service consistently throughout the week as well as throughout the year; while casual riders use the serive mostly on weekends and most active during summer
- Members are generarting higher number of rides while casual riders generates longer rides

How can Cyclistic use digital media to influence casual riders to become members?

- They should launch summer passes and attract them to become a regular member for straight four months and then slowly attract them to buy a new extention pass for fall season
- They should offer a discounted price to casual riders to buy summer pass, and even more lucrative offer to buy extention to fall season

Why would casual riders buy Cyclistic annual memberships?

- Discounted offers should encorage them to buy season pass, which should bend their mind to think about using bike share on regular basis
- When members accept lucrative offer to extend pass to fall season, they would get used to using the service for semi-annual baiss
- Riders who will be using bike service for exercise will surely use this to stay fit and will nurture this new habbit
- Riders will also realise the potential to use this service for their commute and might think of switching to this mode of transportation to save money and will become a member