Cyclistic Case Study

Jessica Y. Yang

2023-01-24

Three Main Questions

These are the three main questions that will guide the future marketing program:

- 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?

Deliverables That Will Be Produced

I will produce a report with the following deliverables in it:

- A clear statement of the business task
- A description of all data sources used
- Documentation of any cleaning or manipulation of data
- A summary of my analysis
- Supporting visualizations and key findings
- My top 3 recommendations based on my analysis

Section Guides

The deliverables will be produced through the following sections:

- Ask
- Prepare
- Process
- Analyze
- Share
- Act

Ask

Guiding questions:

- What is the problem you are trying to solve?
  - This case study focuses on creating marketing strategies to influence casual riders to convert to become Cyclistic members.
- How can your insights drive business decisions?
  - My insights will be able to assist Cyclistic with the best marketing strategy to convert casual riders to become Cyclistic members through several important data and plots.

Key tasks:

- The business task
  - The business task for this case study is to discover how Cyclistic's casual riders and members use their rental bikes differently and produce a strong and strategic marketing tactic to assist casual riders in purchasing a membership with Cyclistic.
- Key stakeholders
  - Cyclistic executive team
  - Director of Marketing in Cyclistic (Lily Moreno)
  - Cyclistic's marketing analytics team

Prepare

Guiding questions:

- Where is your data located?
  - I retrieved and downloaded the previous 12 months of Cyclistic trip data from a public data website. The data has been made available by Motivate International Inc. under this license.
- How is the data organized?
  - The data is produced in 4 different .csv file, each having 3 months of data in it. The organization of each .csv file will be listed below.
- Are there issues with bias or credibility in this data? Does your date ROCCC?
  - R - Reliable: The data is credible because it is public data. It does not have any bias in it because it does not provide any rider's personal information. This data is originally based on the case study 'Sophisticated, Clear, and Polished': Divvy and Data Visualization (Case Study) by Kevin Hartman
  - O - Original: The data is original and can be validated with the original source even though Cyclistic is a fictional company.
  - C - Comprehensive: The data is comprehensive because it has all of the critical information needed to answer the questions proposed and I am able to find solutions for them.

  - C - Current: The data is not up to date, but I have acquired the most recent 12 months of data to do this analysis. The last update is on May 27th, 2020.
  - C - Cited: Since this is a project from the Google Analytics Certificate, I think the source is and should be from a credible organization.

Key tasks:

- Download data and store it appropriately
  - Data is downloaded and stored in a folder on my comptuer called "Cyclistic Case Study".
  - I have combined the 12 months of data into a .zip folder called "Trips Q1 to Q4.zip" and uploaded it to RStudio's project folder for use.
- Determine the credibility of the data
  - As mentioned above, the data is credible because it is public data.
- Identify how data is organized
  - The data is represented in 4 different .csv files and has these columns: rider's ID, session start and end time, bike's ID, trip duration, start and end station names, station's ID, member/casual rider, gender, and birth year.
- Import the data

```
##I installed these packages.
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/Admin/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Admin\AppData\Local\Temp\RtmporDKFE\downloaded_packages
```

```
##I loaded these packages.
library(tidyverse)
```

```
## ── Attaching packages
## ─────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.1
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ──────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(ggplot2)
```

```
library(readxl)
library(dplyr)
library(lubridate)
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
##I used readr to read rectangular data from .csv files
##I named the files by their year and quarter
Q1_2019 <- read_csv("Divvy_Trips_2019_Q1/Divvy_Trips_2019_Q1.csv")
```

```
## 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.
```

```
Q2_2019 <- read_csv("Divvy_Trips_2019_Q2/Divvy_Trips_2019_Q2.csv")
```

```
## Rows: 1108163 Columns: 12
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (4): 03 - Rental Start Station Name, 02 - Rental End Station Name,
User...
## dbl  (5): 01 - Rental Details Rental ID, 01 - Rental Details Bike ID, 03 -
R...
## num  (1): 01 - Rental Details Duration In Seconds Uncapped
## dttm (2): 01 - Rental Details Local Start Time, 01 - Rental Details Local
En...
##
## 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.
```

```
Q3_2019 <- read_csv("Divvy_Trips_2019_Q3/Divvy_Trips_2019_Q3.csv")
```

```
## Rows: 1640718 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.

Q4_2019 <- read_csv("Divvy_Trips_2019_Q4/Divvy_Trips_2019_Q4.csv")

## Rows: 704054 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.
```

Process

Guiding questions:

- What tools are you choosing and why?
  - I am using R for my main statistical analytics tool because it has all the functions I need in order to mutate and analyze my data. I will also be able to create data visualizations at the end when needed.

Key tasks:

- Transform the data and document the cleaning

```
##Since the column names are different on the Q2_2019 data set, I assigned
new names to the columns of the data frame.
colnames(Q2_2019) <- c("trip_id", "start_time", "end_time", "bikeid",
"tripduration", "from_station_id", "from_station_name", "to_station_id",
"to_station_name", "usertype", "gender", "birthyear")

##Duplicate the "start_time" column and name it "month" so I can clearly see
which month the corresponding data is for. I also relocated the new "month"
column to be behind the "end_time" column.
Q1_2019 <- Q1_2019 %>%
  mutate(month = start_time) %>%
  relocate(month, .after = end_time)
```

```r
Q2_2019 <- Q2_2019 %>%
  mutate(month = start_time) %>%
  relocate(month, .after = end_time)
Q3_2019 <- Q3_2019 %>%
  mutate(month = start_time) %>%
  relocate(month, .after = end_time)
Q4_2019 <- Q4_2019 %>%
  mutate(month = start_time) %>%
  relocate(month, .after = end_time)

##Duplicate the "start_time" column and name it "date" so I can clearly see
which month the corresponding data is for. I also relocated the new "date"
column to be behind the "month" column.
Q1_2019 <- Q1_2019 %>%
  mutate(date = start_time) %>%
  relocate(date, .after = month)
Q2_2019 <- Q2_2019 %>%
  mutate(date = start_time) %>%
  relocate(month, .after = month)
Q3_2019 <- Q3_2019 %>%
  mutate(date = start_time) %>%
  relocate(date, .after = month)
Q4_2019 <- Q4_2019 %>%
  mutate(date = start_time) %>%
  relocate(date, .after = month)

##Extract numeric month date from month column.
Q1_2019$month = format(as.Date(Q1_2019$month, format = "%m"), "%m")
Q2_2019$month = format(as.Date(Q2_2019$month, format = "%m"), "%m")
Q3_2019$month = format(as.Date(Q3_2019$month, format = "%m"), "%m")
Q4_2019$month = format(as.Date(Q4_2019$month, format = "%m"), "%m")

##Extract numeric date from the date column.
Q1_2019$date = format(as.Date(Q1_2019$date, format = "%d"), "%d")
Q2_2019$date = format(as.Date(Q2_2019$date, format = "%d"), "%d")
Q3_2019$date = format(as.Date(Q3_2019$date, format = "%d"), "%d")
Q4_2019$date = format(as.Date(Q4_2019$date, format = "%d"), "%d")

##Transform the data to fix the data type inconsistencies found during the
import process so that the data can be combined into one large data frame.
Q1_2019 <- mutate(Q1_2019, from_station_id = as.numeric(from_station_id),
to_station_id = as.numeric(to_station_id))
Q2_2019 <- mutate(Q2_2019, from_station_id = as.numeric(from_station_id),
to_station_id = as.numeric(to_station_id))
Q3_2019 <- mutate(Q3_2019, from_station_id = as.numeric(from_station_id),
to_station_id = as.numeric(to_station_id))
Q4_2019 <- mutate(Q4_2019, from_station_id = as.numeric(from_station_id),
to_station_id = as.numeric(to_station_id))
```

```
##Join all data frames vertically using the rbind function.
cyclistsdata <- rbind(Q1_2019, Q2_2019, Q3_2019, Q4_2019)

##Add a column for the day of the week (week starts Sunday).
cyclistsdata <- cyclistsdata %>%
  mutate(day_of_week = start_time) %>%
  relocate(day_of_week, .after = date)
cyclistsdata$day_of_week <- wday(cyclistsdata$day_of_week)

##Remove unnecessary column in data frames ("tripduration", "birthyear",
"gender").
cyclistsdata <- cyclistsdata %>%
  select(-c(tripduration, birthyear, gender))

##Add column to calculate the total ride length in seconds.
cyclistsdata$ride_length <- as.numeric(difftime(cyclistsdata$end_time,
cyclistsdata$start_time))

##Remove all NA rows as they do not have full data.
cyclistsdatav2 <- drop_na(cyclistsdata) ##Data frame does not have any NA as
the number of observations is still the same.

#Remove all rows for when the bikes were taken out of docks for quality
checks and remove all rows that has ride_length as negative.
cyclistsdatav3 <- cyclistsdatav2[!(cyclistsdatav2$from_station_name == "HQ
QR" | cyclistsdatav2$to_station_id == "671" | cyclistsdatav2$to_station_id ==
"361" | cyclistsdatav2$ride_length < 0), ] ##Data frame had 175 rows that
were taken out.
```

After transforming the data, it is now time to analyze the data.

##Analyze

Guiding questions:

- How should you organize your data to perform analysis on it?
  - I have to make sure all the columns are correctly named and are the same (columns do not have to be in the same order but names have be exactly the same)

```
##Inspect the new data frame that has been created

colnames(cyclistsdatav3) ##Lists of column names

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "month"             "date"              "day_of_week"
##  [7] "bikeid"            "from_station_id"   "from_station_name"
## [10] "to_station_id"     "to_station_name"   "usertype"
## [13] "ride_length"

nrow(cyclistsdatav3) ##The number of rows in the data frame
```

```
## [1] 3817829
```

```
dim(cyclistsdatav3) ##Dimensions of the data frame
```

```
## [1] 3817829      13
```

```
head(cyclistsdatav3) ##First 6 rows of the data frame
```

```
## # A tibble: 6 × 13
##    trip_id start_time          end_time            month date  day_of_w…¹
bikeid
##      <dbl> <dttm>              <dttm>              <chr> <chr>      <dbl>
<dbl>
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07 01    01             3
2167
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34 01    01             3
4386
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12 01    01             3
1524
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28 01    01             3
252
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56 01    01             3
1170
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09 01    01             3
2437
## # … with 6 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>,
## #   ride_length <dbl>, and abbreviated variable name ¹day_of_week
```

```
tail(cyclistsdatav3) ##Last 6 rows of the data frame
```

```
## # A tibble: 6 × 13
##    trip_id start_time          end_time            month date  day_of_w…¹
bikeid
##      <dbl> <dttm>              <dttm>              <chr> <chr>      <dbl>
<dbl>
## 1 25962899 2019-12-31 23:54:54 2020-01-01 00:22:02 12    31             3
5996
## 2 25962900 2019-12-31 23:56:13 2020-01-01 00:15:45 12    31             3
2196
## 3 25962901 2019-12-31 23:56:34 2020-01-01 00:22:08 12    31             3
4877
## 4 25962902 2019-12-31 23:57:05 2020-01-01 00:05:46 12    31             3
863
## 5 25962903 2019-12-31 23:57:11 2020-01-01 00:05:45 12    31             3
2637
## 6 25962904 2019-12-31 23:57:17 2019-12-31 23:59:18 12    31             3
5930
## # … with 6 more variables: from_station_id <dbl>, from_station_name <chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>,
## #   ride_length <dbl>, and abbreviated variable name ¹day_of_week
```

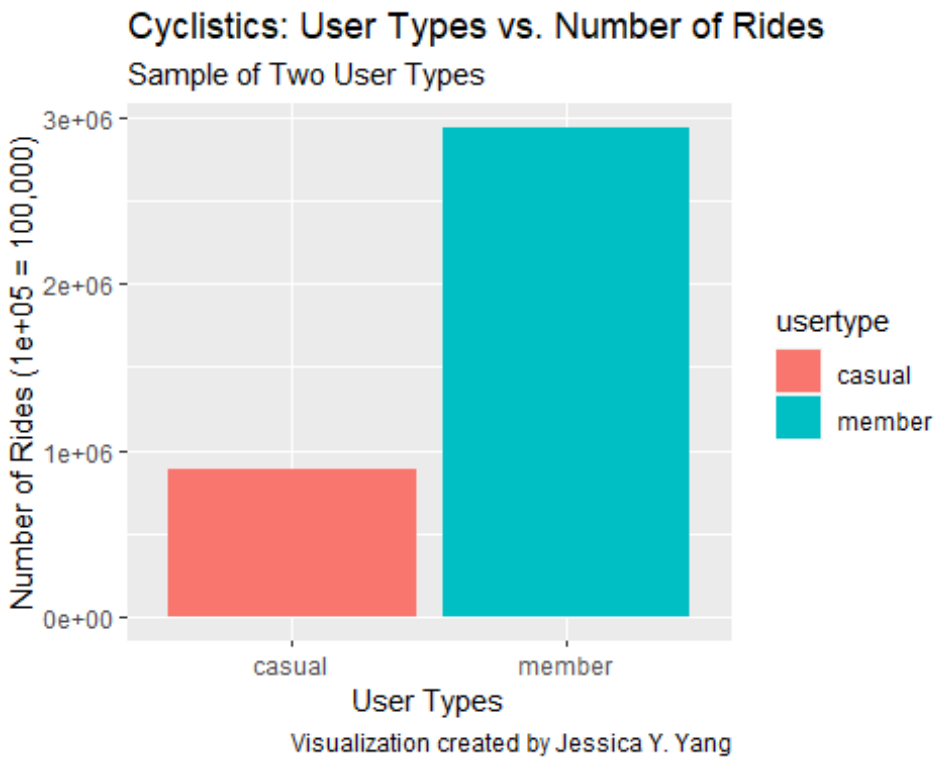
```
str(cyclistsdatav3) ##List of columns and their data types

## tibble [3,817,829 × 13] (S3: tbl_df/tbl/data.frame)
##  $ trip_id          : num [1:3817829] 21742443 21742444 21742445 21742446
21742447 ...
##  $ start_time       : POSIXct[1:3817829], format: "2019-01-01 00:04:37"
"2019-01-01 00:08:13" ...
##  $ end_time         : POSIXct[1:3817829], format: "2019-01-01 00:11:07"
"2019-01-01 00:15:34" ...
##  $ month            : chr [1:3817829] "01" "01" "01" "01" ...
##  $ date             : chr [1:3817829] "01" "01" "01" "01" ...
##  $ day_of_week      : num [1:3817829] 3 3 3 3 3 3 3 3 3 3 ...
##  $ bikeid           : num [1:3817829] 2167 4386 1524 252 1170 ...
##  $ from_station_id  : num [1:3817829] 199 44 15 123 173 98 98 211 150 268
...
##  $ from_station_name: chr [1:3817829] "Wabash Ave & Grand Ave" "State St &
Randolph St" "Racine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id    : num [1:3817829] 84 624 644 176 35 49 49 142 148 141
...
##  $ to_station_name  : chr [1:3817829] "Milwaukee Ave & Grand Ave"
"Dearborn St & Van Buren St (*)" "Western Ave & Fillmore St (*)" "Clark St &
Elm St" ...
##  $ usertype         : chr [1:3817829] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ ride_length      : num [1:3817829] 6.5 7.35 13.82 29.72 6.07 ...

summary(cyclistsdatav3) ##Statistical summary of data

##     trip_id           start_time
##  Min.   :21742443   Min.   :2019-01-01 00:04:37.00
##  1st Qu.:22873747   1st Qu.:2019-05-29 15:46:38.00
##  Median :23962250   Median :2019-07-25 17:49:26.00
##  Mean   :23915588   Mean   :2019-07-19 21:43:54.38
##  3rd Qu.:24963667   3rd Qu.:2019-09-15 04:28:13.00
##  Max.   :25962904   Max.   :2019-12-31 23:57:17.00
##     end_time                       month               date
##  Min.   :2019-01-01 00:11:07.00   Length:3817829     Length:3817829
##  1st Qu.:2019-05-29 16:07:25.00   Class :character   Class :character
##  Median :2019-07-25 18:10:00.00   Mode  :character   Mode  :character
##  Mean   :2019-07-19 22:07:10.93
##  3rd Qu.:2019-09-15 08:08:04.00
##  Max.   :2020-01-10 01:06:36.00
##   day_of_week        bikeid     from_station_id from_station_name
##  Min.   :1.000   Min.   :   1   Min.   :  1.0   Length:3817829
##  1st Qu.:2.000   1st Qu.:1727   1st Qu.: 77.0   Class :character
##  Median :4.000   Median :3451   Median :174.0   Mode  :character
##  Mean   :4.064   Mean   :3380   Mean   :201.7
##  3rd Qu.:6.000   3rd Qu.:5046   3rd Qu.:289.0
##  Max.   :7.000   Max.   :6946   Max.   :673.0
##  to_station_id   to_station_name      usertype          ride_length
```

```
##  Min.   :  1.0   Length:3817829     Length:3817829     Min.   :     1.02
##  1st Qu.: 77.0   Class :character   Class :character   1st Qu.:     6.85
##  Median :174.0   Mode  :character   Mode  :character   Median :    11.82
##  Mean   :202.6                                         Mean   :    23.28
##  3rd Qu.:291.0                                         3rd Qu.:    21.40
##  Max.   :673.0                                         Max.   :177200.37
```

Key tasks:

- Aggregate your data so it's useful and accessible

```
unique(cyclistsdatav3$usertype)

## [1] "Subscriber" "Customer"

##Right now the usertype fields are "Subscriber" and "Customer" but I will
rename it to "member" and "casual", respectively instead.
cyclistsdatav3 <- cyclistsdatav3 %>%
  mutate(usertype = recode(usertype, "Subscriber" = "member", "Customer" =
"casual"))

##Check to see if the proper usertype is changed successfully.
table(cyclistsdatav3$usertype)

##
##  casual  member
##  880537 2937292

##Creates plot to visualize how many total rides each user type has ridden
throughout the year.
cyclistsdatav3 %>%
  group_by(usertype) %>%
  summarize(ride_count = length(trip_id)) %>%
  ggplot(aes(x = usertype, y = ride_count, fill = usertype)) +
geom_col(position = "dodge") + labs(title = "Cyclistics: User Types vs.
Number of Rides", subtitle = "Sample of Two User Types", caption =
"Visualization created by Jessica Y. Yang", x = "User Types", y = "Number of
Rides (1e+05 = 100,000)")
```



- From the above graph, we can see that members rent the bikes at least 3 times more frequently than casual riders do.

```
##Descriptive analysis on ride_length (in minutes).
mean(cyclistsdatav3$ride_length) #Average length of a ride

## [1] 23.27578

median(cyclistsdatav3$ride_length) #Midpoint number in the ascending array of
ride lengths

## [1] 11.81667

max(cyclistsdatav3$ride_length) #Longest ride

## [1] 177200.4

min(cyclistsdatav3$ride_length) #Shortest ride

## [1] 1.016667

##Or use the summary function to show all attributes.
summary(cyclistsdatav3$ride_length)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##      1.02      6.85     11.82     23.28     21.40 177200.37
```

```r
##Compare members vs casual users usage.
aggregate(cyclistsdatav3$ride_length ~ cyclistsdatav3$usertype, FUN = mean)
##casual > member
```

```
##   cyclistsdatav3$usertype cyclistsdatav3$ride_length
## 1                  casual                   54.09194
## 2                  member                   14.03776
```

```r
aggregate(cyclistsdatav3$ride_length ~ cyclistsdatav3$usertype, FUN = median)
##casual > member
```

```
##   cyclistsdatav3$usertype cyclistsdatav3$ride_length
## 1                  casual                   25.83333
## 2                  member                    9.80000
```

```r
aggregate(cyclistsdatav3$ride_length ~ cyclistsdatav3$usertype, FUN = max)
##casual > member
```

```
##   cyclistsdatav3$usertype cyclistsdatav3$ride_length
## 1                  casual                   177200.4
## 2                  member                   101607.1
```

```r
aggregate(cyclistsdatav3$ride_length ~ cyclistsdatav3$usertype, FUN = min)
##casual = member
```

```
##   cyclistsdatav3$usertype cyclistsdatav3$ride_length
## 1                  casual                   1.016667
## 2                  member                   1.016667
```

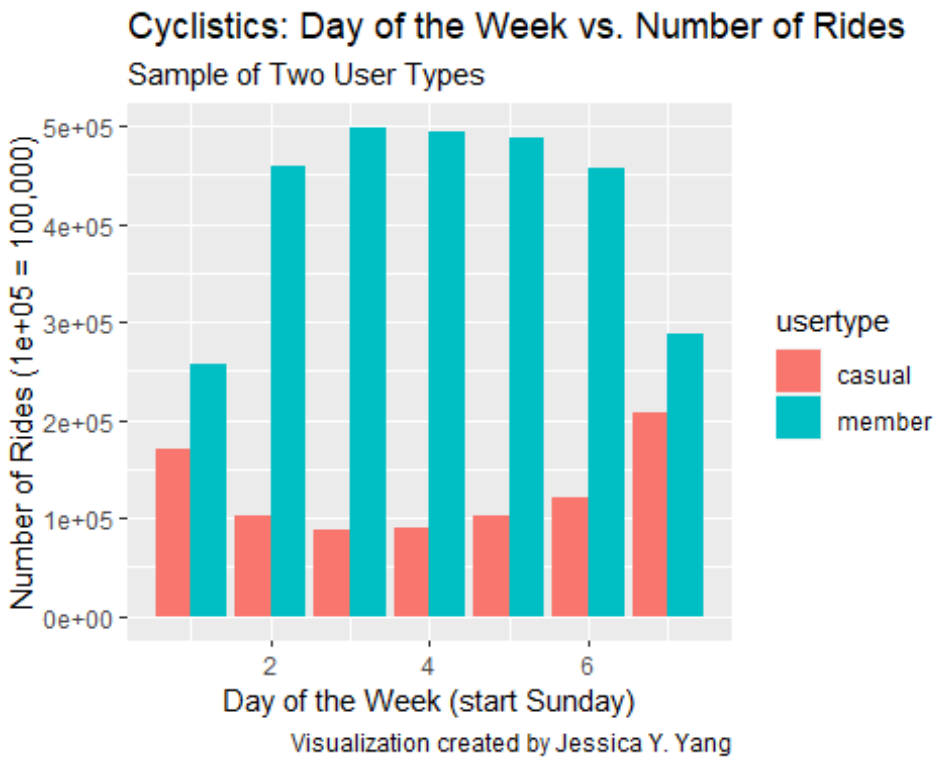
```r
##See the average ride time by each day for members vs. casual.
aggregate(cyclistsdatav3$ride_length ~ cyclistsdatav3$usertype +
cyclistsdatav3$day_of_week, FUN = mean)
```

```
##    cyclistsdatav3$usertype cyclistsdatav3$day_of_week
## 1                   casual                         1
## 2                   member                         1
## 3                   casual                         2
## 4                   member                         2
## 5                   casual                         3
## 6                   member                         3
## 7                   casual                         4
## 8                   member                         4
## 9                   casual                         5
## 10                  member                         5
## 11                  casual                         6
## 12                  member                         6
## 13                  casual                         7
## 14                  member                         7
##    cyclistsdatav3$ride_length
## 1                    53.91131
## 2                    15.30157
## 3                    50.15133
```

```
## 4                    13.62445
## 5                    56.37403
## 6                    13.67516
## 7                    54.99280
## 8                    13.36431
## 9                    59.48967
## 10                   13.70678
## 11                   54.39567
## 12                   13.87889
## 13                   51.99404
## 14                   16.17114
```

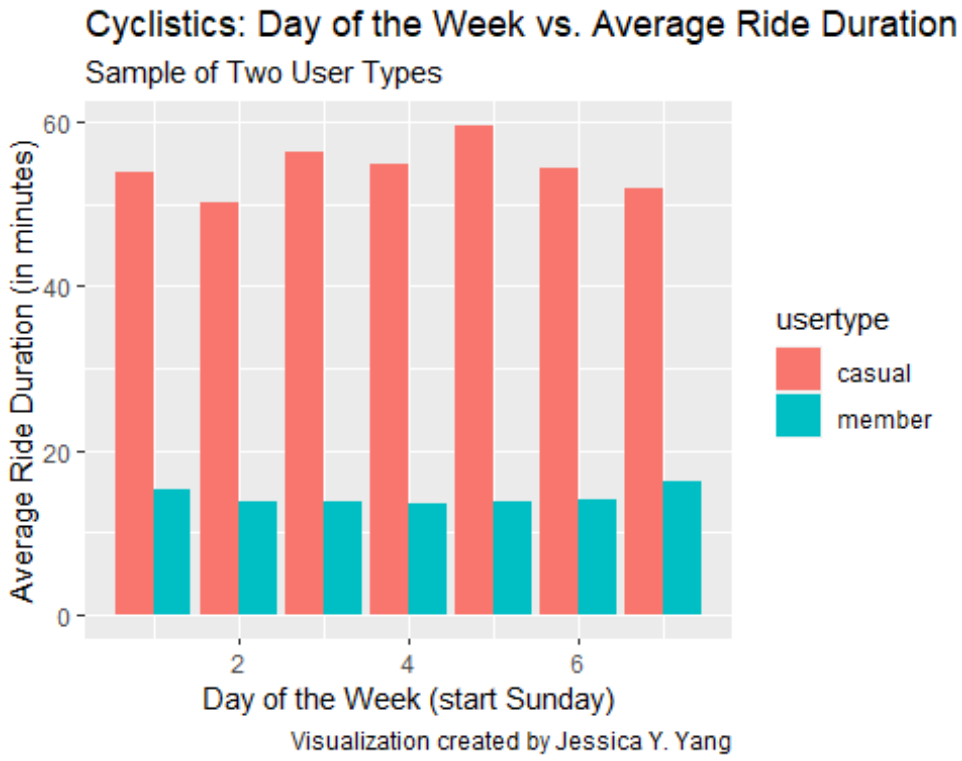
Analyze

```
##Analyze ridership data by type and day of week and visualize it.
cyclistsdatav3 %>%
  group_by(usertype, day_of_week) %>% ##Groups by usertype and day of week
  summarize(number_of_rides = n(), ##Calculates the number of rides and
average duration
            average_duration = mean(ride_length)) %>% ##Calculates the
average duration
  arrange(usertype, day_of_week) %>% ##Sorts by usertype and day of week
  ggplot(aes(x = day_of_week, y = number_of_rides, fill = usertype)) +
geom_col(position = "dodge") + labs(title = "Cyclistics: Day of the Week vs.
Number of Rides", subtitle = "Sample of Two User Types", caption =
"Visualization created by Jessica Y. Yang", x = "Day of the Week (start
Sunday)", y = "Number of Rides (1e+05 = 100,000)") ##Creates plot to
visualize the difference between the number of rides of a casual rider vs. a
member on each day of the week.

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



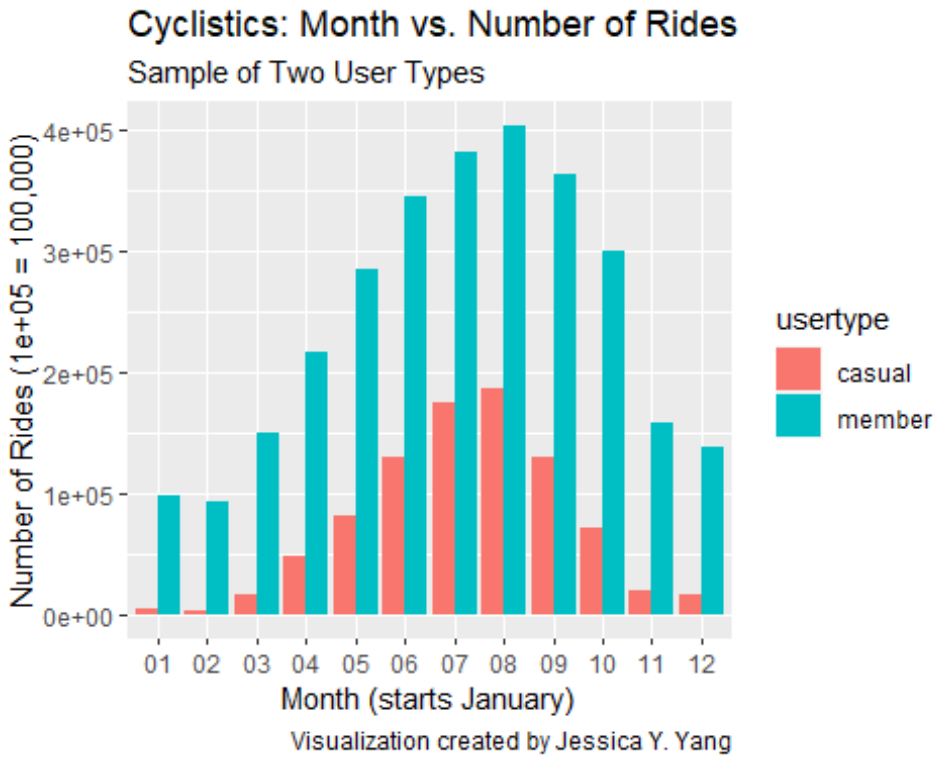
- Members use the service for more times in the week than casual members do.
- The number of rides for members are low on the weekends and high during the weekdays. This makes sense because members will use the bike rental service for commutes (ex. school or work) during the weekdays.
- Casual riders use the bike rental service more on the weekends than on the weekdays. They are probably using it for sightseeing as most people are not working on the weekends.

```
##Visualize the average duration for both casual and members on each day of the week.
cyclistsdatav3 %>%
  group_by(usertype, day_of_week) %>% ##Groups by usertype and day of week
  summarize(number_of_rides = n(), ##Calculates the number of rides and average duration
            average_duration = mean(ride_length)) %>% ##Calculates the average duration
  arrange(usertype, day_of_week) %>% ##Sorts by usertype and day of week
  ggplot(aes(x = day_of_week, y = average_duration, fill = usertype)) +
geom_col(position = "dodge") + labs(title = "Cyclistics: Day of the Week vs. Average Ride Duration (in minutes)", subtitle = "Sample of Two User Types", caption = "Visualization created by Jessica Y. Yang", x = "Day of the Week (start Sunday)", y = "Average Ride Duration (in minutes)")

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



- Casual riders use the bike rental service for a longer period of time per rental than members do. This might be because members usually know which destination they want to go to so they go straight to the destination. However, casual riders might be doing some sightseeing around the city so they do not have a fixed destination.

```
##Visualize the total rides taken by members and casuals by month.
cyclistsdatav3 %>%
  group_by(usertype, month) %>% ##Groups by usertype and month
  summarize(number_of_rides = n(),
            .groups = "drop") %>%
  arrange(usertype, month) %>%
  ggplot(aes(x = month, y = number_of_rides, fill = usertype)) +
geom_col(position = "dodge") + labs(title = "Cyclistics: Month vs. Number of
Rides", subtitle = "Sample of Two User Types", caption = "Visualization
created by Jessica Y. Yang", x = "Month (starts January)", y = "Number of
Rides (1e+05 = 100,000)")
```



- There is a bell curve in both casual riders and members. It starts off low in January and peaks in August, and goes back down in December. I think the reason why it is higher during the months of June to September is because of school holidays. With school holidays, there will be more people outside to rent the bikes. Sometimes, if they are doing a family or friends activity, they can rent with their families or friends which will thus increase the number of rides.
- How do your findings relate to your original question? And what story does your data tell?
  - As stated in the "Ask" section, the business task for this case study is to discover how Cyclistic's casual riders and members use their rental bikes differently. Through my visualizations, I can confidently say that casual riders and annual members have different uses for bike rentals. Annual members are more likely to use it during the weekdays probably for commute while casual riders are more likely to use it on the weekends for leisure. The average casual rider ride duration is longer than that of an average annual member. Regardless of being a member or not the peak month to rent bikes is August. Since Chicago experiences inclement weather, the usage of bikes during the winter season (November through March) is evidently lower than that of the Summer (June to September).
- Is your presentation accessible to your audience?
  - Yes, it is accessible to my audience.

Act

Final conclusion

- Annual members use the bikes more frequently than casual riders do.
- Regardless of being a member or just a casual rider, the peak month that bikes were rented out is during the Summer season.
- Casual riders travel for a longer time period.
- Members ride less during the weekends while casual riders ride more during the weekends.

Top 3 recommendations

- Encourage annual members to ride more on the weekends by giving them discounts or extend their membership for a period of time.
- Release some sort of flash sale that have full annual member benefits for casual riders so they can acquire more bikes and indulge in the benefits of being a member (which might actually help convert casual riders into annual members after they lived through the benefits)
- Offer a weekend-only membership at a different price point to entice casual riders towards a full membership since they ride more during the weekends. This weekend-only membership only unlock bikes on Friday, Saturday, and Sunday.