Cyclistic Case Study – Data Analysis

J. Daniel Marin

1. Business Task

The marketing director at Cyclistic believes that converting casual riders into annual members is key to the company's future success. Annual members generate more consistent revenue, and understanding how their behavior differs from casual riders can help shape targeted marketing efforts.

As a junior data analyst on the marketing analytics team, I have been tasked with analyzing historical ride data to determine how annual members and casual riders use Cyclistic bikes differently.

The insights from this analysis will support the marketing team's decisions on designing campaigns that encourage casual riders to become annual members. The key stakeholders for this project are the marketing director (Lily Moreno), the Cyclistic marketing analytics team, and the Cyclistic executive team.

2. Data Description

For this case study, I used two datasets from Divvy's public data portal:

- `Divvy_Trips_2019_Q1.csv`
- `Divvy_Trips_2020_Q1.csv`

These datasets represent historical bike-share trips in Chicago during Q1 of 2019 and 2020. Although Cyclistic is a fictional company, these files are used as realistic data for analysis.

Source and Licensing

The data was downloaded from Divvy's official S3 repository. It is publicly available and free to use for educational and non-commercial purposes. The datasets do not contain any personally identifiable information (PII), ensuring compliance with privacy regulations.

File Structure and Content

Each file is in `.csv` format and contains individual trip records. After loading both datasets into R using `read_csv()`, I inspected their structure with `colnames()` and `glimpse()`:

- `df_2019` contains **12 columns**, including fields like `trip_id`, `start_time`, `end_time`, `bikeid`, `usertype`, and demographic variables like `gender` and `birthyear`.
- `df_2020` contains **13 columns**, with updated naming conventions like `ride_id`, `started_at`, `ended_at`, `member_casual`, and additional variables like `rideable_type` and station coordinates (`start_lat`, `start_lng`, `end_lat`, `end_lng`).

This revealed structural inconsistencies between the two datasets that will need to be addressed before they can be combined.

Initial Observations and Issues

- Column names differ significantly between years. For example, `usertype` in 2019 was replaced by `member_casual` in 2020, and time columns were renamed from `start_time`/`end_time` to `started_at`/`ended_at`.
- The 2020 dataset includes new fields (`rideable_type`, station coordinates), which are not present in 2019.
- The 2019 dataset contains demographic columns like `gender` and `birthyear`, which are absent in 2020.
- The number of observations is also different: 365,069 rows in 2019 and 426,887 in 2020.

Relevance to the Business Task

These datasets provide the necessary variables to compare **casual riders vs annual members** in terms of:

- Ride duration (can be calculated from timestamps)
- Time and day of usage
- Frequency
- Station usage and rideable types (for 2020)

This information will directly support the business goal of understanding behavioral differences and designing targeted marketing strategies.

Data Credibility and Integrity (ROCCC)

To assess the quality and trustworthiness of the data, the ROCCC framework was applied:

- **Reliable**: The data is automatically collected by Divvy's systems during rentals, minimizing manual input errors.
- **Original**: Downloaded directly from Divvy's official S3 repository.
- **Comprehensive**: Includes complete ride records for Q1 of 2019 and 2020. However, 2019 includes demographic fields (`gender`, `birthyear`) not available in 2020, while 2020 includes geographical coordinates not present in 2019.
- **Current**: Covers recent data relevant for historical comparison and behavioral insights.
- **Cited**: The source is publicly available and properly referenced with a direct link to the official provider.

Missing Data Summary

An inspection using `colSums(is.na())` revealed the following:

- In `df_2019`, there are 19,711 missing values for `gender` and 18,023 for `birthyear`.
- In `df_2020`, only one missing value was found in `end_station_name`, `end_station_id`, `end_lat`, and `end_lng` each.

These results indicate that while 2020 data is almost complete, the demographic fields in 2019 may be unreliable due to the high number of missing values. This limitation will be considered in future steps.

3. Data Cleaning and Preparation

To prepare the data for analysis, I followed the "Process" phase of the data analysis roadmap. This involved checking for inconsistencies, renaming columns, handling missing values, and merging the datasets into one clean data frame ready for analysis.

Tools used:

I used **RStudio** with the following packages: - `tidyverse`: for data wrangling and manipulation - `lubridate`: for handling date-time formats

```
Sys.setlocale("LC_TIME", "English")
```

```
## [1] "English_United States.1252"
```

```
library(tidyverse)
library(lubridate)
library(hms)
library(ggplot2)
```

```
df_2019 <- read_csv("csv/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.
```

```
df_2020 <- read_csv("csv/Divvy_Trips_2020_Q1.csv")
```

```
## Rows: 426887 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## 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.
```

```
# Check column names for consistency
colnames(df_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(df_2020)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"
##  [4] "ended_at"           "start_station_name" "start_station_id"
##  [7] "end_station_name"   "end_station_id"     "start_lat"
## [10] "start_lng"          "end_lat"            "end_lng"
## [13] "member_casual"
```

```
glimpse(df_2019)
```

```
## Rows: 365,069
## Columns: 12
## $ trip_id           <dbl> 21742443, 21742444, 21742445, 21742446, 21742447, 21~
## $ start_time        <dttm> 2019-01-01 00:04:37, 2019-01-01 00:08:13, 2019-01-0~
## $ end_time          <dttm> 2019-01-01 00:11:07, 2019-01-01 00:15:34, 2019-01-0~
## $ bikeid            <dbl> 2167, 4386, 1524, 252, 1170, 2437, 2708, 2796, 6205,~
## $ tripduration      <dbl> 390, 441, 829, 1783, 364, 216, 177, 100, 1727, 336, ~
## $ from_station_id   <dbl> 199, 44, 15, 123, 173, 98, 98, 211, 150, 268, 299, 2~
## $ from_station_name <chr> "Wabash Ave & Grand Ave", "State St & Randolph St", ~
## $ to_station_id     <dbl> 84, 624, 644, 176, 35, 49, 49, 142, 148, 141, 295, 4~
## $ to_station_name   <chr> "Milwaukee Ave & Grand Ave", "Dearborn St & Van Bure~
## $ usertype          <chr> "Subscriber", "Subscriber", "Subscriber", "Subscribe~
## $ gender            <chr> "Male", "Female", "Female", "Male", "Male", "Female"~
## $ birthyear         <dbl> 1989, 1990, 1994, 1993, 1994, 1983, 1984, 1990, 1995~
```

```
glimpse(df_2020)
```

```
## Rows: 426,887
## Columns: 13
## $ ride_id            <chr> "EACB19130B0CDA4A", "8FED874C809DC021", "789F3C21E4~
## $ rideable_type      <chr> "docked_bike", "docked_bike", "docked_bike", "docke~
## $ started_at         <dttm> 2020-01-21 20:06:59, 2020-01-30 14:22:39, 2020-01-~
## $ ended_at           <dttm> 2020-01-21 20:14:30, 2020-01-30 14:26:22, 2020-01-~
## $ start_station_name <chr> "Western Ave & Leland Ave", "Clark St & Montrose Av~
## $ start_station_id   <dbl> 239, 234, 296, 51, 66, 212, 96, 96, 212, 38, 117, 1~
## $ end_station_name   <chr> "Clark St & Leland Ave", "Southport Ave & Irving Pa~
## $ end_station_id     <dbl> 326, 318, 117, 24, 212, 96, 212, 212, 96, 100, 632,~
## $ start_lat          <dbl> 41.9665, 41.9616, 41.9401, 41.8846, 41.8856, 41.889~
## $ start_lng          <dbl> -87.6884, -87.6660, -87.6455, -87.6319, -87.6418, -~
## $ end_lat            <dbl> 41.9671, 41.9542, 41.9402, 41.8918, 41.8899, 41.884~
## $ end_lng            <dbl> -87.6674, -87.6644, -87.6530, -87.6206, -87.6343, -~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

Here we notice that the columns have different names, so the next cleaning step would be to unify them with the corresponding column

Standardize column names

```
#Here we are renaming the columns in df_2019 to be able to merge with df_2020
df_2019_clean <- df_2019 %>%
  rename(
    ride_id = trip_id,
```

```r
    rideable_type = bikeid,
    started_at = start_time,
    ended_at = end_time,
    start_station_name = from_station_name,
    start_station_id = from_station_id,
    end_station_name = to_station_name,
    end_station_id = to_station_id,
    member_casual = usertype
  ) %>%
  mutate(
    start_lat = NA,
    start_lng = NA,
    end_lat = NA,
    end_lng = NA
  )
```

```r
#Here we realize that they use different terms to refer to the same value, so we harmonize them so that

df_2019_clean <- df_2019_clean %>%
  mutate(member_casual = case_when(
    member_casual == "Subscriber" ~ "member",
    member_casual == "Customer" ~ "casual",
    TRUE ~ member_casual
  ))

# Fix incompatible data types before merging
df_2019_clean$ride_id <- as.character(df_2019_clean$ride_id)
df_2020$ride_id <- as.character(df_2020$ride_id)

#Set 2019 rideable_type to "unknown_bike"
df_2019_clean$rideable_type <- "unknown_bike"
```

```r
#The two tables are merged to create the all_trips table

all_trips <- bind_rows(df_2019_clean, df_2020)

#Then creating the ride_length in HH:MM:SS format and create a column called day_of_week, and calculate

all_trips <- all_trips %>%
  mutate(ride_length = as_hms(ended_at - started_at))

all_trips <- all_trips %>%
  mutate(day_of_week = wday(started_at, label = TRUE, abbr = FALSE))
```

```r
# Remove invalid or misleading rides:
# - Negative durations (ride_length < 0)
# - Test station entries like "HQ QR"
all_trips <- all_trips %>%
  filter(ride_length > as_hms("00:00:00")) %>%
  filter(start_station_name != "HQ QR")

# Create additional date-related columns for later analysis
all_trips <- all_trips %>%
  mutate(
    date = as.Date(started_at),
    month = format(date, "%m"),
    day = format(date, "%d"),
    year = format(date, "%Y")
  )
```

4. Analysis

This section focuses on identifying trends and behavioral differences between annual members and casual riders using aggregated metrics such as ride duration, day of the week, and frequency of use.

```
# Average, median, max, min ride length by user type
all_trips %>%
  group_by(member_casual) %>%
  summarise(
    average_duration_mins = mean(as.numeric(ride_length) / 60, na.rm = TRUE),
    median_duration_mins = median(as.numeric(ride_length) / 60, na.rm = TRUE),
    max_duration_mins = max(as.numeric(ride_length) / 60, na.rm = TRUE),
    min_duration_mins = min(as.numeric(ride_length) / 60, na.rm = TRUE)
  )
```

```
## # A tibble: 2 x 5
##   member_casual average_duration_mins median_duration_mins max_duration_mins
##   <chr>                         <dbl>                <dbl>             <dbl>
## 1 casual                         89.5                23.2            177200.
## 2 member                         13.3                 8.47           101607.
## # i 1 more variable: min_duration_mins <dbl>
```

```
# Number of rides and average duration per day of week
all_trips %>%
  group_by(member_casual, day_of_week) %>%
  summarise(
    number_of_rides = n(),
    average_duration = mean(as.numeric(ride_length), na.rm = TRUE)
  ) %>%
  arrange(member_casual, day_of_week)
```

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

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual day_of_week number_of_rides average_duration
##    <chr>         <ord>                 <int>            <dbl>
##  1 casual        Sunday                18652            5061.
##  2 casual        Monday                 5591            4752.
##  3 casual        Tuesday                7311            4562.
##  4 casual        Wednesday              7690            4480.
##  5 casual        Thursday               7147            8452.
##  6 casual        Friday                 8013            6091.
##  7 casual        Saturday              13473           4951.
##  8 member        Sunday                60197            973.
##  9 member        Monday               110430            822.
## 10 member        Tuesday              127974            769.
## 11 member        Wednesday            121902            712.
## 12 member        Thursday             125228            707.
## 13 member        Friday               115168            797.
## 14 member        Saturday              59413            974.
```

```
# Number of rides and average duration per month
all_trips %>%
  group_by(member_casual, month) %>%
  summarise(
    number_of_rides = n(),
    average_duration = mean(as.numeric(ride_length), na.rm = TRUE)
  ) %>%
  arrange(member_casual, month)
```

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

```
## # A tibble: 6 x 4
## # Groups:   member_casual [2]
##   member_casual month number_of_rides average_duration
##   <chr>         <chr>           <int>            <dbl>
## 1 casual        01              12387            7151.
## 2 casual        02              14952            8128.
## 3 casual        03              40538            3813.
## 4 member        01             234769             782.
## 5 member        02             220262             784.
## 6 member        03             265281             816.
```

```
# Top 10 most used start stations by type
all_trips %>%
  group_by(member_casual, start_station_name) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  slice_max(order_by = count, n = 10)
```

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

```
## # A tibble: 20 x 3
## # Groups:   member_casual [2]
##    member_casual start_station_name           count
##    <chr>         <chr>                        <int>
##  1 casual        Streeter Dr & Grand Ave       2749
##  2 casual        Lake Shore Dr & Monroe St     2732
##  3 casual        Shedd Aquarium                1832
##  4 casual        Millennium Park               1406
##  5 casual        Michigan Ave & Oak St         1017
##  6 casual        Michigan Ave & Washington St   839
##  7 casual        Dusable Harbor                  832
##  8 casual        Adler Planetarium               827
##  9 casual        Theater on the Lake             795
## 10 casual        Lake Shore Dr & North Blvd      606
## 11 member        Canal St & Adams St           13799
## 12 member        Clinton St & Washington Blvd 13434
## 13 member        Clinton St & Madison St      12891
## 14 member        Kingsbury St & Kinzie St      8720
## 15 member        Columbus Dr & Randolph St     8515
## 16 member        Canal St & Madison St         7957
## 17 member        Franklin St & Monroe St       7010
## 18 member        Michigan Ave & Washington St  6686
## 19 member        Larrabee St & Kingsbury St    6467
## 20 member        Clinton St & Lake St          6439
```

```
# Filtrar solo usuarios casuales y meses de enero a marzo
casual_monthly_trends <- all_trips %>%
  filter(member_casual == "casual",
         month %in% c("01", "02", "03"),
         year %in% c("2019", "2020")) %>%
  group_by(year, month) %>%
  summarise(
    number_of_rides = n(),
    average_duration_mins = mean(as.numeric(ride_length) / 60, na.rm = TRUE)
  ) %>%
  arrange(year, month)
```

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

```
# Ver resultados
casual_monthly_trends
```

```
## # A tibble: 6 x 4
## # Groups:   year [2]
##   year  month number_of_rides average_duration_mins
##   <chr> <chr>           <int>                 <dbl>
## 1 2019  01               4602                  47.3
## 2 2019  02               2638                 146.
## 3 2019  03              15923                  52.3
## 4 2020  01               7785                 162.
## 5 2020  02              12314                 133.
## 6 2020  03              24615                  70.8
```

4.1 Analysis Summary

After cleaning and analyzing the combined dataset for Q1 of 2019 and 2020, we identified key differences in behavior between casual riders and annual members. These insights are crucial for designing marketing strategies to convert more casual users into long-term members.

4.1.1 Ride Duration

- Casual riders tend to have much longer ride durations (average: approximately 89.5 minutes) compared to members (average: approximately 13.25 minutes).
- The median duration for casuals is also significantly higher, suggesting that they use bikes more for leisure than for transportation.
- This is supported by the high maximum duration outliers among casual users, indicating sporadic but very long rides.

4.1.2 Usage by Day of Week

- Casual riders ride more on weekends, particularly Saturdays and Sundays, which suggests recreational use.
- Members ride more on weekdays, with peaks from Monday through Friday, which is consistent with commuting behavior.

4.1.3 Usage by Month

- Casual riders show higher variability across months:
  - In February, their ride duration peaks (average of around 145 to 161 minutes).
  - In March, there is a significant increase in total ride count year over year (from 15,923 in 2019 to 24,615 in 2020).
- Members show a more stable monthly pattern in both ride count and duration, reflecting habitual or routine use.

4.1.4 Start Station Analysis

- The most common start stations for casual riders are located near tourist attractions and lakefront areas (e.g., Millennium Park, Shedd Aquarium, Streeter Dr & Grand Ave).
- In contrast, members frequently start their rides from transportation hubs and downtown locations (e.g., Canal St & Adams St, Clinton St & Washington Blvd), which aligns with commuting purposes.

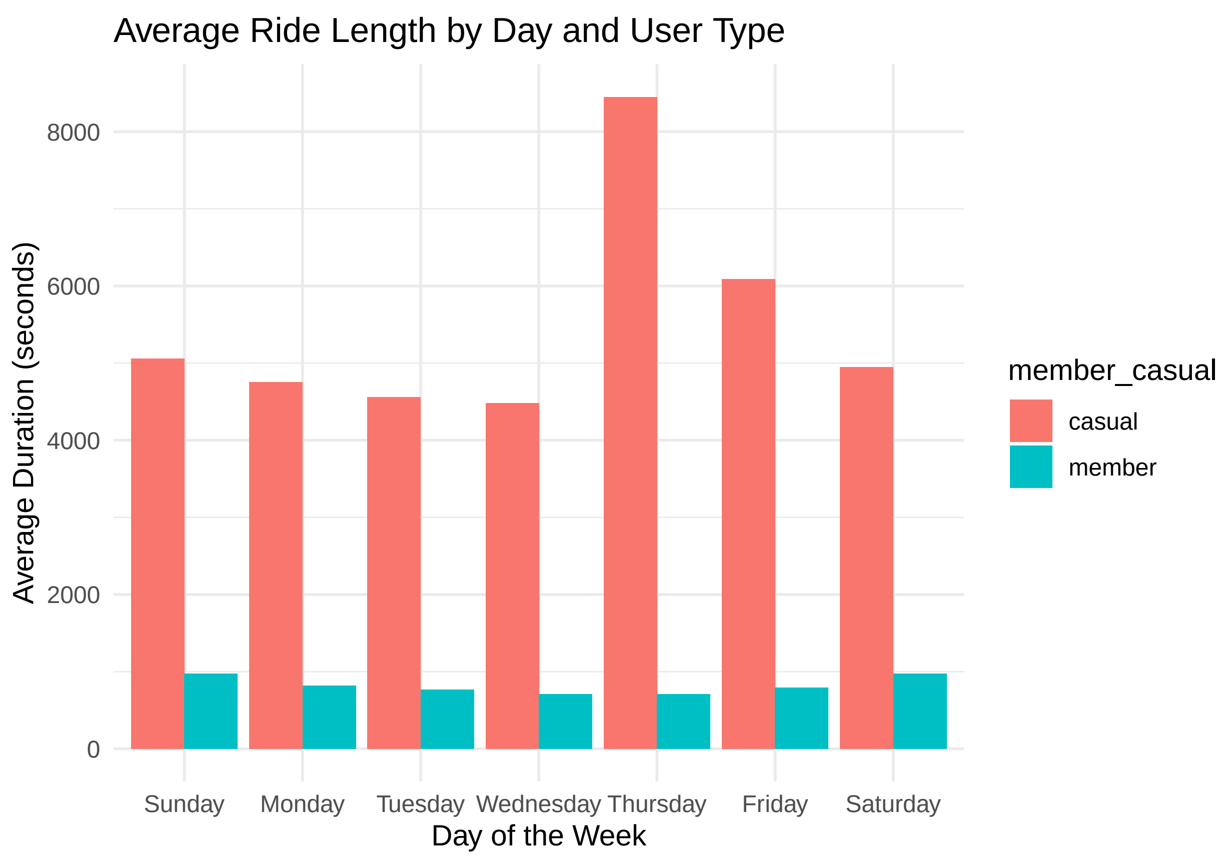
4.1.5 Key Takeaways

- Casual users tend to use the service for leisure, while members show patterns consistent with daily commuting.
- Marketing efforts should focus on:
  - Promoting membership advantages to casual riders, especially on weekends and in the spring.
  - Emphasizing convenience and reliability for commuters in densely populated work areas.

5. Share: Data Visualization

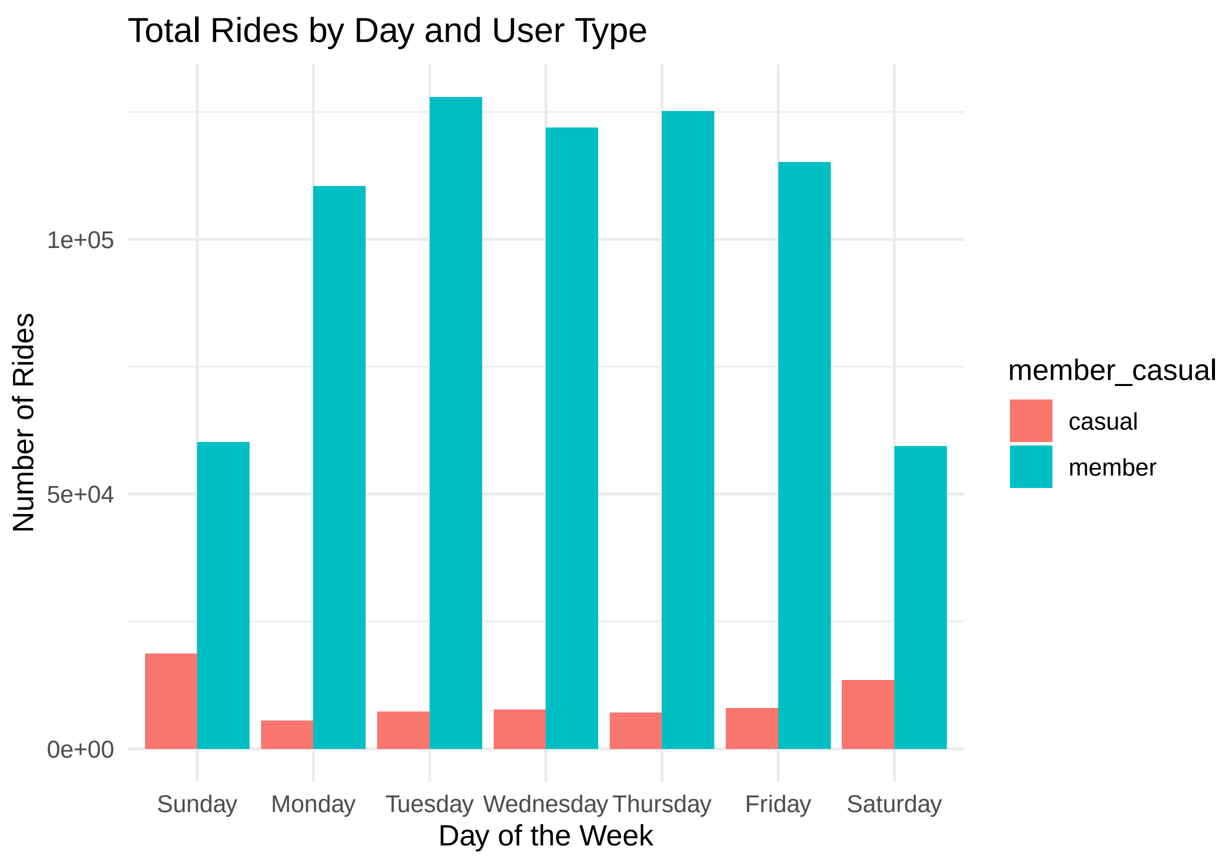
In this section, I created visualizations to better understand how casual riders and annual members use Cyclistic bikes differently.

5.1 Average Ride Length by User Type



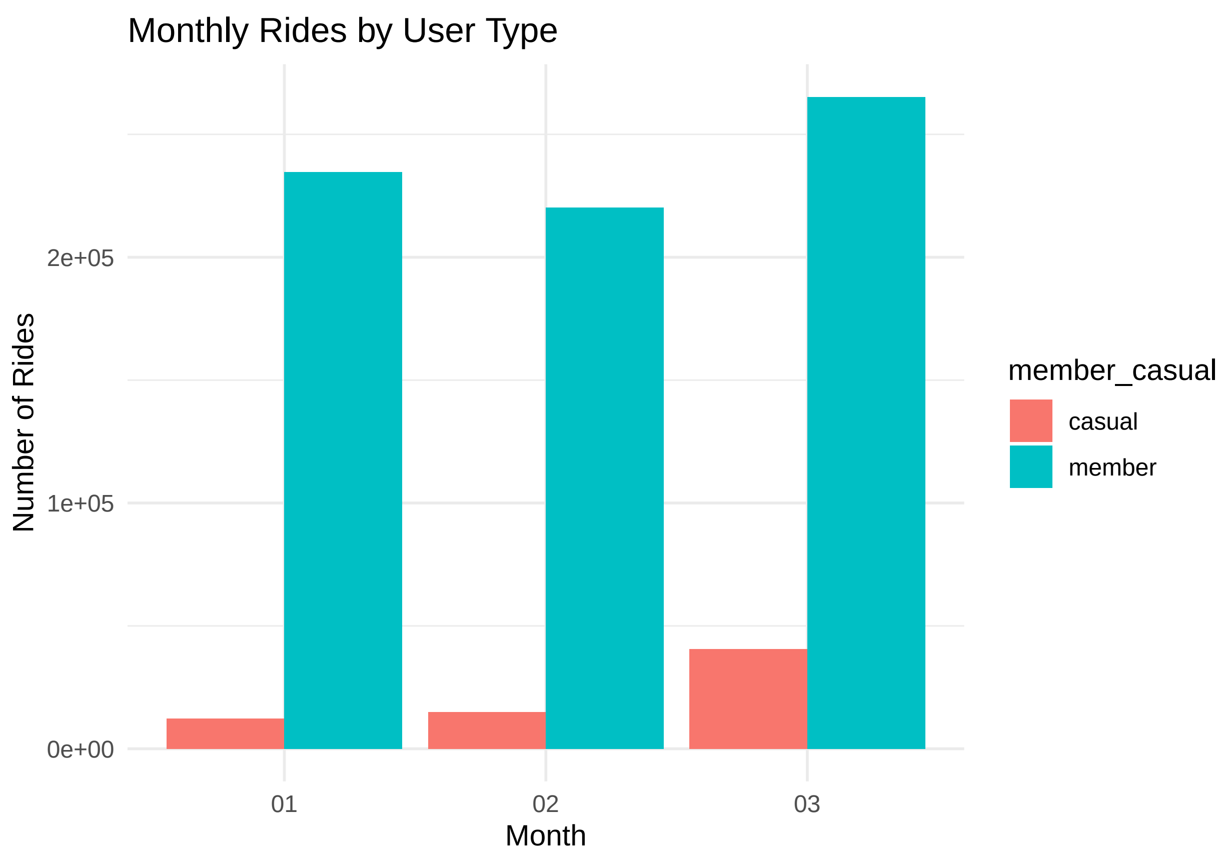
Description: This chart shows that casual riders tend to have longer ride durations than members across all days of the week.

5.2 Number of Rides by Day and User Type



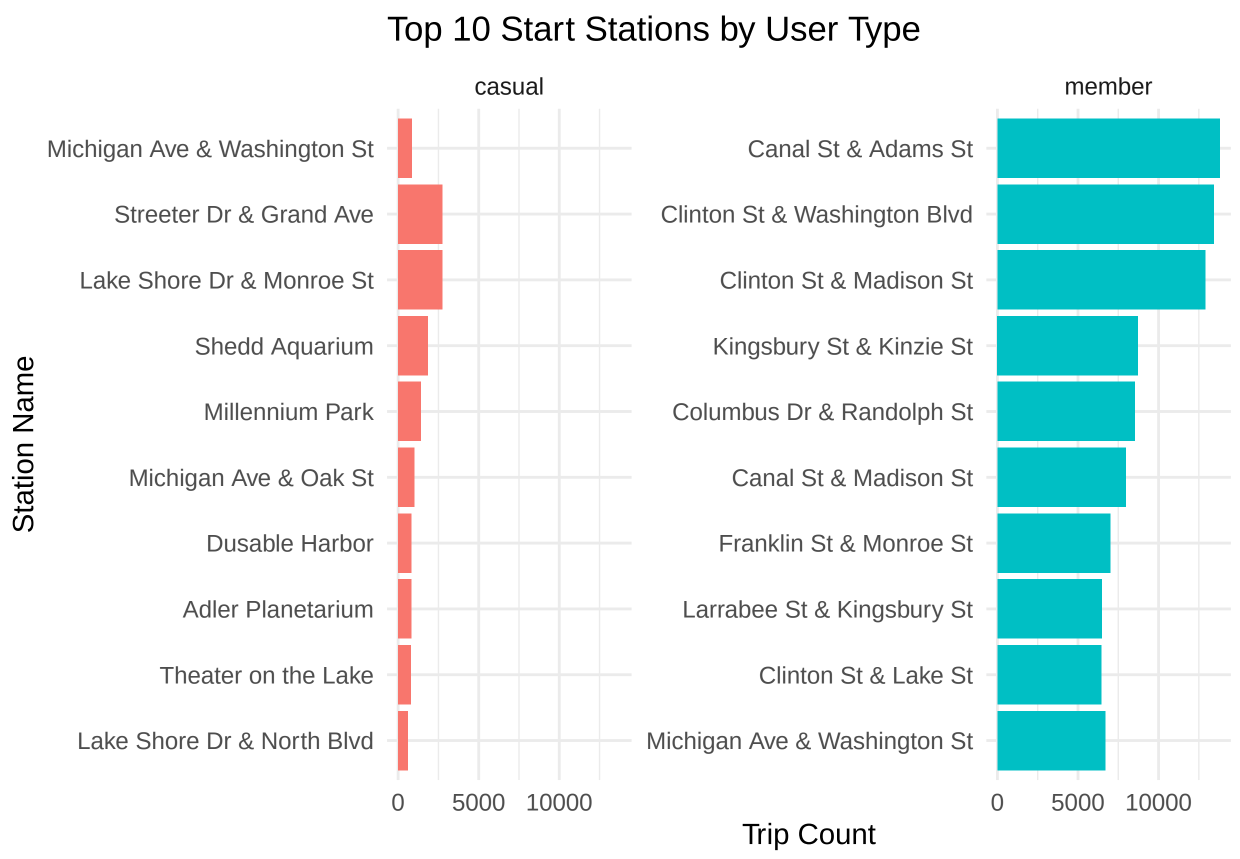
This chart displays how members take more rides on weekdays, while casual riders increase the use of the service on weekends.

5.3 Rides by Month for Each User Type



This chart compares monthly ride counts and shows a clear increase in rides during March for both user types.

5.4 Top Start Stations by User Type



This bar chart shows the most used start stations for both user types. The preferences appear scattered with no strong overlap.

6. Act

Final Recommendations

Based on the analysis, I propose the following recommendations:

1. **Launch weekend-focused campaigns for casual users.**
   Casual riders show the highest usage on Saturdays and Sundays. Marketing efforts (such as discounts or weekend passes) should target these days to increase engagement.
2. **Highlight membership benefits for weekday commuters.**
   Annual members consistently ride more during weekdays. Campaigns should emphasize cost savings and convenience for daily travel, especially for work commutes.
3. **Take advantage of seasonal spikes.**
   Both 2019 and 2020 show an increase in casual usage in March. This indicates a seasonal opportunity to launch targeted promotions that could convert casual riders into members.

Potential Next Steps

If more time and resources were available, I would:

- Expand the analysis to cover 12 full months.
- Include weather and demographic data to better understand ride behavior.

Application of Insights

These insights can support the marketing team in designing more personalized and effective strategies to increase membership rates while optimizing user experience for both segments.