Cyclistic Case Study

“Ahmed saber”

2023-01-06

Google Capstone Project - Cyclystic Case Study

This note book is created to share the process used in approaching the Google Capstone Project

Case Study 1: How Does a Bike-Share Navigate Speedy Success?

Background and Setting

- **Cyclistic** is a Chicago based Bike-share program with a fleet of 5,824 bicycles that are geo-tracked and locked into a network of 692 stations.
- **The bikes**can be unlocked from one station and returned to any other station in the system at anytime.
- **Cyclistic** sets itself apart by not only offering standard top seated bikes but also offers reclining bikes, hand tricycles, and cargo bikes.
- **Cyclistic** business model provides a more inclusive program to people with disabilities and riders who are not able to use a standard two-wheeled bike.
- **Cyclistic** has two classifications for its users:
    - Customers who purchase single ride or full-day passes are referred to as **Casual Riders**
    - Customers who purchase annual memberships are referred to as **Members**

Objectives and Goals

Lily Moreno - Director of marketing - has set a clear goal: Design marketing strategies aimed at converting Casual Riders into Annual Members .

- **The Goal of the analytics team is to understand:**

1. How do Annual Members and Casual Riders use Cyclistic bikes differently?
2. Why would Casual Riders buy Cyclistic Annual Memberships?
3. How can Cyclistic use digital media to influence casual riders to become members?

This presentation aims to answer the question:**How do Annual Members and Casual Riders use Cyclistic bikes differently?**

The Data

- Data for this case study has been obtained from Dibbybikes Trip Records available through click here (https://www.coursera.org/learn/google-data-analytics-capstone/supplement/7PGIT/case-study-1-how-does-a-bike-share-navigate-speedy-success)
- For the purposes of this case study, the datasets are appropriate and will enable us to answer the business questions.
- The data has been made available by Motivate International Inc. under this license (https://ride.divvybikes.com/data-license-agreement)
- The data used for this analysis covers the Trips during a **12-Month period**, spanning between **January-2022** to **December-2022**. *The data includes records fore each registered trip, its exact starting and ending times, starting and

ending stations, and the membership type of the rider (Casual Rider or a Member).

Data Limitations

- **An important limitation in our data, is that due to privacy-related issues no personally identifiable information can be used.**
- The privacy issues means that for our analysis we are not able to connect pass purchases to credit card numbers.
- Resulting in the inability to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes.
- **The data spans over 5,667,717 total rows; But contains missing values for starting stations and ending stations and their IDs.**
- The decision was made to not include these trip entries in the case study analysis, but in actual scenario we need to verify the reason of these blanks to make an informed decision about the measures to take towards the analysis.

Data Handling and Initial Processing

Data was downloaded from the repository in the form of 12 CSV files, each file corresponds to the records of a month in the year 2022. * **The data downloaded has been already processed from the source to:** + Remove trips that are taken by staff as they service and inspect the system + Remove any trips that were below 60 seconds in length (potentially false starts or users trying to re-dock a bike to ensure it was secure).

Initial Setup

- **Installing necessary packages**
- The packages nedded for the cleaning process and through the process are **(Janitor, readr, tidyverse, hms , lubridate, ggplot2)**

Loading our packages:

```
library("janitor")
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v ggplot2 3.4.0      v purrr   1.0.0
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("readr")
library("hms")
library("lubridate")
```

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

```
library("ggplot2")
```

Importing and Appending Data

We will begin by importing the data into our workspace by using read_csv() and then appending them together using rbind()

```
tripdata_01 <- read.csv("202201-divvy-tripdata.csv")
tripdata_02 <- read.csv("202202-divvy-tripdata.csv")
tripdata_03 <- read.csv("202203-divvy-tripdata.csv")
tripdata_04 <- read.csv("202204-divvy-tripdata.csv")
tripdata_05 <- read.csv("202205-divvy-tripdata.csv")
tripdata_06 <- read.csv("202206-divvy-tripdata.csv")
tripdata_07 <- read.csv("202207-divvy-tripdata.csv")
tripdata_08 <- read.csv("202208-divvy-tripdata.csv")
tripdata_09 <- read.csv("202209-divvy-publictripdata.csv")
tripdata_10 <- read.csv("202210-divvy-tripdata.csv")
tripdata_11 <- read.csv("202211-divvy-tripdata.csv")
tripdata_12 <- read.csv("202212-divvy-tripdata.csv")
trips_2022 <- rbind(tripdata_01,tripdata_02,tripdata_03,tripdata_04,tripdata_05,tripdata_06,tripda
ta_07,tripdata_08,tripdata_09,tripdata_10,tripdata_11,tripdata_12)
```

We will use View() and dim() to view and determine the dimensions of our data frame

```
View(trips_2022)
dim(trips_2022)
```

```
## [1] 5667717      13
```

Column Selections and Cleaning

We will then filter out records that are fully empty records, or contain missing entries for the starting station and/or ending station names by using filter() Then we will select the columns that are releveant to our analysis and what we are trying to answer

```
trips_2022 <- trips_2022 %>% drop_na() %>%
  filter(!is.na(end_station_name) | end_station_name != "" ) %>%
  filter(!is.na(start_station_name) | start_station_name != "")
```

Now we will select only the data that is necessary for our analysis

```
trips_2022 <- trips_2022 %>%
  select(-c(start_station_id, end_station_id, start_lat , start_lng,end_lat, end_lng,start_station
_name,end_station_name))
```

Adding New Columns for Our Data

- **We will create some new columns to further enhance our analysis results**
- **We will add some calculated columns and some filtering columns**
- We will use mutate() to manipulate the columns
- We will add start_weekday to find the exact weekday from a date entry
- We will adjust the format of started_at column and ended_at columns to dates
- We will extract the dates of which each trip started or ended

```
trips_2022 <- trips_2022 %>%
  mutate(start_weekday = wday(started_at,label=TRUE,abbr=FALSE),
         started_at=ymd_hms(started_at),ended_at=ymd_hms(ended_at),
         start_date= as.Date(started_at),end_date=as.Date(ended_at))
```

- We will add a column to calculate the duration of each trip in seconds
- We will combine as.numeric() and difftime() from “hms” package
- difftime() calculates the differences between two times in any time unit
- as.numeric() is used to remove the unit and adjust the type of the record to ease calculations

```
trips_2022$trip_duration_secs <- as.numeric(difftime(trips_2022$ended_at,
                                                     trips_2022$started_at,
                                                     units="secs"), units = "secs")
```

- **We will create another set of columns grouping purposes**
- The month in which a trip ended

```
trips_2022$start_month <- month(trips_2022$ended_at)
```

- The hour of the day in which a trip started

```
trips_2022$start_hour <- hour(trips_2022$started_at)
```

- We will add the abbreviation of each month's name for cleaner plots

```
trips_2022$months_abbre <- month.abb[trips_2022$start_month]
```

- We will set the order of the weekdays to start on monday

```
trips_2022$start_weekday <- ordered(trips_2022$start_weekday,
                                    levels=c("Monday", "Tuesday", "Wednesday",
                                             "Thursday", "Friday",
                                             "Saturday", "Sunday"))
```

Analyzing our Data

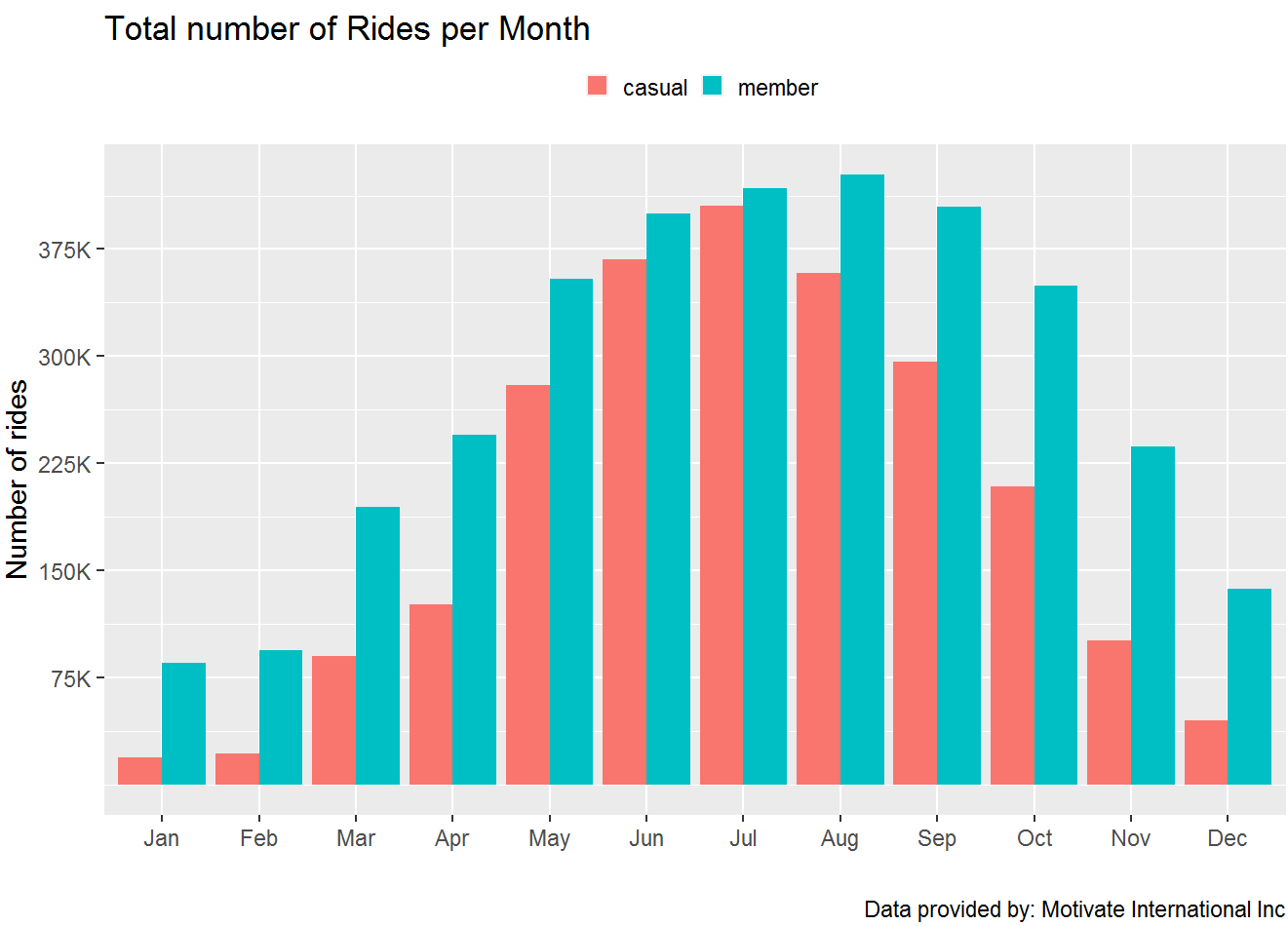
First, we will take a look at how the number of riders differ by the month across the year * We will create a table for the data summary we will obtain using summarize(), We will group our results by the month, month's abbreviation, and membership type. * Then we will order the summary by the numeric represntation of the month

```
trips_monthly <- trips_2022 %>%
  group_by(start_month, member_casual, months_abbre) %>%
  summarize(total_monthly_rides=n()) %>%
  arrange(start_month)
```

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

- **To produce the following plot we use the ggplot as follows**
- Inside the geom_col() we used position="dodge" to avoid making a stacked column chart which results by default
- theme() can be used to customize any aspects of the plot, we used it here to move the legend to the top
- scale_y_continuous() uses this structure to customize what markers are shown on the y-axis and their labels

```
trips_monthly %>%
  ggplot(aes(reorder(x=months_abbre, start_month), y=total_monthly_rides, fill=member_casual)) +
  geom_col(position = "dodge") +
  labs(title="Total number of Rides per Month", caption="Data provided by: Motivate International
 Inc",
       x="", y="Number of rides") +
  guides(fill=guide_legend(title="")) +
  theme(legend.position="top", legend.key.size = unit(0.3, 'cm')) +
  scale_y_continuous(breaks = c(75000, 150000, 225000, 300000, 375000), labels=c("75K", "150K", "2
25K", "300K", "375K"))
```



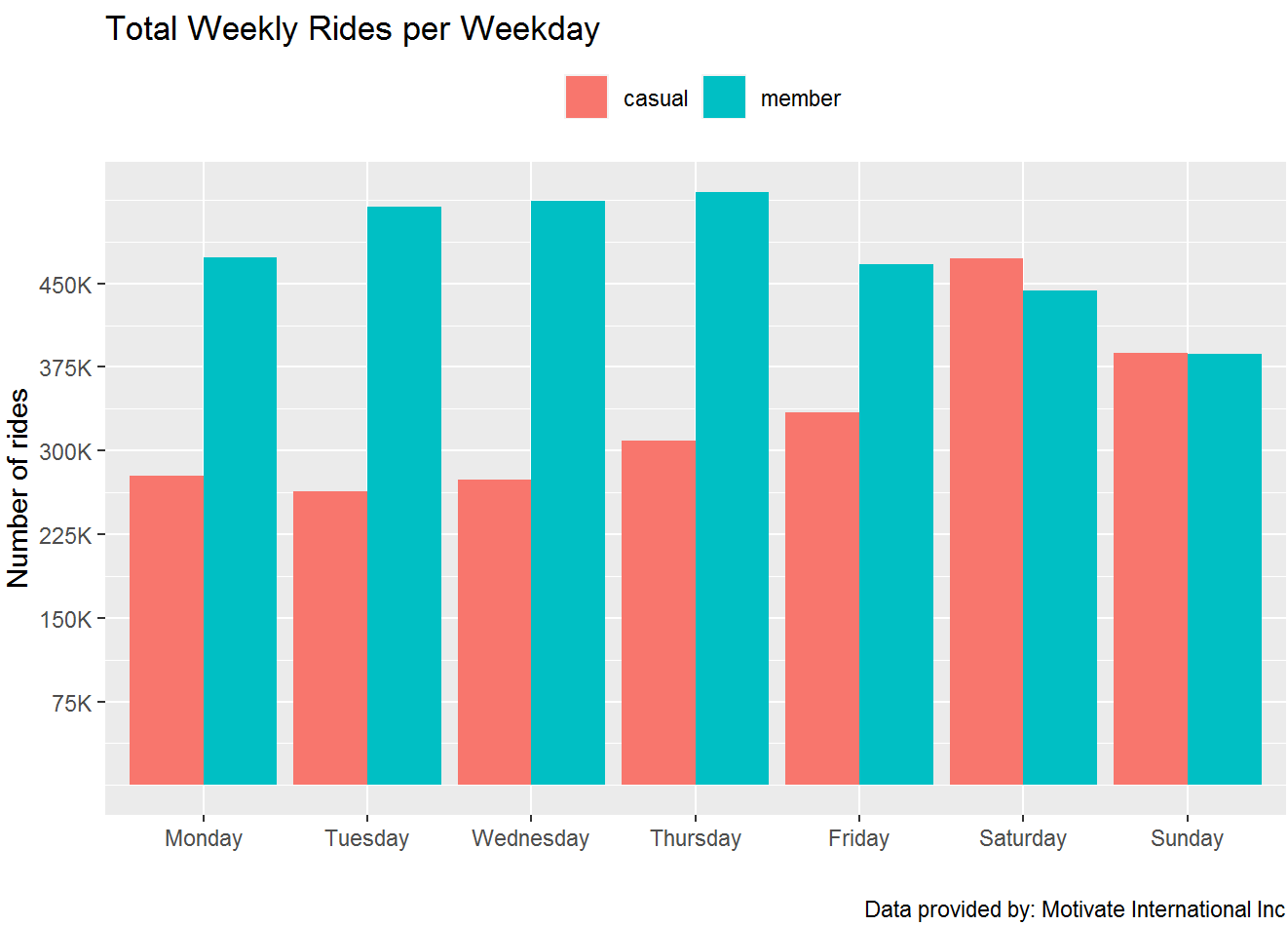
- **Secondly, weekly rides, we will view how the numbers of riders differ by the day of the week across our data set**
- We will create a table for the data summary we will obtain using summarize(), We will group our results by the weekdat at which the trip started, and membership type.
- We already set an order for our week days to start on Monday so we won't need that here

```
trips_weekly <- trips_2022 %>% group_by(start_weekday, member_casual) %>%
  summarize(total_weekly_rides=n())
```

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

- **To produce the following plot we use the ggplot as follows**

```
trips_weekly %>% ggplot(aes(x=start_weekday, y=total_weekly_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title="Total Weekly Rides per Weekday",
       caption="Data provided by: Motivate International Inc", x="", y="Number of rides") +
  guides(fill=guide_legend(title="")) +
  theme(legend.position="top") +
  scale_y_continuous(breaks = c(75000, 150000, 225000, 300000, 375000, 450000),
                     labels=c("75K", "150K", "225K", "300K", "375K", "450K"))
```



- **Now, to take a look at the rides started during each hour of the day**
- We will create a table for the data summary we will obtain using summarize(), We will group our results by the starting hour at which the trip started, and membership type.
- Our data is using 24-hour format so no ordering is required here

```
trips_hourly <- trips_2022 %>% group_by(start_hour, member_casual) %>%
  summarize(trips_at_hour=n())
```

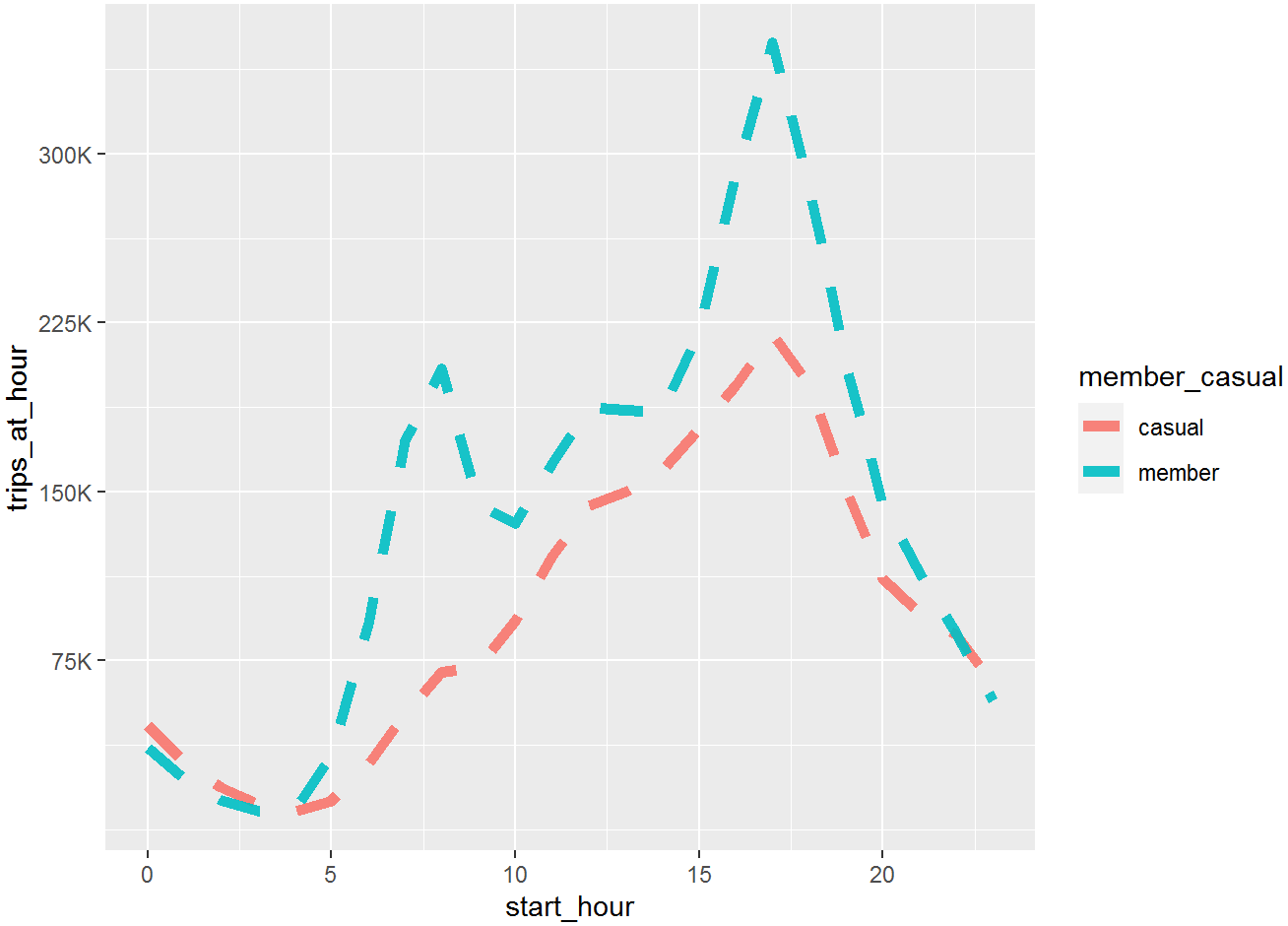
```
## `summarise()` has grouped output by 'start_hour'. You can override using the
## `.groups` argument.
```

- **To produce the plot we use the ggplot as follows**

```
trips_hourly %>% ggplot(aes(x=start_hour, y=trips_at_hour, color=member_casual)) +
  geom_line(position = "dodge", size=2, alpha=0.9, linetype=2) +
  scale_y_continuous(breaks = c(75000, 150000, 225000, 300000, 375000, 450000),
                     labels=c("75K", "150K", "225K", "300K", "375K", "450K"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```

```
## Warning: Width not defined
## i Set with `position_dodge(width = ...)`
```



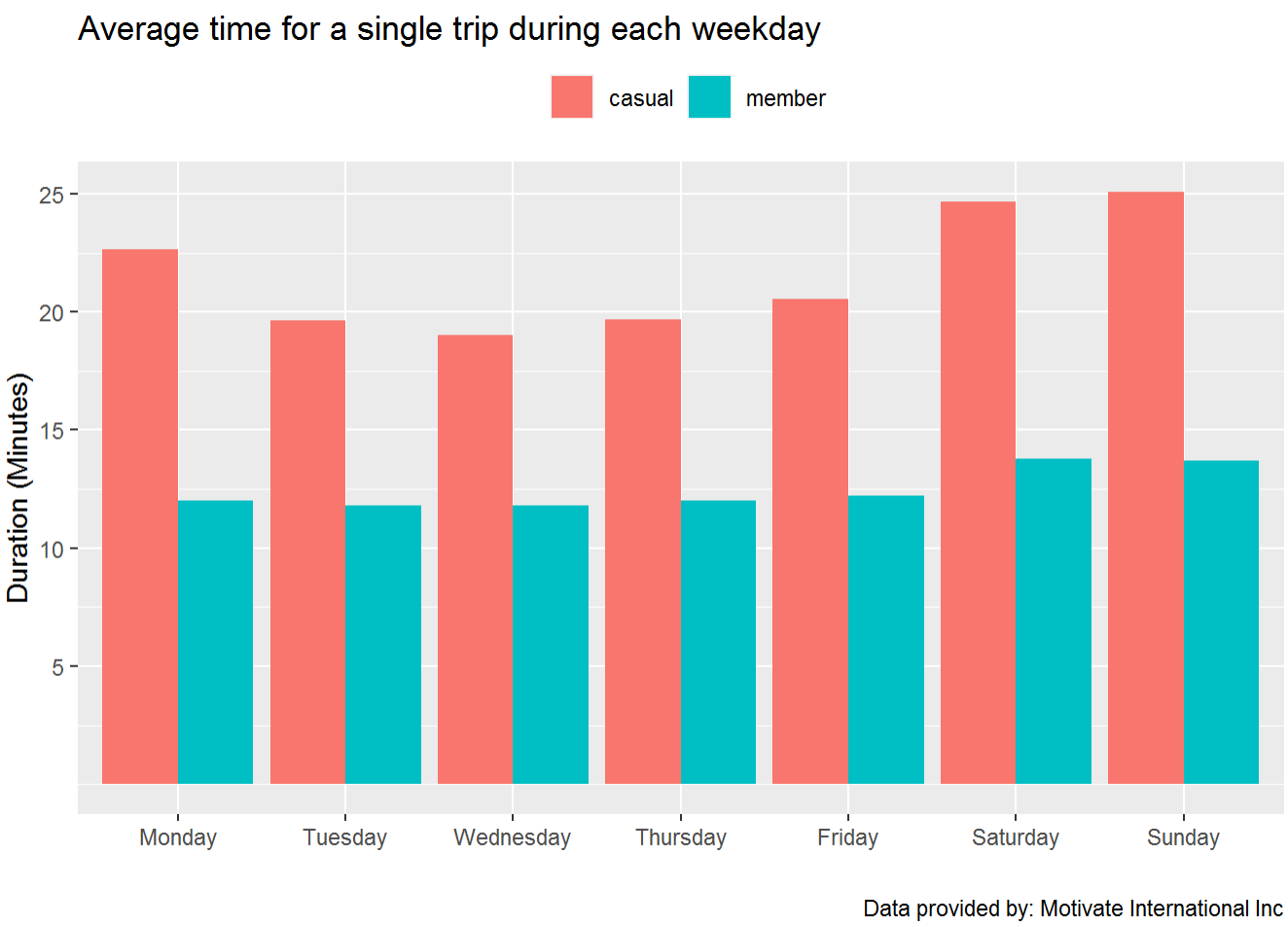
- **Finally, we take a look at the average ride duration during each day of the week**
- We will create a table for the data summary we will obtain using summarize(), We will group our results by the day of the week at which the trip started, and membership type.
- We already set an order for our week to start on monday so no ordering is required here

```
trips_average <- trips_2022 %>% group_by(start_weekday, member_casual) %>%
  summarize(average_trip_mins=mean(trip_duration_secs/60))
```

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

- **To produce the plot we use the ggplot as follows**

```
trips_average %>% ggplot(aes(x=start_weekday, y=average_trip_mins, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title="Average time for a single trip during each weekday",
       caption="Data provided by: Motivate International Inc", x="", y="Duration (Minutes)") +
  guides(fill=guide_legend(title="")) +
  theme(legend.position="top") +
  scale_y_continuous(breaks = c(5, 10, 15, 20, 25, 30, 35),
                     labels=c("5", "10", "15", "20", "25", "30", "35"))
```



Key Findings

1. The number of trips from both user bases increase during the Warm Season in Chicago (June 3rd to September 20th).
2. Casual Riders increases big time more than the number of Members' trips during weekends.
3. The peak number of trips from both Members and Casual Riders occurs during Chicago rush hour (Chicago Rush Hour historically occurs between 7 a.m. to 9 a.m. in the morning and *15 p.m. to 19 p.m in the evening
4. Casual Riders spend an average 50% more time per each trip.

Recomendations

1. The marketing strategy should focus more on the warm season period, since it has the highest trip counts
2. Introducing new pricing plans: A weekend pass would be good initiative to make casual riders become members and buy memberships
3. The average trip time for casual riders is +35min on weekends so a tier of 45min or above is reccomended
4. Use push notifications during rush hours suggesting using a bike today, and highlight or offer competitive rates during these periods.
5. Highlighting the benefits of the different passes to riders with longer trip durations; for example, highlight the price adjustments if Casual Riders were members.

Considerations

- **More data points can be needed to further enhance the scope of the analysis, such as:**
- **Information about the service users** (Gender, Age, Physical health; This can enhance the marketing strategy and improve understanding of the targeted demographic group)
- **Information about the premium plans* (Prices, Tiers, Student incentives; These data points can allow us to better understand potential reasons why users gravitate towards a plan versus another, and the potential growth aspects)

Thank you for you time and patience; any feedback is appreciated. A link of a Google Slides presentation will be linked here to tell the story of our data and findings.