Bike Study

Lucas F

2023-02-20

Dude Where's my Bike?

The setting: I have been tasked by the fictional copany "Cyclistic" to investigate the difference between how members(customers with annual subcriptons) and casual riders (customers who purchase single use or day passes) are utilizing an e-bike service in Chicago. This data will be used (fictionally) to help the marketing team develop new strategies to convert casual riders into members.

The data: The data used will be the previous 12 months of bike-share data collected by the company. This data contains information about ride times, member types, trip dates, bike type and station location information. This data does not include any personal information, so I will not be able to compare the number of trips in a period by unique customers within member types.

Setting up the data in R Studio

I decided to use R because it included the organizing, cleaning and manipulation functionality that I want for this project, that I would use multiple other applications for otherwise. I will start by setting up my environment and combining the 12 months of separate CSV files into a single table that I can work with.

```
#setting up environment
setwd("C:\\Users\\lucas\\Desktop\\Data Analysis\\BikeData\\CSV files")
install.packages('tinytex', repos = "http://cran.us.r-project.org")
install.packages('tidyverse', repos = "http://cran.us.r-project.org")
install.packages('hydroTSM', repos = "http://cran.us.r-project.org")
library(hydroTSM)
library(tidyverse)
library(ggplot2)
library(dplyr)
library(lubridate)

#import & merge data files
Data <- "C:\\Users\\lucas\\Desktop\\Data Analysis\\BikeData\\CSV files"

bike_data_merge <- list.files(path = Data) %>%
  lapply(read_csv) %>%
  bind_rows
```

Now with the data imported and merged, lets take a look at it to make sure everything is organized correctly.

```
#Gotta take a look at that sweet sweet data
head(bike_data_merge)
```

```
## # A tibble: 6 x 13
##   ride_id       ridea~1 started_at          ended_at            start~2 start~3
##   <chr>         <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 E1E065E7ED285~ classi~ 2022-02-19 18:08:41 2022-02-19 18:23:56 State ~ TA1305~
## 2 1602DCDC5B30F~ classi~ 2022-02-20 17:41:30 2022-02-20 17:45:56 Halste~ TA1309~
## 3 BE7DD2AF4B55C~ classi~ 2022-02-25 18:55:56 2022-02-25 19:09:34 State ~ TA1305~
## 4 A1789BDF84441~ classi~ 2022-02-14 11:57:03 2022-02-14 12:04:00 Southp~ 13235
## 5 07DE78092C62F~ classi~ 2022-02-16 05:36:06 2022-02-16 05:39:00 State ~ TA1305~
## 6 9A2F204F04AB7~ classi~ 2022-02-07 09:51:57 2022-02-07 10:07:53 St. Cl~ 13016
## # ... with 7 more variables: end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, and abbreviated variable names 1: rideable_type,
## #   2: start_station_name, 3: start_station_id
```

```
tail(bike_data_merge)
```

```
## # A tibble: 6 x 13
##   ride_id       ridea~1 started_at          ended_at            start~2 start~3
##   <chr>         <chr>   <dttm>              <dttm>              <chr>   <chr>
## 1 A3DC3E8358DB1~ electr~ 2023-01-17 18:36:00 2023-01-17 19:00:26 Clark ~ TA1307~
## 2 A303816F2E8A3~ electr~ 2023-01-11 17:46:23 2023-01-11 17:57:31 Clark ~ TA1307~
## 3 BCDBB142CC610~ classi~ 2023-01-30 15:08:10 2023-01-30 15:33:26 Wester~ TA1307~
## 4 7D1C7CA805171~ classi~ 2023-01-06 19:34:50 2023-01-06 19:50:01 Clark ~ TA1307~
## 5 1A4EB636346DF~ classi~ 2023-01-13 18:59:24 2023-01-13 19:14:44 Clark ~ TA1307~
## 6 069971675AC7D~ electr~ 2023-01-02 13:48:29 2023-01-02 13:59:29 Clark ~ TA1307~
## # ... with 7 more variables: end_station_name <chr>, end_station_id <chr>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>, and abbreviated variable names 1: rideable_type,
## #   2: start_station_name, 3: start_station_id
```

```
colnames(bike_data_merge)
```

```
##  [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"
```

```
str(bike_data_merge)
```

```
## spc_tbl_ [5,754,248 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5754248] "E1E065E7ED285C02" "1602DCDC5B30FFE3" "BE7DD2AF4B55C4AF" "A178
##  $ rideable_type     : chr [1:5754248] "classic_bike" "classic_bike" "classic_bike" "classic_bike" .
##  $ started_at        : POSIXct[1:5754248], format: "2022-02-19 18:08:41" "2022-02-20 17:41:30" ...
##  $ ended_at          : POSIXct[1:5754248], format: "2022-02-19 18:23:56" "2022-02-20 17:45:56" ...
##  $ start_station_name: chr [1:5754248] "State St & Randolph St" "Halsted St & Wrightwood Ave" "State
##  $ start_station_id  : chr [1:5754248] "TA1305000029" "TA1309000061" "TA1305000029" "13235" ...
##  $ end_station_name  : chr [1:5754248] "Clark St & Lincoln Ave" "Southport Ave & Wrightwood Ave" "Car
##  $ end_station_id    : chr [1:5754248] "13179" "TA1307000113" "13011" "13323" ...
##  $ start_lat         : num [1:5754248] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:5754248] -87.6 -87.6 -87.6 -87.7 -87.6 ...
##  $ end_lat           : num [1:5754248] 41.9 41.9 41.9 42 41.9 ...
##  $ end_lng           : num [1:5754248] -87.6 -87.7 -87.6 -87.6 -87.6 ...
##  $ member_casual     : chr [1:5754248] "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>
```

Now we have a few different views of the data to see how the table is organized, the different data types currently assigned to each column, how the columns are named..etc.

This is where I will start cleaning/renaming/organizing the data so that I can work with it more efficiently and without worrying about inaccurate data getting in the way of my analysis,

```
#Creating a column for ride length, and month/day/year/day_of_week for future use
bike_data_add <- mutate(bike_data_merge,
                        Ride_Time = difftime(ended_at,
                                             started_at,
                                             units = "secs"))
```

```
#Removing columns I don't need
data_trimmed <- subset(bike_data_add, select = -c(start_station_id,
                                                  end_station_name,
                                                  end_station_id,
                                                  start_lat,
                                                  start_lng,
                                                  end_lat, end_lng))
```

```
#Renaming columns to be a bit easier to work with.
data_renamed <- rename(data_trimmed,
                       ID = ride_id,
                       Type = rideable_type,
                       Start_Date = started_at,
                       End_Date = ended_at,
                       Member_Type = member_casual)
```

```
#getting rid of NA values, rides with 0 or negative durations and any
#stations that are actually the QA recall designation.
data_cleaned <- data_renamed[!(data_renamed$start_station_name == "HQ QR" |
                                 data_renamed$Ride_Time <= 0 |
                                 data_renamed$Ride_Time == "NA"),] %>%
                                 arrange(Member_Type)
```

The next thing I want to do is create a few columns for ***date***, ***month***, ***day***, ***year*** and ***day of week***.

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

At this point, lets take a look at the table again to make sure all of these changes have worked correctly.

```
#Just looking at the sweet sweet data again, I can do what I want
str(data_cleaned)
```

```
## tibble [5,753,714 x 12] (S3: tbl_df/tbl/data.frame)
##  $ ID                : chr [1:5753714] "BA1742BD29B72980" "75AEFCD8AF0C5E05" "C920B5A0FE63CF76" "D0C
##  $ Type              : chr [1:5753714] "classic_bike" "docked_bike" "docked_bike" "classic_bike" ...
##  $ Start_Date        : POSIXct[1:5753714], format: "2022-02-28 16:27:51" "2022-02-10 15:33:04" ...
##  $ End_Date          : POSIXct[1:5753714], format: "2022-02-28 16:34:16" "2022-02-10 16:17:39" ...
##  $ start_station_name: chr [1:5753714] "State St & Randolph St" "Wacker Dr & Washington St" "Wacker 
##  $ Member_Type       : chr [1:5753714] "casual" "casual" "casual" "casual" ...
##  $ Ride_Time         : ’difftime’ num [1:5753714] 385 2675 1075 885 ...
##   ..- attr(*, "units")= chr "secs"
##  $ date              : Date[1:5753714], format: "2022-02-28" "2022-02-10" ...
##  $ month             : chr [1:5753714] "02" "02" "02" "02" ...
##  $ day               : chr [1:5753714] "28" "10" "21" "20" ...
##  $ year              : chr [1:5753714] "2022" "2022" "2022" "2022" ...
##  $ day_of_week       : chr [1:5753714] "Monday" "Thursday" "Monday" "Sunday" ...
```

```
head(data_cleaned)
```

```
## # A tibble: 6 x 12
##   ID       Type  Start_Date          End_Date            start~1 Membe~2 Ride_~3
##   <chr>    <chr> <dttm>              <dttm>              <chr>   <chr>   <drtn>
## 1 BA1742B~ clas~ 2022-02-28 16:27:51 2022-02-28 16:34:16 State ~ casual   385 s~
## 2 75AEFCD~ dock~ 2022-02-10 15:33:04 2022-02-10 16:17:39 Wacker~ casual  2675 s~
## 3 C920B5A~ dock~ 2022-02-21 11:38:50 2022-02-21 11:56:45 Wacker~ casual  1075 s~
## 4 D0CC48B~ clas~ 2022-02-20 19:20:11 2022-02-20 19:34:56 Frankl~ casual   885 s~
## 5 0AB6272~ clas~ 2022-02-28 08:16:19 2022-02-28 08:23:08 Michig~ casual   409 s~
## 6 071345F~ clas~ 2022-02-27 15:23:15 2022-02-27 15:39:49 Halste~ casual   994 s~
## # ... with 5 more variables: date <date>, month <chr>, day <chr>, year <chr>,
## #   day_of_week <chr>, and abbreviated variable names 1: start_station_name,
## #   2: Member_Type, 3: Ride_Time
```

```
tail(data_cleaned)
```

```
## # A tibble: 6 x 12
##   ID    Type  Start_Date End_Date start_station_name Member_Type Ride_Time
##   <chr> <chr> <dttm>     <dttm>   <chr>              <chr>       <drtn>
## 1 <NA>  <NA>  NA         NA       <NA>               <NA>        NA secs
## 2 <NA>  <NA>  NA         NA       <NA>               <NA>        NA secs
## 3 <NA>  <NA>  NA         NA       <NA>               <NA>        NA secs
## 4 <NA>  <NA>  NA          NA       <NA>               <NA>        NA secs
## 5 <NA>  <NA>  NA          NA       <NA>               <NA>        NA secs
## 6 <NA>  <NA>  NA          NA       <NA>               <NA>        NA secs
## # ... with 5 more variables: date <date>, month <chr>, day <chr>, year <chr>,
## #   day_of_week <chr>
```

Now I just want to make a small data type adjustment and sweep for NA values.

```
#clarifying a data type to make sure it can be used for calculations
data_cleaned$Ride_Time <- as.numeric(as.character(data_cleaned$Ride_Time))

#clearing up rows with NA values and confirming they are gone
data_final <- na.omit(data_cleaned)

#Checking to make sure it worked.
sum(is.na(data_final$Ride_Time))
```

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

Lets start looking at some numbers!

Alright, now that the data has been wrangled and cleaned, we can start running some calculations to see what it tells us about the difference between members and non members. The following chunk will display the values by rider type for mean, median, min and max ride times.

```
aggregate(data_final$Ride_Time ~ data_final$Member_Type, FUN = mean)
```

```
##   data_final$Member_Type data_final$Ride_Time
## 1                 casual            1893.7953
## 2                 member             764.6686
```

```
aggregate(data_final$Ride_Time ~ data_final$Member_Type, FUN = median)
```

```
##   data_final$Member_Type data_final$Ride_Time
## 1                 casual                  809
## 2                 member                  532
```

```
aggregate(data_final$Ride_Time ~ data_final$Member_Type, FUN = max)
```

```
##   data_final$Member_Type data_final$Ride_Time
## 1                 casual              2483235
## 2                 member                93594
```

```
aggregate(data_final$Ride_Time ~ data_final$Member_Type, FUN = min)
```

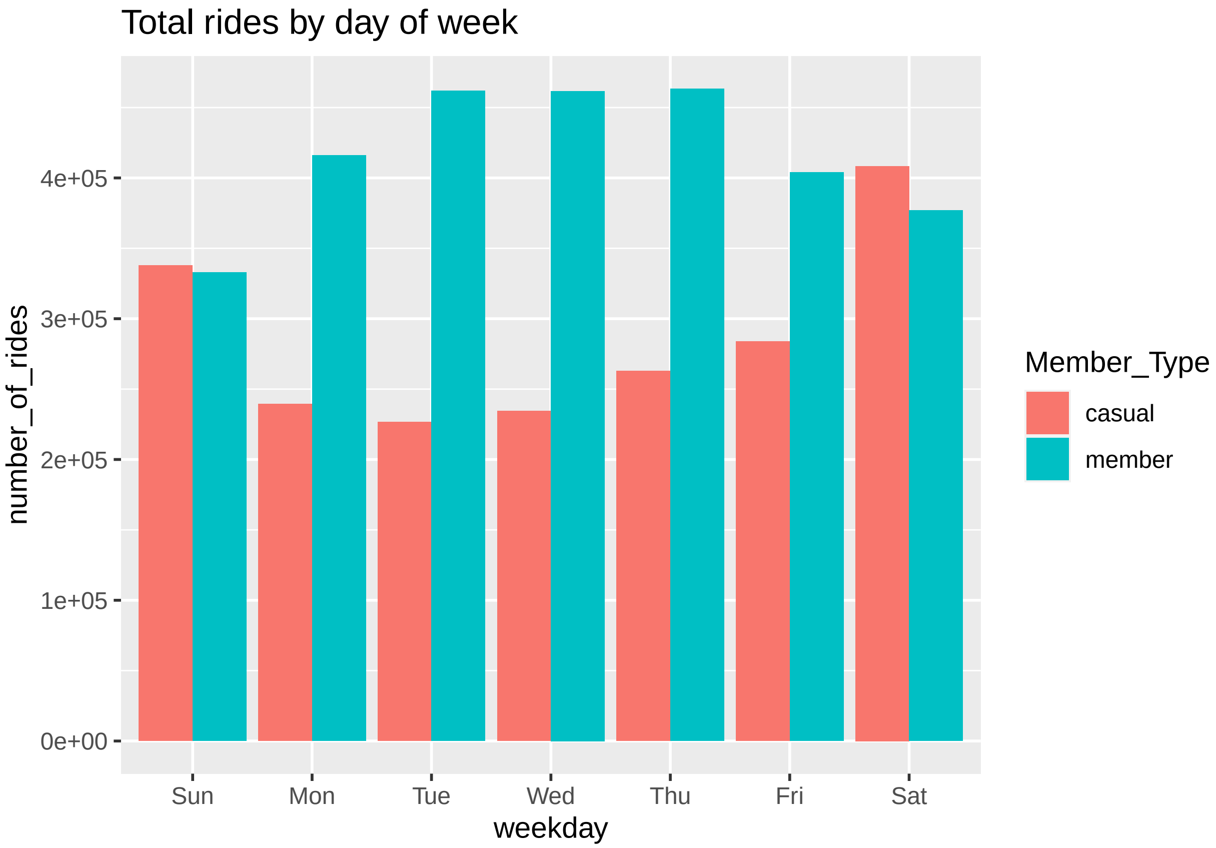
```
##   data_final$Member_Type data_final$Ride_Time
## 1                 casual                    1
## 2                 member                    1
```

So what we initially see here is that generally speaking; casual riders have longer ride times for both mean, median and max when compared to members, both having the same minimum ride time of 1 second (This may warrant looking into as well separately).

Visualizing this data helps to put it into a better perspective, especially when we start looking at the data on a weekday basis.

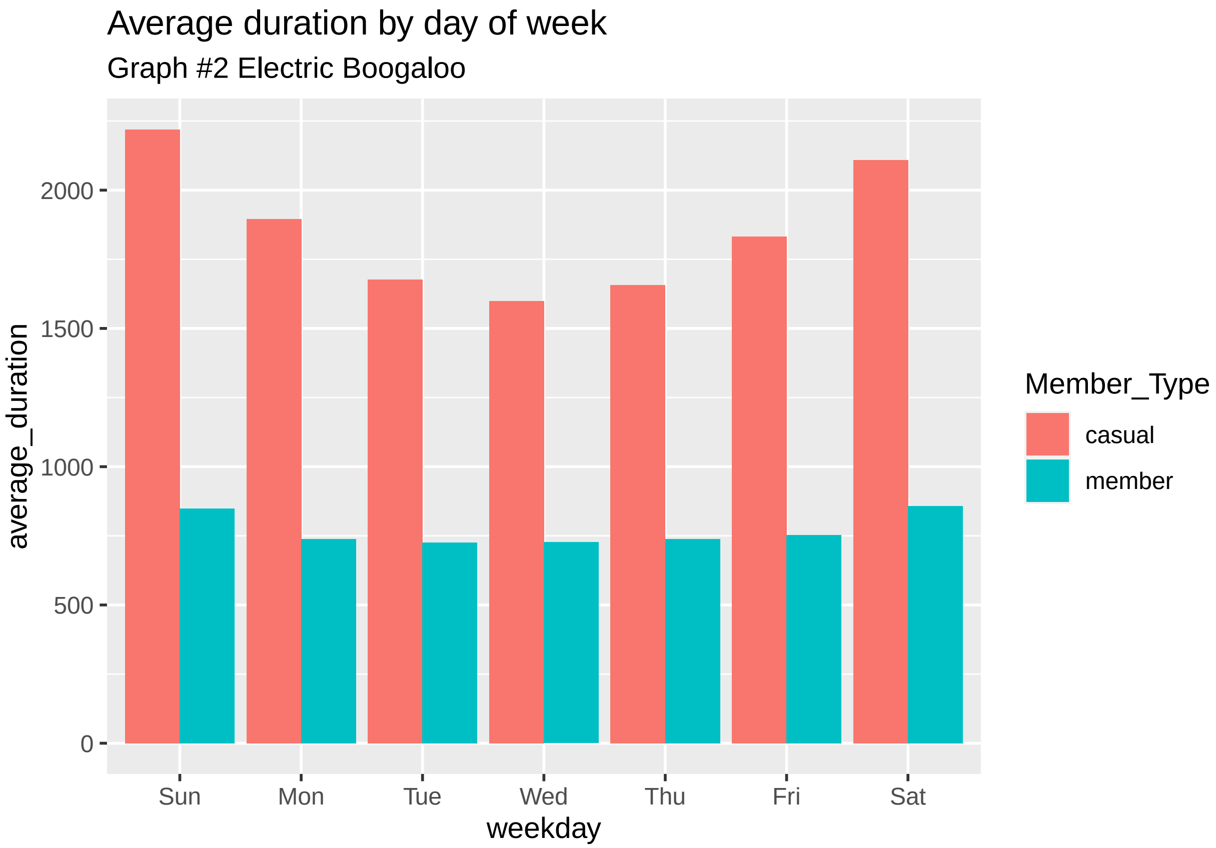
```
data_final %>%
  mutate(weekday = wday(Start_Date, label = TRUE)) %>%
  group_by(Member_Type, weekday) %>%
  summarise(number_of_rides = n()) %>%
  arrange(Member_Type, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = Member_Type)) +
  geom_col(position = "dodge")+
  labs(title = 'Total rides by day of week')
```

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



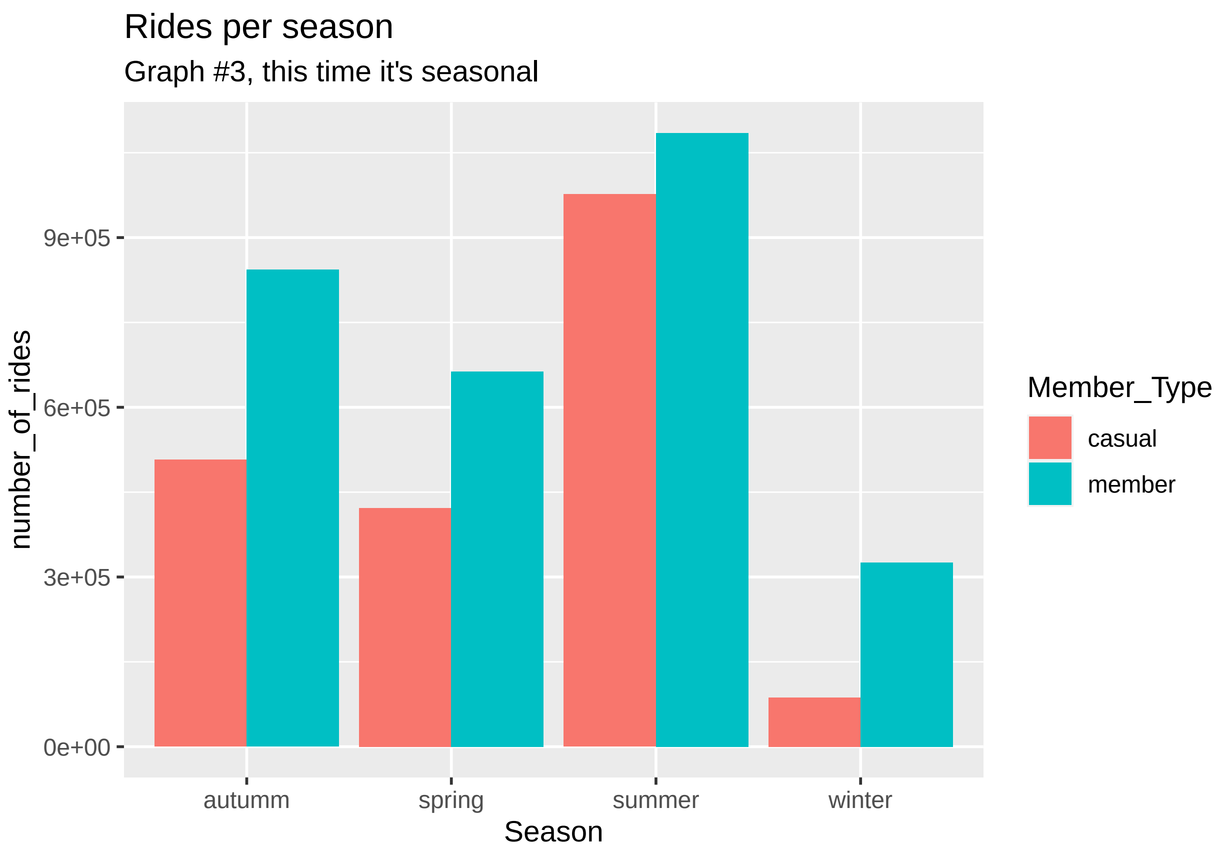
```
data_final %>%
  mutate(weekday = wday(Start_Date, label = TRUE)) %>%
  group_by(Member_Type, weekday) %>%
  summarise(average_duration = mean(Ride_Time)) %>%
  arrange(Member_Type, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = Member_Type)) +
  geom_col(position = "dodge")+
  labs(title = 'Average duration by day of week', subtitle =
        'Graph #2 Electric Boogaloo')
```

```
## ‘summarise()‘ has grouped output by ’Member_Type’. You can override using the
## ‘.groups‘ argument.
```



```
data_final %>%
  mutate(Season = time2season(Start_Date, out.fmt = "seasons")) %>%
  group_by(Member_Type, Season) %>%
  summarise(number_of_rides = n()) %>%
  arrange(Member_Type, Season) %>%
  ggplot(mapping = aes(x = Season, y = number_of_rides, fill = Member_Type))+
  geom_col(position = "dodge")+
  labs(title = 'Rides per season', subtitle = "Graph #3, this time it's seasonal")
```

```
## ‘summarise()‘ has grouped output by ’Member_Type’. You can override using the
## ‘.groups‘ argument.
```



## Summary

From these graphs we can conclude that Members have a higher usage of the bike-share service during weekdays, most likely for commuting to work, only barely outpaced on the weekend by casual riders.

We see that the member trip duration is also very consistent throughout the week which supports the commuting conclusion.

On the other hand, we have the casual riders taking longer rides compared to members with the longest trips being closer to the weekends. This indicates that casual riders are using the service less for a specific recurring purpose and more for leisure travel.

The final observation is that the number of casual riders sharply drops off during winter months compared to a less stark drop off for members.

Recommendations

- Focus marketing strategies towards weekend and leisure riders to inform of membership benefits.
- Consider stations purposefully connected to popular biking trails and parks to increase visibility of the service to those riders.
- New membership tier specifically for seasonal use at a reduced rate from the regular membership.