Cyclistic Notebook

2024-08-16

Introduction

The company Cyclistic bases their information off of Divvy. In this case study, I am tasked to analyze Q1 2019 and Q1 2020 data to find patterns on how to convert casual (free, no membership) riders to members (a yearly paid fee). I had to trim some extra data out to meet posit.cloud's free trial limit of 1GB RAM when knitting and running.

Make sure you run this code to get the necessary functions!

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("conflicted")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readr)
library(conflicted)

conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
conflict_prefer("lag", "dplyr")
```

```
## [conflicted] Will prefer dplyr::lag over any other package.
```

Load the Datasets and check column names to see if they match

```
divvy_2019 <- read.csv("Divvy_Trips_2019_Q1.csv")
divvy_2020 <- read.csv("Divvy_Trips_2020_Q1.csv")
colnames(divvy_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(divvy_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"
```

Data Cleaning

```
# make sure to check afterwards if the column names match!
(divvy_2019 <- rename(divvy_2019,
                  ride_id = trip_id,
                  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))


#convert data-type
divvy_2019 <- mutate(divvy_2019, ride_id = as.character(ride_id),
                     rideable_type = as.character(rideable_type))

#stack now that the data is cleaned & matching datatypes
all_trips <- bind_rows(divvy_2019, divvy_2020)

#remove cols to make it even, "-c" means to exclude
all_trips <- all_trips %>%
  select(-c(start_lat, start_lng, end_lat, end_lng, birthyear, gender, "tripduration"))
```

Check Results

```
table(all_trips$member_casual)
```

```
##
##     casual   Customer     member Subscriber
##      48480      23163     378407     341906
```

```
#member_casual has 4 different values, needs to be consolidated into two.
```

```r
all_trips <- all_trips %>%
  mutate(member_casual = recode(member_casual, "Subscriber" = "member", "Customer" = "casual"))
```

Adding date and ride_length so that I can view values over time on a graph

```r
all_trips$date <- as.Date(all_trips$started_at)

all_trips$month <- format(as.Date(all_trips$date), "%m")

all_trips$day <- format(as.Date(all_trips$date), "%d")

all_trips$year <- format(as.Date(all_trips$date), "%Y")

all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")


all_trips$ride_length <- difftime(all_trips$ended_at, all_trips$started_at)


#Make sure the data-type is correct for future use
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length)
```

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

More Data Cleaning – Remove invalid values (negative ride length, or when rides went back to HQ)

```r
all_trips_v2 <- all_trips[!(all_trips$start_station_name == "HQ QR" | all_trips$ride_length<0),]
```

Analysis: C = Casual, M = Member.

```r
#value ~ type, find whatever FUN =, in this case, mean of ride length per member_casual. ride_length = 

#C - 5373, M - 795
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean)
#C - 1393, M - 508
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median)
#C - 10,632,022 M - 6,096,428
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max)
#C - 2, M - 1
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min)


#average ride_length depending on day of week per member-type
#lowest: C Tuesday @ 4561.8039, M Thursday @ 707.2093
#highest: C Thursday @ 8451.6669, M Saturday @ 974.0730
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week,
          FUN = mean)
```

```r
#chronological ordering
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday",
```

```
                                                                    "Tuesday", "Wednesday",
                                                                    "Thursday", "Friday",
                                                                    "Saturday"))


aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week,
          FUN = mean)
```

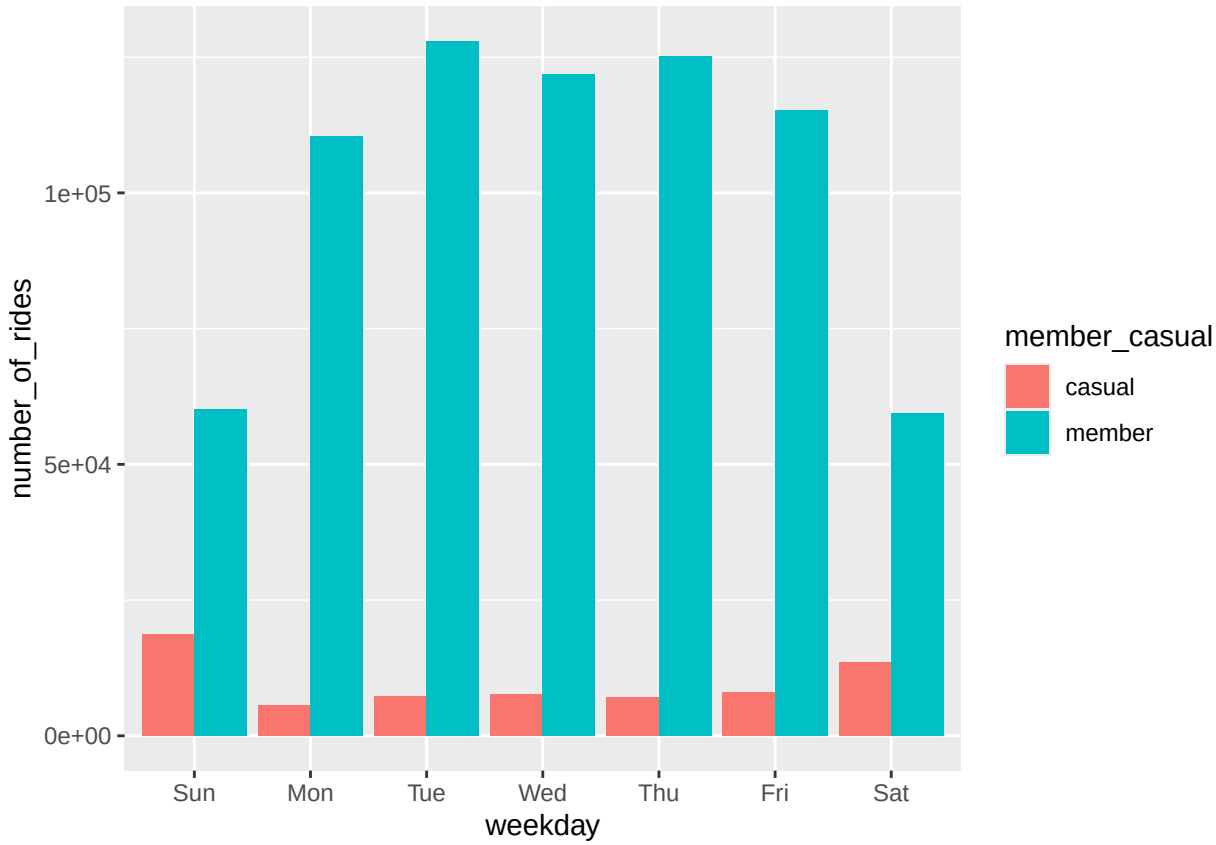
```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Sunday                5061.3044
## 2                      member                   Sunday                 972.9383
## 3                      casual                   Monday                4752.0504
## 4                      member                   Monday                 822.3112
## 5                      casual                  Tuesday                4561.8039
## 6                      member                  Tuesday                 769.4416
## 7                      casual                Wednesday                4480.3724
## 8                      member                Wednesday                 711.9838
## 9                      casual                 Thursday                8451.6669
## 10                     member                 Thursday                 707.2093
## 11                     casual                   Friday                6090.7373
## 12                     member                   Friday                 796.7338
## 13                     casual                 Saturday                4950.7708
## 14                     member                 Saturday                 974.0730
```

Graphing

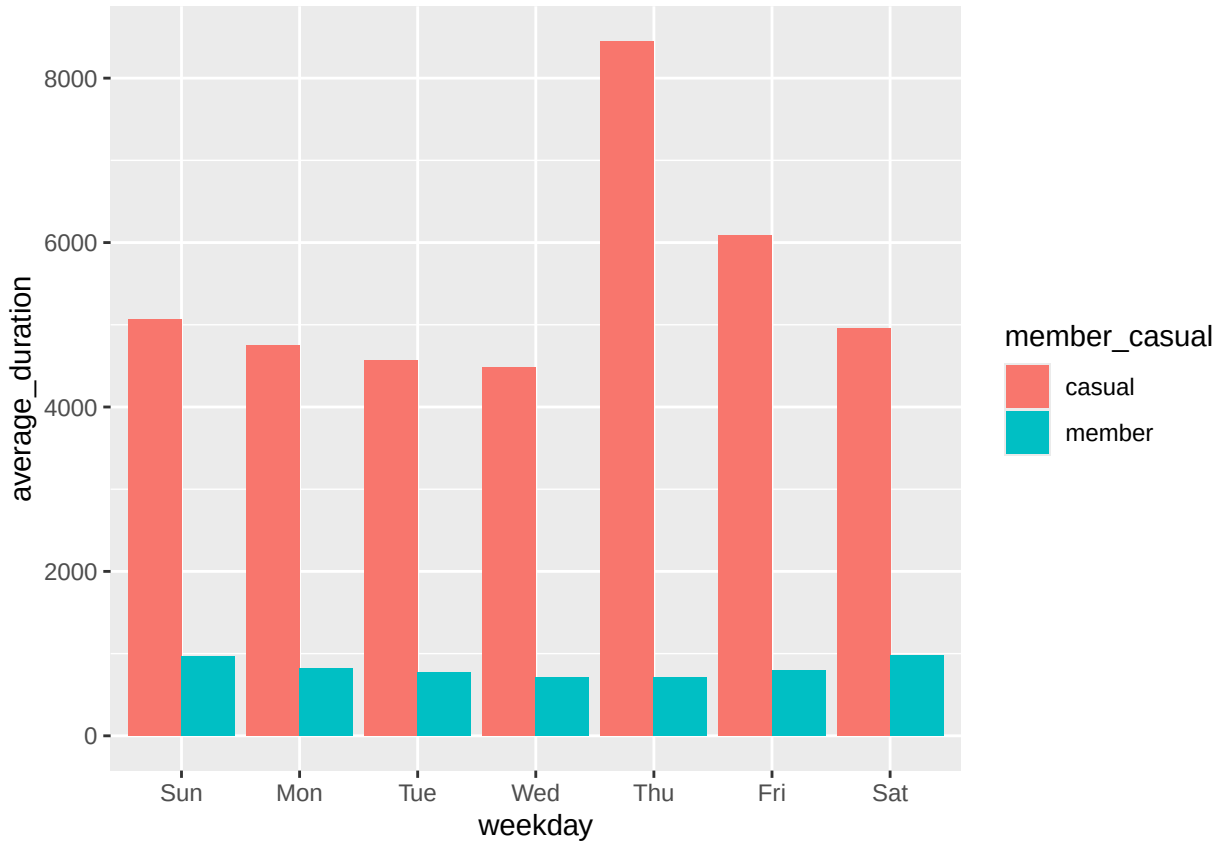
```
graph_trips <- all_trips_v2 %>%
  #makes weekday col using started_at, TRUE = return name of day in full
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  #groups by usertype and weekday
  group_by(member_casual, weekday) %>%
  #n() counts number of rows
  summarize(number_of_rides = n(),
  average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday)
```

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

```
graph_trips %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")
```




```
graph_trips %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge")
```



Export for further Analysis in Tableau

```
counts <- aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual +
                      all_trips_v2$day_of_week, FUN = mean)
#for some reason my code of write.csv(counts, "C://Users/me/Desktop/myfile.csv") does not work,
#alternative is use another device to try to download, but I will upload to cloud,
#then just export it.
write.csv(counts, "/cloud/project/avg_ride_length.csv")


counts_2 <- aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual +
                      all_trips_v2$day_of_week, FUN = length)

write.csv(counts_2, "/cloud/project/ride_counts.csv")
```