Cyclistic Capstone

leab38

2022-09-05

Cyclistic Data Analysis Capstone Project

This is the capstone data analysis for the Cyclistic project as part of the Google Data Analytics Professional Certificate on Coursera.

```
library(tidyverse)
library(lubridate)
library(scales)
library(tinytex)
```

Executive Summary

The structure of this report will follow the 6-stage data analysis process outlined by Google in the course. Those stages are: 1. Ask 2. Prepare 3. Process 4. Analyze 5. Share 6. Act

Also, for those who want a quick tl;dr without reading through the full report, my summary of findings are as follows:

1. As expected, members ride more consistently throughout the week and throughout the year. With greatest ridership from March - November and Tuesday - Thursday.
2. Also, as expected, casual riders are most often riding on the weekends and in the summer months.
3. Interestingly, casual riders have a much longer average ride length than members.

Recommendation: Consider offering different types of memberships that capture this alternate usage of the bikes to encourage casual users to become regular members.

I go into greater detail on the findings and recommendations in the section labeled "Act".

Ask

The Ask stage is designed to understand what questions are we trying to answer with the data. According to the project as defined by the case study, the following questions will guide the future marketing program: 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 data analysis is expected to answer the first question only. "How do annual members and casual riders use Cyclistic bikes differently?"

Prepare

The Prepare stage focuses on the data to be used. We need to understand where the data is located, how it is organized, if there are any issues with the data. In this case study, the data has been provided by the "Cyclistic" organization.

###Download / store data #### Create file name function In previous years, the data was provided on a quarterly basis. However, for 2021-2022, to gather the last 12 months of data, 12 zip files with 12 csv files must be downloaded. In order to simplify the import of 12 csv files worth of data, I created a function to build the filename, so that I can iterate on the files to import them to R.

To repeat this in your own environment, make sure to update "filefolder" to be the file location where the unzipped folders are stored. The zip files must be unzipped (on Windows) before the below will run successfully.

```r
# Function to create the file name once we know the date (yyyymm)
create_filename <- function(date) {
  #folder variable located in hidden chunk
  filefolder=folder
  foldername=paste(date,"-divvy-tripdata",sep = "")
  filename=paste(foldername,".csv",sep = "")
  filelocation=paste(filefolder,foldername,"/",filename,sep="")
  return(filelocation)
}
```

Iterate to import files Using a for loop with a start point (mine was August 2021), iterate over 12 files to import into R.

```r
# Initialize year/month for starting file
year=2021
month=08

# Iterate over files - remember they need to be unzipped

for (i in 1:12) {
  if (month < 10) {
    month = paste('0',month,sep = '')
  }
  date=paste(year,month,sep="")
  if (i==1) {
    df_name <- list(paste0('cycle_data_',month))
    assign(paste0('cycle_data_',month),read_csv(create_filename(date)))
    } else {
      df_name<-append(df_name,paste0('cycle_data_',month))
      assign(paste0('cycle_data_',month),read_csv(create_filename(date)))
      }

  # Adjust month/year, based on December/not-December
  month=as.integer(month)
  month
  if (month==12) {
    year = year + 1
    month = 1
  } else {
    month = month+1
  }
}
print("File import complete!")
```

```
## [1] "File import complete!"
```

Identify how the data is organized part I

Another step of the Prepare phase is to identify how the data is organized. To do this, we look at the structure of the data. Since we have 12 files that need to be merged, the first thing I want to look at is the column names and compare to see if we have any differences between them. We got lucky on this one.. the files have all the same columns with the same names that means we can easily merge the data frames into a single data frame.

```
# Print column names for all existing data frames
for (name in df_name){
  print(colnames(eval(parse(text=name))))
}
```

```
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
##  [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"
```

Merge the data frames

```r
# Merge data frames into single data frame
i=1
for (name in df_name){
  if (i==1){
    all_trips<-eval(parse(text=name))
  } else {
    all_trips<-bind_rows(all_trips,eval(parse(text=name)))
  }
  i=i+1
}
```

Identify how the data is organized part II

Since I was starting with 12 separate data frames, I prioritized getting those data frames into one. Now we can do some more identifying. For this we're going to look at the structure and a summary of the columns to understand how the data frame is structured. The "str" function also lets us verify the data types assigned to each column, which will be useful later on in the analysis. I will also do a quick first look at total counts for members versus casual riders.

```r
# Verify basic information for all_trips
str(all_trips)
```

```
## spec_tbl_df [5,901,463 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5901463] "99103BB87CC6C1BB" "EAFCCCFB0A3FC5A1" "9EF4F46C57AD234D" "5834
##  $ rideable_type     : chr [1:5901463] "electric_bike" "electric_bike" "electric_bike" "electric_bik
##  $ started_at        : POSIXct[1:5901463], format: "2021-08-10 17:15:49" "2021-08-10 17:23:14" ...
##  $ ended_at          : POSIXct[1:5901463], format: "2021-08-10 17:22:44" "2021-08-10 17:39:24" ...
##  $ start_station_name: chr [1:5901463] NA NA NA NA ...
##  $ start_station_id  : chr [1:5901463] NA NA NA NA ...
##  $ end_station_name  : chr [1:5901463] NA NA NA NA ...
##  $ end_station_id    : chr [1:5901463] NA NA NA NA ...
##  $ start_lat         : num [1:5901463] 41.8 41.8 42 42 41.8 ...
##  $ start_lng         : num [1:5901463] -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num [1:5901463] 41.8 41.8 42 42 41.8 ...
##  $ end_lng           : num [1:5901463] -87.7 -87.6 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr [1:5901463] "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>
```

```r
summary(all_trips)
```

```
##    ride_id          rideable_type        started_at                
##  Length:5901463     Length:5901463     Min.   :2021-08-01 00:00:04.00  
##  Class :character   Class :character   1st Qu.:2021-09-27 12:35:12.50  
##  Mode  :character   Mode  :character   Median :2022-02-14 14:10:08.00  
##                                        Mean   :2022-01-31 21:50:42.24  
##                                        3rd Qu.:2022-06-05 15:29:40.50  
##                                        Max.   :2022-07-31 23:59:58.00  
##                                                                        
##     ended_at                      start_station_name start_station_id  
##  Min.   :2021-08-01 00:03:11.00   Length:5901463     Length:5901463    
##  1st Qu.:2021-09-27 12:54:02.50   Class :character   Class :character  
##  Median :2022-02-14 14:20:23.00   Mode  :character   Mode  :character  
##  Mean   :2022-01-31 22:10:35.61                                        
##  3rd Qu.:2022-06-05 15:54:48.00                                        
##  Max.   :2022-08-04 13:53:01.00                                        
```

```
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:5901463     Length:5901463     Min.   :41.64   Min.   :-87.84
##  Class  :character  Class  :character  1st Qu.:41.88   1st Qu.:-87.66
##  Mode   :character  Mode   :character  Median :41.90   Median :-87.64
##                                        Mean   :41.90   Mean   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :45.64   Max.   :-73.80
##
##     end_lat         end_lng       member_casual
##  Min.   :41.39   Min.   :-88.97   Length:5901463
##  1st Qu.:41.88   1st Qu.:-87.66   Class  :character
##  Median :41.90   Median :-87.64   Mode   :character
##  Mean   :41.90   Mean   :-87.65
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.37   Max.   :-87.50
##  NA’s   :5590    NA’s   :5590
```

```
colnames(all_trips)
```

```
##  [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"
```

```
# Look at values/counts for member_casual/rideable_type
all_trips %>%
  count(member_casual, rideable_type)
```

```
## # A tibble: 5 x 3
##   member_casual rideable_type       n
##   <chr>         <chr>           <int>
## 1 casual        classic_bike  1132892
## 2 casual        docked_bike    226728
## 3 casual        electric_bike 1162606
## 4 member        classic_bike  1922749
## 5 member        electric_bike 1456488
```

Process

The Process phase is when we choose our tools, verify the data’s integrity, ensure the data is clean and ready to analyze.

I chose to use R in order to take advantage of using R Markdown to create my report. This allows me to both clean and process the data and then visualize it for the “Share” phase.

To start off, I will create some calculated fields that will help later on in the analysis.

Calculated Fields

So, when looking at the data at the beginning, I knew I wanted to be able to group on datetime features like day of the week and month, as well as gather information on the duration of the rides being taken. To do this, I needed to create new columns with calculated fields, which took information from existing columns to make this information easier to filter.

```
# Create calculated field for day, month, year, day of the week
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")

# Create calculated field for length of ride in seconds
all_trips$ride_length<-difftime(all_trips$ended_at,all_trips$started_at)

# Convert "ride_length" from factor to numeric
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length)
```

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

Identify how the data is organized part III

Now that there are more columns, I need to re-verify the structure and take a glance at the aggregate function summary.

```
# Verify structure
str(all_trips)
```

```
## spec_tbl_df [5,901,463 x 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:5901463] "99103BB87CC6C1BB" "EAFCCCFB0A3FC5A1" "9EF4F46C57AD234D" "5834
##  $ rideable_type     : chr [1:5901463] "electric_bike" "electric_bike" "electric_bike" "electric_bike
##  $ started_at        : POSIXct[1:5901463], format: "2021-08-10 17:15:49" "2021-08-10 17:23:14" ...
##  $ ended_at          : POSIXct[1:5901463], format: "2021-08-10 17:22:44" "2021-08-10 17:39:24" ...
##  $ start_station_name: chr [1:5901463] NA NA NA NA ...
##  $ start_station_id  : chr [1:5901463] NA NA NA NA ...
##  $ end_station_name  : chr [1:5901463] NA NA NA NA ...
##  $ end_station_id    : chr [1:5901463] NA NA NA NA ...
##  $ start_lat         : num [1:5901463] 41.8 41.8 42 42 41.8 ...
##  $ start_lng         : num [1:5901463] -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num [1:5901463] 41.8 41.8 42 42 41.8 ...
##  $ end_lng           : num [1:5901463] -87.7 -87.6 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr [1:5901463] "member" "member" "member" "member" ...
##  $ date              : Date[1:5901463], format: "2021-08-10" "2021-08-10" ...
##  $ month             : chr [1:5901463] "08" "08" "08" "08" ...
##  $ day               : chr [1:5901463] "10" "10" "21" "21" ...
##  $ year              : chr [1:5901463] "21" "21" "21" "21" ...
##  $ day_of_week       : chr [1:5901463] "Tuesday" "Tuesday" "Saturday" "Saturday" ...
##  $ ride_length       : num [1:5901463] 415 970 973 918 522 ...
##  - 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>
```

```
# Descriptive analysis of ride length
summary(all_trips$ride_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8245     370     657    1193    1189 2497750
```

Cleaning up the data

Now that is interesting, I see that the "ride_length" column I created has a negative value for its minimum. That means some rides have a start time that is after the end time. I want to count all rows with a ride_length of 0 or less and see what we're dealing with. I have a lot of lines of data, so if the impact would be minimal to the dataset, I'll go ahead and remove these rows.

```
# Count the number of trips with negative ride_lengths
all_trips %>%
  count(ride_length<=0)
```

```
## # A tibble: 2 x 2
##   `ride_length <= 0`       n
##   <lgl>                <int>
## 1 FALSE              5900827
## 2 TRUE                   636
```

Aha great! So that's not many compared to the full size of the dataset, so I'm going to go ahead and remove these. If this were a company dataset, I would definitely want to dig in a bit to understand how this happened. Based on the case study details, I think these might be related to bikes being taken out of circulation for quality control.

```
# Create secondary data frame removing ride_length<0
all_trips_v2 <- all_trips[!(all_trips$ride_length<=0),]
```

Analyze

Now that the data is stored, merged, and cleaned, we are moving on to the Analyze phase. This includes organizing the data (this one is definitely an iterative process!), formatting the data, and identifying trends, relationships, or even, if we're lucky, surprises in the data. My first comparison is an aggregate one. This will compare the mean, median, max, and min ride lengths by whether the person using the bike was a member or a casual user.

```r
# Compare members and casual users
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual
          ,FUN=mean)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                1752.9596
## 2                     member                 776.0127
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual
          ,FUN=median)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                      864
## 2                     member                      541
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual
          ,FUN=max)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                  2497750
## 2                     member                    93594
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual
          ,FUN=min)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        1
## 2                     member                        1
```

Trend watch!

So one trend we notice here is that the casual users of the bikes are going for much longer rides.

Another thing we can look at is how long are the rides by the day of the week.

```r
# See the average ride time by each day for members vs casual users
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                   Friday                1644.5874
## 2                      member                   Friday                 756.3981
## 3                      casual                   Monday                1783.4532
## 4                      member                   Monday                 754.0940
## 5                      casual                 Saturday                1910.3282
## 6                      member                 Saturday                 868.5031
## 7                      casual                   Sunday                2038.3645
## 8                      member                   Sunday                 877.9249
## 9                      casual                 Thursday                1571.9136
## 10                     member                 Thursday                 744.4289
## 11                     casual                  Tuesday                1527.3643
## 12                     member                  Tuesday                 728.8203
## 13                     casual                Wednesday                1500.2268
## 14                     member                Wednesday                 730.4306
```

I see the days are out of order. Let's fix that!

```
# Order days of the week
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "W

# Compare weekday/membership type
all_trips_v2 %>%
  mutate(weekday = wday(started_at,label=TRUE)) %>%
  group_by(member_casual,weekday) %>%
  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.
```

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun              475539            2038.
##  2 casual        Mon              299626            1783.
##  3 casual        Tue              273782            1527.
##  4 casual        Wed              281757            1500.
##  5 casual        Thu              316087            1572.
##  6 casual        Fri              347599            1645.
##  7 casual        Sat              527499            1910.
##  8 member        Sun              417918             878.
##  9 member        Mon              472359             754.
## 10 member        Tue              523347             729.
## 11 member        Wed              522582             730.
## 12 member        Thu              522627             744.
## 13 member        Fri              466647             756.
## 14 member        Sat              453458             869.
```

Looking at this, we can see that casual riders ride most on Saturday and Sunday (not too surprising) and they ride longer distances, which we saw before. Members ride most Tuesday through Thursday and only take mildly longer rides on the weekend, if they cycle on the weekends. I also think it's interesting to see that the change in how many rides users take per day does not change much for members, but is much more variable for casual members depending on the day of the week.

Share

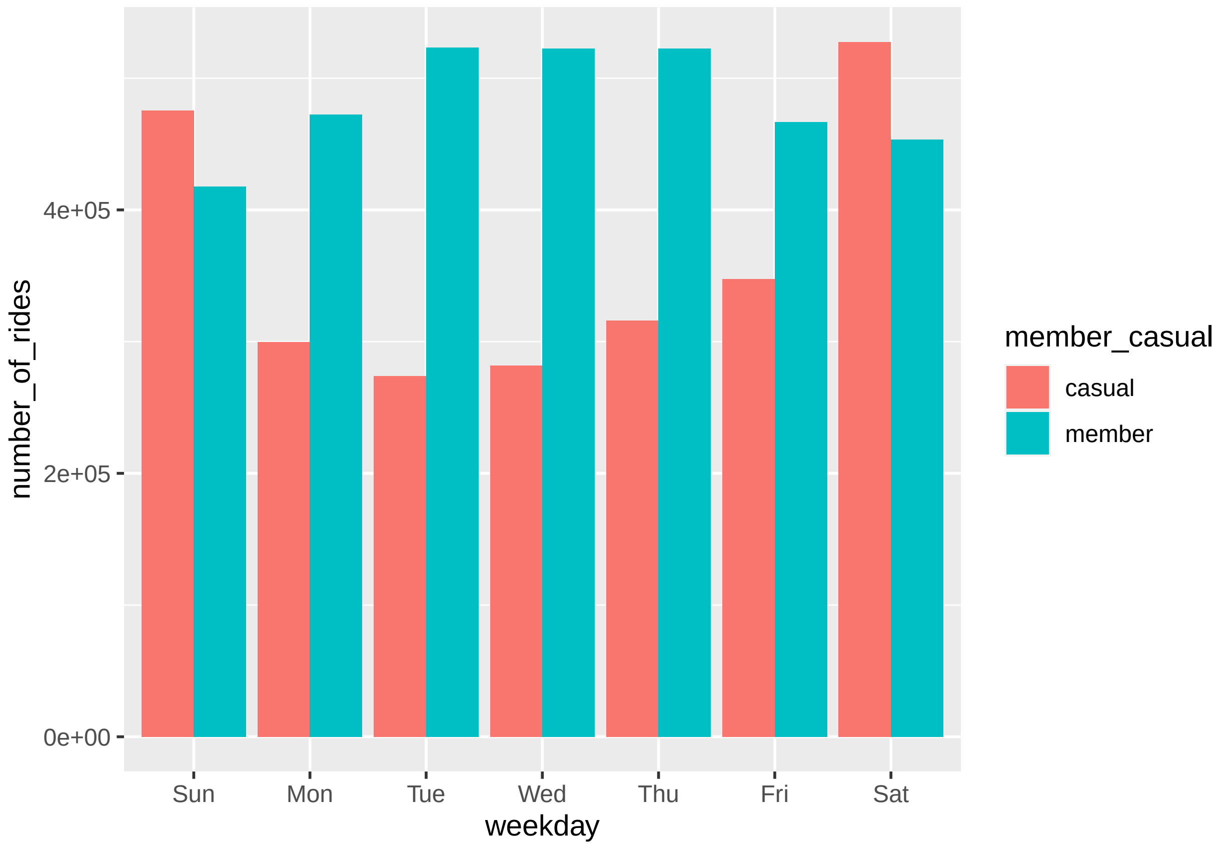
In the Share phase, we create effective visualizations and find the best way to share our findings. I have a couple of ggplots to show you below and then there will be a brief conclusion of what I learned during this data analysis project.

Weekday ridership plot

The first plot is a comparison of member/casual riders number of riders by the day of the week.

```
# Visualize weekday ridership with ggplot
all_trips_v2 %>%
  mutate(weekday = wday(started_at,label=TRUE)) %>%
  group_by(member_casual,weekday) %>%
  summarize(number_of_rides=n(),average_duration=mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x=weekday,y=number_of_rides,fill=member_casual)) + geom_col(position="dodge")
```

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

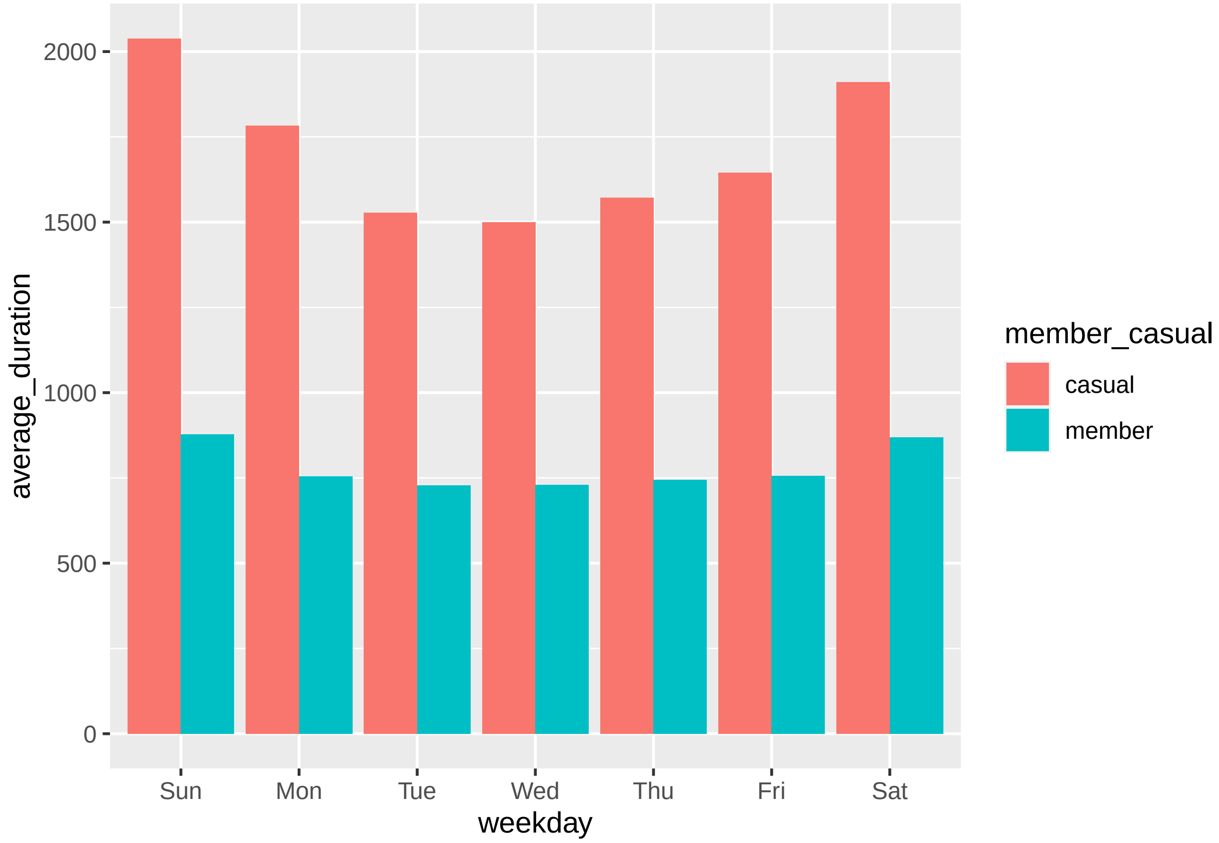


Weekday length of ride comparison

My second plot compares the length of the rides by the day of the week and membership type.

```
# Visualize ride length
all_trips_v2 %>%
  mutate(weekday = wday(started_at,label=TRUE)) %>%
  group_by(member_casual,weekday) %>%
  summarize(number_of_rides=n(),average_duration=mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x=weekday,y=average_duration,fill=member_casual)) + geom_col(position="dodge")
```

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

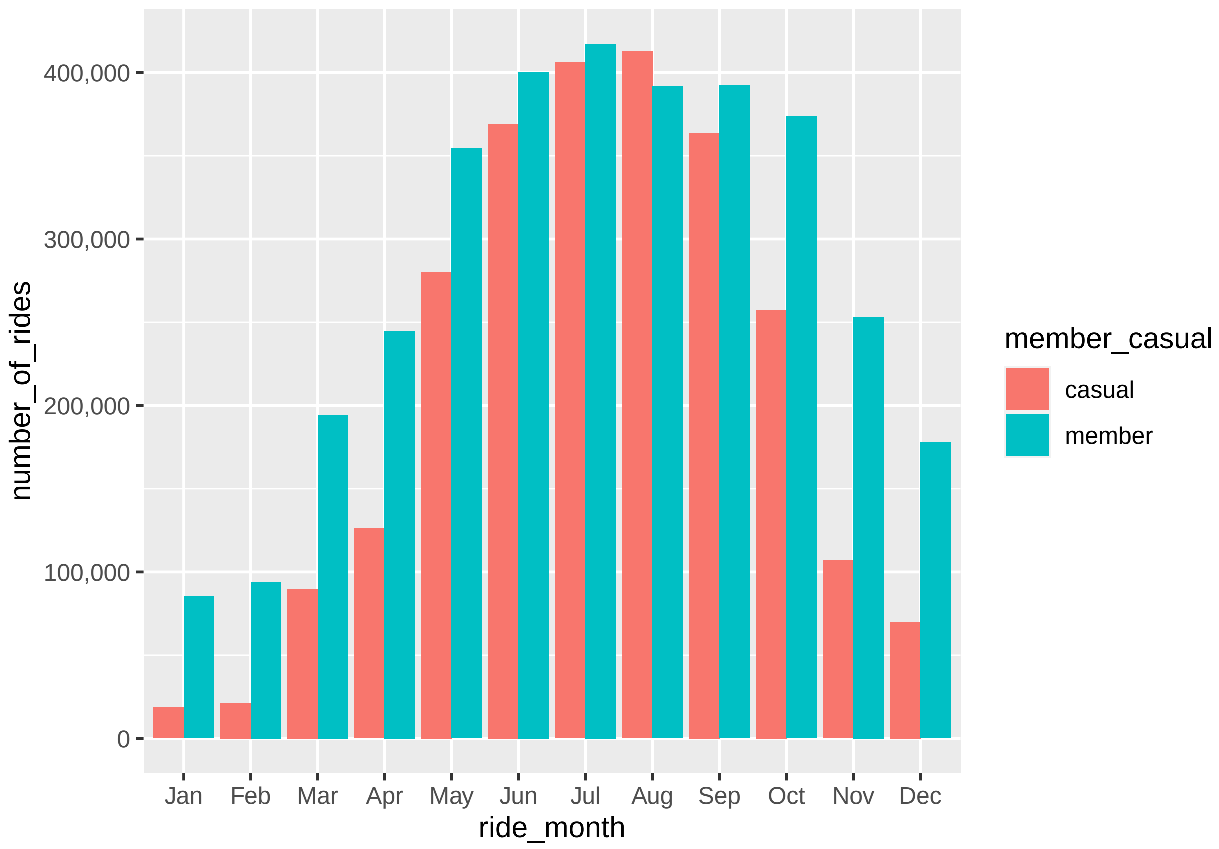


Monthly ridership comparison

My third visualization compares the number of riders by membership type riding every month.

```
# Visualize monthly ridership with ggplot dodge
all_trips_v2 %>%
  mutate(ride_month = month(started_at,label=TRUE)) %>%
  group_by(member_casual,ride_month) %>%
  summarize(number_of_rides=n(),average_duration=mean(ride_length)) %>%
  arrange(member_casual, ride_month) %>%
  ggplot(aes(x=ride_month,y=number_of_rides,fill=member_casual)) + geom_col(position="dodge") +
  scale_y_continuous(labels=comma)
```

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

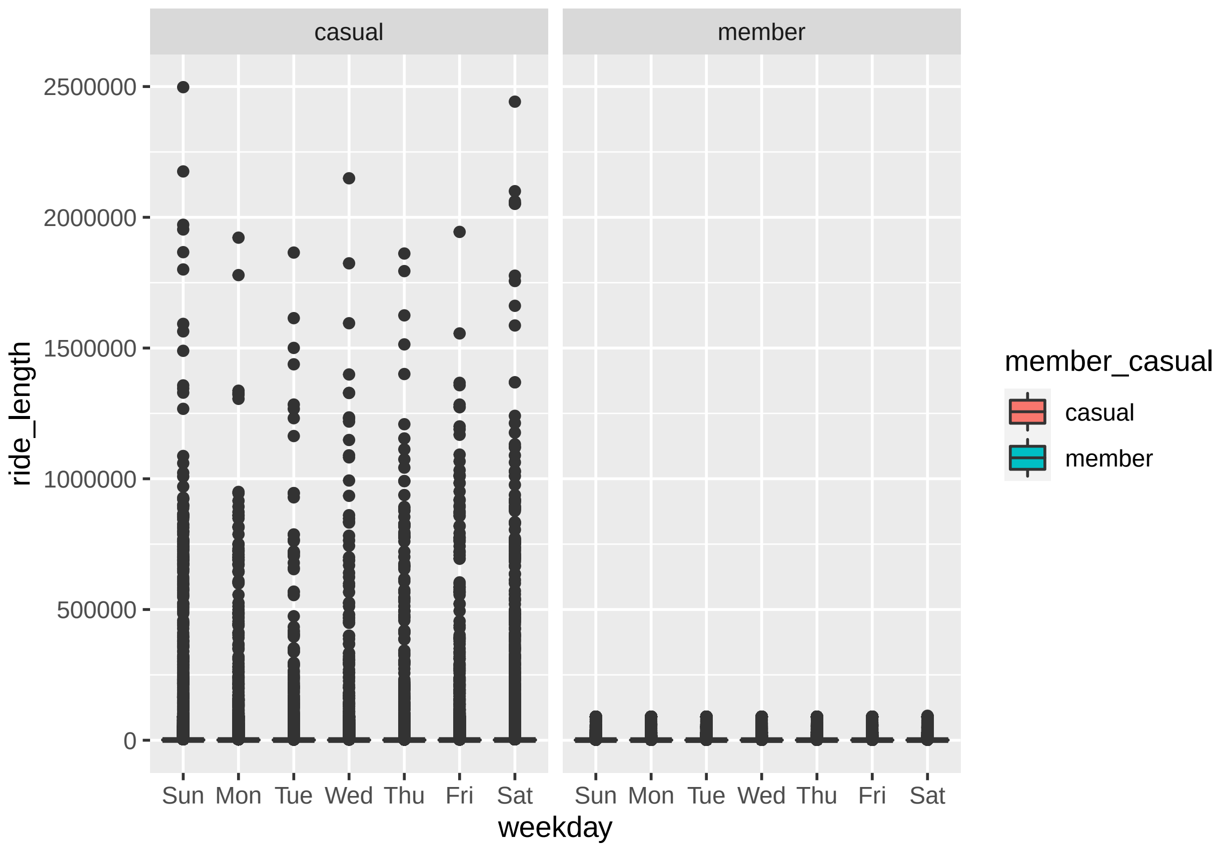


Length of ride boxplot comparison

My second plot compares the length of the rides by the day of the week and membership type.

```
# Visualize ride length box plot

all_trips_v2 %>%
  mutate(weekday = wday(started_at,label=TRUE)) %>%
  group_by(member_casual,weekday) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x=weekday,y=ride_length,fill=member_casual)) + geom_boxplot() + facet_wrap(~member_casual)
```



Act

The Act phase is where recommendations for actions are made. In this case study, I was focused on answering the question: How do annual members and casual riders use Cyclistic bikes differently?

I noticed a few differences, some expected, some less so.

1. Cyclistic members ride more consistently throughout the week, but ridership is greatest Tuesday - Thursday.
2. Casual riders are most active on the weekends.
3. Casual riders cycle much longer distances on average.
4. Over the year, as expected, members ride more every month.
5. Members are more active even into the colder months of the year.
6. Casual riders ride the most in the summer/nicer weather months (May - October).
7. In 2021-2022, casual riders even matched or exceeded total rides by members in the months of June, July, and August.
8. Looking at the boxplot we can see that the ride lengths have a lot of outliers above the whiskers. A further study would be good to look at individual members and their ridership to understand if the long rides are being done by a few members or if this is standard for casual users.

Recommendations:

1. Consider offering a member package that gives value to folks who want to do fewer rides with longer distances. This could take the form of a 10 ride pass rather than a monthly pass.
2. Consider offering a member package for summer user only, for example May - September.

If you would like to check out my notebook and run it for yourself, you can find it on GitHub here.