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

Saurabh Ghadge

10/02/2022

In this case study, I am going to perform data analysis for a fictional bike-share company in order to help them attract more riders. Along the way, I am also goin to perform numerous real-world tasks of a junior data analyst by following the steps of the data analysis process: Ask, Prepare, Process, Analyze, Share, and Act.

Scenario

You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

1) Ask:

Three 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?

I am assigned to answer the first question: How do annual members and casual riders use Cyclistic bikes differently?
Key tasks

- 1. Identify the business task
- 2. Consider key stakeholders

Buiseness Task Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Prepare:

I will use Cyclistic's historical trip data to analyze and identify trends(here (https://divvy-tripdata.s3.amazonaws.com/index.html)).This is public data that you can use to explore how different customer types are using Cyclistic bikes.(Note: The datasets have a different name because Cyclistic is a fictional company. For the purposes of this case study).

Key tasks - 1. Download data and store it appropriately. - 2. Identify how it's organized. - 3. Sort and filter the data. I downloaded the data for last 12 months on my local machine. In following steps we will read the data and make the columns in consistent format.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
setwd("E:/divvy_tripdata12mon")
temp = list.files(pattern = "*.csv")
df4 <- read_csv(temp[1])
```

```
## Rows: 84776 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): ride_id, rideable_type, started_at, ended_at, start_station_name, e...
## dbl (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, en...
```

```
##
## 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.
```

```
head(df4,2)
```

```
## # A tibble: 2 x 13
##   ride_id  rideable_type started_at  ended_at  start_station_n~ start_station_id
##   <chr>    <chr>         <chr>       <chr>     <chr>                       <dbl>
## 1 A847FAD~ docked_bike   26-04-2020~ 26-04-20~ Eckhart Park                   86
## 2 5405B80~ docked_bike   17-04-2020~ 17-04-20~ Drake Ave & Ful~              503
## # ... with 7 more variables: end_station_name <chr>, end_station_id <dbl>,
## #   start_lat <dbl>, start_lng <dbl>, end_lat <dbl>, end_lng <dbl>,
## #   member_casual <chr>
```

started_at and ended_ at should be in *datetime* but here, they are in character type so we will convert them into datetime.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
df4$started_at <- dmy_hm(df4$started_at)
df4$ended_at <- dmy_hm(df4$ended_at)
```

```
setwd("E:/divvy_tripdata12mon")
temp = list.files(pattern = "*.csv")
df5 <- read_csv(temp[2]);head(df5,2)
```

```
## Rows: 200274 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 02668A~ docked_bike   2020-05-27 10:03:52 2020-05-27 10:16:49 Franklin St & J~
## 2 7A50CC~ docked_bike   2020-05-25 10:47:11 2020-05-25 11:05:40 Clark St & Wrig~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df6 <- read_csv(temp[3]);head(df6,2)
```

```
## Rows: 343005 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 8CD5DE~ docked_bike   2020-06-13 23:24:48 2020-06-13 23:36:55 Wilton Ave & Be~
## 2 9A191E~ docked_bike   2020-06-26 07:26:10 2020-06-26 07:31:58 Federal St & Po~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df7 <- read_csv(temp[4]);head(df7,2)
```

```
## Rows: 551480 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 762198~ docked_bike   2020-07-09 15:22:02 2020-07-09 15:25:52 Ritchie Ct & Ba~
## 2 BEC9C9~ docked_bike   2020-07-24 23:56:30 2020-07-25 00:20:17 Halsted St & Ro~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df8 <- read_csv(temp[5]);head(df8,2)
```

```
## Rows: 622361 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 322BD2~ docked_bike   2020-08-20 18:08:14 2020-08-20 18:17:51 Lake Shore Dr &~
## 2 2A3AEF~ electric_bike 2020-08-27 18:46:04 2020-08-27 19:54:51 Michigan Ave & ~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df9 <- read_csv(temp[6]);head(df9,2)
```

```
## Rows: 532958 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 2B22BD~ electric_bike 2020-09-17 14:27:11 2020-09-17 14:44:24 Michigan Ave & ~
## 2 A7FB70~ electric_bike 2020-09-17 15:07:31 2020-09-17 15:07:45 W Oakdale Ave &~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df10 <- read_csv(temp[7]);head(df10,2)
```

```
## Rows: 388653 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 ACB6B4~ electric_bike 2020-10-31 19:39:43 2020-10-31 19:57:12 Lakeview Ave & ~
## 2 DF450C~ electric_bike 2020-10-31 23:50:08 2020-11-01 00:04:16 Southport Ave &~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df11 <- read_csv(temp[8]);head(df11,2)
```

```
## Rows: 259716 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 BD0A6F~ electric_bike 2020-11-01 13:36:00 2020-11-01 13:45:40 Dearborn St & E~
## 2 96A7A7~ electric_bike 2020-11-01 10:03:26 2020-11-01 10:14:45 Franklin St & I~
## # ... with 8 more variables: start_station_id <dbl>, end_station_name <chr>,
## #   end_station_id <dbl>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df12 <- read_csv(temp[9]);head(df12,2)
```

```
## Rows: 131573 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 70B6A9~ classic_bike  2020-12-27 12:44:29 2020-12-27 12:55:06 Aberdeen St & J~
## 2 158A46~ electric_bike 2020-12-18 17:37:15 2020-12-18 17:44:19 <NA>
## # ... with 8 more variables: start_station_id <chr>, end_station_name <chr>,
## #   end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df1 <- read_csv(temp[10]);head(df1,2)
```

```
## Rows: 96834 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 E19E6F~ electric_bike 2021-01-23 16:14:19 2021-01-23 16:24:44 California Ave ~
## 2 DC88F2~ electric_bike 2021-01-27 18:43:08 2021-01-27 18:47:12 California Ave ~
## # ... with 8 more variables: start_station_id <chr>, end_station_name <chr>,
## #   end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df2 <- read_csv(temp[11]);head(df2,2)
```

```
## Rows: 49622 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 89E7AA~ classic_bike  2021-02-12 16:14:56 2021-02-12 16:21:43 Glenwood Ave & ~
## 2 0FEFDE~ classic_bike  2021-02-14 17:52:38 2021-02-14 18:12:09 Glenwood Ave & ~
## # ... with 8 more variables: start_station_id <chr>, end_station_name <chr>,
## #   end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

```
df3 <- read_csv(temp[12]);head(df3,2)
```

```
## Rows: 228496 Columns: 13
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): ride_id, rideable_type, start_station_name, start_station_id, end_...
## dbl  (4): start_lat, start_lng, end_lat, end_lng
## 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.
```

```
## # A tibble: 2 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 CFA86D~ classic_bike  2021-03-16 08:32:30 2021-03-16 08:36:34 Humboldt Blvd &~
## 2 30D9DC~ classic_bike  2021-03-28 01:26:28 2021-03-28 01:36:55 Humboldt Blvd &~
## # ... with 8 more variables: start_station_id <chr>, end_station_name <chr>,
## #   end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

All other column formats are seen to be consistent.So we will now bind these dataframes into single data frame for further Analysis.

```
divvy_trip <- rbind(df4,df5,df6,df7,df8,df9,df10,df11,df12,df1,df2,df3)
remove(df4,df5,df6,df7,df8,df9,df10,df11,df12,df1,df2,df3)
head(divvy_trip)
```

```
## # A tibble: 6 x 13
##   ride_id rideable_type started_at          ended_at            start_station_n~
##   <chr>   <chr>         <dttm>              <dttm>              <chr>
## 1 A847FA~ docked_bike   2020-04-26 17:45:00 2020-04-26 18:12:00 Eckhart Park
## 2 5405B8~ docked_bike   2020-04-17 17:08:00 2020-04-17 17:17:00 Drake Ave & Ful~
## 3 5DD24A~ docked_bike   2020-04-01 17:54:00 2020-04-01 18:08:00 McClurg Ct & Er~
## 4 2A59BB~ docked_bike   2020-04-07 12:50:00 2020-04-07 13:02:00 California Ave ~
## 5 27AD30~ docked_bike   2020-04-18 10:22:00 2020-04-18 11:15:00 Rush St & Hubba~
## 6 356216~ docked_bike   2020-04-30 17:55:00 2020-04-30 18:01:00 Mies van der Ro~
## # ... with 8 more variables: start_station_id <chr>, end_station_name <chr>,
## #   end_station_id <chr>, start_lat <dbl>, start_lng <dbl>, end_lat <dbl>,
## #   end_lng <dbl>, member_casual <chr>
```

2) Process:

Key tasks

- 1. Check the data for errors.
- 2. Choose your tools.
- 3. Transform the data so you can work with it effectively.
- 4. Document the cleaning process
  first of all we will check for missing Values. We will remove them as it will not make any sense if try to impute with another measure or values for missing one.

```
divvy_trip <- divvy_trip[complete.cases(divvy_trip),]
```

We are now going to introduced some calculated columns..

- ride_length (time taken by ride)
- day_of_week (day of ride)
- month (month of ride)

```
divvy_trip <- divvy_trip %>% mutate(ride_length = ended_at - started_at) #in seconds
```

Here, are some ride length are negative which makes no sense, Which will surely can be a data entry error. So,we are going to filter out those rows.

```
divvy_trip <- divvy_trip %>% filter ( ride_length > 0)
divvy_trip$ride_length <- seconds_to_period(divvy_trip$ride_length)
```

Now our filtered data has dimension ..

```
dim(divvy_trip)
```

```
## [1] 3283363      14
```

We lost about 7% of data in total :(, but not worry we have enough large data for our analysis. Now we will calculate other columns.

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

Analyze:

Key tasks

- 1. Aggregate your data so it's useful and accessible.
- 2. Organize and format your data.
- 3. Perform calculations.
- 4. Identify trends and relationships.

```
summary(divvy_trip$ride_length)
```

```
##                   Min.                 1st Qu.                  Median
##                   "1S"                 "8M 5S"                "14M 46S"
##                   Mean                 3rd Qu.                    Max.
## "28M 3.41014258856762S"               "26M 56S"        "40d 18H 40M 0S"
```

This is basic summary of column ride_length, From which we observe that minimum ride length is of only 1 seconds while the maximum ride_length is of 40 days,18 hour and 40 minutes long. The overall mean of ride_length is 28 Minutes.

```
member_casual <- divvy_trip %>%
        group_by(member_casual) %>%
        summarize(mean_rlen = ms(mean(ride_length)))
member_casual
```

```
## # A tibble: 2 x 2
##   member_casual mean_rlen
##   <chr>         <Period>
## 1 casual        28M 87728S
## 2 member        28M 44159S
```

For both member and casual riders shows a same average ride length.
We will next try to summarize by day wise ride_length average of casual and member riders.

```
day_wise <- divvy_trip %>%
        group_by(day_of_week,member_casual) %>%
        summarize(mean_rlen = ms(mean(ride_length)))
```

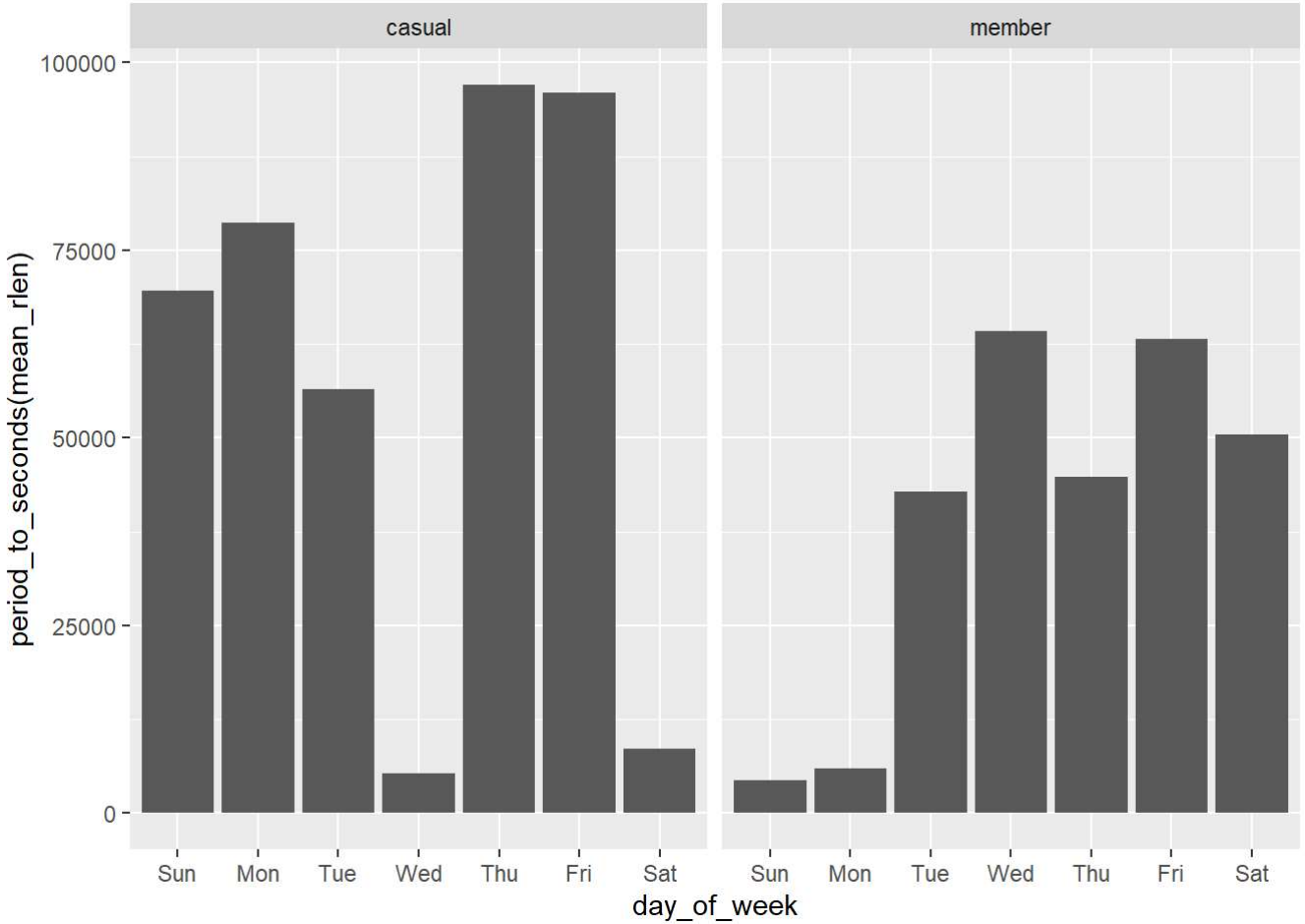
```
## `summarise()` has grouped output by 'day_of_week'. You can override using the `.groups` argument.
```

```
day_wise
```

```
## # A tibble: 14 x 3
## # Groups:   day_of_week [7]
##    day_of_week member_casual mean_rlen
##    <ord>       <chr>         <Period>
##  1 Sun         casual        28M 67926S
##  2 Sun         member        28M 2739S
##  3 Mon         casual        28M 77037S
##  4 Mon         member        28M 4329S
##  5 Tue         casual        28M 54875S
##  6 Tue         member        28M 41105S
##  7 Wed         casual        29M 3568S
##  8 Wed         member        28M 62549S
##  9 Thu         casual        28M 95301S
## 10 Thu         member        28M 43077S
## 11 Fri         casual        28M 94363S
## 12 Fri         member        28M 61474S
## 13 Sat         casual        29M 6858S
## 14 Sat         member        28M 48793S
```

These values are sort of difficult to understand as they all nearly looking same. We will plot a plot taking days on x axis and mean seconds ride_length on y_ axis by breaking them by casual rider and member rider.

```
ggplot(data = day_wise,mapping = aes(day_of_week,period_to_seconds(mean_rlen))) + geom_bar(stat = 'identity') + facet_grid(.
~member_casual)
```



We see that average ride_length for casual riders is longer than that of member riders. Also Casual riders are tends to have more average ride_length on Thursday and Friday while member riders on wed and friday.
We can further broken down ride_length by *month*.

```
month_ <- divvy_trip %>%
        group_by(month,member_casual) %>%
        summarize(mean_rlen = ms(mean(ride_length,na.rm = TRUE)))
```

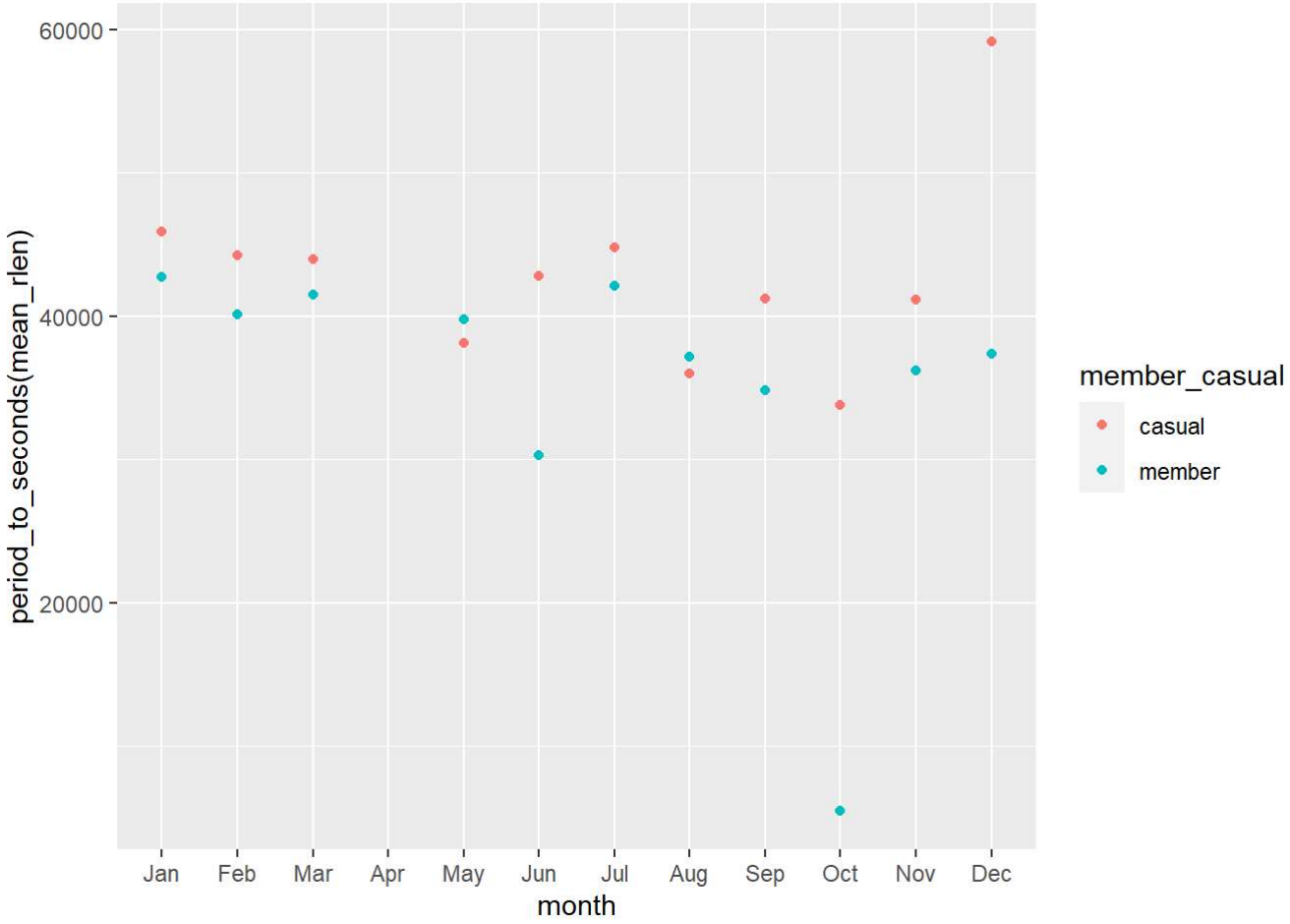
```
## Warning in .parse_hms(..., order = "MS", quiet = quiet): Some strings failed to
## parse, or all strings are NAs

## Warning in .parse_hms(..., order = "MS", quiet = quiet): Some strings failed to
## parse, or all strings are NAs
```

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

```
ggplot(data = month_,mapping = aes(month,period_to_seconds(mean_rlen),color = member_casual)) + geom_point()
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



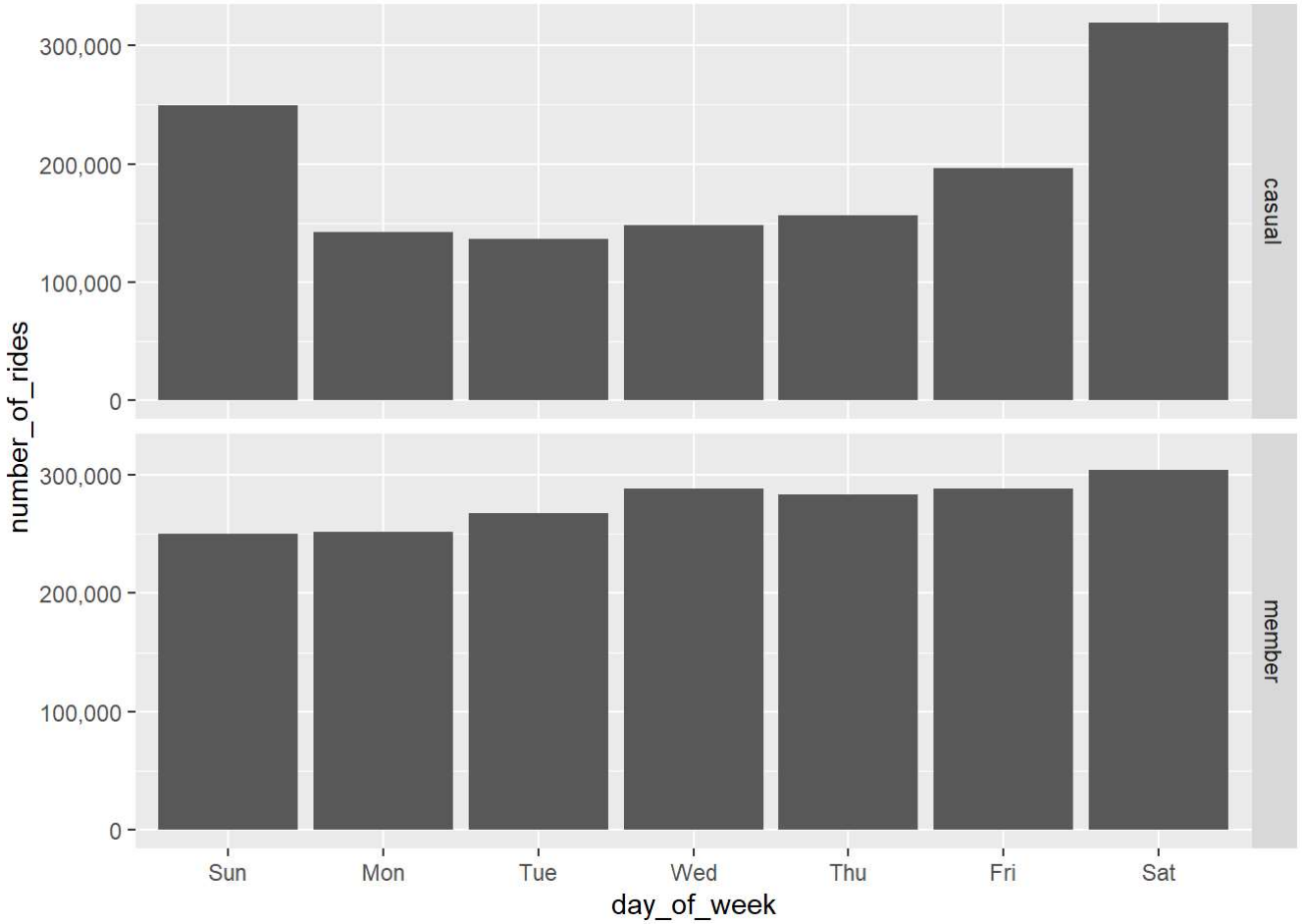
From plot for casual riders month wise average ride length is higher than member riders and also both shows slightly decrease in ride_length over month.

Now it's important to check how many rides are there by month and day for casual and member rider.

```
day_count <- divvy_trip %>%
        group_by(day_of_week,member_casual) %>%
        summarize(number_of_rides = n()) %>%
        arrange(member_casual)
```

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

```
ggplot(day_count,mapping = aes (day_of_week,number_of_rides)) + geom_bar(stat = 'identity') + scale_y_continuous(labels = sc
ales::comma) + facet_grid(member_casual~.)
```

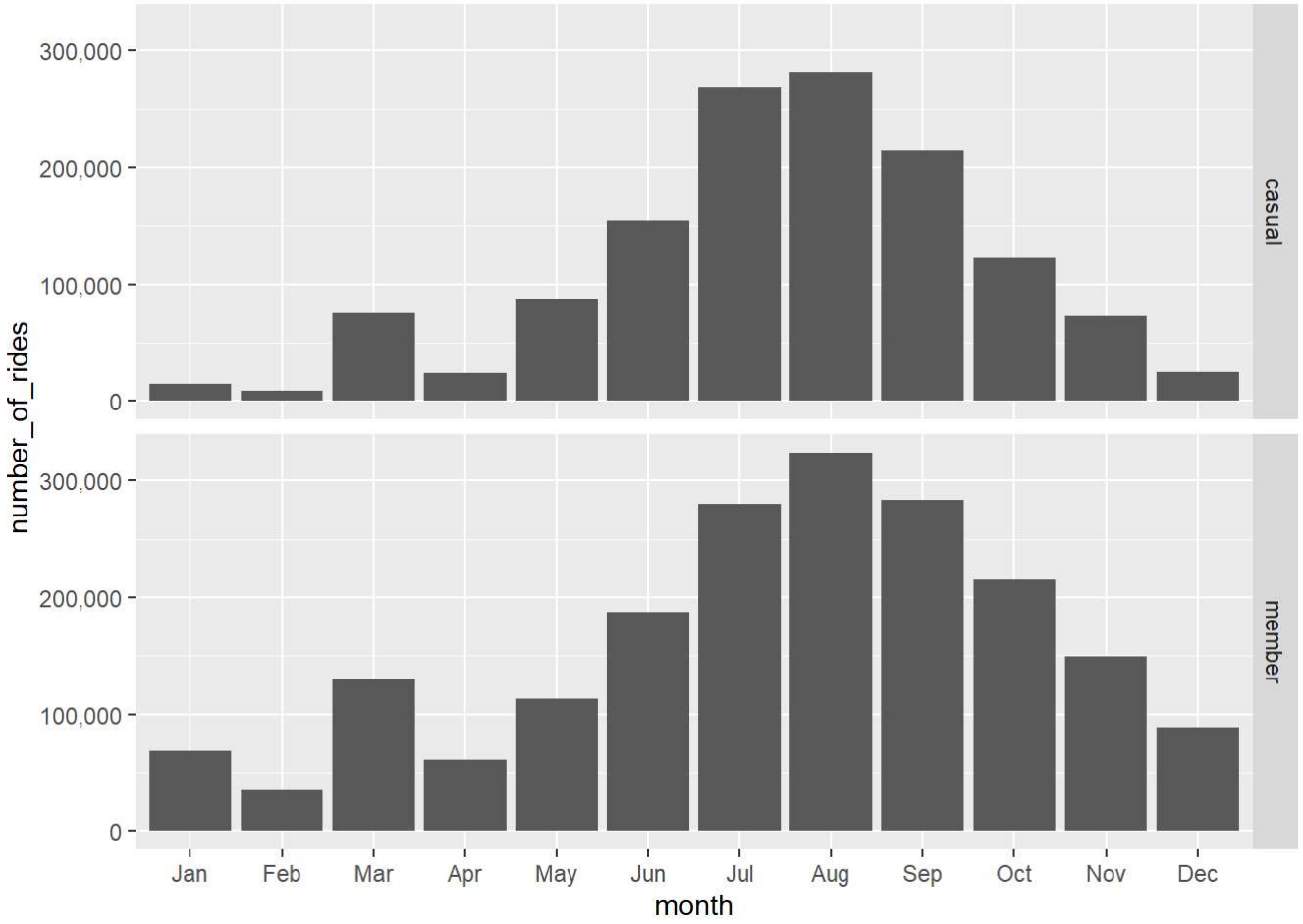


For casual riders number of rides is far less as compared to member riders.Next we are going to check for monthwise number of rides for both type of risers.

```
month_count <- divvy_trip %>%
        group_by(month,member_casual) %>%
        summarize(number_of_rides = n())
```

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

```
ggplot(month_count,mapping = aes(month,number_of_rides)) + geom_bar(stat = 'identity') + scale_y_continuous(labels = scale
s::comma) + facet_grid(member_casual~.)
```



We see that for both riders number of rides from April month are start to increases, for August number of rides are higher among all months and then it starts to fall down.
Now we will try to find out relation between start_station and member_casual riders..

```
start_station <- divvy_trip %>%
        group_by(start_station_id,member_casual) %>%
        summarize(number_of_rides = n(),mean_rlen = ms(mean(ride_length))) %>%
        arrange(desc(number_of_rides),desc(mean_rlen))
```

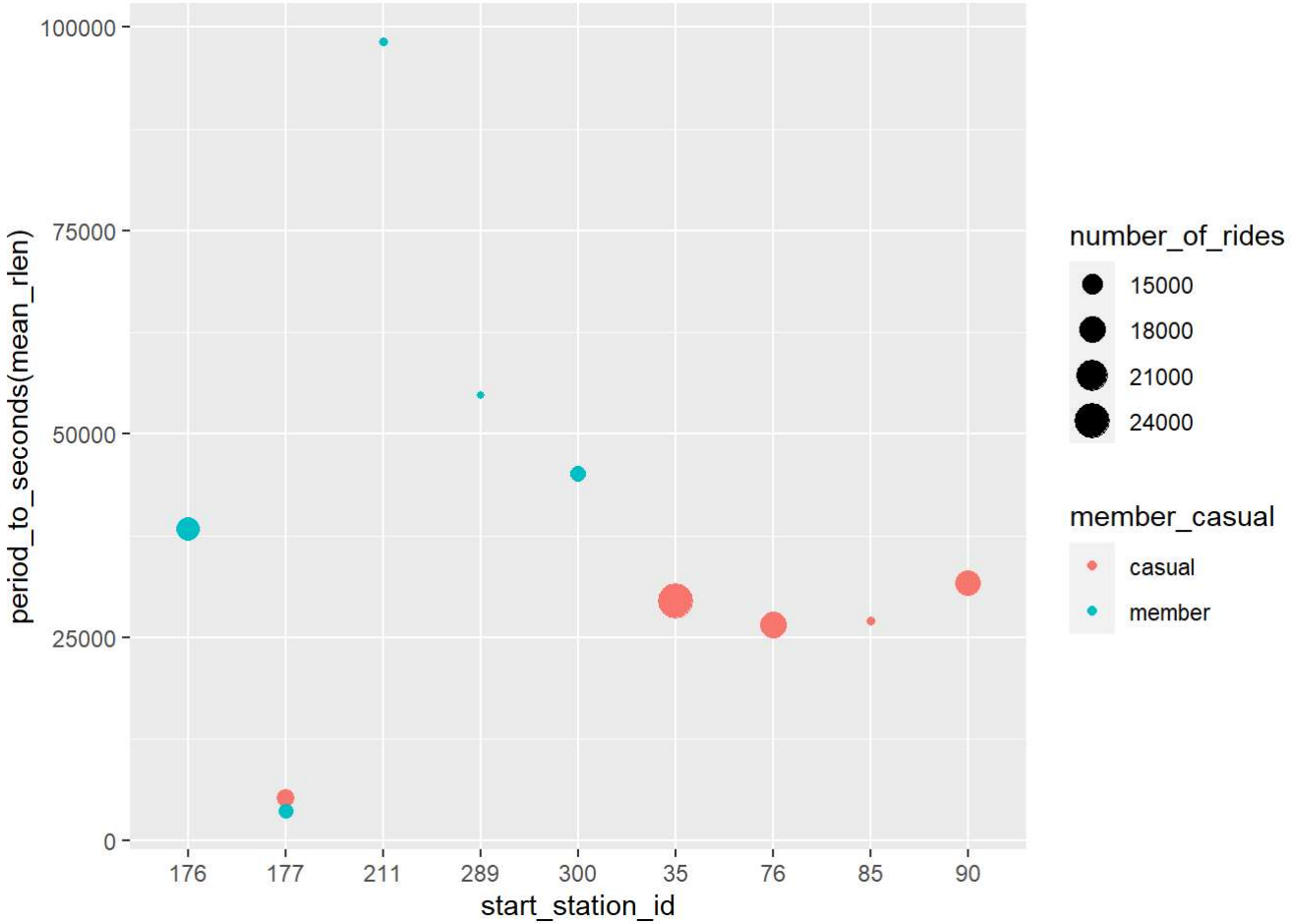
```
## `summarise()` has grouped output by 'start_station_id'. You can override using the `.groups` argument.
```

```
start_station[1:10,]
```

```
## # A tibble: 10 x 4
## # Groups:   start_station_id [9]
##    start_station_id member_casual number_of_rides mean_rlen
##    <chr>            <chr>                  <int> <Period>
##  1 35               casual                 24131 29M 27686S
##  2 76               casual                 17856 29M 24877S
##  3 90               casual                 17331 29M 29923S
##  4 176              member                 16466 28M 36572S
##  5 177              casual                 13824 29M 3572S
##  6 300              member                 13276 28M 43402S
##  7 177              member                 13228 29M 1902S
##  8 211              member                 12380 27M 96591S
##  9 85               casual                 12376 29M 25275S
## 10 289              member                 12341 28M 53091S
```

Here,we are seeing 10 highest number of rides at stations with their riders type and average ride length.

```
ggplot(data = start_station[1:10,],mapping = aes(start_station_id,period_to_seconds(mean_rlen),size = number_of_rides,color
 = member_casual)) + geom_point()
```



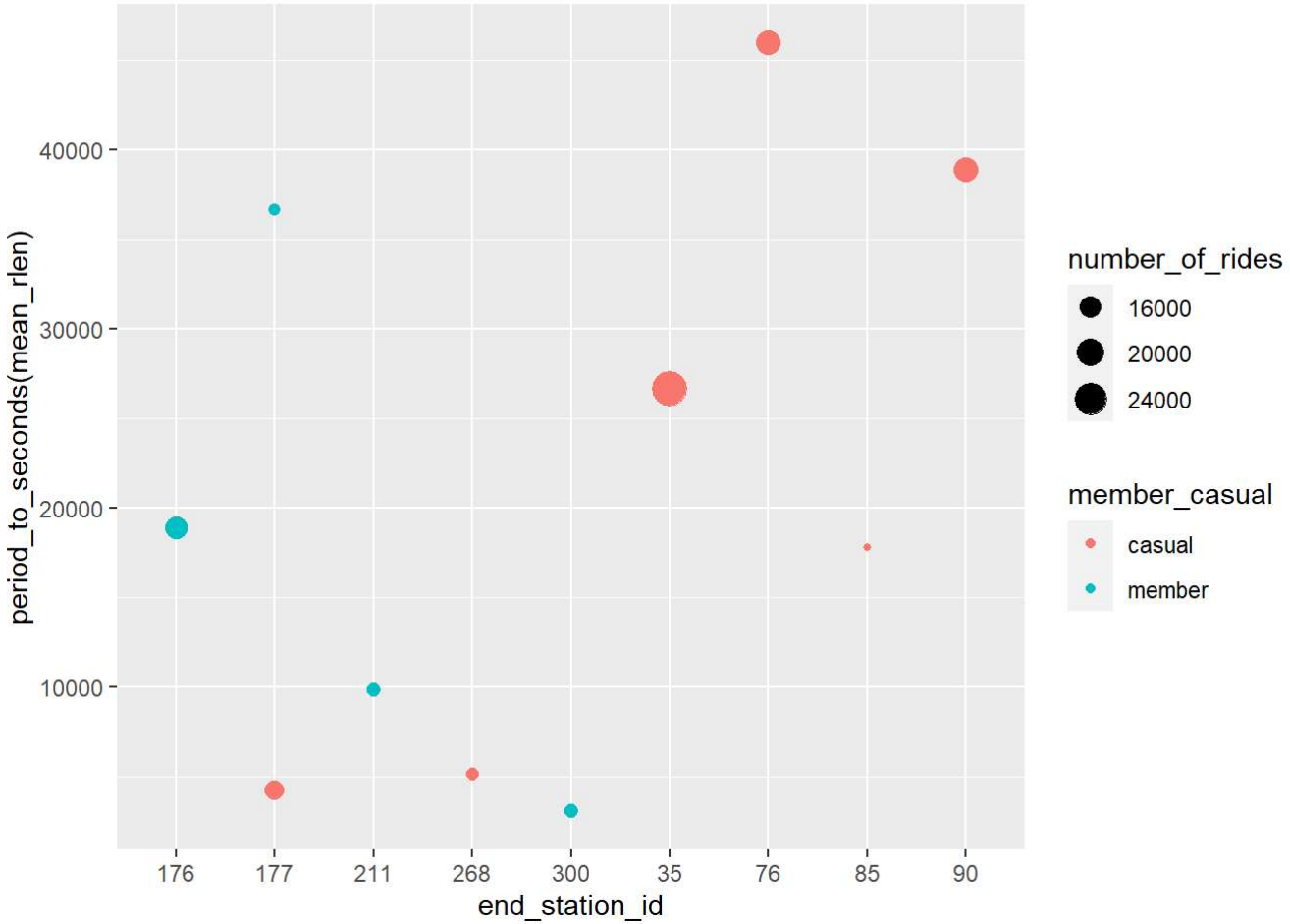
From this plot now we can visually clarify that at station 35 most of rides were happened and they are by casual members.Also at station 177 number of rides for both riders are constant.It seems that member riders ride length is pretty much higher than casual riders.

Similarly, we can check at which station most of the rides were ended and by which types of riders(member/Casual).

```
end_station <- divvy_trip %>%
        group_by(end_station_id,member_casual) %>%
        summarize(number_of_rides = n(),mean_rlen = ms(mean(ride_length))) %>%
        arrange(desc(number_of_rides),desc(mean_rlen))
```

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

```
ggplot(data = end_station[1:10,],mapping = aes(end_station_id,period_to_seconds(mean_rlen),size = number_of_rides,color = me
mber_casual)) + geom_point()
```



At station 35 most of casual riders rides were ended and also their ride length is also seems to be maximum than member riders, which is completely different from start_station_id where ride_length of member riders were seen to be maximum, also lots of member riders ended their rides at station 176.

Next phase of analysis i.e. *Share* and *Act* can be found on attached ppt…