Divvy Customer Analysis

Eugene Bezuglov

2023-05-28

Contents

Introduction

W. Edwards Deming, a famous American scientist, once said: ***Without data, you're just another person with an opinion****. This project is my attempt to not become yet another person.*

This is the R part of my final project on the road to getting the Google Data Analytics Professional Certificate. The Tableau part is available here, and the link to the SQL part will be added later. It's also available on Kaggle.

I'll be analyzing the data of a bike-sharing company, Cyclistic, which is a fictional name for Divvy, a real Chicago-based bike-sharing company. So, from now on I'll refer to Cyclistic as Divvy.

Key Goals

Divvy offers several pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Divvy members.

Since annual members are much more profitable than casual users, the stakeholders believe their marketing campaign should be focused on converting casual riders into members. My key goal is to answer this question:

How do annual members and casual riders use Divvy bikes differently? And to support my findings with a detailed report, which you're reading.

Data Description

The data for this project is a public data set provided by Divvy, a Chicagoland's bike share system. It has been made available by Motivate International Inc. under this license. No riders' personally identifiable information is provided.

Part 1: Data Wrangling and Cleaning

For this project I downloaded 12 months of data. It comes from the internal sources and doesn't require extensive cleaning; however, some issues still exist, and in this section I address them, describing my step-by-step process of working with the data.

Uploading and Merging Data

The data I was given comes in monthly csv files, so at first I upload it into separate tibbles:

```r
# Import CSV into tibbles
feb2022 <- read_csv("./data/202202-divvy-tripdata.csv")
mar2022 <- read_csv("./data/202203-divvy-tripdata.csv")
apr2022 <- read_csv("./data/202204-divvy-tripdata.csv")
may2022 <- read_csv("./data/202205-divvy-tripdata.csv")
jun2022 <- read_csv("./data/202206-divvy-tripdata.csv")
jul2022 <- read_csv("./data/202207-divvy-tripdata.csv")
aug2022 <- read_csv("./data/202208-divvy-tripdata.csv")
sep2022 <- read_csv("./data/202209-divvy-publictripdata.csv")
oct2022 <- read_csv("./data/202210-divvy-tripdata.csv")
nov2022 <- read_csv("./data/202211-divvy-tripdata.csv")
```

```
dec2022 <- read_csv("./data/202212-divvy-tripdata.csv")
jan2023 <- read_csv("./data/202301-divvy-tripdata.csv")
```

You might have a look at one of the tibbles to see how the data is organized:

```
glimpse(feb2022)
```

```
## Rows: 115,609
## Columns: 13
## $ ride_id            <chr> "E1E065E7ED285C02", "1602DCDC5B30FFE3", "BE7DD2AF4B~
## $ rideable_type      <chr> "classic_bike", "classic_bike", "classic_bike", "cl~
## $ started_at         <dttm> 2022-02-19 18:08:41, 2022-02-20 17:41:30, 2022-02-~
## $ ended_at           <dttm> 2022-02-19 18:23:56, 2022-02-20 17:45:56, 2022-02-~
## $ start_station_name <chr> "State St & Randolph St", "Halsted St & Wrightwood ~
## $ start_station_id   <chr> "TA1305000029", "TA1309000061", "TA1305000029", "13~
## $ end_station_name   <chr> "Clark St & Lincoln Ave", "Southport Ave & Wrightwo~
## $ end_station_id     <chr> "13179", "TA1307000113", "13011", "13323", "TA13070~
## $ start_lat          <dbl> 41.88462, 41.92914, 41.88462, 41.94815, 41.88462, 4~
## $ start_lng          <dbl> -87.62783, -87.64908, -87.62783, -87.66394, -87.627~
## $ end_lat            <dbl> 41.91569, 41.92877, 41.87926, 41.95283, 41.88584, 4~
## $ end_lng            <dbl> -87.63460, -87.66391, -87.63990, -87.64999, -87.635~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

The variables are self-explanatory so I won't elaborate on that topic.

Next, I check if the new tibbles are row-bindable. If tibbles have different number of columns, **rbind()** throws an error, whereas **bind_rows()** assigns NA's. Hence, using **rbind** bind method. Returns TRUE if the columns are identical.

```
compare_df_cols_same(feb2022, mar2022, apr2022, may2022, jun2022, jul2022,
                     aug2022,sep2022, oct2022, nov2022, dec2022, jan2023,
                     bind_method = c("rbind"))
```

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

Next, I combine single tibbles into one tibble. This time I use bind_rows because it's faster.

```
full_tripdata <- bind_rows(feb2022, mar2022, apr2022, may2022, jun2022, jul2022,
                           aug2022, sep2022, oct2022, nov2022, dec2022, jan2023)
```

And remove single tibbles.

```
rm(feb2022, mar2022, apr2022, may2022, jun2022, jul2022,
         aug2022, sep2022, oct2022, nov2022, dec2022, jan2023)
```

Handling Duplicated, Missing, and Irrelevant Values

In the new *full_tripdata tibble*, ride_id variable is the primary key, so I verify it contains no duplicates.

```
get_dupes(full_tripdata, ride_id)
```

```
## No duplicate combinations found of: ride_id
```

```
## # A tibble: 0 x 14
## # i 14 variables: ride_id <chr>, dupe_count <int>, rideable_type <chr>,
## #   started_at <dttm>, ended_at <dttm>, start_station_name <chr>,
## #   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>
```

After introducing electric bikes, Divvy Bikes renamed docked_bike to classic_bike in their database. Therefore, I replace docked_bike to classic_bike in the rideable_type observations.

```
full_tripdata <- full_tripdata %>%
  mutate(rideable_type = if_else(rideable_type == "docked_bike", "classic_bike", rideable_type))
```

And verify the replacement was successful.

```
table(full_tripdata$rideable_type)
```

```
##
##  classic_bike electric_bike
##       2814692       2939556
```

Next, I check if any observations are missing.

```
sapply(full_tripdata, function(x) sum(is.na(x)))
```

```
##            ride_id      rideable_type         started_at           ended_at
##                  0                  0                  0                  0
## start_station_name   start_station_id   end_station_name     end_station_id
##             843525             843525             902655             902655
##          start_lat          start_lng            end_lat            end_lng
##                  0                  0               5899               5899
##      member_casual
##                  0
```

Some coordinates are missing, as well as ids and names of stations. Can the missing values be found based on the data we have? For example, it's intuitive that we could fill in the missing *start_station_name* and *start_station_id* based on the combination of matching *start_lat* and *start_lng* from another row. However, the answer is **no**. Turns out that all of the observations with missing station names and ids have their lat and lng written in the 2 decimal places precision, whereas at least 5 decimal places precision is required to locate the station correctly.

Here's the code to show it:

(nchar = 8 is for fetching coordinates in xx.xxxxx format, which is 8 characters)

```r
full_tripdata %>%
  filter(is.na(start_station_name),
         is.na(start_station_id),
         nchar(as.character(start_lat)) == 8)
```

```
## # A tibble: 0 x 13
## # i 13 variables: ride_id <chr>, rideable_type <chr>, started_at <dttm>,
## #   ended_at <dttm>, start_station_name <chr>, 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>
```

Yet, I choose not to remove the rows with NAs for station names and ID's because we can still get some insights from such rows. Moreover, removing such rows galore could lower the reliability of the data.

Aside from NA's one more issue exists — there are more station names than station ID's:

```r
full_tripdata %>%
  summarise (
    start_id = start_station_id %>% n_distinct,
    end_id = end_station_id %>% n_distinct,
    start_name = start_station_name %>% n_distinct,
    end_name = end_station_name %>% n_distinct
)
```

```
## # A tibble: 1 x 4
##   start_id end_id start_name end_name
##      <int>  <int>      <int>    <int>
## 1     1312   1317       1680     1697
```

I create separate tibbles with duplicate naming for start stations and end stations, and then manually explore them:

```r
full_tripdata %>%
  group_by(start_station_id) %>%
  filter(n_distinct(start_station_name) > 1) %>%
  select(start_station_id, start_station_name) %>%
  distinct() %>%
  arrange(desc(start_station_id))
```

```
## # A tibble: 721 x 2
## # Groups:   start_station_id [335]
##    start_station_id start_station_name
##    <chr>            <chr>
##  1 TA1309000042     Lincoln Ave & Belmont Ave
##  2 TA1309000042     Lincoln Ave & Belmont Ave (Temp)
##  3 TA1307000138     Lincoln Ave & Roscoe St
##  4 TA1307000138     Wood St & Webster Ave
##  5 TA1306000026     Racine Ave & Fullerton Ave
##  6 TA1306000026     Racine Ave & Fullerton Ave (Temp)
##  7 TA1306000015     Morgan St & Lake St
##  8 TA1306000015     Sangamon St & Lake St
##  9 KA1504000168     Western & 28th - Velasquez Institute Vaccination Site
## 10 KA1504000168     Western Ave & 28th St
## # i 711 more rows
```

```
full_tripdata %>%
  group_by(end_station_id) %>%
  filter(n_distinct(end_station_name) > 1) %>%
  select(end_station_id, end_station_name) %>%
  distinct() %>%
  arrange(desc(end_station_id))
```

```
## # A tibble: 741 x 2
## # Groups:   end_station_id [343]
##    end_station_id end_station_name
##    <chr>          <chr>
##  1 chargingstx1   Bissell St & Armitage Ave*
##  2 chargingstx1   Bissell St & Armitage Ave - Charging
##  3 TA1309000042   Lincoln Ave & Belmont Ave
##  4 TA1309000042   Lincoln Ave & Belmont Ave (Temp)
##  5 TA1307000138   Lincoln Ave & Roscoe St
##  6 TA1307000138   Wood St & Webster Ave
##  7 TA1306000026   Racine Ave & Fullerton Ave
##  8 TA1306000026   Racine Ave & Fullerton Ave (Temp)
##  9 TA1306000015   Morgan St & Lake St
## 10 TA1306000015   Sangamon St & Lake St
## # i 731 more rows
```

There are over 700 non-unique names. After manually exploring them, I found several patterns:

- Stations with the same ID sometimes have their name started with "Public Rack -" or "City Rack -",and sometimes not. I explored System Map on DivvyBikes website and figured out that these must be the same stations.
- Some stations have " (Temp)" added to their name. Looks like DivvyBikes added temporary stations for whatever reason, yet for the purpose of the current analysis it's safe to merge such stations into one.
- "DIVVY 001 - Warehouse test station" and "Hubbard Bike-checking (LBS-WH-TEST)" are maintenance stations, I remove them.
- Two stations have "amp;" string piece attached after the ampersand sign, I remove it.
- Three stations have the vaccination site tag added to their name. I believe these were temporary stations during the covid vaccination campaign, and for the purpose of this analysis it's safe to remove this tag and merge two separate stations into one.
- One station has a duplicate with "- Charging" tag attached, I remove it.
- One station had "*" after its name, I remove it.

Fixing these issues:

```
full_tripdata <- full_tripdata %>%
  mutate(start_station_name = str_remove(start_station_name, "Public Rack - ")) %>%
  mutate(start_station_name = str_remove(start_station_name, "City Rack - ")) %>%
  mutate(start_station_name = str_remove(start_station_name, "amp;")) %>%
  mutate(start_station_name = str_trim(str_remove(start_station_name, " \\(Temp\\)"))) %>%
  mutate(start_station_name = str_remove(start_station_name, "\\*")) %>%
  mutate(start_station_name = str_remove(start_station_name, " - Charging")) %>%
  mutate(start_station_name = str_remove(start_station_name, " (May)")) %>%
  mutate(start_station_name = str_replace(start_station_name, "Western & 28th - Velasquez Institute Vac
  mutate(start_station_name = str_replace(start_station_name, "Halsted & 63rd - Kennedy-King Vaccinatio
  mutate(start_station_name = str_replace(start_station_name, "Broadway & Wilson - Truman College Vacci
  filter(start_station_id != "DIVVY 001 - Warehouse test station" &
           start_station_id != "Hubbard Bike-checking (LBS-WH-TEST)" |
           is.na(start_station_id)) %>%
  mutate(end_station_name = str_remove(end_station_name, "Public Rack - ")) %>%
  mutate(end_station_name = str_remove(end_station_name, "City Rack - ")) %>%
  mutate(end_station_name = str_remove(end_station_name, "amp;")) %>%
  mutate(end_station_name = str_trim(str_remove(end_station_name, " \\(Temp\\)"))) %>%
  mutate(end_station_name = str_remove(end_station_name, "\\*")) %>%
  mutate(end_station_name = str_remove(end_station_name, " - Charging")) %>%
  mutate(end_station_name = str_remove(end_station_name, " (May)")) %>%
  mutate(end_station_name = str_replace(end_station_name, "Western & 28th - Velasquez Institute Vaccina
  mutate(end_station_name = str_replace(end_station_name, "Halsted & 63rd - Kennedy-King Vaccination Si
  mutate(end_station_name = str_replace(end_station_name, "Broadway & Wilson - Truman College Vaccinati
  filter(end_station_id != "DIVVY 001 - Warehouse test station" &
           end_station_id != "Hubbard Bike-checking (LBS-WH-TEST)" |
           is.na(end_station_id))
```

Now I check for the duplicate station names once more:

```
full_tripdata %>%
  group_by(start_station_id) %>%
  filter(n_distinct(start_station_name) > 1) %>%
  select(start_station_id, start_station_name) %>%
  distinct() %>%
  arrange(desc(start_station_id))
```

```
## # A tibble: 209 x 2
## # Groups:   start_station_id [102]
##    start_station_id start_station_name
##    <chr>            <chr>
##  1 TA1307000138     Lincoln Ave & Roscoe St
##  2 TA1307000138     Wood St & Webster Ave
##  3 TA1306000015     Morgan St & Lake St
##  4 TA1306000015     Sangamon St & Lake St
##  5 954              Prairie Ave & Garfield Blvd
##  6 954              Prairie Ave & Garfield Blvd N
##  7 951              Woodlawn Ave & 63rd St - NE
##  8 951              Woodlawn Ave & 63rd St N
##  9 950              Woodlawn Ave & 63rd St - SE
## 10 950              Woodlawn Ave & 63rd St S
## # i 199 more rows
```

```
full_tripdata %>%
  group_by(end_station_id) %>%
  filter(n_distinct(end_station_name) > 1) %>%
  select(end_station_id, end_station_name) %>%
  distinct() %>%
  arrange(desc(end_station_id))
```

```
## # A tibble: 207 x 2
## # Groups:   end_station_id [101]
##    end_station_id end_station_name
##    <chr>          <chr>
##  1 TA1307000138   Lincoln Ave & Roscoe St
##  2 TA1307000138   Wood St & Webster Ave
##  3 TA1306000015   Morgan St & Lake St
##  4 TA1306000015   Sangamon St & Lake St
##  5 954            Prairie Ave & Garfield Blvd
##  6 954            Prairie Ave & Garfield Blvd N
##  7 951            Woodlawn Ave & 63rd St - NE
##  8 951            Woodlawn Ave & 63rd St N
##  9 950            Woodlawn Ave & 63rd St - SE
## 10 950            Woodlawn Ave & 63rd St S
## # i 197 more rows
```

The number of non-unique station names dropped from over 700 to slightly over 200. Now, the majority of such stations are nearby stations with the same ID but different names, such as *"Woodlawn Ave & 63rd St N"* and *"Woodlawn Ave & 63rd St - NE"*.

Some stations still have the same ID but completely different names. I could dive further into fixing this problem by using station coordinates together with DivvyBikes System Map, but it is irrelevant to the purpose of the current project. Moreover, if I worked for DivvyBikes I would have access to more data to fix this problem without making any guesses. So here I leave it as is.

Let's have a look at the count of distinct station names and IDs once again:

```
full_tripdata %>%
  summarise (
    start_id = start_station_id %>% n_distinct,
    end_id = end_station_id %>% n_distinct,
    start_name = start_station_name %>% n_distinct,
    end_name = end_station_name %>% n_distinct
)
```

```
## # A tibble: 1 x 4
##   start_id end_id start_name end_name
##      <int>  <int>      <int>    <int>
## 1     1310   1316       1380     1385
```

There are more distinct IDs and names for end stations. Let's have a closer look by filtering end station names that are not in start station names:

```
full_tripdata %>%
  filter(!is.na(end_station_name)
         & !(full_tripdata$end_station_name %in% full_tripdata$start_station_name)
         ) %>%
  distinct(end_station_name)
```

```
## # A tibble: 13 x 1
##    end_station_name
##    <chr>
##  1 Linder Ave & Archer Ave
##  2 The Montessori School of Englewood
##  3 James Madison School
##  4 Torrence Ave & 98th St
##  5 Foster Ave & Drake Ave
##  6 Normal Blvd & 61st Pl
##  7 Paul Revere Elementary School
##  8 May St & 87th St
##  9 Lotus Ave & Harrison St
## 10 Ashland Ave & 45th St  S
## 11 Houston Ave & 130th St
## 12 Bishop St & 81st St
## 13 Homan Ave & 62nd Pl
```

Station names look perfectly fine; maybe these are just unpopular. Anyway, let's explore this situation in all possible projections:

```
# Distinct start_station_name that is not in end_station_name
full_tripdata %>%
  filter(!is.na(start_station_name)
         & !(full_tripdata$start_station_name %in% full_tripdata$end_station_name)
  ) %>%
  distinct(start_station_name)
```

```
## # A tibble: 8 x 1
##   start_station_name
##   <chr>
## 1 N Shore Channel Trail & Argyle Ave
## 2 WEST CHI-WATSON
## 3 Oglesby Ave & 105th St
## 4 Lamon Ave & Archer Ave
## 5 Hale Ave & 111th St
## 6 S Kostner Ave & W 18th Pl
## 7 Houston Ave & 131st St
## 8 Ellis Ave & Doty Ave
```

```
# Distinct start_station_id that is not in end_station_id
full_tripdata %>%
  filter(!is.na(start_station_id)
         & !(full_tripdata$start_station_id %in% full_tripdata$end_station_id)
  ) %>%
  distinct(start_station_id)
```

```
## # A tibble: 4 x 1
##   start_station_id
##   <chr>
## 1 DIVVY 001
## 2 849
## 3 780
## 4 901
```

```
# Distinct end_station_id that is not in start_station_id
full_tripdata %>%
  filter(!is.na(end_station_id)
         & !(full_tripdata$end_station_id %in% full_tripdata$start_station_id)
  ) %>%
  distinct(end_station_id)
```

```
## # A tibble: 10 x 1
##    end_station_id
##    <chr>
##  1 782
##  2 741
##  3 732
##  4 1017
##  5 908
##  6 731
##  7 1024
##  8 936
##  9 884
## 10 1037
```

All names and IDs look fine, aside from "DIVVY CASSETTE REPAIR MOBILE STATION" ID, which must belong to a maintenance station. It's better to remove it from the data.

```
full_tripdata <- full_tripdata %>%
  filter(end_station_id != "DIVVY CASSETTE REPAIR MOBILE STATION"
         | is.na(end_station_id)
         )
```

Finally, the data looks clean enough. It's time to add new variables for the future analysis.

Adding New Variables

I add new columns for *date*, *year*, *month*, *week*, *day of week*, and *hour* of when each ride started.

```
full_tripdata$date <- as.Date(full_tripdata$started_at)
full_tripdata$year <- format(as.Date(full_tripdata$started_at), "%Y")
full_tripdata$month <- format(as.Date(full_tripdata$started_at), "%m")
full_tripdata$week <- week(as.POSIXct(full_tripdata$started_at))
full_tripdata$day_of_week <- format(as.Date(full_tripdata$date), "%A")
full_tripdata$hour <- hour(as.POSIXct(full_tripdata$started_at))
```

I also add a variable that takes the value 1 is the trip is round, and 0 in all the other cases.

```
full_tripdata$round_trip <- ifelse((full_tripdata$start_station_id == full_tripdata$end_station_id), 1,
```

Finally, I add a variable for ride duration in seconds.

```
full_tripdata$ride_length <- as.numeric(difftime(full_tripdata$ended_at,full_tripdata$started_at), unit
```

After introducing new variables, I examine a brief statistical summary of the data to get a better understanding of it:

```
summary(full_tripdata)
```

```
##    ride_id          rideable_type        started_at
##  Length:5752026     Length:5752026     Min.   :2022-02-01 00:03:18.00
##  Class :character   Class :character   1st Qu.:2022-06-02 15:19:06.50
##  Mode  :character   Mode  :character   Median :2022-07-27 23:13:45.50
##                                        Mean   :2022-07-29 13:41:40.95
##                                        3rd Qu.:2022-09-22 21:07:59.00
##                                        Max.   :2023-01-31 23:56:09.00
##
##     ended_at                      start_station_name start_station_id
##  Min.   :2022-02-01 00:09:37.00   Length:5752026     Length:5752026
##  1st Qu.:2022-06-02 15:38:48.75   Class :character   Class :character
##  Median :2022-07-27 23:31:35.00   Mode  :character   Mode  :character
##  Mean   :2022-07-29 14:00:58.39
##  3rd Qu.:2022-09-22 21:26:05.25
##  Max.   :2023-02-04 04:27:03.00
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:5752026     Length:5752026     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.   :42.07   Max.   :-87.52
##
##     end_lat         end_lng       member_casual           date
##  Min.   : 0.00   Min.   :-88.14   Length:5752026     Min.   :2022-02-01
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character   1st Qu.:2022-06-02
##  Median :41.90   Median :-87.64   Mode  :character   Median :2022-07-27
##  Mean   :41.90   Mean   :-87.65                      Mean   :2022-07-28
##  3rd Qu.:41.93   3rd Qu.:-87.63                      3rd Qu.:2022-09-22
##  Max.   :42.37   Max.   :  0.00                      Max.   :2023-01-31
##  NA's   :5899    NA's   :5899
##      year              month                 week      day_of_week
##  Length:5752026     Length:5752026     Min.   : 1.0   Length:5752026
##  Class :character   Class :character   1st Qu.:21.0   Class :character
##  Mode  :character   Mode  :character   Median :29.0   Mode  :character
##                                        Mean   :28.7
##                                        3rd Qu.:37.0
##                                        Max.   :53.0
##
##       hour         round_trip       ride_length
##  Min.   : 0.00   Min.   :0.0       Min.   :-621201
##  1st Qu.:11.00   1st Qu.:0.0       1st Qu.:    346
##  Median :15.00   Median :0.0       Median :    612
##  Mean   :14.21   Mean   :0.1       Mean   :   1157
##  3rd Qu.:18.00   3rd Qu.:0.0       3rd Qu.:   1100
##  Max.   :23.00   Max.   :1.0       Max.   :2483235
##                  NA's   :1315919
```

Clearly, there are some outliers in the data that require attention.

Removing Outliers

The first issue is that ride_length obviously can't be negative or zero, yet some observations are.

```
sum(full_tripdata$ride_length <= 0)
```

```
## [1] 533
```

This might be due to time miscalibration across stations or bikes, depending on how this process is organized. Can't solve this issue without additional data, so I exclude rides with negative or zero duration.

```
full_tripdata <- full_tripdata[!(full_tripdata$ride_length <= 0),]
```

The second issue is a remarkably long max ride_length (in a scale of weeks).

Even if it's indeed true that some bike enthusiasts rent bikes for extremely long stretches of time, for the purpose of this analysis it's reasonable to remove such outliers.

Moreover, one-second or ten-second rides don't make much sense either. Whether they stem from customers changing their minds, errors in clock calibration, or Lance Armstrong renting bikes, is unsolvable without additional information.

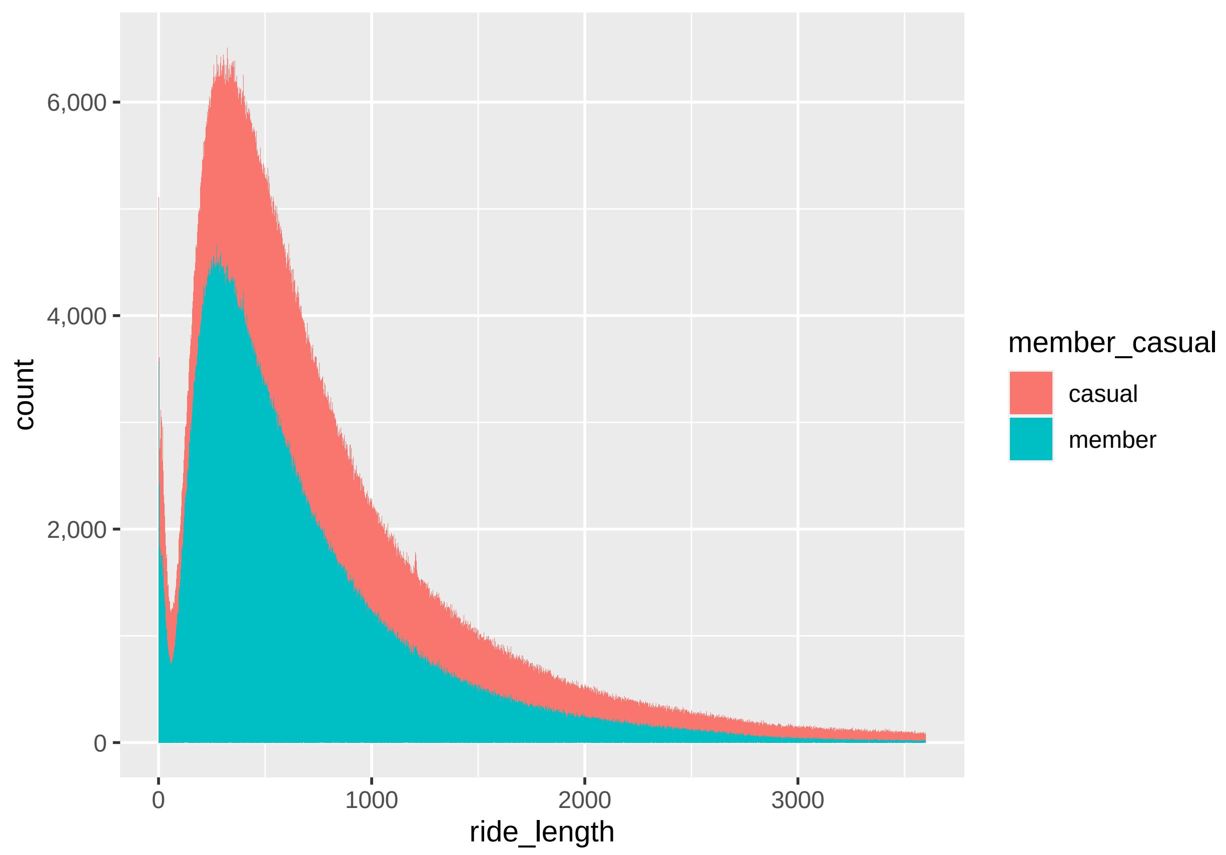
Hence, data trimming. First, I examine the deciles. (Mainly to show my potential employers I'm familiar with basic statistics)

```
quantile(full_tripdata$ride_length, probs = seq(0,1,0.1))
```

```
##      0%     10%     20%     30%     40%     50%     60%     70%     80%     90% 
##       1     205     300     393     494     612     760     962    1278    1905 
##    100% 
## 2483235
```

From the deciles it seems there's some right skew in the curve. It's better to visualize it anyway:

```
ggplot(full_tripdata, aes(x = ride_length, fill = member_casual)) +
  geom_histogram(binwidth = 1, position = "stack") +
  scale_y_continuous(labels = comma) +
  xlim(0,3600)
```



The distribution also has outliers close to a zero point. As I mentioned before, this problem is unsolvable without additional data, so I need to remove them.

The distribution is asymmetrical, so I can't use the Empirical Rule (aka the 68-95-99.7 rule).

Instead, I'll use the interquartile range (IQR) with 3.0 coefficient to remove extreme outliers on the right, and my common sense to remove the outliers on the left.

So, my common sense tells me it's appropriate to remove all the outliers that fall to the left of the lowest point at the inward curve in the first decile.

After experimenting with xlim() values, I can see that the lowest point is located somewhere in the ride_length < 100 range. Precisely, the lowest point has the minimum count(ride_length) where ride_length is less than 100.

```
full_tripdata %>%
  filter(ride_length < 100) %>%
  count(ride_length) %>%
  arrange(n) %>%
  print(n = 1)
```

```
## # A tibble: 99 x 2
##   ride_length     n
##         <dbl> <int>
## 1          65  1178
## # i 98 more rows
```

The cut-off point on the left is 65, so I remove the corresponding rides.

```r
full_tripdata <- full_tripdata[!(full_tripdata$ride_length < 65),]
```

Now, as I've just mentioned, I use IQR with 3.0 coefficient to determine the cut-off point for the outliers on the right. To get the value, I add 3 x IQR to the third quartile.

```r
quantile(full_tripdata$ride_length, probs = 0.75) + 3*IQR(full_tripdata$ride_length)
```

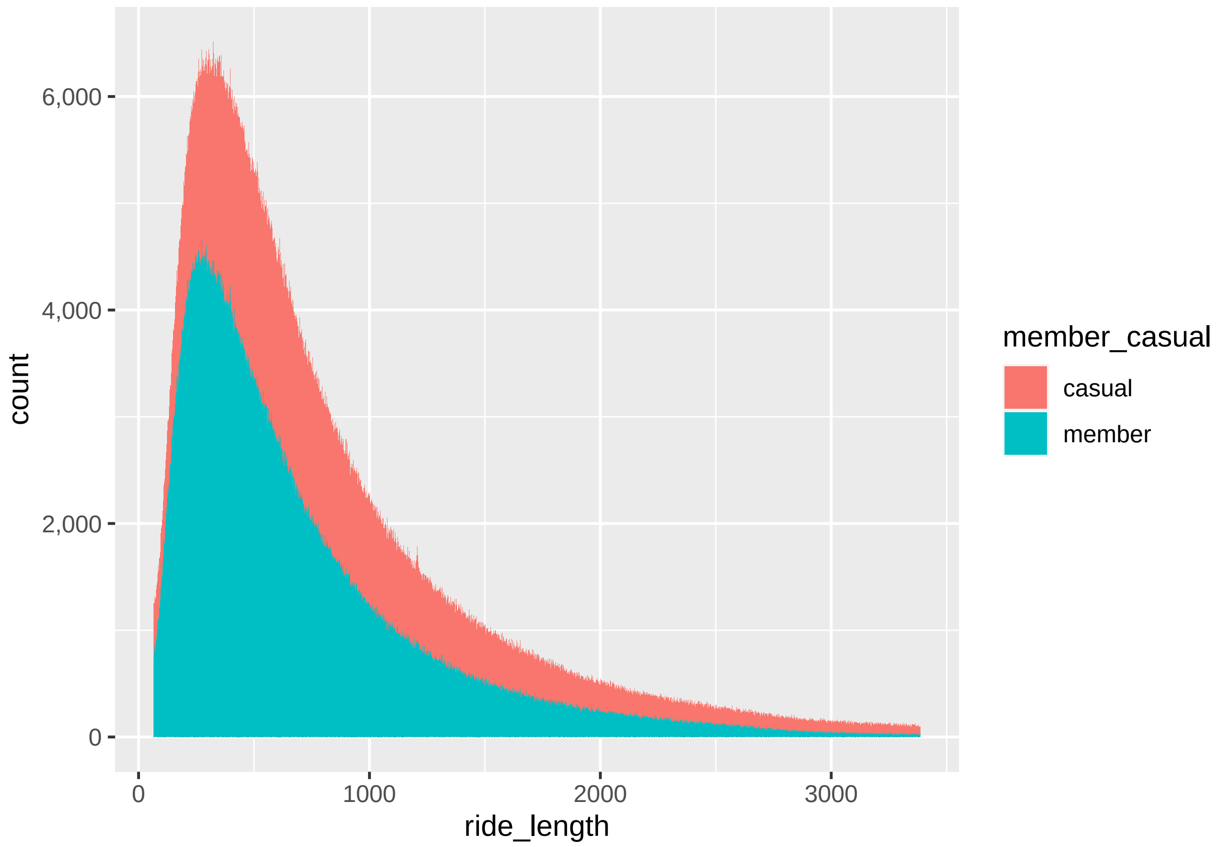
```
##  75%
## 3386
```

Removing the outliers on the right:

```r
full_tripdata <- full_tripdata[!(full_tripdata$ride_length > 3386),]
```

I check the distribution once again to verify it looks more reliable for the current analysis goal.

```r
ggplot(full_tripdata, aes(x = ride_length, fill = member_casual)) +
  geom_histogram(binwidth = 1, position = "stack") +
  scale_y_continuous(labels = comma)
```



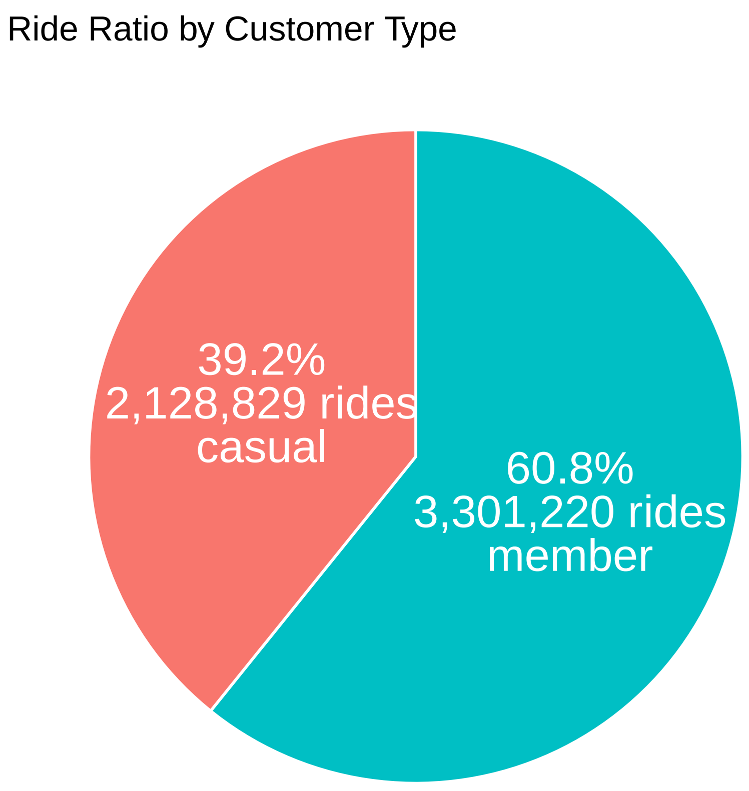
Part 2: Data Analysis and Visualization

A reminder: the purpose of this entire analysis is to find insights on how **casual** Divvy customers differ from annual **members**.

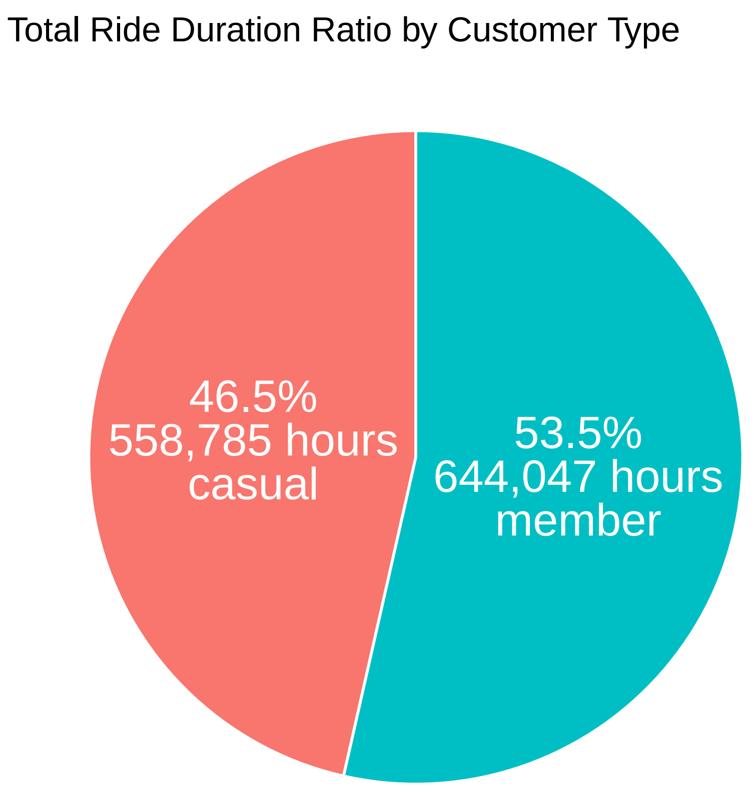
General View

For starters, let's have the most general view on the data:

```r
pie_percent_rides <- full_tripdata %>%
  count(member_casual) %>%
  rename(group = member_casual) %>%
  rename(value = n) %>%
  arrange(desc(group)) %>%
  mutate(prop = value / sum(value) *100)

ggplot(pie_percent_rides, aes(x="", y=value, fill=group)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
  geom_text(aes(label = paste0(round(prop, 2), "%\n", comma(value), " rides\n", group)),
            position = position_stack(vjust = 0.5),
            color = "white", size=6, lineheight=0.8) +
  labs(title = "Ride Ratio by Customer Type") +
  guides(fill = guide_legend(title = "Customer Type"))
```



```r
pie_ride_length_ratio <- full_tripdata %>%
  group_by(member_casual) %>%
  summarise(total_duration = sum(ride_length) / 3600) %>%
  mutate(prop = total_duration / sum(total_duration) * 100)

ggplot(pie_ride_length_ratio, aes(x="", y=total_duration, fill=member_casual)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
  geom_text(aes(label = paste0(round(prop, 1), "%\n", comma(round(total_duration,0)),
                               " hours\n", member_casual)),
              position = position_stack(vjust = 0.5),
              color = "white", size=6, lineheight=0.8) +
  labs(title = "Total Ride Duration Ratio by Customer Type") +
  guides(fill = guide_legend(title = "Customer Type"))
```



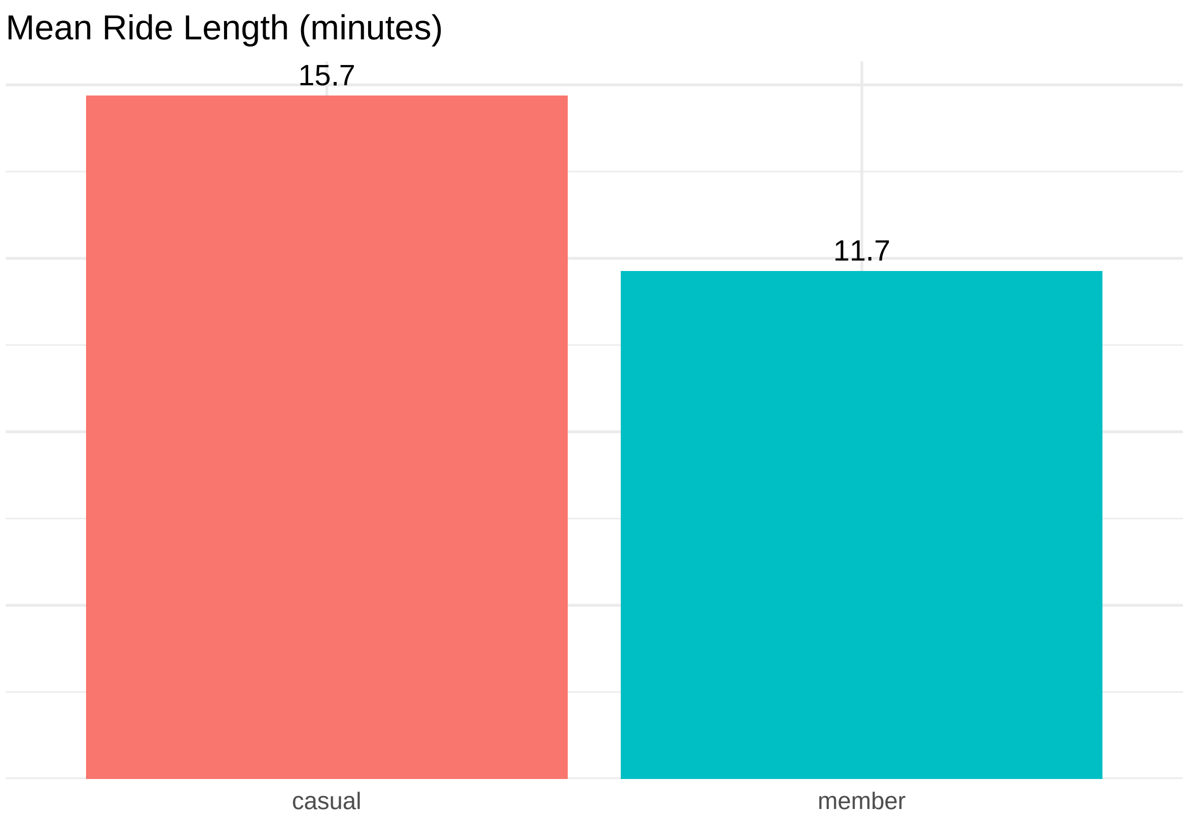
This doesn't give us much insight but helps appreciate the sheer amount of data — over five million rides and over a million hours. Also, it's already evident from these pie charts that casual riders have longer mean ride time.

Let's look at the precise numbers.

Mean Numbers

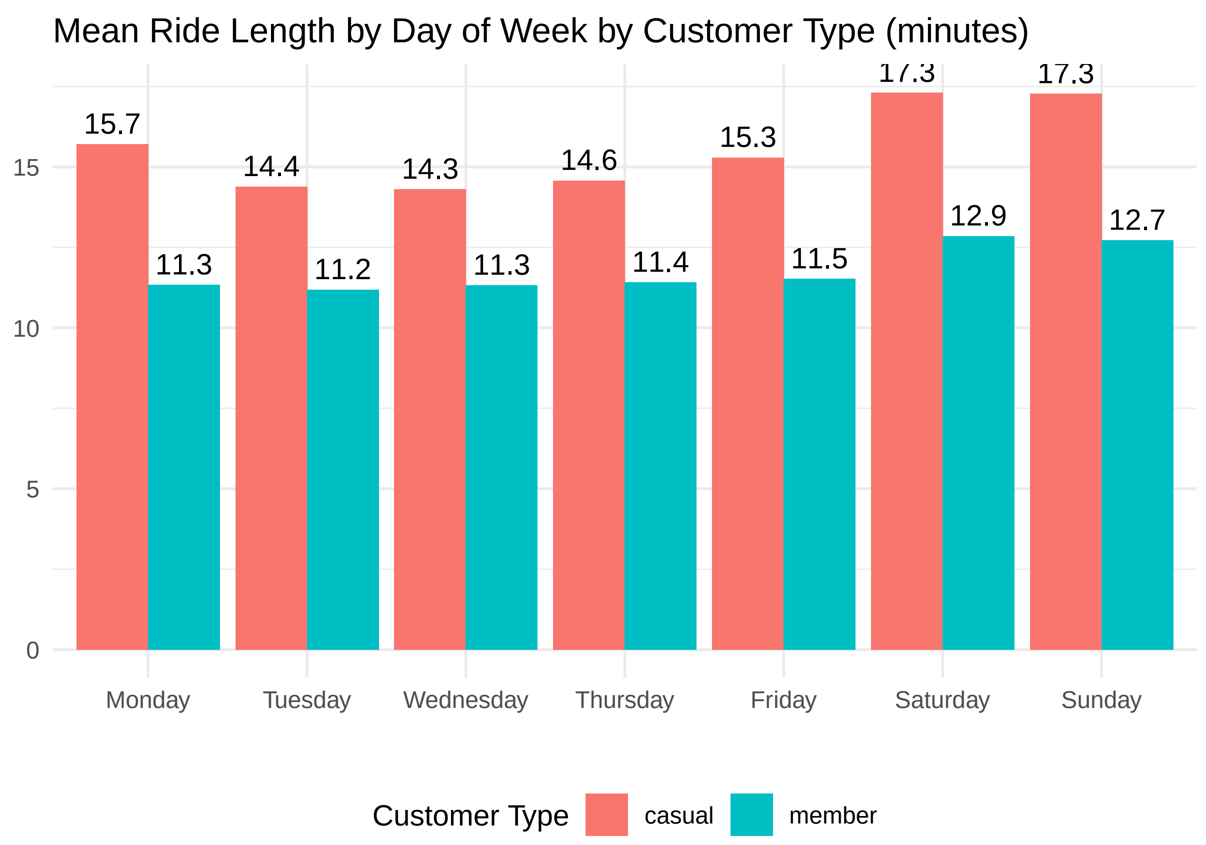
First, let's check general mean ride length:

```
bar_mean_length <- full_tripdata %>%
  group_by(member_casual) %>%
  summarise(mean_length = mean(ride_length) / 60)

ggplot(bar_mean_length, aes(x = member_casual, y = mean_length, fill = member_casual)) +
  theme_minimal() +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = comma(mean_length)), vjust = -0.5) +
  scale_y_continuous(labels = NULL, expand = expansion(mult = c(0, 0.05))) +
  theme(legend.position = "none") +
  labs(
    title = "Mean Ride Length (minutes)",
    x = "",
    y = "",
    fill = "Customer Type"
  )
```



Now, let's check mean ride length distribution across the days of week:

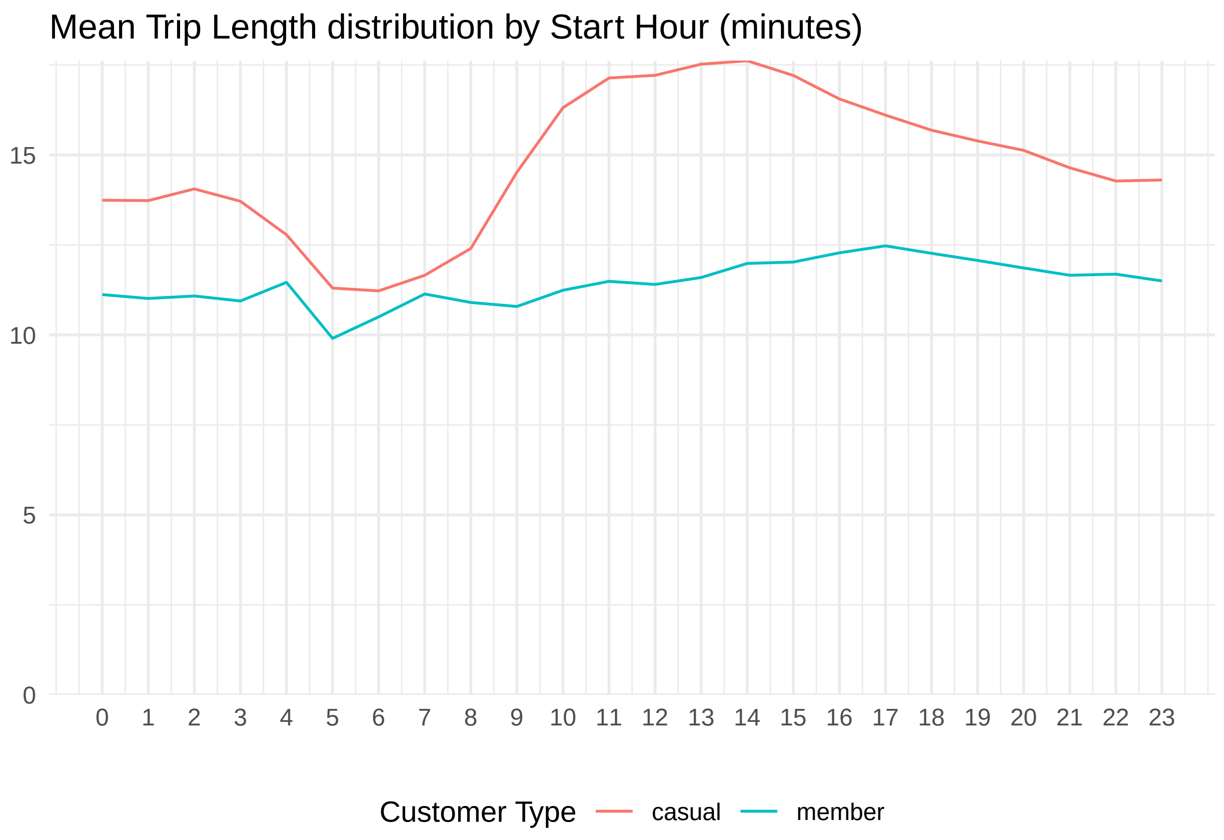
```
bar_mean_length_per_week <- full_tripdata %>%
  group_by(day_of_week, member_casual) %>%
  summarise(mean_length = mean(ride_length) / 60) %>%
  mutate(day_of_week = factor(day_of_week, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Fr
  arrange(day_of_week)

ggplot(bar_mean_length_per_week, aes(x=day_of_week, y=mean_length, fill=member_casual)) +
  theme_minimal() +
  geom_bar(stat="identity", position = "dodge") +
  geom_text(aes(label = round(mean_length,1)), position = position_dodge(width = 0.9), vjust = -0.5) +
  scale_y_continuous(labels = comma) +
  theme(legend.position="bottom") +
  labs(title = "Mean Ride Length by Day of Week by Customer Type (minutes)",
       x = "",
       y = "",
       fill = "Customer Type")
```



Evidently, mean ride lengths are higher on weekends for both groups. However, **members** tend to have a more even distribution of mean ride lengths across weekdays than **casual** customers.

Might it be because members use bikes mainly to commute, while casuals use it for a wider variety of reasons? Anyway, let's see how mean ride lengths are distributed across the hours when the rides started:

```
line_ride_length_hour <- full_tripdata %>%
  group_by(hour, member_casual) %>%
  summarise(mean_ride_length = mean(ride_length) / 60) %>%
  arrange(hour)

ggplot(line_ride_length_hour, aes(x = hour, y = mean_ride_length, color = member_casual)) +
  geom_line() +
  scale_x_continuous(breaks = unique(line_ride_length_hour$hour)) +
  scale_y_continuous(expand = c(0, 0),
                     limits = c(0, max(line_ride_length_hour$mean_ride_length))) +
  theme_minimal() +
  labs(title = "Mean Trip Length distribution by Start Hour (minutes)",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```



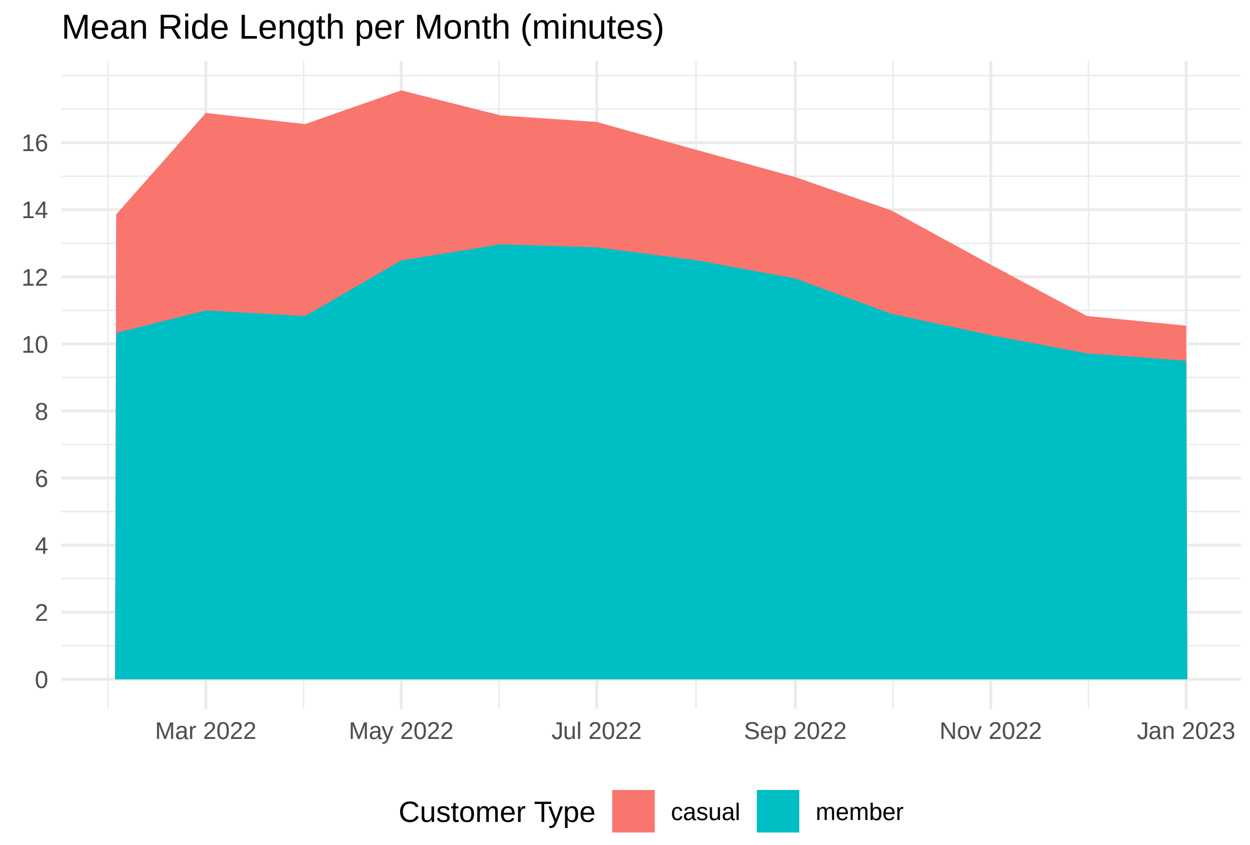
The two graphs follow the somewhat similar pattern, however, the spread of values for casuals is significantly larger.

Why is the spread? Is it due to members being more efficient with riding a bicycle, so they need less time than casuals to travel equal distance during busy hours?

This hypothesis is worth testing; however, the current database doesn't have the data on distance. I can make some assumptions based on start station and end station coordinates, but such assumptions won't be accurate. Also, this hypothesis can be tested based on stats for distinct customers, but there's no such data available.

To finish with mean values, here's mean ride length per month:

```
area_mean_ride_month <- full_tripdata %>%
  group_by(month = floor_date(date, unit = "month"), member_casual) %>%
  summarise(mean_length = mean(ride_length) / 60) %>%
  arrange(month)

ggplot(area_mean_ride_month, aes(x = month, y = mean_length, fill = member_casual)) +
  geom_area(position = "identity") +
  theme_minimal() +
  theme(legend.position = "bottom",
        axis.title.x = element_blank(),
        axis.title.y = element_blank()) +
  scale_x_date(date_breaks = "2 month", date_labels = "%b %Y") +
  scale_y_continuous(breaks = seq(0, max(area_mean_ride_month$mean_length), 2)) +
  labs(title = "Mean Ride Length per Month (minutes)",
       fill = "Customer Type")
```



Clearly, the warmer the weather, the longer the rides. Also, it's evident that in December and January the difference in mean ride length between customer groups is minimal.

Is it because in cold weather all customers mainly use bikes to just commute, whereas in good weather they slow down to enjoy the ride? Then why do casuals slow down more?

Or, is it because warm weather attracts more less-skilled riders, making the gap in mean ride length between groups grow?

Or, is it because in warm weather customers travel longer distances and also members are more experienced riders, so the gap tends to grow wider with the more distance traveled?

My common sense tells me all of the factors above take place, but these are too many hypotheses to test with so limited data. Anyway, the fact is that for some reason in warmer months **casual** riders tend to have proportionally longer rides than **members**, and in colder months this gap is much smaller.

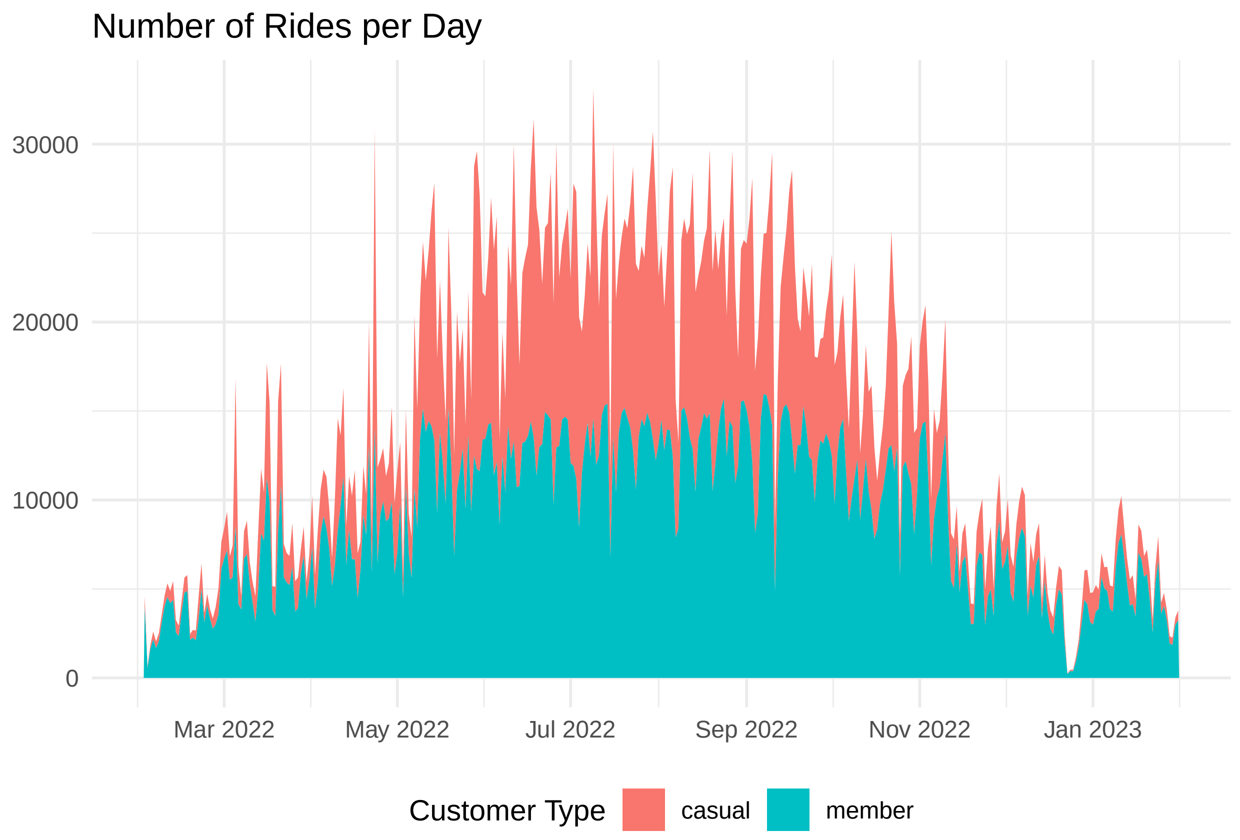
Mean numbers are informative, but what if such a gap is due to the sample size? What if only a tiny fraction of customers that use bikes in winter are casual riders?

To answer that, I need to analyse ridership from various angles.

Ridership Analysis

Let's start with a daily ridership distribution:

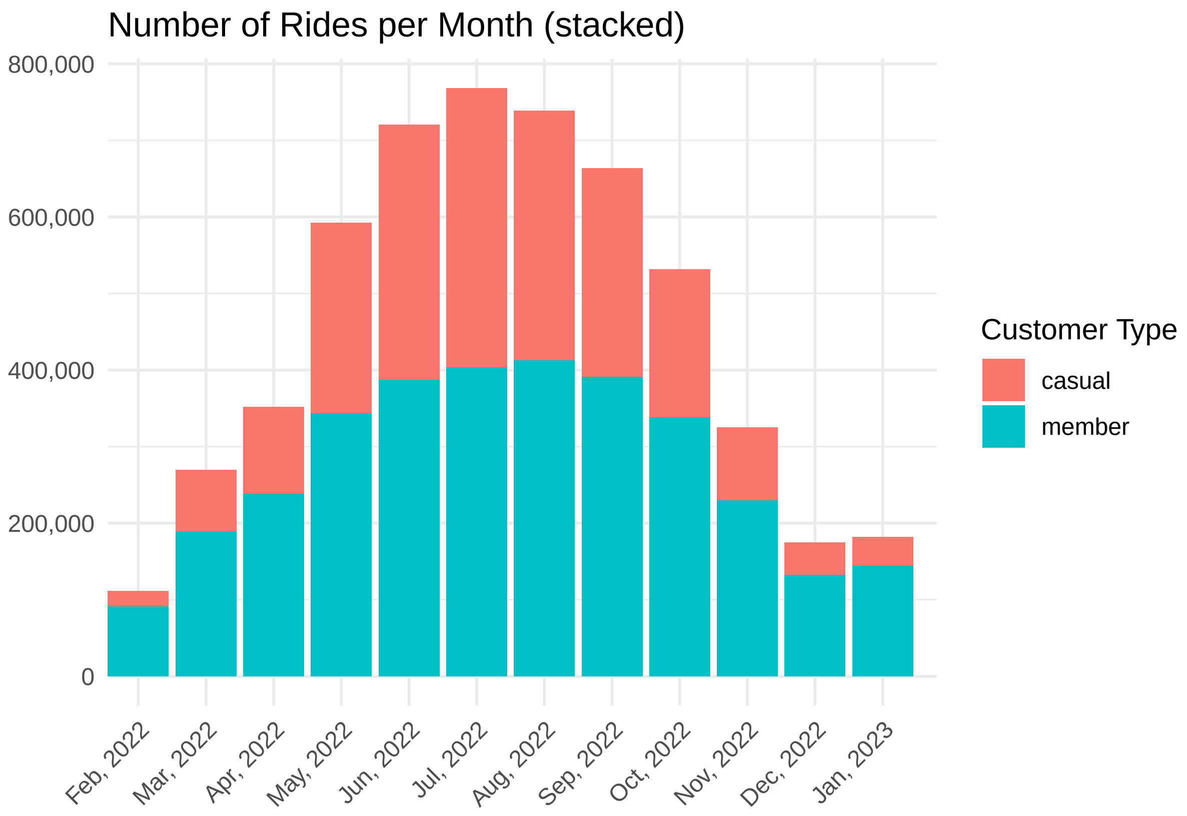
```r
area_ride_count_date <- full_tripdata %>%
  group_by(date,member_casual) %>%
  summarise(ride_count = n()) %>%
  arrange(date)

ggplot(area_ride_count_date, aes(x = date, y = ride_count, fill = member_casual)) +
  geom_area() +
  theme_minimal() +
  theme(legend.position="bottom",
        axis.title.x = element_blank(),
        axis.title.y = element_blank()) +
  scale_x_date(breaks = "2 month", labels = date_format("%b %Y")) +
  labs(title = "Number of Rides per Day",
       fill = "Customer Type")
```



Truly a terrifyingly looking plot. To improve readability, I aggregate it by month:

```r
rides_month <- full_tripdata %>%
  group_by(month, member_casual, year) %>%
  summarise(rides_count = n()) %>%
  arrange(year)
```

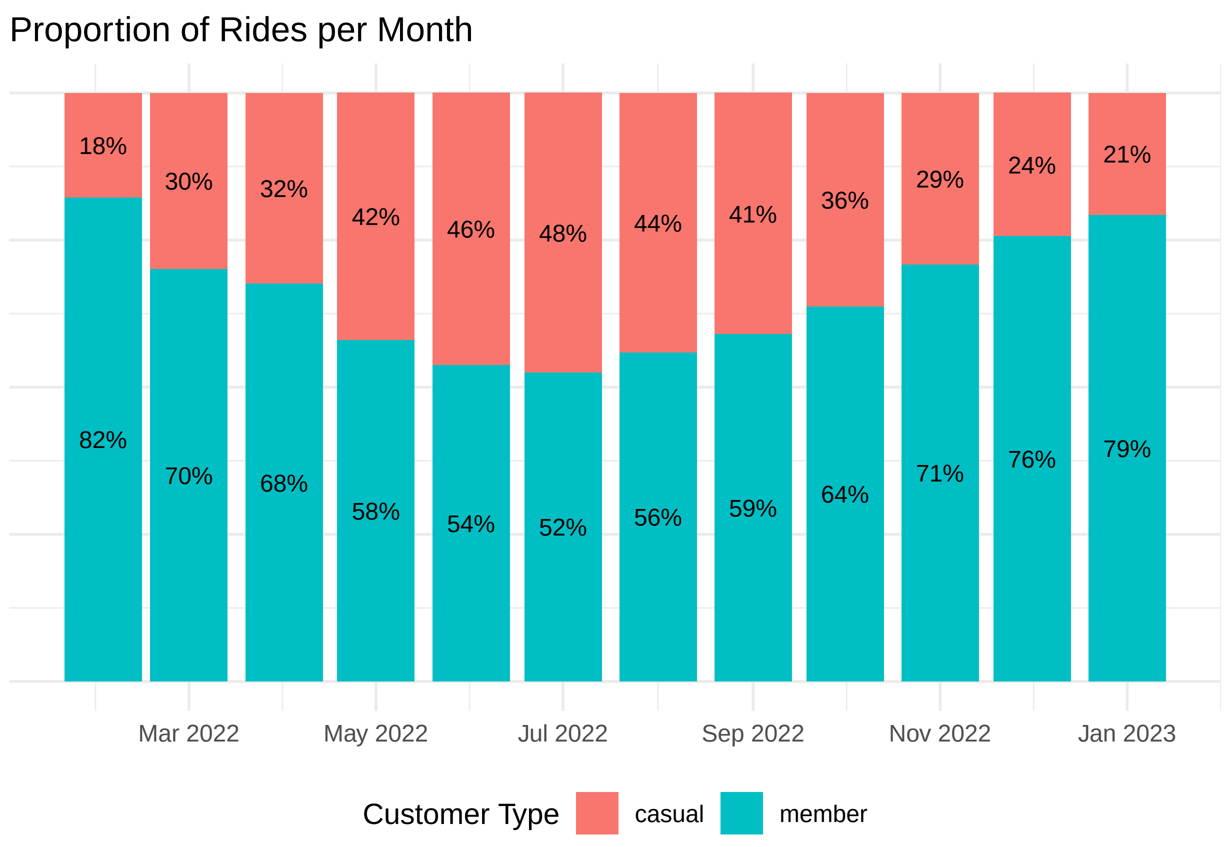
```r
ggplot(rides_month, aes(x = month, y = rides_count, fill = member_casual)) +
  geom_col(position = "stack") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  scale_x_discrete(limits = rides_month$month,
                   label = paste0(month.abb[as.integer(rides_month$month)],", ",
                                  rides_month$year)) +
  labs(title = "Number of Rides per Month (stacked)",
       x = "",
       y = "",
       fill = "Customer Type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From this stacked area chart it's clear that in December, January, and February casual riders are indeed significantly less present in the ridership proportion.

For a more precise proportion, I build a filled column chart:

```r
col_ride_count_month <- full_tripdata %>%
  group_by(month = floor_date(date, unit = "month"), member_casual) %>%
  summarise(ride_count = n()) %>%
  arrange(month) %>%
  group_by(month) %>%
  mutate(ride_count_prop = ride_count / sum(ride_count))

ggplot(col_ride_count_month, aes(x = month, y = ride_count_prop, fill = member_casual)) +
  geom_col(position = "fill") +
  geom_text(aes(label = paste0(round(ride_count_prop * 100), "%")),
            position = position_fill(vjust = 0.5), size = 3) +
  theme_minimal() +
  theme(legend.position = "bottom",
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.y = element_blank()) +  # Remove Y-axis title and text
  scale_x_date(date_labels = "%b %Y", date_breaks = "2 month") +
  labs(title = "Proportion of Rides per Month",
       fill = "Customer Type")
```



In December, January, and February only roughly 1 out of each 5 rides is by casual riders. Adding the previous mean ride length analysis to this, I reckon that during these months casual riders mostly use bikes for the same purpose, as members.

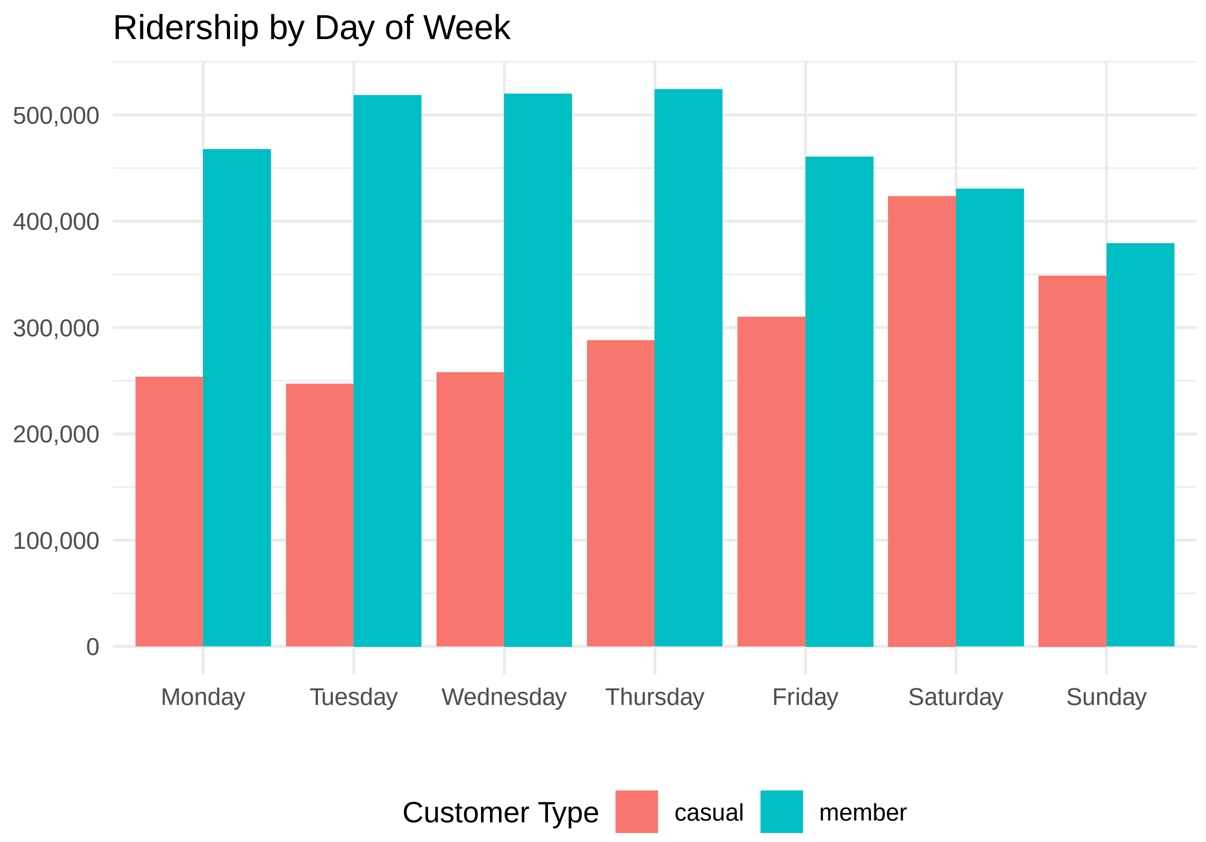
Hence, to turn such casuals into members, they should be marketed accordingly. Why don't they buy the annual membership subscription?

- *If they don't see enough value for their dollar*, Divvy could send them infographics, showing how much they personally could benefit from buying annual membership.
- *If they don't ride bikes too often*, they could be lured into riding more often by providing personal discounts (maybe during certain hours and so on), and after they adapt to riding often enough so the annual subscription is worth buying, send them infographics from the previous paragraph.

The last insight seems good, but it's not enough. For more food for thought I need to look at the ridership numbers.

Here's the bar plot showing the number of rides grouped by day of week:

```r
bar_trips_count_week_per_membership <- full_tripdata %>%
  group_by(day_of_week, member_casual) %>%
  summarise(trips = sum(!is.na(ride_id))) %>%
  mutate(day_of_week = factor(day_of_week,
                              levels = c("Monday", "Tuesday", "Wednesday",
                                         "Thursday", "Friday", "Saturday",
                                         "Sunday"))) %>%
  arrange(day_of_week)

ggplot(bar_trips_count_week_per_membership, aes(x=day_of_week, y=trips, fill=member_casual)) +
  theme_minimal() +
  geom_bar(stat="identity", position = "dodge") +
  scale_y_continuous(labels = comma) +
  theme(legend.position="bottom") +
  labs(title = "Ridership by Day of Week",
       x = "",
       y = "",
       fill = "Customer Type")
```

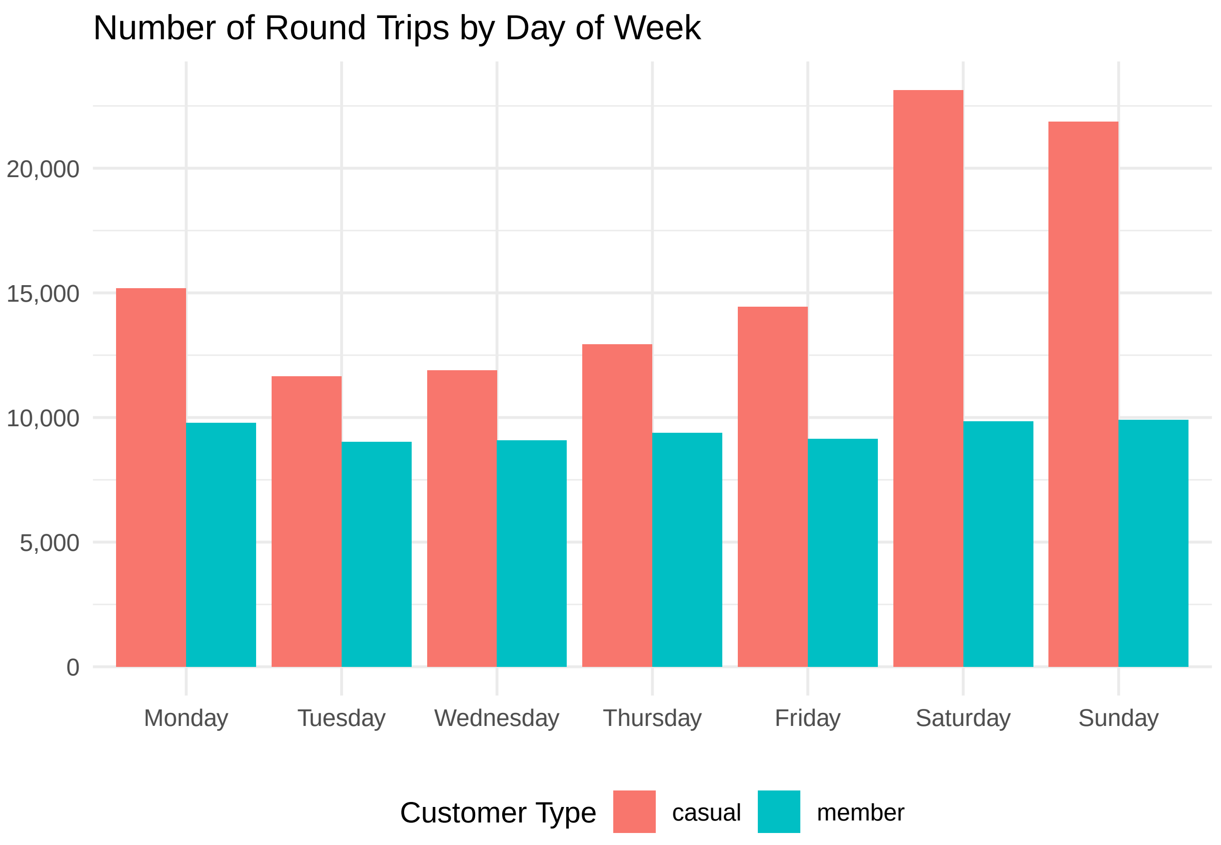


The bar shows evident differences — **casual** ridership grows on weekends, whereas **member** ridership drops.

It's intuitive that on weekends people commute less and ride for fun more. From *Mean Ride Length by Day of Week* plot I already know that mean ride length grows on weekends for the both groups.

Now, I assume that if on weekends customers ride for fun more, it should result in a higher number of round trips.

```
bar_round_trips_dow <- full_tripdata %>%
  group_by(day_of_week, member_casual) %>%
  summarise(num_round_trip = sum(round_trip, na.rm = TRUE)) %>%
  mutate(day_of_week = factor(day_of_week,
                              levels = c("Monday", "Tuesday", "Wednesday",
                                         "Thursday", "Friday", "Saturday",
                                         "Sunday"))) %>%
  arrange(day_of_week)


ggplot(bar_round_trips_dow, aes(x = day_of_week, y = num_round_trip, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  labs(title = "Number of Round Trips by Day of Week",
       fill = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```

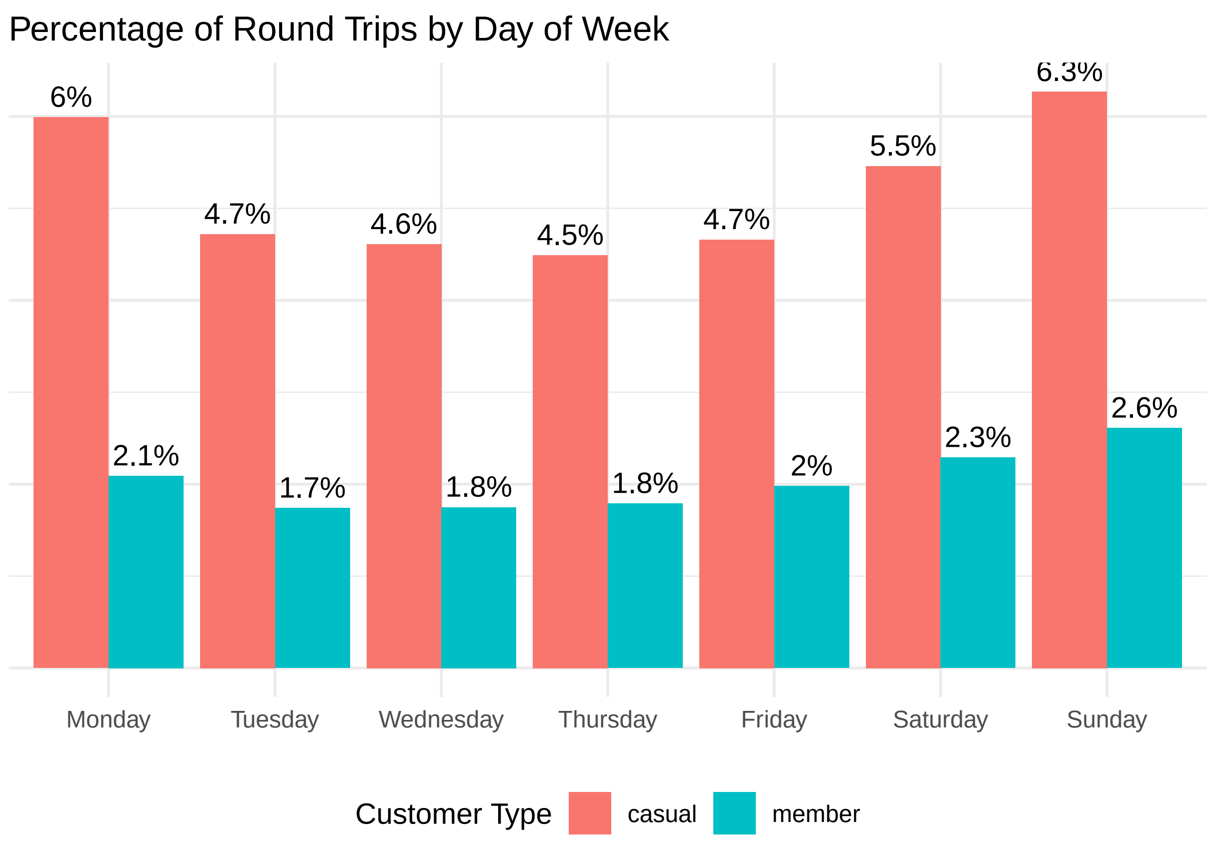


The number of round trips is higher on weekends, which proves my hypothesis of the growing number of recreational rides on Saturday and Sunday. Or, does it?

For **members**, yes, because overall member ridership drops on weekends, yet the number of round trips grows. But what about **casuals**, whose overall ridership also grows on weekends?

To answer this, I'm building a *Ratio of Round Trips by Day of Week* graph, which shows what percentage of total rides for each day are round.

```
round_trips_dow <- full_tripdata %>%
  group_by(day_of_week, member_casual) %>%
  summarise(num_round_trip = sum(round_trip, na.rm = TRUE), total_trips = n(),
            ratio = round((num_round_trip / total_trips)*100,2)) %>%
  mutate(day_of_week = factor(day_of_week,
                              levels = c("Monday", "Tuesday", "Wednesday",
                                         "Thursday", "Friday", "Saturday",
                                         "Sunday"))) %>%
  arrange(day_of_week)

ggplot(round_trips_dow, aes(x = day_of_week, y = ratio, fill = member_casual)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = paste0(round(ratio, 1), "%")),
            position = position_dodge(width = 0.9), vjust = -0.5) +
  theme_minimal() +
  scale_y_continuous(labels = NULL) +
  labs(title = "Percentage of Round Trips by Day of Week",
       fill = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```

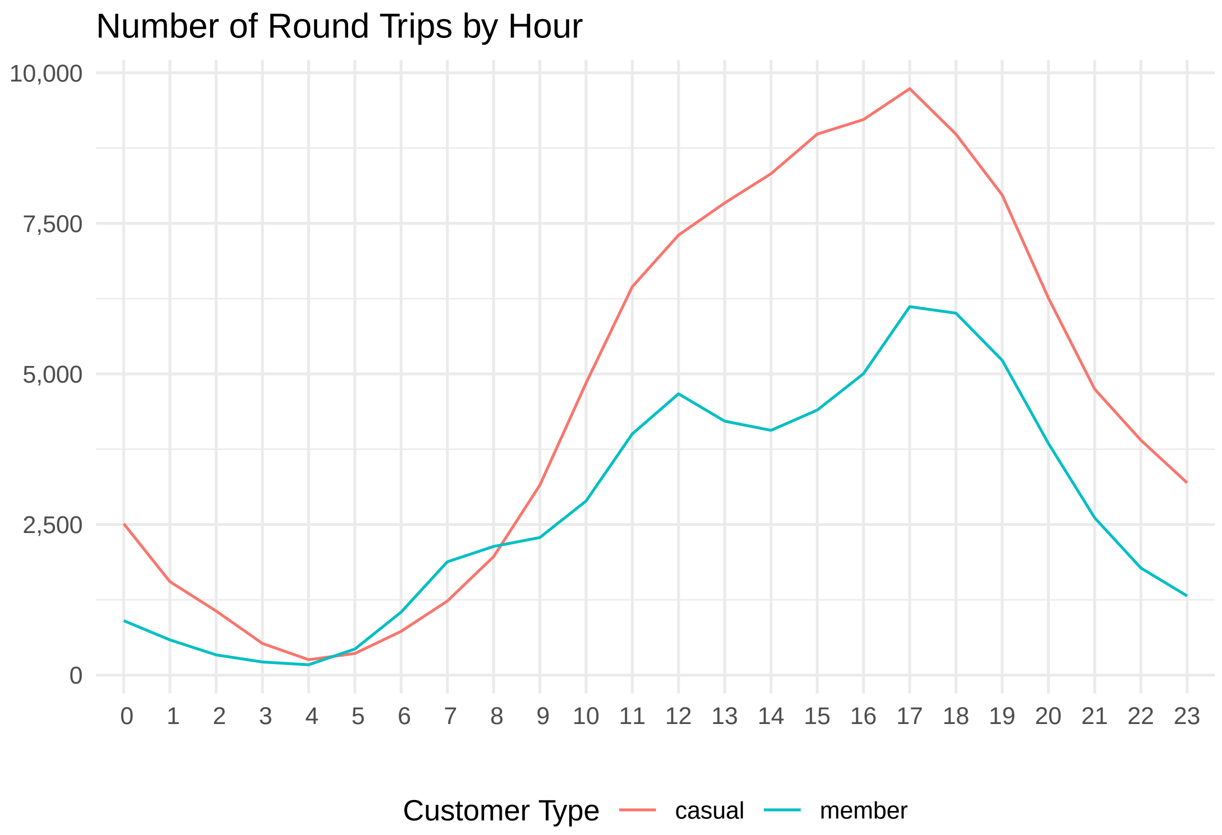


Now it's evident that the number of round trips grows on weekends for both groups. For some reason, it also grows on Mondays. Why is that? Do people in Chicago typically have Monday as a weekend? Or do they enjoy their round trips on Saturday and Sunday, so they decide to have more on Monday? These are questions for further investigation, but are out of the scope of this analysis.

However, I have one more insight — **casual** riders make more round trips, than **members**. Moreover, it's interesting to look at how round trips are distributed across the hours of when such rides start.

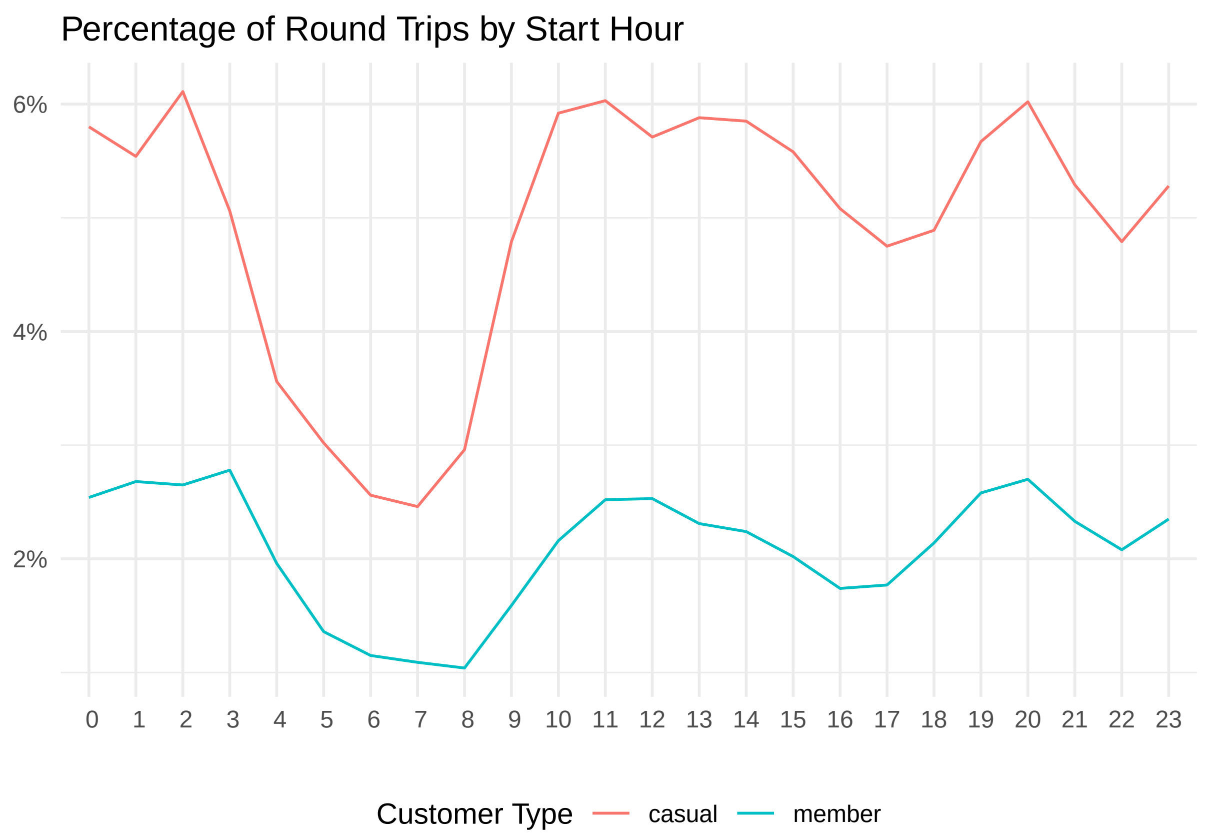
Absolute numbers:

```r
area_round_trips <- full_tripdata %>%
  group_by(hour, member_casual) %>%
  summarise(num_round_trip = sum(round_trip, na.rm = TRUE)) %>%
  arrange(hour)

ggplot(area_round_trips, aes(x = hour, y = num_round_trip, color = member_casual)) +
  geom_line() +
  scale_x_discrete(limits = area_round_trips$hour) +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  labs(title = "Number of Round Trips by Hour",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```



Percentages:

```r
line_round_trips <- full_tripdata %>%
  group_by(hour, member_casual) %>%
  summarise(num_round_trip = sum(round_trip, na.rm = TRUE), total_trips = n(),
            ratio = round((num_round_trip / total_trips)*100,2)) %>%
  arrange(hour)

ggplot(line_round_trips, aes(x = hour, y = ratio, color = member_casual)) +
  geom_line() +
  scale_x_discrete(limits = line_round_trips$hour) +
  theme_minimal() +
  scale_y_continuous(labels = function(x) paste0(x, "%")) +  # Add % sign to Y-axis labels
  labs(title = "Percentage of Round Trips by Start Hour",
       color = "Customer Type",
       x = element_blank(),
       y = "") +
  theme(legend.position = "bottom")
```



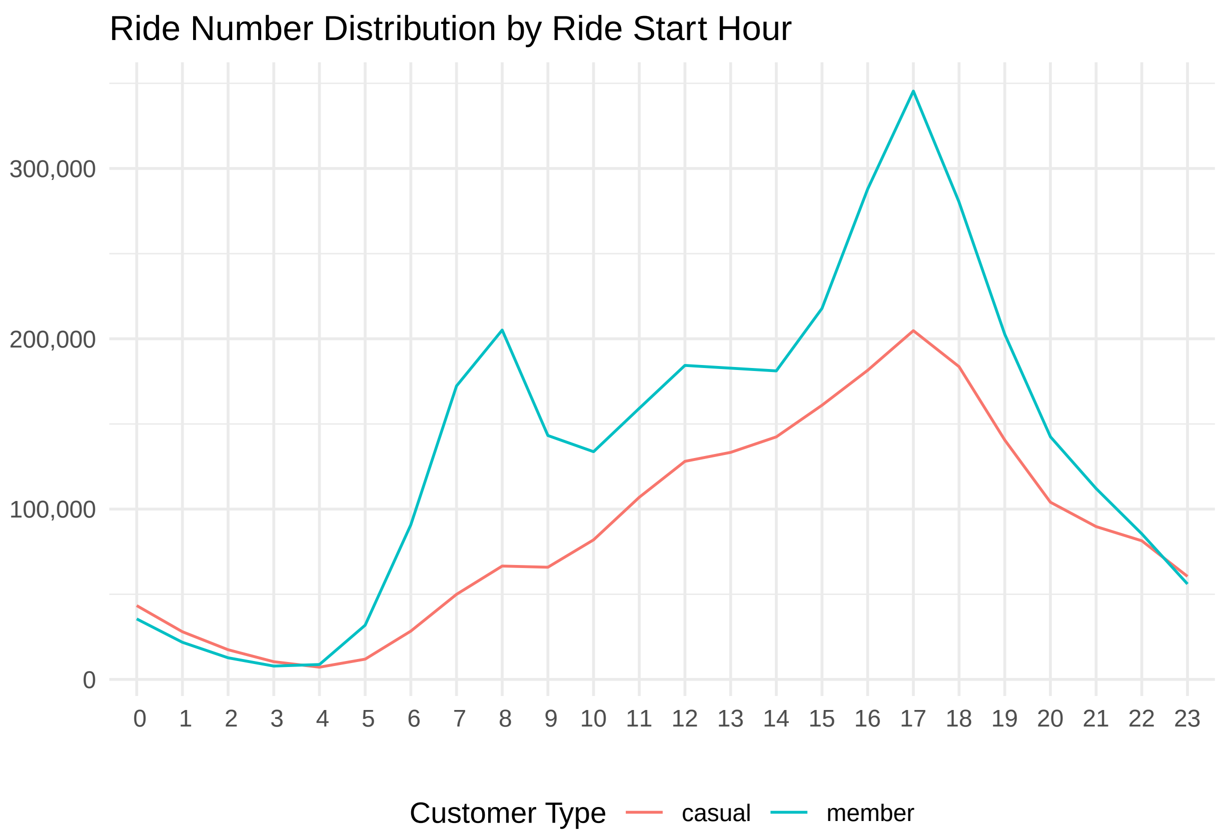
Nothing special here, both groups show the same pattern where the percentage of round trips drops at late-late night and during early commute hours.

After looking at how round trips are distributed across the hours of when they start, the next logical progression is to do this to all rides.

Absolute numbers:

(How many rides took place at each hour)

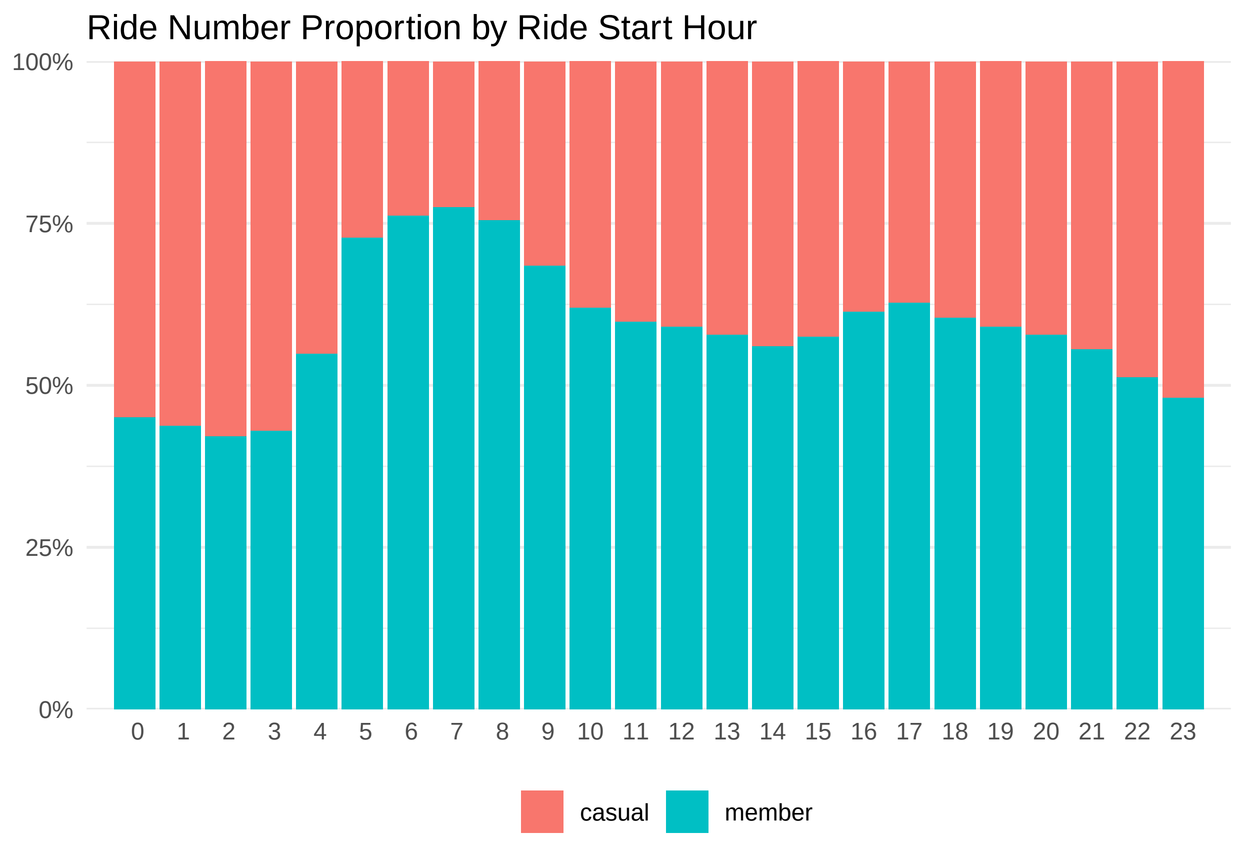
```
line_ride_count_hour <- full_tripdata %>%
  group_by(hour, member_casual) %>%
  summarise(ride_count = n()) %>%
  arrange(hour)

ggplot(line_ride_count_hour, aes(x = hour, y = ride_count, color = member_casual)) +
  geom_line() +
  scale_x_discrete(limits = line_ride_count_hour$hour) +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  labs(title = "Ride Number Distribution by Ride Start Hour",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```



Proportion:

(Out of all rides for each hour, what is the proportional contribution by members and casuals)

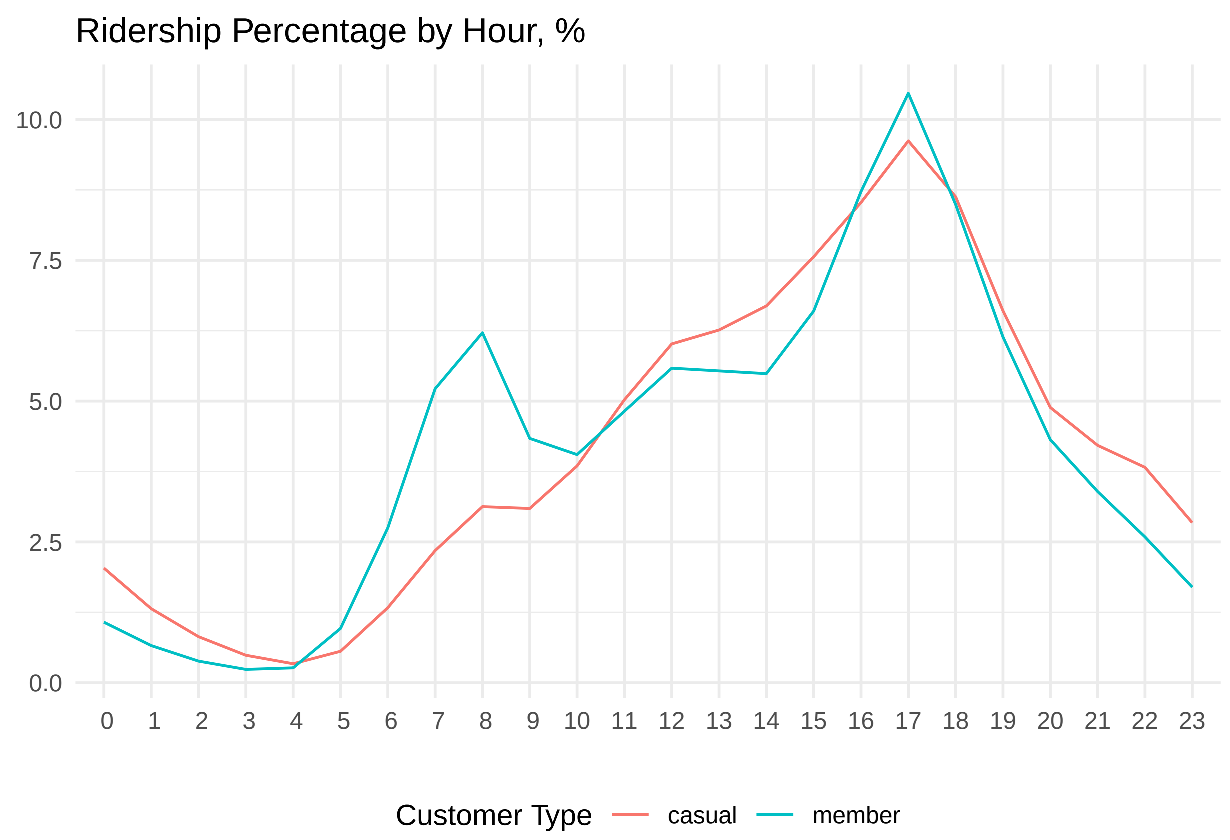
```
ggplot(full_tripdata, aes(x = hour, fill = member_casual)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = function(x) paste0(x * 100, "%"), expand = c(0, 0)) +
  scale_x_discrete(limits = full_tripdata$hour) +
  labs(x = NULL, y = NULL, title = "Ride Number Proportion by Ride Start Hour") +
  guides(fill = guide_legend(title = NULL)) +
  theme_minimal() +
  theme(legend.position = "bottom")
```



Percentages:

(What percentage of total rides is contributed by members and casuals at each hour)

```
line_ratio_ride_hour <- full_tripdata %>%
  group_by(hour, member_casual) %>%
  summarise(ratio_ride_count = n()) %>%
  arrange(hour) %>%
  group_by(member_casual) %>%
  mutate(total = sum(ratio_ride_count),
         prop = ratio_ride_count / total * 100)

ggplot(line_ratio_ride_hour, aes(x = hour, y = prop, color = member_casual)) +
  geom_line() +
  scale_x_discrete(limits = line_ratio_ride_hour$hour) +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  labs(title = "Ridership Percentage by Hour, %",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.position = "bottom")
```

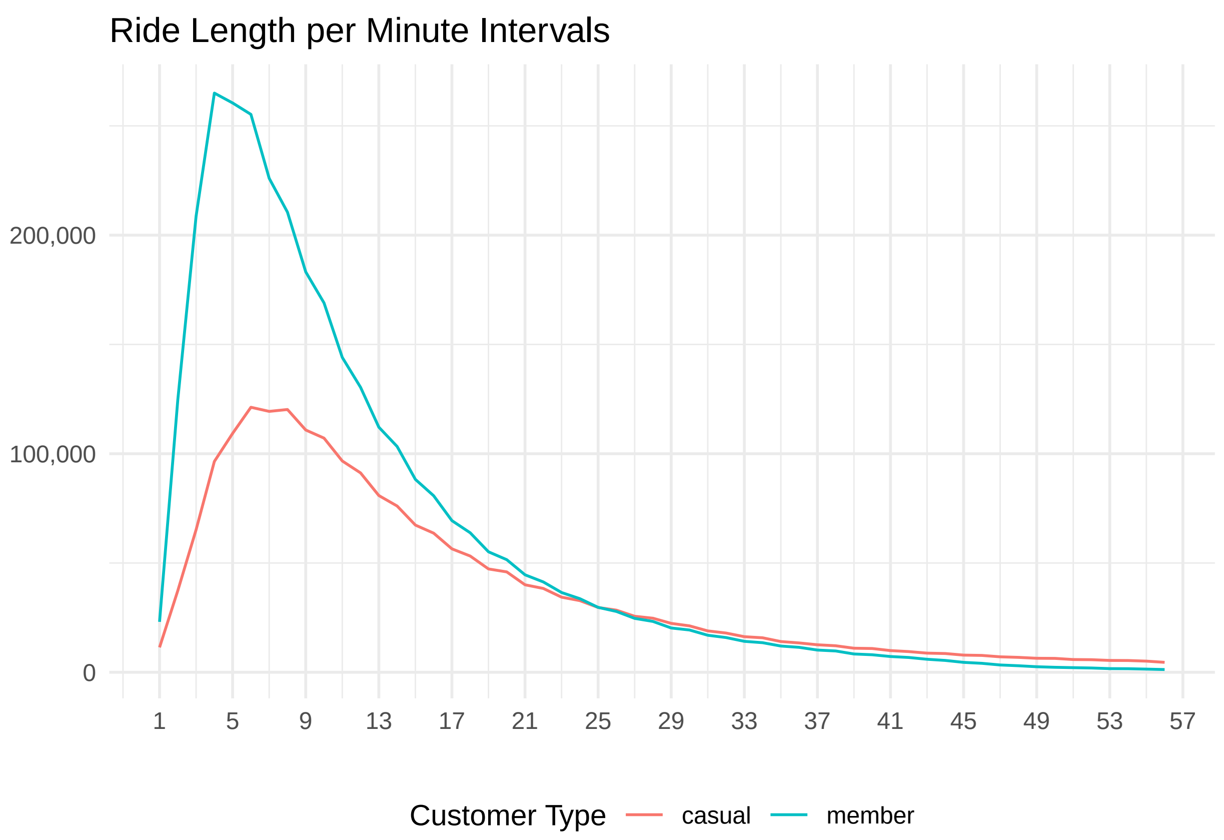


Both groups have a spike in rides at around 5 PM, which might be a sign of them using bikes to commute from work. However, **members** also have a spike at around 8 AM, whereas casual riders don't have it.

It's possible that **casual** riders don't find bikes a good enough option to commute **to** work. Why? It's a question for further investigation. Yet, this is one more insight.

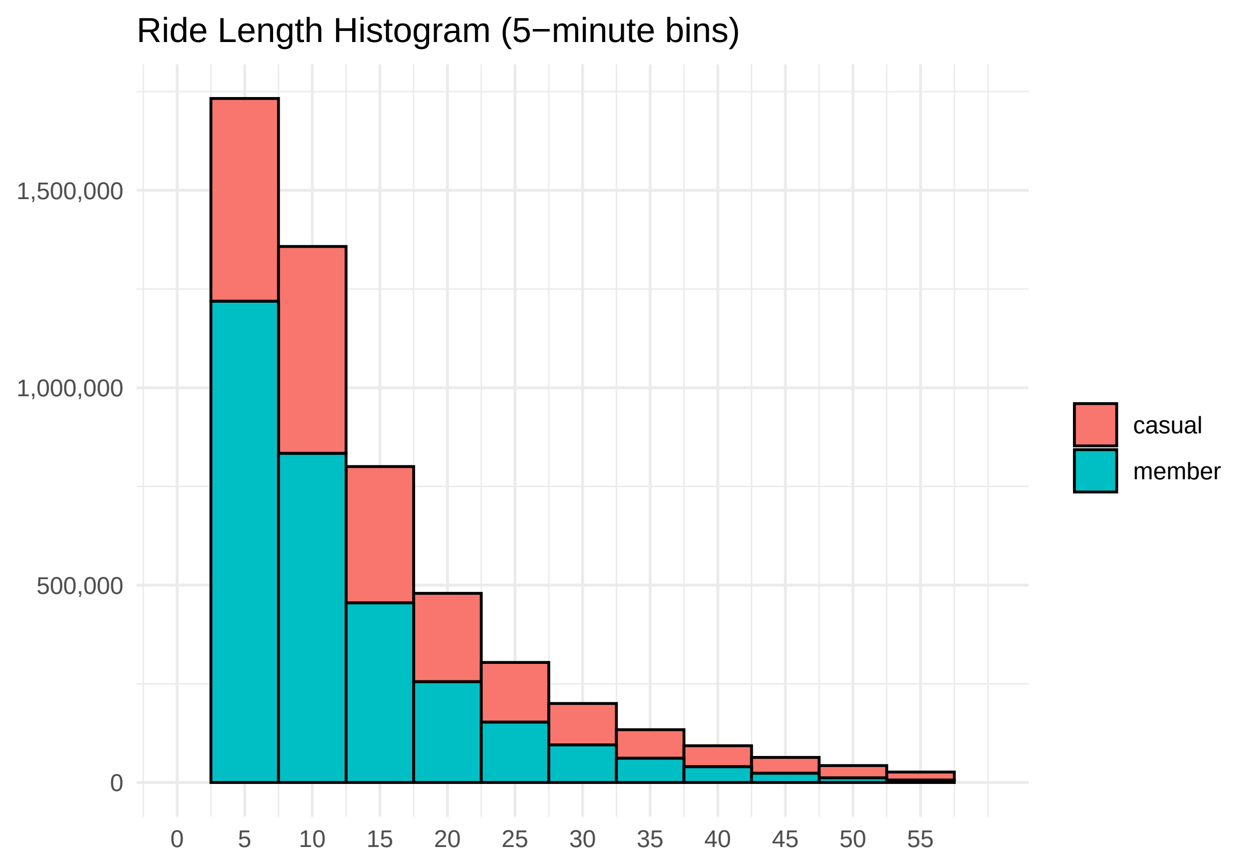
Now, to get a better grasp of how ridership is distributed duration-wise , I look at a couple more plots. First, ride length distribution per minute intervals:

```r
trip_length_minutes <- full_tripdata %>%
  group_by(ride_length, member_casual) %>%
  summarise(num_rides = n(),
            ride_length_minutes = round(ride_length/60)) %>%
  group_by(ride_length_minutes, member_casual) %>%
  mutate(total_rides = n())

ggplot(trip_length_minutes, aes(x = ride_length_minutes, y = total_rides,
                                color = member_casual)) +
  geom_line() +
  scale_y_continuous(label = comma) +
  scale_x_continuous(breaks = seq(1,58,4)) +
  theme_minimal() +
  theme(legend.position = "bottom") +
  labs(title = "Ride Length per Minute Intervals",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank())
```



Second, ride length histogram with 5-minute bins:

```r
ggplot(full_tripdata, aes(x = ride_length/60, fill = member_casual)) +
  geom_histogram(binwidth = 5, color = "black", position = "stack") +
  scale_x_continuous(limits = c(0, 60), breaks = seq(0, 55, 5)) +
  scale_y_continuous(label = comma) +
  theme_minimal() +
  labs(title = "Ride Length Histogram (5-minute bins)",
       fill = "Customer Type") +
  theme(legend.title = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank())
```



For precise numbers I look at the deciles once again:

```
full_tripdata %>%
  group_by(member_casual) %>%
  summarise(
    `10%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.1)), format = "%Mm %Ss"),
    `20%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.2)), format = "%Mm %Ss"),
    `30%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.3)), format = "%Mm %Ss"),
    `40%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.4)), format = "%Mm %Ss"),
    `50%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.5)), format = "%Mm %Ss"),
    `60%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.6)), format = "%Mm %Ss"),
    `70%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.7)), format = "%Mm %Ss"),
    `80%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.8)), format = "%Mm %Ss"),
    `90%` = format(as.POSIXct(0, origin = "1970-01-01") +
                     round(quantile(ride_length, probs = 0.9)), format = "%Mm %Ss")
  )
```

```
## # A tibble: 2 x 10
##   member_casual ‘10%‘   ‘20%‘   ‘30%‘   ‘40%‘   ‘50%‘   ‘60%‘  ‘70%‘ ‘80%‘ ‘90%‘
##   <chr>         <chr>   <chr>   <chr>   <chr>   <chr>   <chr>  <chr> <chr> <chr>
## 1 casual        04m 32s 06m 23s 08m 09s 10m 03s 12m 15s 14m 5~ 18m ~ 23m ~ 32m ~
## 2 member        03m 23s 04m 39s 05m 55s 07m 18s 08m 55s 10m 5~ 13m ~ 17m ~ 24m ~
```

Casuals: *50%* of rides are shorter than 12m 15s; *90%*, 32m 45s.

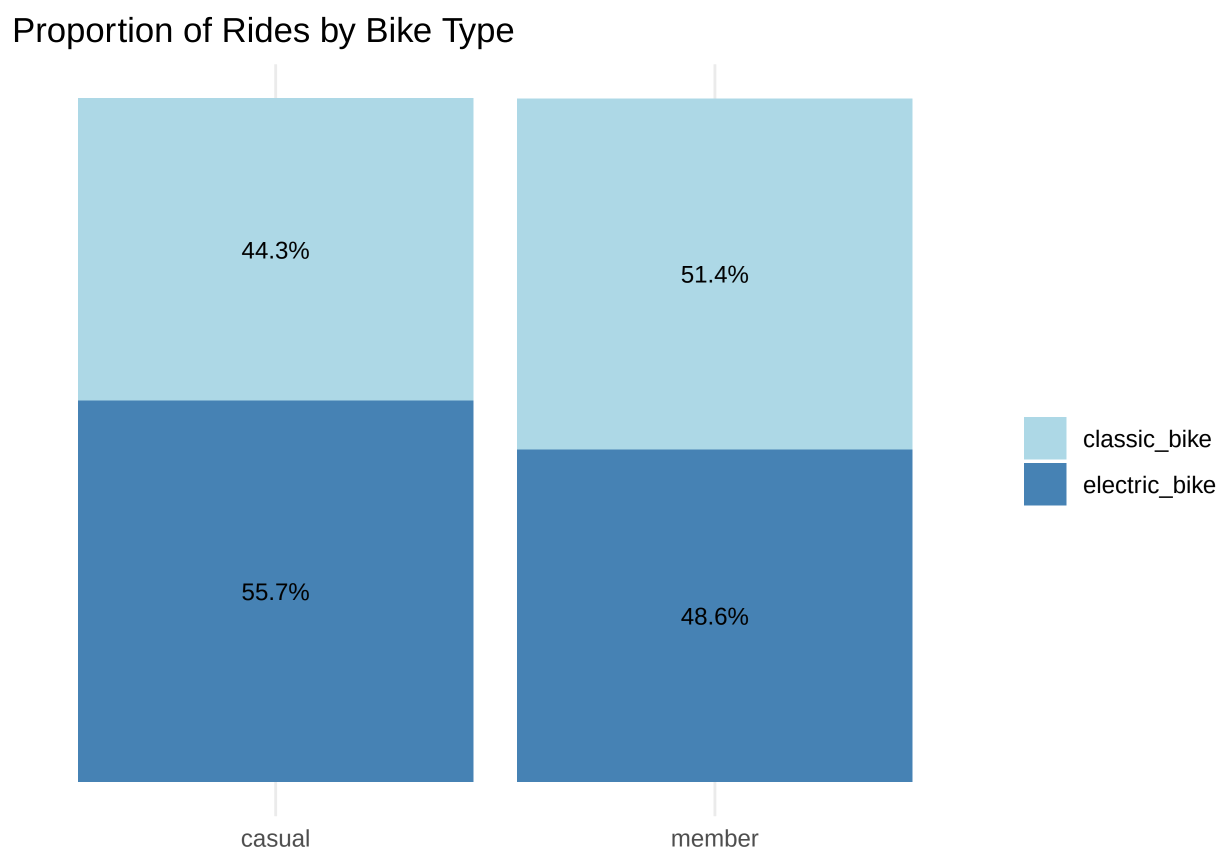
Members: *50%* of rides are shorter than 8m 55s; *90%*, 24m 12s.

As for the longer rides, casual riders expectedly have more of them than members, likely because of a Divvy's 45 minutes limit per ride in their annual subscription.

Bike Type Analysis

Now I want to look at the data in terms of the types of bikes used for rides:

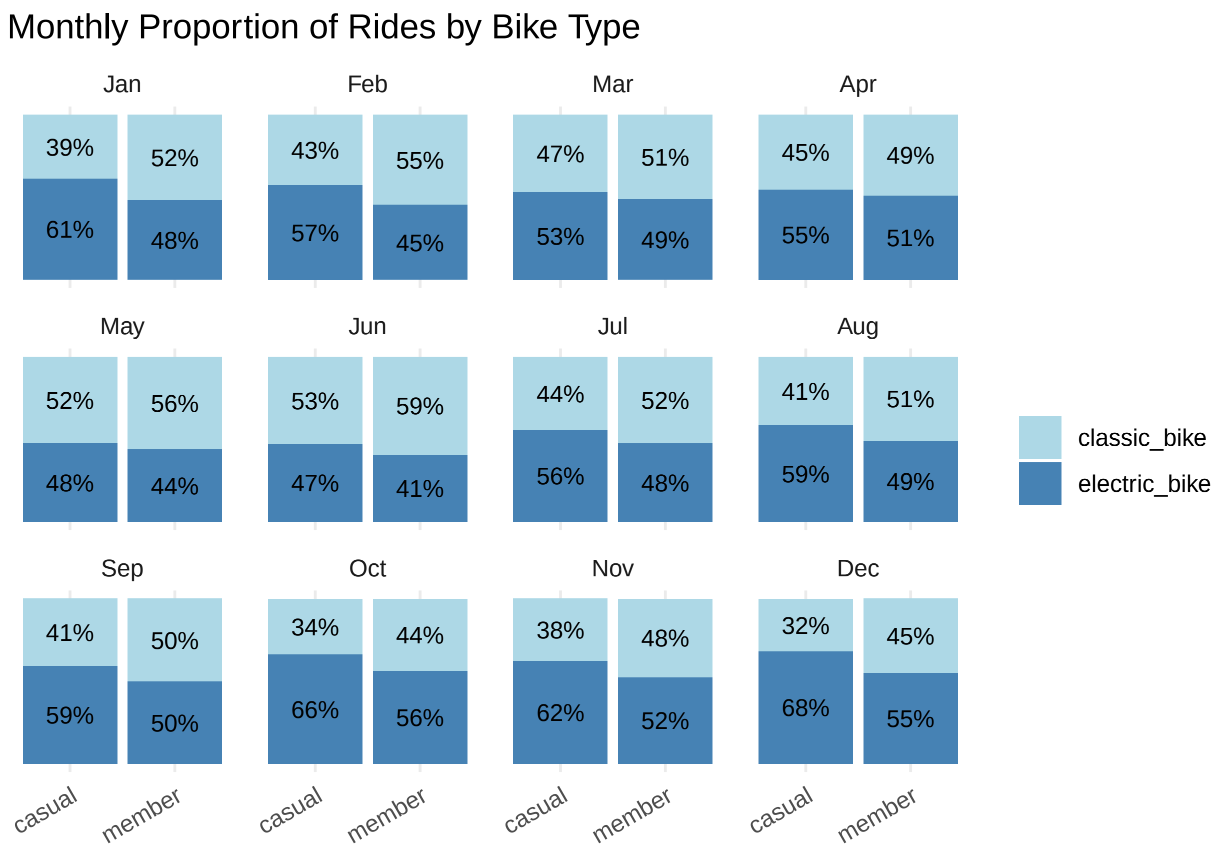
```r
bike_type_prop <- full_tripdata %>%
  group_by(member_casual, rideable_type) %>%
  summarise(count = n()) %>%
  mutate(proportion = count / sum(count))

ggplot(bike_type_prop, aes(x = member_casual, y = proportion, fill = rideable_type)) +
  theme_minimal() +
  geom_bar(stat = "identity") +
  geom_text(aes(label = percent(proportion)),
            position = position_stack(vjust = 0.5),
            size = 3) +
  scale_fill_manual(values = c("lightblue", "steelblue")) +
  scale_y_continuous(breaks = NULL) +
  labs(y = NULL,
       x = NULL,
       fill = "",
       title = "Proportion of Rides by Bike Type")
```



This plot shows that **casual** riders prefer electric bikes over classic, whereas **members** slightly prefer classic bikes over electic.

To look at this proportion in dynamics, I applied a monthly facet wrap:

```
bike_type_prop_month <- full_tripdata %>%
  group_by(member_casual, rideable_type, month) %>%
  summarise(count = n()) %>%
  group_by(month, member_casual) %>%
  mutate(sum_count = sum(count),
         proportion = count / sum_count)

ggplot(bike_type_prop_month, aes(x = member_casual, y = proportion, fill = rideable_type)) +
  theme_minimal() +
  geom_bar(stat = "identity", position = "fill") +
  geom_text(aes(label = percent(proportion, accuracy = 1)),
            position = position_stack(vjust = 0.5),
            size = 3) +
  scale_fill_manual(values = c("lightblue", "steelblue")) +
  scale_y_continuous(breaks = NULL, labels = scales::percent) +
  labs(y = NULL,
       x = NULL,
       fill = "",
       title = "Monthly Proportion of Rides by Bike Type") +
  facet_wrap(~ month,
             labeller = labeller(month = c("01" = "Jan", "02" = "Feb", "03" = "Mar",
                                           "04" = "Apr", "05" = "May", "06" = "Jun",
                                           "07" = "Jul", "08" = "Aug", "09" = "Sep",
                                           "10" = "Oct", "11" = "Nov", "12" = "Dec"))) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



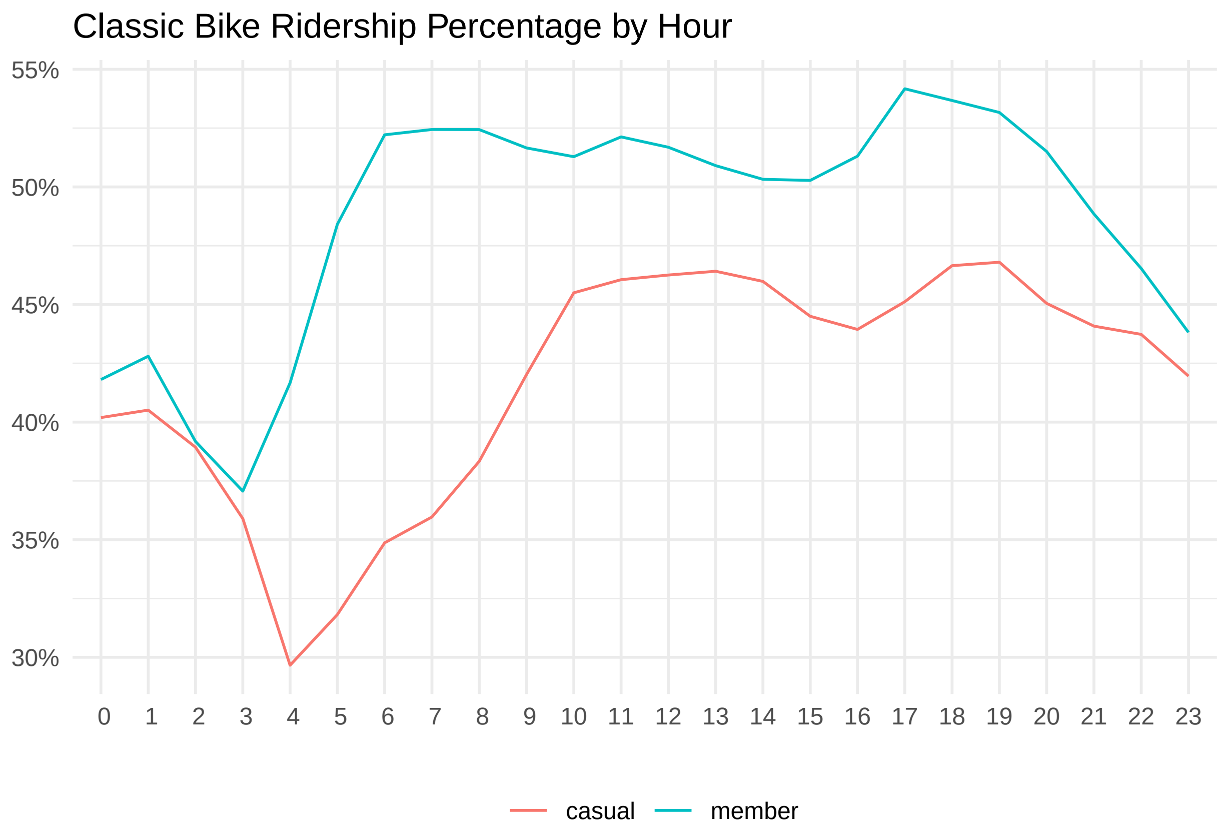
Monthly analysis shows that in a **casual** group classic bike popularity is constantly lower than that of electric bikes, except for May and June. Maybe this is due to people trying to lose weight for summer and then giving this idea up; maybe not, there's not enough data to be certain.

In **members** group classic bikes are constantly equally or slightly more popular except for October, November, and December. Intuitively, this is due to the weather situation.

Another interesting story to observe is how bike type popularity is distributed across hours. Here's a plot of classic bike ridership percentage across hours:

(A percentage value on Y-axis indicates that out of all rides at this hour in this member group, x% of rides were on classic bikes)

```
ratio_rideable_hour <- full_tripdata %>%
  group_by(hour, member_casual, rideable_type) %>%
  summarise(ratio_rideable_hour = n()) %>%
  arrange(hour) %>%
  group_by(member_casual, hour) %>%
  mutate(total = sum(ratio_rideable_hour),
         prop = ratio_rideable_hour / total * 100) %>%
  filter(rideable_type == "classic_bike")

ggplot(ratio_rideable_hour, aes(x = hour, y = prop, color = member_casual)) +
  geom_line() +
  scale_x_discrete(limits = ratio_rideable_hour$hour) +
  theme_minimal() +
  scale_y_continuous(labels = function(x) paste0(x, "%")) +
  labs(title = "Classic Bike Ridership Percentage by Hour",
       color = "Customer Type",
       x = element_blank(),
       y = element_blank()) +
  theme(legend.title = element_blank(),
        legend.position = "bottom")
```



One interesting observation is that the number of rides starts growing at around 4AM, and so does classic bike popularity, and that is true for both customer groups. However, I can't think of any decent insight based on this observation. *(Maybe riders are tired at night and tend to use electric bikes, or maybe at night they ride from stations where the availability of electric bikes is higher. I'm leaving this question unanswered because it's out of the scope of the current analysis)*

In general, it's evident that **classic bikes are more popular among members than casuals**, which is one more insight. But whether it is due to pricing, availability, or fitness aspect is due to further analysis, which is out of the scope of the data.

Part 3: Spatial Analysis

The best way to present the data for spatial analysis is to plot it on an actual map, which I will do.

But first, let's have a look at the top 10 start and end stations for both customer groups. *(Rides with missing station IDs and names are filtered out)*

Top 10 start stations:

```
full_tripdata %>%
  filter(!is.na(start_station_name), !is.na(start_station_id)) %>%
  group_by(start_station_id, start_station_name, member_casual) %>%
  summarise(rides = n()) %>%
  group_by(start_station_id, start_station_name) %>%
  mutate(total_rides_station = sum(rides)) %>%
  group_by(start_station_id, start_station_name) %>%
  mutate(percentage_of_total_rides_station = round(rides / sum(rides) * 100, 2)) %>%
  group_by(member_casual) %>%
  mutate(percentage_of_total_rides_by_member_casual = round(rides / sum(rides) * 100, 2)) %>%
  arrange(member_casual, desc(rides)) %>%
  group_by(member_casual) %>%
  slice(1:10)
```

```
## # A tibble: 20 x 7
## # Groups:   member_casual [2]
##    start_station_id start_station_name   member_casual rides total_rides_station
##    <chr>            <chr>                <chr>         <int>               <int>
##  1 13022            Streeter Dr & Grand~ casual        47888               64187
##  2 13300            DuSable Lake Shore ~ casual        26279               35196
##  3 LF-005           DuSable Lake Shore ~ casual        20872               36708
##  4 13042            Michigan Ave & Oak ~ casual        20703               34608
##  5 13008            Millennium Park      casual        20035               29161
##  6 15544            Shedd Aquarium       casual        18127               22836
##  7 TA1308000001     Theater on the Lake  casual        16190               30248
##  8 TA1308000050     Wells St & Concord ~ casual        15482               36588
##  9 13146            Clark St & Armitage~ casual        12713               27797
## 10 TA1307000039     Clark St & Elm St    casual        12422               34314
## 11 KA1503000043     Kingsbury St & Kinz~ member        24685               33184
## 12 TA1307000039     Clark St & Elm St    member        21892               34314
## 13 TA1308000050     Wells St & Concord ~ member        21106               36588
## 14 KA1503000071     University Ave & 57~ member        20539               25704
## 15 WL-012           Clinton St & Washin~ member        20044               25682
## 16 KA1503000014     Ellis Ave & 60th St  member        20028               24661
## 17 13332            Loomis St & Lexingt~ member        18985               21940
## 18 KA1504000135     Wells St & Elm St    member        18939               30913
## 19 TA1305000032     Clinton St & Madiso~ member        18706               25832
## 20 13137            Broadway & Barry Ave member        17626               29026
## # i 2 more variables: percentage_of_total_rides_station <dbl>,
## #   percentage_of_total_rides_by_member_casual <dbl>
```

Top 10 end stations:

```
full_tripdata %>%
  filter(!is.na(end_station_name), !is.na(end_station_id)) %>%
  group_by(end_station_id, end_station_name, member_casual) %>%
  summarise(rides = n()) %>%
  group_by(end_station_id, end_station_name) %>%
  mutate(total_rides_station = sum(rides)) %>%
  group_by(end_station_id, end_station_name) %>%
  mutate(percentage_of_total_rides_station = round(rides / sum(rides) * 100, 2)) %>%
  group_by(member_casual) %>%
  mutate(percentage_of_total_rides_by_member_casual = round(rides / sum(rides) * 100, 2)) %>%
  arrange(member_casual, desc(rides)) %>%
  group_by(member_casual) %>%
  slice(1:10)
```

```
## # A tibble: 20 x 7
## # Groups:   member_casual [2]
##    end_station_id end_station_name       member_casual rides total_rides_station
##    <chr>          <chr>                  <chr>         <int>               <int>
##  1 13022          Streeter Dr & Grand A~ casual        48790               63457
##  2 13300          DuSable Lake Shore Dr~ casual        23686               33665
##  3 LF-005         DuSable Lake Shore Dr~ casual        23090               38478
##  4 13042          Michigan Ave & Oak St  casual        21492               34691
##  5 13008          Millennium Park        casual        21378               29657
##  6 TA1308000001   Theater on the Lake    casual        16630               29735
##  7 15544          Shedd Aquarium         casual        16608               22004
##  8 TA1308000050   Wells St & Concord Ln  casual        14882               36660
##  9 13146          Clark St & Armitage A~ casual        12760               27785
## 10 13179          Clark St & Lincoln Ave casual        12558               25890
## 11 KA1503000043   Kingsbury St & Kinzie~ member        24586               32113
## 12 TA1307000039   Clark St & Elm St      member        22307               33806
## 13 TA1308000050   Wells St & Concord Ln  member        21778               36660
## 14 KA1503000071   University Ave & 57th~ member        21113               26381
## 15 WL-012         Clinton St & Washingt~ member        20869               25688
## 16 KA1503000014   Ellis Ave & 60th St    member        19914               24424
## 17 TA1305000032   Clinton St & Madison ~ member        19536               26240
## 18 13332          Loomis St & Lexington~ member        18866               21601
## 19 KA1504000135   Wells St & Elm St      member        18722               29984
## 20 13137          Broadway & Barry Ave   member        17988               29416
## # i 2 more variables: percentage_of_total_rides_station <dbl>,
## #   percentage_of_total_rides_by_member_casual <dbl>
```

Top start stations are simultaneously top end stations for both groups, but there are strong differences in the top stations for different customer groups. Notably, the most popular station for **casual** riders accounts for more than 2.5% of all casual rides. This station is most likely located at a popular place for leisure.

Now, maps. Since Google Maps require API key to get a map, and Leaflet doesn't have any decent options to plot routes, I use Stamen maps.

Fetching a Map

I fetch a Chicago map by manually entering coordinates for a boundary box. (*I do it manually to get a better bbox than a default one, and for obvious reasons I spare the boring process of getting those coordinates*)

```
chicago_map <- get_stamenmap(bbox = c(-87.75, 41.78, -87.55, 41.97),
                             source = "stamen",
                             maptype = "toner-lite",
                             zoom = 12)
```

```
## i Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.
```

Heatmaps

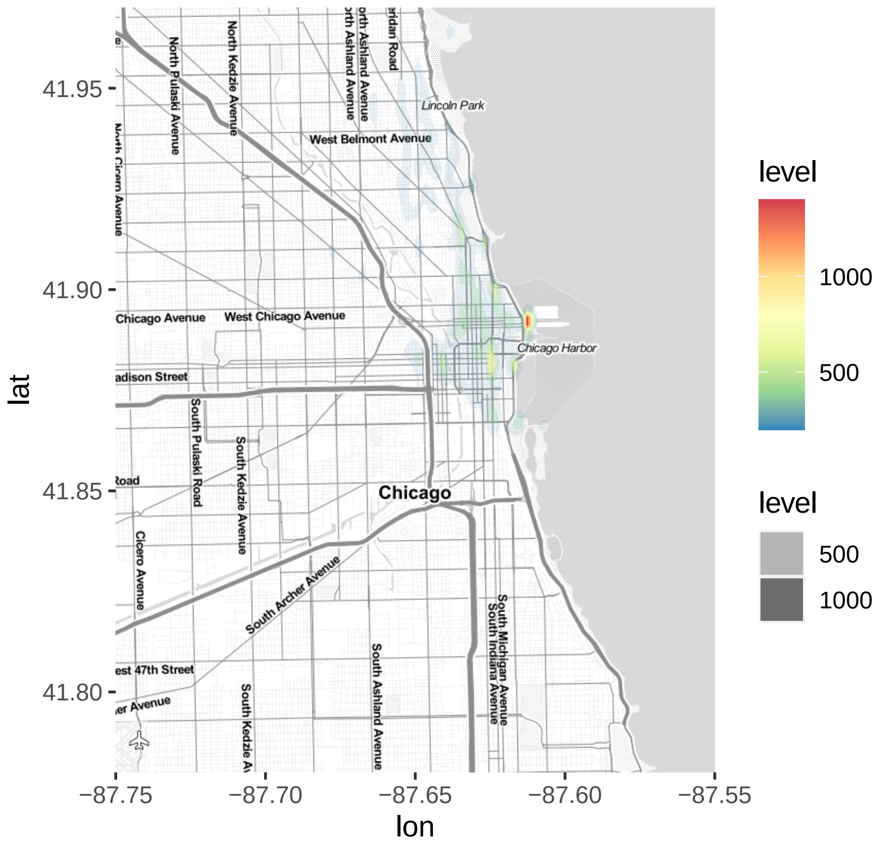
I created separate tibbles for casual and member heatmaps and filtered out rows with NA values in station names and IDs because (as I explained in Part 1) such rows contain broken coordinates.

```r
# Casual rides heatmap tibble
full_tripdata_casual <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "casual"
  )
```

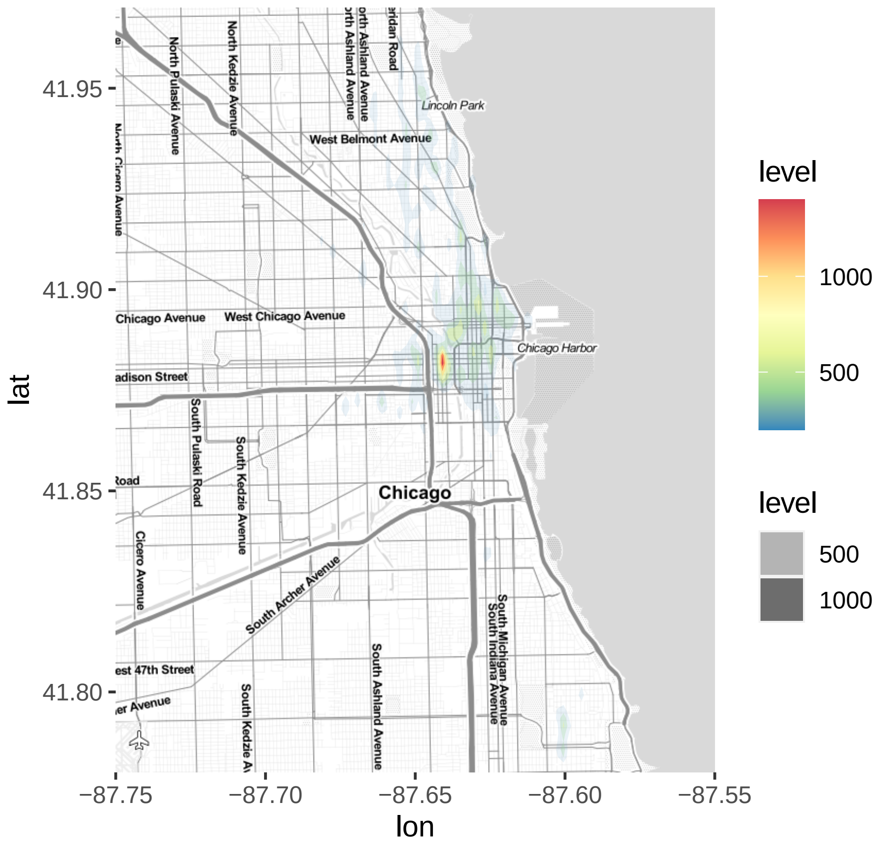
```r
# Member rides heatmap tibble
full_tripdata_member <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "member"
  )
```

Plotting heatmaps:

```r
# casual rides heatmap
ggmap(chicago_map) +
  stat_density2d(data=full_tripdata_casual,
                 aes(x=start_lng, y=start_lat, fill=..level.., alpha= ..level..),
                 geom="polygon") +
  scale_fill_gradientn(colours=rev(brewer.pal(7, "Spectral")))
```



```r
# Member rides heatmap
ggmap(chicago_map) +
  stat_density2d(data=full_tripdata_member,
                 aes(x=start_lng, y=start_lat, fill=..level.., alpha= ..level..),
                 geom="polygon") +
  scale_fill_gradientn(colours=rev(brewer.pal(7, "Spectral")))
```



The difference is explicit: **casual** rides start mostly at the harbor area, whereas **member** rides start at busy city areas.

However, heatmaps aren't too informative when researching routes. From heatmaps we know that members and casuals start their rides at different stations, but where do they ride?

To explore this, I'm plotting routes.

Routes

I introduce two new tibbles:

```
# Top 500 Casual Routes
routes_casual <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "casual") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 500) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

```
# Top 500 Member Routes
routes_member <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "member") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 500) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

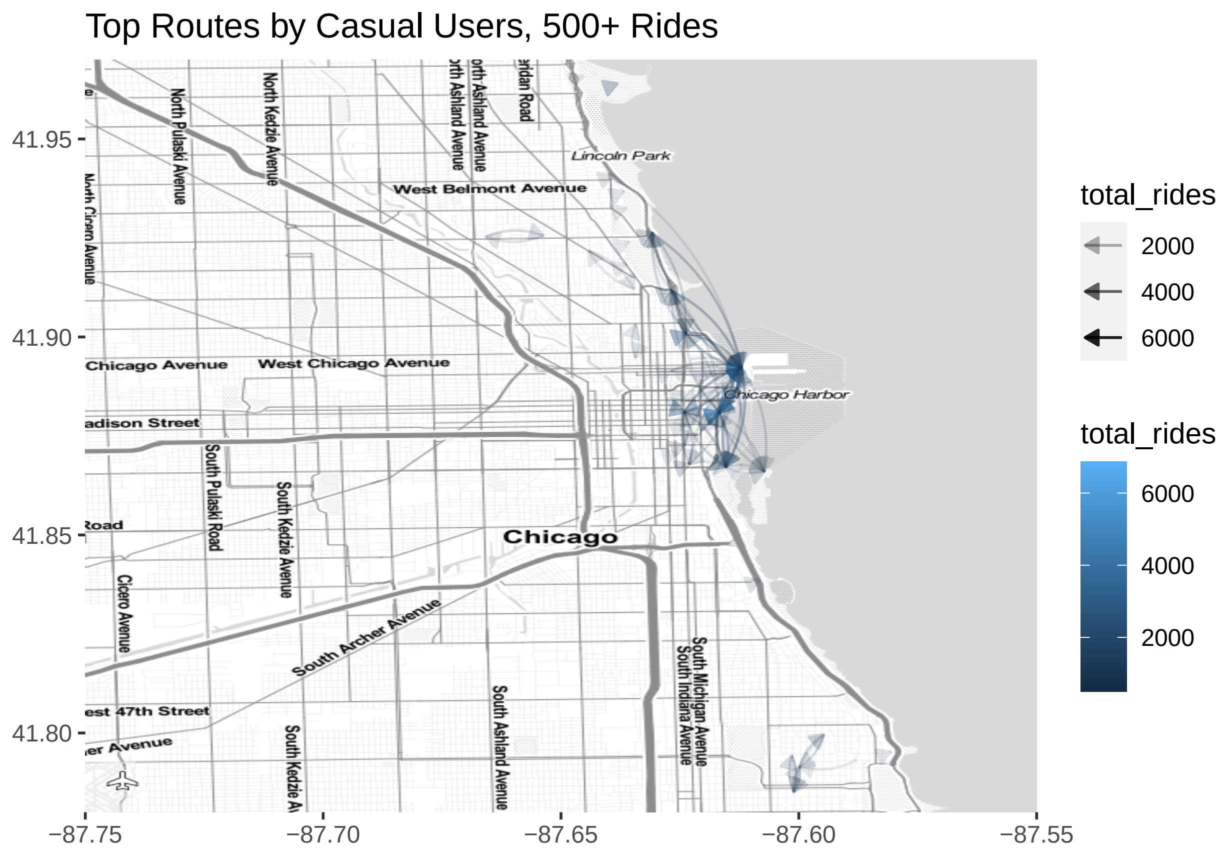
Some explanation of what I did:

- I filtered out NA values in station names and IDs because such rows have only double decimal places precision for lat and lng, which can heavily skew the data. Alternatively, I could filter out rows with coordinates that have less than 5 decimal places precision, the result would be the same.
- I grouped data by station IDs + station names. Grouping by IDs or names alone would be wrong because, as I mentioned above, some stations have the same ID but different names, and are, in fact, different although closely located stations. In simpler words, for the *stations* table the primary key would be a combination of ID and NAME.
- Original data has coordinates for rides but not for bike stations, and despite rides start and end only at bike stations, coordinates for rides to (or from) the same station differ. Therefore, I assign the median coordinates of all the rides for a station as the coordinates of this station.
- Filtered out routes with less than 500 rides.
- Geom_curve function doesn't allow start and end coordinates to be the same, so I would need to exclude round trips in order to make the viz work. I don't want to do it, so I incremented start coordinates for round trips by 0.00001, which won't affect the plot.

Finally, plotting.

Top 500 Casual Routes:

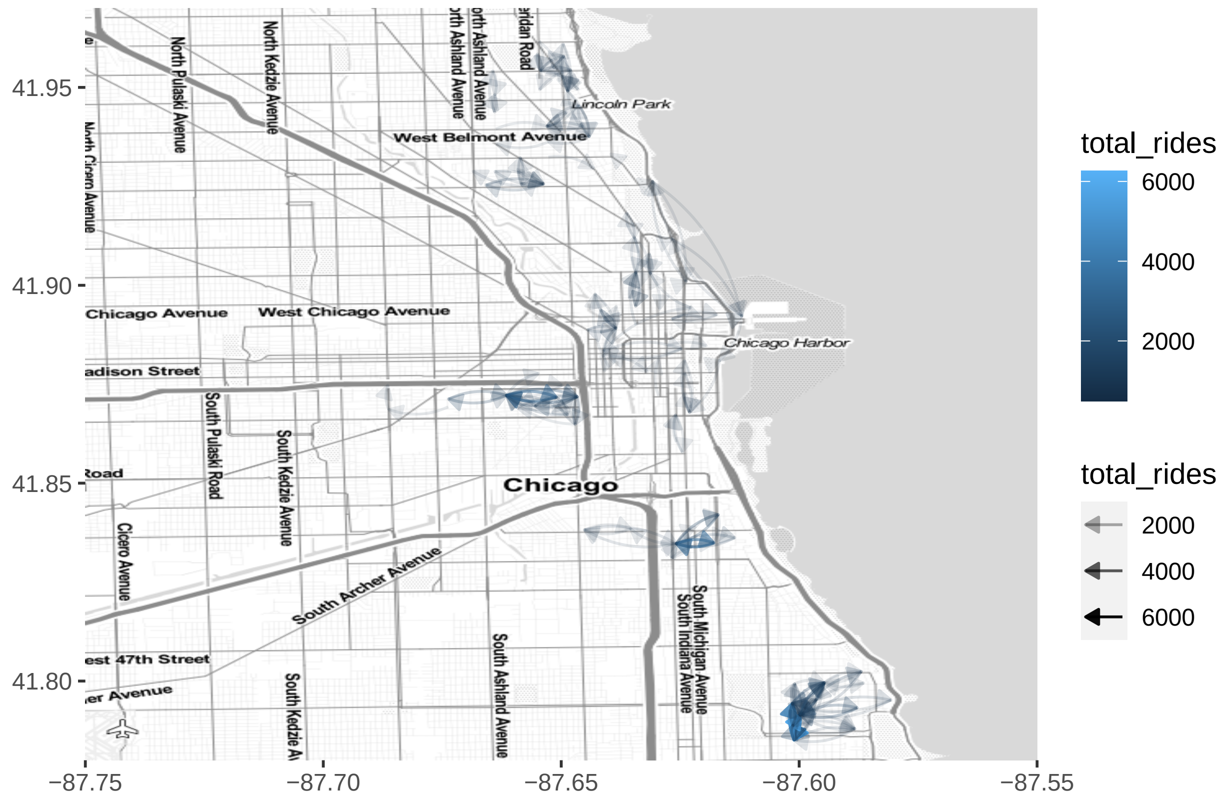
```
ggmap(chicago_map) +
  geom_curve(routes_casual,
             mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                           alpha= total_rides, color = total_rides),
             size = 0.5, curvature = .2,
             arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
  coord_cartesian() +
  labs(title = "Top Routes by Casual Users, 500+ Rides",x=NULL,y=NULL) +
  theme(legend.position="right")
```



Top 500 Member Routes:

```
ggmap(chicago_map) +
  geom_curve(routes_member,
             mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                           alpha= total_rides, color = total_rides),
             size = 0.5, curvature = .2,
             arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
  coord_cartesian() +
  labs(title = "Top Routes by Members, 500+ Rides",x=NULL,y=NULL) +
  theme(legend.position="right")
```

Top Routes by Members, 500+ Rides



There's not a single doubt that **casual** rides are concentrated around places for leisure, namely Chicago Harbor, and **member** rides are concentrated around the busy city areas.

Let's break the data down by the bike type. To make the data easier to follow, I create separate plots for members and for casuals. Since plots are also separate for classic and electric bikes, I lower the cap for a route to be considered popular to 250 rides.

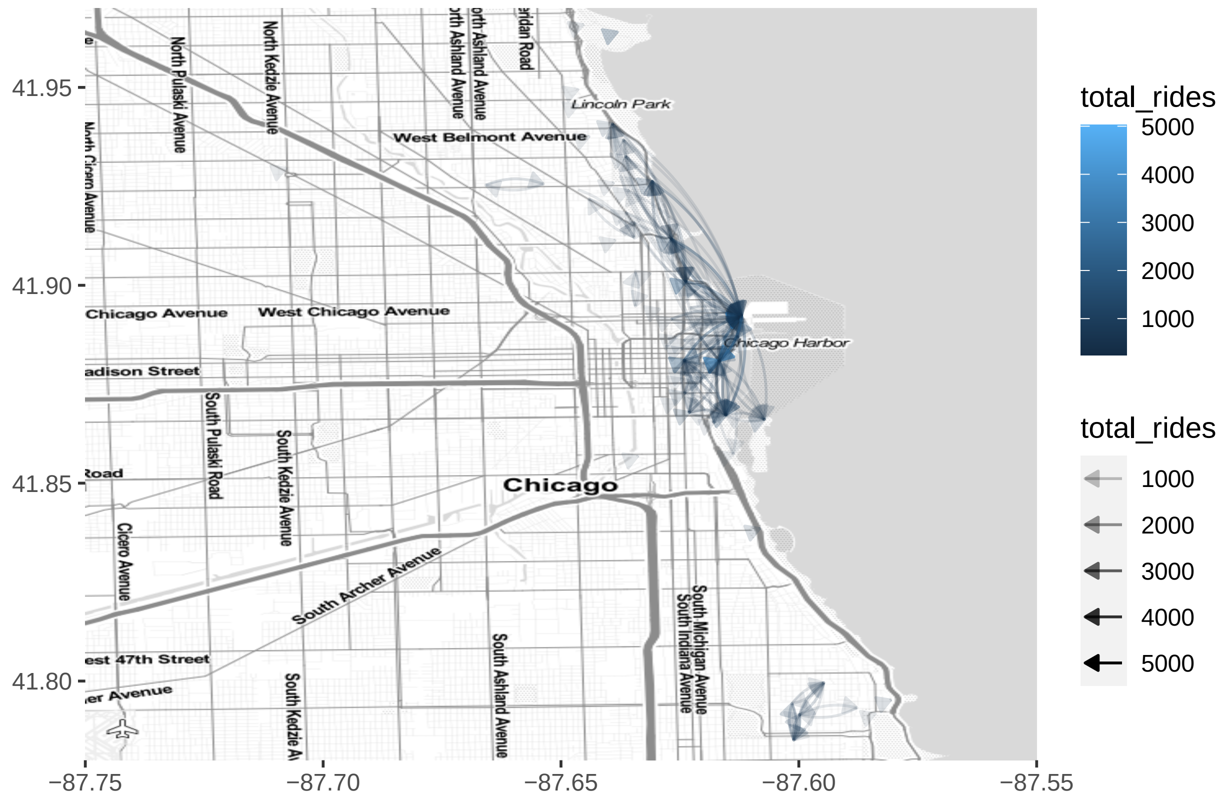
Casuals:

```r
# Casual Top Routes Classic Bike (250+ rides)
routes_casual_classic <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "casual",
         rideable_type == "classic_bike") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 250) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

```r
# Casual Top Routes Electric Bike (250+ rides)
routes_casual_electric <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "casual",
         rideable_type == "electric_bike") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 250) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

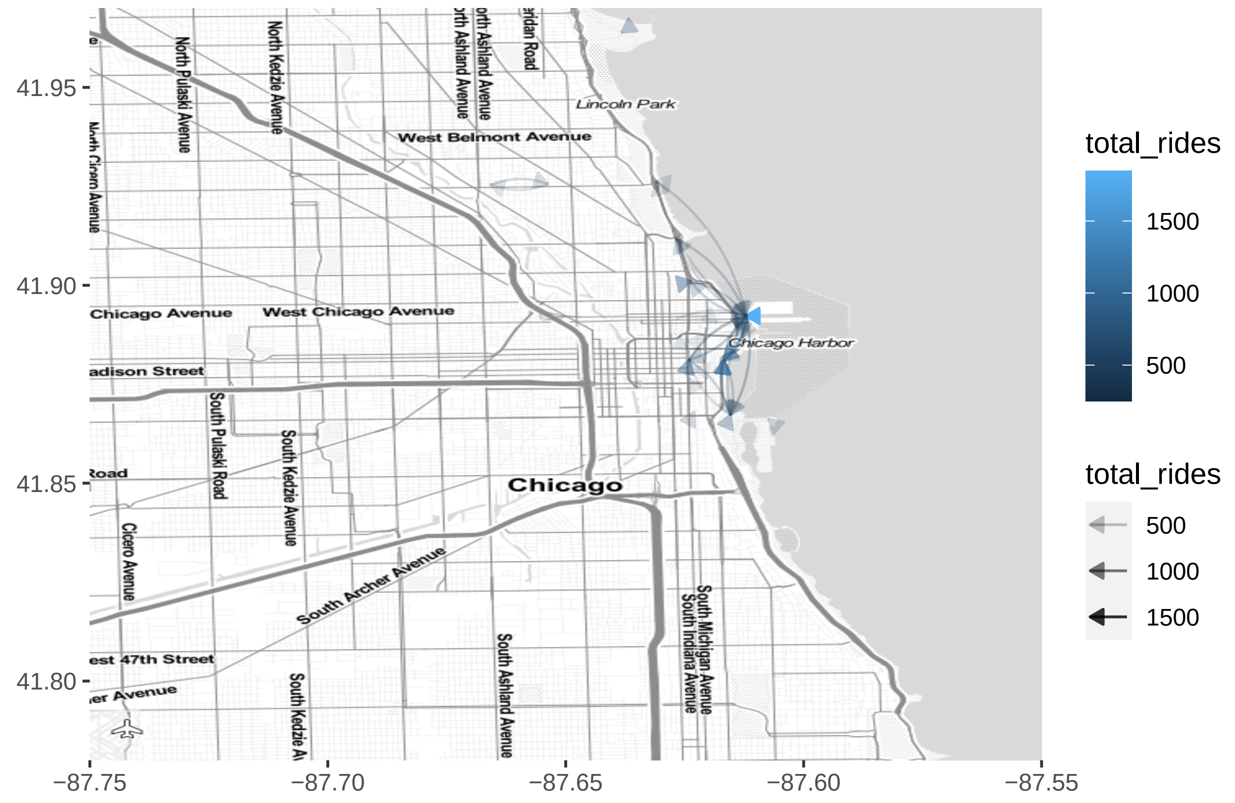
```r
ggmap(chicago_map) +
  geom_curve(routes_casual_classic,
             mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                           alpha= total_rides, color = total_rides),
             size = 0.5, curvature = .2,
             arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
  coord_cartesian() +
  labs(title = "Top Routes by Casuals, Classic Bike, 250+ Rides",x=NULL,y=NULL) +
  theme(legend.position="right")
```

Top Routes by Casuals, Classic Bike, 250+ Rides



```
ggmap(chicago_map) +
  geom_curve(routes_casual_electric,
             mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                           alpha= total_rides, color = total_rides),
             size = 0.5, curvature = .2,
             arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
  coord_cartesian() +
  labs(title = "Top Routes by Casuals, Electric Bike, 250+ Rides",x=NULL,y=NULL) +
  theme(legend.position="right")
```

Top Routes by Casuals, Electric Bike, 250+ Rides



Interesting observation: casual riders on classic bikes take more one-way trips, whereas on electric bikes they take more round trips (light blue arrow without a tail at the harbor).

Members:

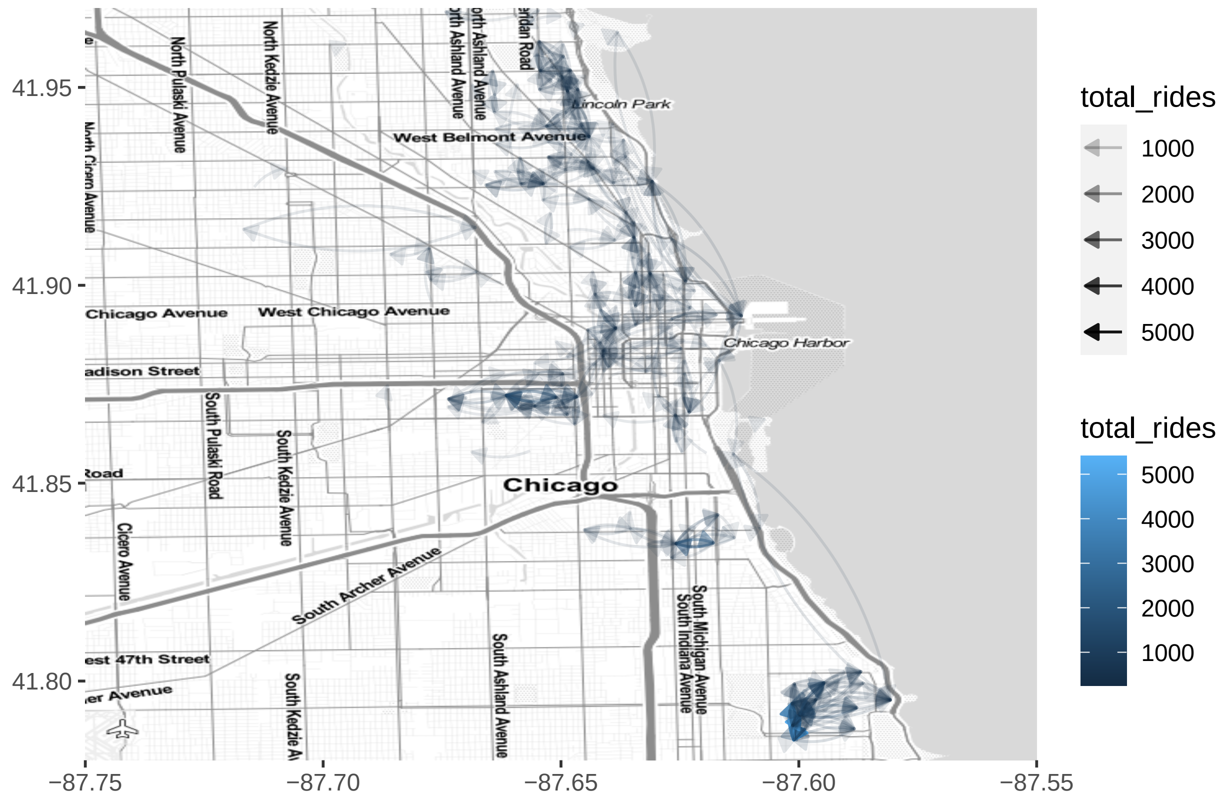
```
# Member Top Routes Classic Bike (250+ rides)
routes_member_classic <-  full_tripdata %>%
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "member",
         rideable_type == "classic_bike") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 250) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

```
# Member Top Routes Electric Bike (250+ rides)
routes_member_electric <-  full_tripdata %>%
```

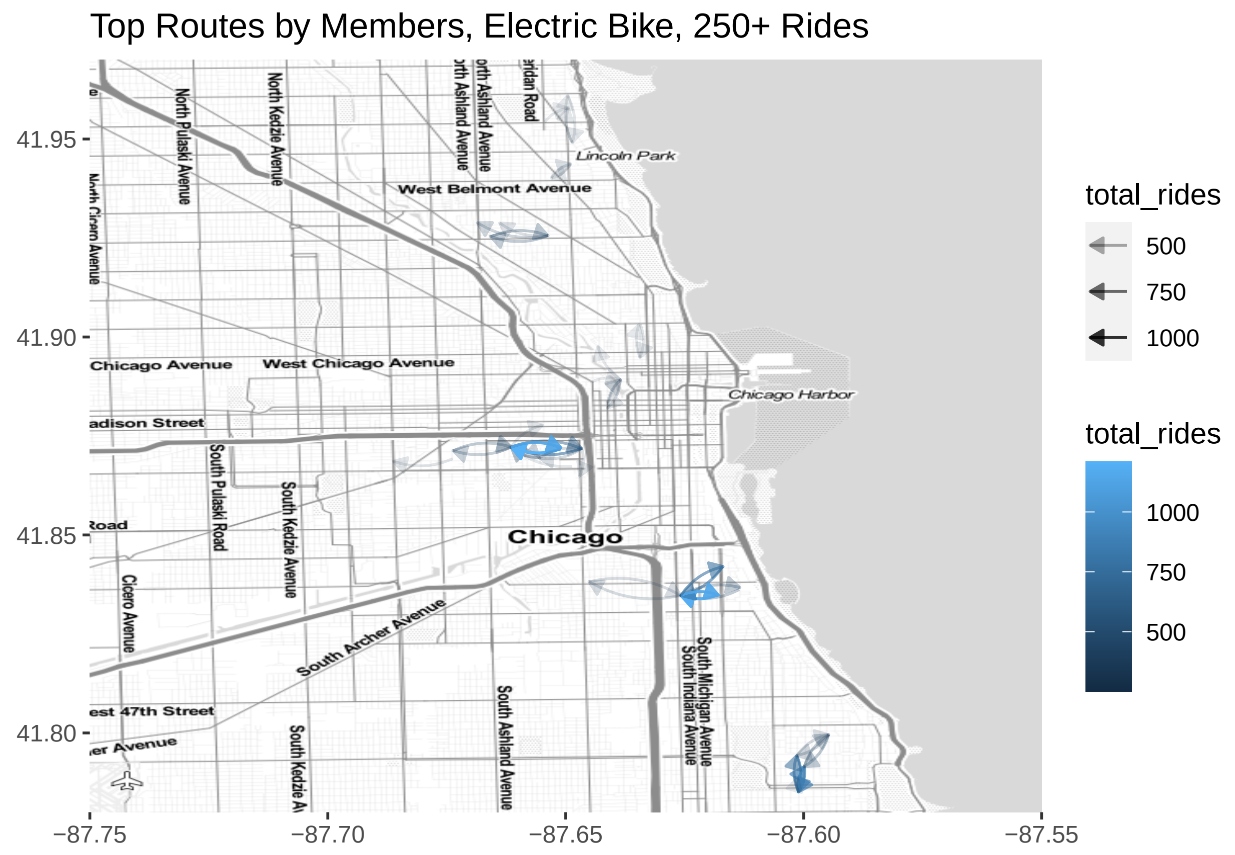
```r
  filter(!is.na(start_station_name),
         !is.na(start_station_id),
         !is.na(end_station_name),
         !is.na(end_station_id),
         member_casual == "member",
         rideable_type == "electric_bike") %>%
  group_by(start_station_id, start_station_name, end_station_id, end_station_name) %>%
  summarise( start_lat = median(start_lat),
             start_lng = median(start_lng),
             end_lat = median(end_lat),
             end_lng = median(end_lng),
             total_rides = n()
  ) %>%
  filter(total_rides > 250) %>%
  mutate(start_lat = if_else(start_lat == end_lat, start_lat + 0.00001, start_lat)) %>%
  mutate(start_lng = if_else(start_lng == end_lng, start_lng + 0.00001, start_lng))
```

```r
ggmap(chicago_map) +
  geom_curve(routes_member_classic,
             mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                           alpha= total_rides, color = total_rides),
             size = 0.5, curvature = .2,
             arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
  coord_cartesian() +
  labs(title = "Top Routes by Members, Classic Bike, 250+ Rides",x=NULL,y=NULL) +
  theme(legend.position="right")
```

Top Routes by Members, Classic Bike, 250+ Rides



```
ggmap(chicago_map) +
 geom_curve(routes_member_electric,
           mapping = aes(x = start_lng, y = start_lat, xend = end_lng, yend = end_lat,
                         alpha= total_rides, color = total_rides),
           size = 0.5, curvature = .2,
           arrow = arrow(length=unit(0.2,"cm"), ends="first", type = "closed")) +
 coord_cartesian() +
 labs(title = "Top Routes by Members, Electric Bike, 250+ Rides",x=NULL,y=NULL) +
 theme(legend.position="right")
```



Looks like the Harbor area is not popular in members on electric bikes. Also, members mostly use electric bikes to commute short distances at a small number of busy city areas, whereas they use classic bikes to commute all over Chicago, including longer rides from distant areas.

Summary

There are evident differences in how casual riders and annual members use bikes differently:

1. Casual customers take longer rides.

- **Casuals**: *Mean ride* is *15.7* minutes. *50%* of rides are shorter than *12m 15s*; 90%, *32m 45s*.
- **Members**: *Mean ride* is *11.7* minutes. *50%* of rides are shorter than *8m 55s*; 90%, *24m 12s*.

2. **Casual** ridership increases during weekends, **member** ridership decreases.
3. Casual customers make more round trips. Percentage of round trips: 5.2% for **casuals**, 2% for **members**.
4. Both groups seem to use bikes to commute *from* work, but **casual** riders don't seem to actively use bikes to commute *to* work.
5. **Casual** customers prefer electric bikes, **members** slightly prefer classic bikes.

- **Casuals**: *55.7%* of rides were on electric bikes; classic — *44.3%*.

- **Members**: *48.6%* of rides were on electric bikes; classic — *51.4%*.

6. Most popular routes for **casual** users start near harbor area, and also end at (or close to) harbor area. The second most popular group of routes follows along the shore towards Lincoln Park. Only a small fraction of rides are in busy city areas. Conversely, most popular routes for **members** are in busy city areas, with only a minuscule fraction or rides leading to the harbor.

It's safe to say that casual customers prefer weekends and ride mostly for leisure, whereas members mostly use bikes to commute. Member ridership is more consistent throughout the year, whereas casual ridership has a sharp increase during May to October.

Recommendations

1. **Create additional benefits for hesitant casual customers**. A hesitant casual customer might be one whose ridership pattern resembles that of a member. *Top Routes by Casuals, Classic Bike, 250+ Rides* plot shows there's such a group of casual customers. Think of ways to target them.
2. **Lure casual customers into trying bikes as means of day-to-day transportation.** Identify casual customers who use bikes quite regularly and create a marketing campaign to target them. Think of a powerful message for this group; e.g. connect riding a bike with health benefits, which results in the overall improved quality of life (stress reduction, weight loss, improved mood, saving money on medicine and so on).

Actually, such a campaign could be effective on a larger scale, but trying it on a small relevant group first would be more efficient in terms of using the budget.

3. **Focus on the leisure aspect**. This recommendation somewhat resembles the second one, but here the focus is not on trying to lure casual customers into commuting, but to nudge them into taking more leisure rides. The message should also focus on connecting riding a bike with improving the quality of life; however, the target group could differ.

Speaking of the target groups, for the second recommendation the target group could be those who use bikes for leisure AND sometimes use bikes to commute. For the third recommendation, the group could be those who use bikes for leisure BUT don't use them to commute. Of course, "sometimes" and "don't use" should be defined in numbers, it's not necessarily that "don't use" = 0 and "sometimes" is anything > 1. And, of course, there should be multiple groups for each recommendation for the A/B testing.