Cyclistic bike-share analysis

Aleksandr Popov

2023-09-20

How Does a Bike-Share Navigate Speedy Success?

Analysis data about casual riders vs. annual members.

Ask

Scenario:

Cyclistic is a bike-share company in Chicago. The director of marketing, Lily Moreno, believes the company's future success depends on maximizing the number of annual memberships. The marketing analyst team need to understand how casual riders (with single-ride and full-day passes) and annual members use Cyclistic bikes differently. From these insights, the marketing analyst team will design a new marketing strategy to convert casual riders into annual members.

The main aim of the project:

Design marketing strategies aimed at converting casual riders into annual members.

Three questions to answer:

- *How do annual members and casual riders use Cyclistic bikes differently?*
- Why would casual riders buy Cyclistic annual memberships?
- How can Cyclistic use digital media to influence casual riders to become members?

Stakeholders:

- Lily Moreno - the director of marketing

- Cyclistic executive team - this team will decide whether to approve the recommended marketing program.

In order to answer the key business questions, I'll follow the steps of the data analysis process:

- Ask
- Prepare
- Process
- Analyze
- Share
- Act

Deliverable:
[✓] _A clear statement of the business task_

Prepare

The data has been made available by Motivate International Inc. (under the licence) and it is located here: link (https://divvy-tripdata.s3.amazonaws.com/index.html). So according to the licence we can say that the data is reliable and original. We have some restrictions on working with the data: data-privacy issues prohibit from using riders' personally identifiable information.

The data is organized by periods, that included in its file names. Data source contains data from 2013 till July 2023:

- 2013 year: data for the whole year is in one file;
- 2014-2017 years: one file contains two quarters;
- 2018-1 quarter 2020: one file contains a quarter;
- April 2020-July 2023: one file contains a month.

In my case study I'll explore only data for entire 2022 year. It'll make our analysis more comprehensive, because we'll cover all the year and all seasons. Also I chosen 2022 year because it's the last full year we have, so we can say that our data is current as well.

Let's download data for every month of 2022 year: we'll get 12 zip files for every month. Create folder for our project, subfolder for ZIP files and put downloaded files into it. After extracting all files we'll get 12 csv files. Create one more subfolder for CSV files and save them into it.

So we have 12 csv files (1GB in total). Let's use RStudio to read these files and look at its content.

At first, load libraries that we will use in our project (here we're loading all libraries that we need for further steps).

```
#Load Libraries
library("tidyverse")
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.3     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become e
rrors
```

```
library("ggplot2")
library("lubridate")
library("maps")
```

```
##
## Присоединяю пакет: 'maps'
##
## Следующий объект скрыт от 'package:purrr':
##
##     map
```

```
library("scales")
```

```
##
## Присоединяю пакет: 'scales'
##
## Следующий объект скрыт от 'package:purrr':
##
##     discard
##
## Следующий объект скрыт от 'package:readr':
##
##     col_factor
```

```
library("ggmap")
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
## ℹ Google's Terms of Service: <https://mapsplatform.google.com>
## ℹ Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```
library("tinytex")
```

Show all files in our CSV directory:

```
#find all files in directory with datasets and put filenames into a list
datasets_dir <- paste(getwd(), "/Datasets/csv/2022", sep = "")
files_list <- dir(datasets_dir)
files_list
```

```
##  [1] "202201-divvy-tripdata.csv"       "202202-divvy-tripdata.csv"
##  [3] "202203-divvy-tripdata.csv"       "202204-divvy-tripdata.csv"
##  [5] "202205-divvy-tripdata.csv"       "202206-divvy-tripdata.csv"
##  [7] "202207-divvy-tripdata.csv"       "202208-divvy-tripdata.csv"
##  [9] "202209-divvy-publictripdata.csv" "202210-divvy-tripdata.csv"
## [11] "202211-divvy-tripdata.csv"       "202212-divvy-tripdata.csv"
```

We can open these files in spreadsheets (Excel or Google Sheets) to make a description of the data, but the files are too big to use spreadsheets, so we will use R.

Let's read the files using `read_csv` function and union all files into one dataframe. Check if every file has the same columns, using `colnames` function.

Now we have one dataset for all 2022 year, it'll help us to analyze data faster and more efficient. Let's take a look at the dataset. Show column names and datatypes with `glimpse` function.

```
#show columns and formats
glimpse(tripdata_2022)
```

```
## Rows: 5,667,717
## Columns: 13
## $ ride_id            <chr> "C2F7DD78E82EC875", "A6CF8980A652D272", "BD0F91DFF7…
## $ rideable_type      <chr> "electric_bike", "electric_bike", "classic_bike", "…
## $ started_at         <dttm> 2022-01-13 11:59:47, 2022-01-10 08:41:56, 2022-01-…
## $ ended_at           <dttm> 2022-01-13 12:02:44, 2022-01-10 08:46:17, 2022-01-…
## $ start_station_name <chr> "Glenwood Ave & Touhy Ave", "Glenwood Ave & Touhy A…
## $ start_station_id   <chr> "525", "525", "TA1306000016", "KA1504000151", "TA13…
## $ end_station_name   <chr> "Clark St & Touhy Ave", "Clark St & Touhy Ave", "Gr…
## $ end_station_id     <chr> "RP-007", "RP-007", "TA1307000001", "TA1309000021",…
## $ start_lat          <dbl> 42.01280, 42.01276, 41.92560, 41.98359, 41.87785, 4…
## $ start_lng          <dbl> -87.66591, -87.66597, -87.65371, -87.66915, -87.624…
## $ end_lat            <dbl> 42.01256, 42.01256, 41.92533, 41.96151, 41.88462, 4…
## $ end_lng            <dbl> -87.67437, -87.67437, -87.66580, -87.67139, -87.627…
## $ member_casual      <chr> "casual", "casual", "member", "casual", "member", "…
```

It has 13 columns and 5 667 717 rows. Here is description for every column:

1. ride_id (string) - identifier for a ride;
2. rideable_type (string) - bike type (electric_bike, classic_bike, docked_bike)
3. started_at (datetime) - trip start date and time
4. ended_at (datetime) - trip end date and time
5. start_station_name (string) - start trip station name
6. start_station_id (string) - start trip station ID
7. end_station_name (string) - end trip station name
8. end_station_id (string) - end trip station ID
9. start_lat (double) - start trip latitude
10. start_lng (double) - start trip longitude
11. end_lat (double) - end trip latitude
12. end_lng (double) - end trip longitude
13. member_casual (string) - type of membership (casual/member)

```
#show all types of bikes
tripdata_2022 %>%
  distinct(rideable_type)
```

```
## # A tibble: 3 × 1
##   rideable_type
##   <chr>
## 1 electric_bike
## 2 classic_bike
## 3 docked_bike
```

```
#show all kinds of member_casual column values
tripdata_2022 %>%
  distinct(member_casual)
```

```
## # A tibble: 2 × 1
##   member_casual
##   <chr>
## 1 casual
## 2 member
```

Let's sort our dataframe by `started_at` column and find the range of the dataset (min and max ride date):

```
#sort data by ride date
sorted_tripdata_2022 <- tripdata_2022 %>%
  arrange(started_at)
```

```
#show range for ride dates
cat("MIN date = ", format(date(min(sorted_tripdata_2022$started_at)), format = "%d.%m.%Y"), "; ",
"MAX date = ", format(date(max(sorted_tripdata_2022$started_at)), format = "%d.%m.%Y"))
```

```
## MIN date =  01.01.2022 ;  MAX date =  31.12.2022
```

Deliverable:
[✓] _A description of all data sources used_

Process

We have the large dataset (more than 5 mio rows in total) and it's better to use programming languages (R, Python) or SQL for data cleaning and analysis. We'll continue using R for these purposes.

Let's add a new column with trip duration (numeric) and separate date and time for ride beginning (column `started_at`). We'll use `mutate` function and create a new dataframe not to spoil initial dataframe. Convert trip duration into minutes:

```
#calculate trip duration, trip date and trip time
tripdata_2022_clean <- sorted_tripdata_2022 %>%
  mutate(trip_duration = as.numeric(ended_at - started_at)/60) %>%
  mutate(trip_date = date(started_at)) %>%
  mutate(trip_start_time = hms::as_hms(started_at))
```

Then add a new column for weekday (string):

```
#calculate weekdays for trip date
tripdata_2022_clean <- tripdata_2022_clean %>%
  mutate(trip_weekday = wday(trip_date, label = TRUE, locale = "eb_EN.UTF-8"))
```

Now we can delete unused columns:

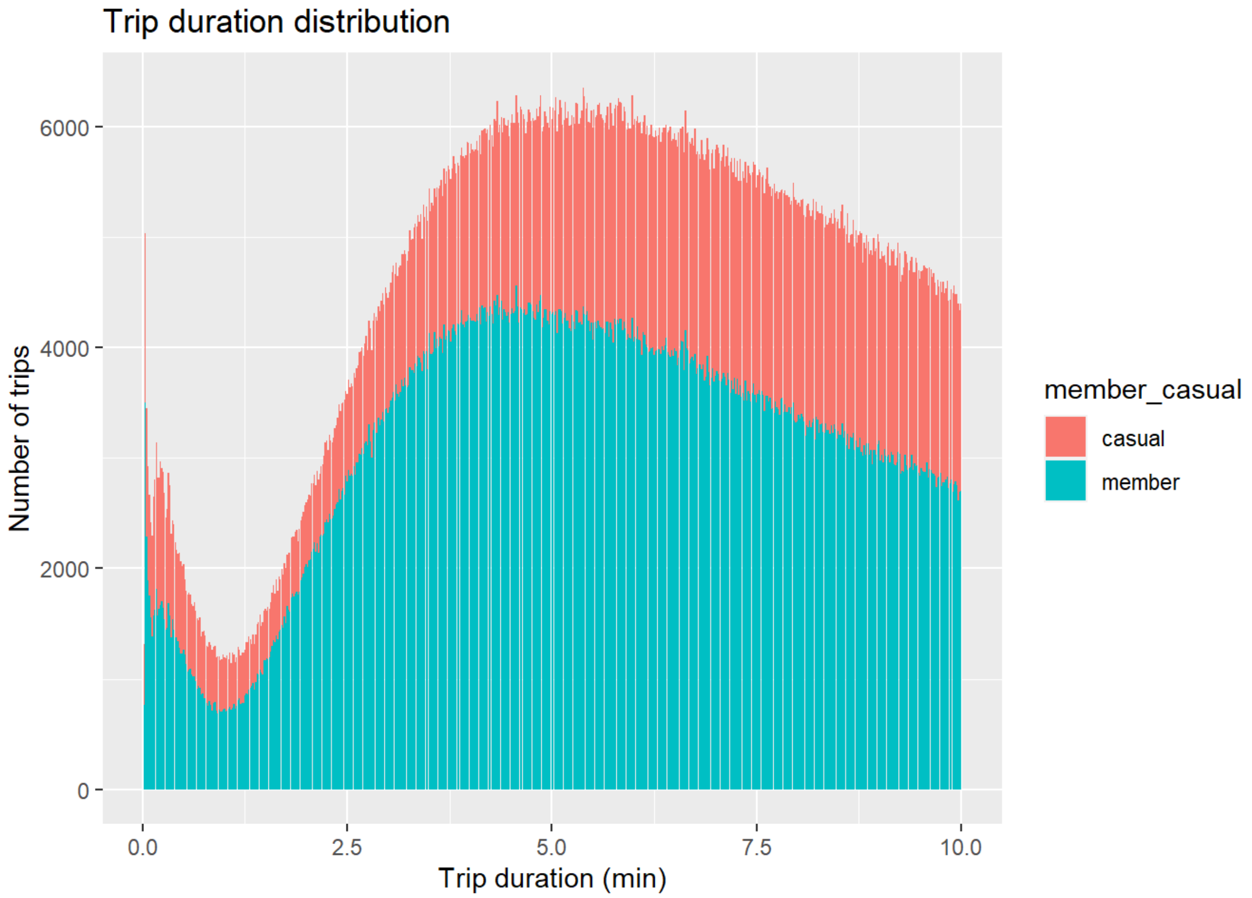
```
#delete unused columns
tripdata_2022_clean <- tripdata_2022_clean %>%
  select(-started_at, -ended_at, -start_station_name, -start_station_id, -end_station_name, -end_s
tation_id)
```

Check data for incorrect values: delete rows where value of `trip_duration` field is less or equal 0.

```
#delete rides with trip duration less then 0
tripdata_2022_clean <- tripdata_2022_clean %>%
  filter(trip_duration > 0)
```

Also we can see that our dataset contains very short rides (even several seconds). Let’s show distribution of number of rides for trip duration.

```
#show trip rides distribution
tripdata_2022_clean %>%
  group_by(trip_duration, member_casual) %>%
  summarize(count_trips = n()) %>%
  filter(trip_duration <= 10) %>%
  ggplot(aes(x=trip_duration, y = count_trips, fill = member_casual)) +
    geom_col() +
    labs(x = "Trip duration (min)", y = "Number of trips", title = "Trip duration distribution")
```



We can assume that rides with length less then 1 minute weren’t the real rides and were started by mistake. So let’s not consider them in our analyze.

```
#delete rides with trip duration less then 1 minute
tripdata_2022_clean <- tripdata_2022_clean %>%
  filter(trip_duration >= 1)
```

At the end we’ll get a clean dataframe. Let’s show the main characteristics of our dataframe and its statistical summary:

```
#show dataframe description
cat("Column names: ", colnames(tripdata_2022_clean), sep=",", "\n")
```

```
## Column names: ,ride_id,rideable_type,start_lat,start_lng,end_lat,end_lng,member_casual,trip_dur
ation,trip_date,trip_start_time,trip_weekday,
```

```
cat("Number of rows", nrow(tripdata_2022_clean), "\n")
```

```
## Number of rows 5546628
```

```
cat("Dimensions: ", dim(tripdata_2022_clean), "\n")
```

```
## Dimensions:  5546628 11
```

```
cat("List of columns with datatypes: ", "\n")
```

```
## List of columns with datatypes:
```

```
print(str(tripdata_2022_clean))
```

```
## tibble [5,546,628 × 11] (S3: tbl_df/tbl/data.frame)
##  $ ride_id        : chr [1:5546628] "98D355D9A9852BE9" "04706CA7F5BD25EE" "42178E850B92597A" "6
B93C46E8F5B114C" ...
##  $ rideable_type  : chr [1:5546628] "classic_bike" "electric_bike" "electric_bike" "classic_bik
e" ...
##  $ start_lat      : num [1:5546628] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng      : num [1:5546628] -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ end_lat        : num [1:5546628] 41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng        : num [1:5546628] -87.6 -87.6 -87.6 -87.6 -87.6 ...
##  $ member_casual  : chr [1:5546628] "casual" "casual" "casual" "casual" ...
##  $ trip_duration  : num [1:5546628] 1.72 3.65 30.97 28.88 28.48 ...
##  $ trip_date      : Date[1:5546628], format: "2022-01-01" "2022-01-01" ...
##  $ trip_start_time: 'hms' num [1:5546628] 00:00:05 00:01:00 00:01:16 00:02:14 ...
##   ..- attr(*, "units")= chr "secs"
##  $ trip_weekday   : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 7 7 7 7 7 7 7 7 7 7 ...
## NULL
```

```
#show data summary
cat("Statistical summary of data: ", "\n")
```

```
## Statistical summary of data:
```

```
summary(tripdata_2022_clean)
```

```
##    ride_id          rideable_type        start_lat       start_lng
##  Length:5546628     Length:5546628     Min.   :41.64   Min.   :-87.84
##  Class :character   Class :character   1st Qu.:41.88   1st Qu.:-87.66
##  Mode  :character   Mode  :character   Median :41.90   Median :-87.64
##                                        Mean   :41.90   Mean   :-87.65
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :45.64   Max.   :-73.80
##
##     end_lat         end_lng       member_casual      trip_duration
##  Min.   : 0.00   Min.   :-88.14   Length:5546628     Min.   :    1.00
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character   1st Qu.:    6.07
##  Median :41.90   Median :-87.64   Mode  :character   Median :   10.52
##  Mean   :41.90   Mean   :-87.65                      Mean   :   19.86
##  3rd Qu.:41.93   3rd Qu.:-87.63                      3rd Qu.:   18.75
##  Max.   :42.37   Max.   : 0.00                       Max.   :41387.25
##  NA's   :5856    NA's   :5856
##    trip_date          trip_start_time   trip_weekday
##  Min.   :2022-01-01   Length:5546628    Sun:759173
##  1st Qu.:2022-05-28   Class1:hms        Mon:735319
##  Median :2022-07-22   Class2:difftime   Tue:766074
##  Mean   :2022-07-19   Mode  :numeric    Wed:781411
##  3rd Qu.:2022-09-15                     Thu:823805
##  Max.   :2022-12-31                     Fri:784690
##                                         Sat:896156
```

```
#show first rows
head(tripdata_2022_clean)
```

```
## # A tibble: 6 × 11
##   ride_id        rideable_type start_lat start_lng end_lat end_lng member_casual
##   <chr>          <chr>             <dbl>     <dbl>   <dbl>   <dbl> <chr>
## 1 98D355D9A9852… classic_bike       41.9     -87.6    41.9   -87.6 casual
## 2 04706CA7F5BD2… electric_bike      41.9     -87.6    41.9   -87.6 casual
## 3 42178E850B925… electric_bike      41.9     -87.6    41.9   -87.6 casual
## 4 6B93C46E8F5B1… classic_bike       41.9     -87.6    41.9   -87.6 casual
## 5 466943353EAC8… classic_bike       41.9     -87.6    41.9   -87.6 casual
## 6 7BFB6F3EAF946… electric_bike      41.9     -87.7    41.9   -87.7 casual
## # ℹ 4 more variables: trip_duration <dbl>, trip_date <date>,
## #   trip_start_time <time>, trip_weekday <ord>
```

Deliverable:
[✓] *Documentation of any cleaning or manipulation of data*

Analyze

Now we have all columns that we need and in appropriate formats, so we can start our analysis. Let’s aggregate our data to make it useful for analysis.

1. Aggregate rides by membership type (`member_casual`) and calculate average, median, minimum and maximum trip duration.

```
#aggr data by member_casual
tripdata_2022_clean %>%
  group_by(membership = member_casual) %>%
  summarise(mean_trip = mean(trip_duration), median_trip = median(trip_duration), max_trip = max(t
rip_duration), min_trip = min(trip_duration), count_trips = n())
```

```
## # A tibble: 2 × 6
##   membership mean_trip median_trip max_trip min_trip count_trips
##   <chr>          <dbl>       <dbl>    <dbl>    <dbl>       <int>
## 1 casual          29.8       13.3    41387.        1     2274142
## 2 member          13.0        9.03    1560.        1     3272486
```

- **Users with annual membership made more trips during the 2022 year than casual riders (3.3 mio trips vs. 2.3 mio trips), but their trips are much shorter. Average trip for members is about 13 minutes vs. 30 minutes for casual riders. Median trip (show more frequent trip length and decrease influence of too short or too long trips) is also longer for casual riders: 13 minutes vs. 9 minute.**

Let's use `filter` function to create separated dataframes for casual riders and for members. Further, we can use it to find differences between two types of riders.

```
#only casual riders
tripdata_2022_casual <- filter(tripdata_2022_clean, member_casual == "casual")
```

```
#only annual members
tripdata_2022_member <- filter(tripdata_2022_clean, member_casual == "member")
```

2. Look at the rides distribution for both types of riders by months:

```
#aggr by months
aggr_month <- tripdata_2022_clean %>%
  group_by(member_casual, month = month(trip_date, label = TRUE, locale = "eb_EN.UTF-8")) %>%
  summarize(mean_trip = mean(trip_duration), median_trip = median(trip_duration), max_trip = max(t
rip_duration), min_trip = min(trip_duration), count_trips = n())
```

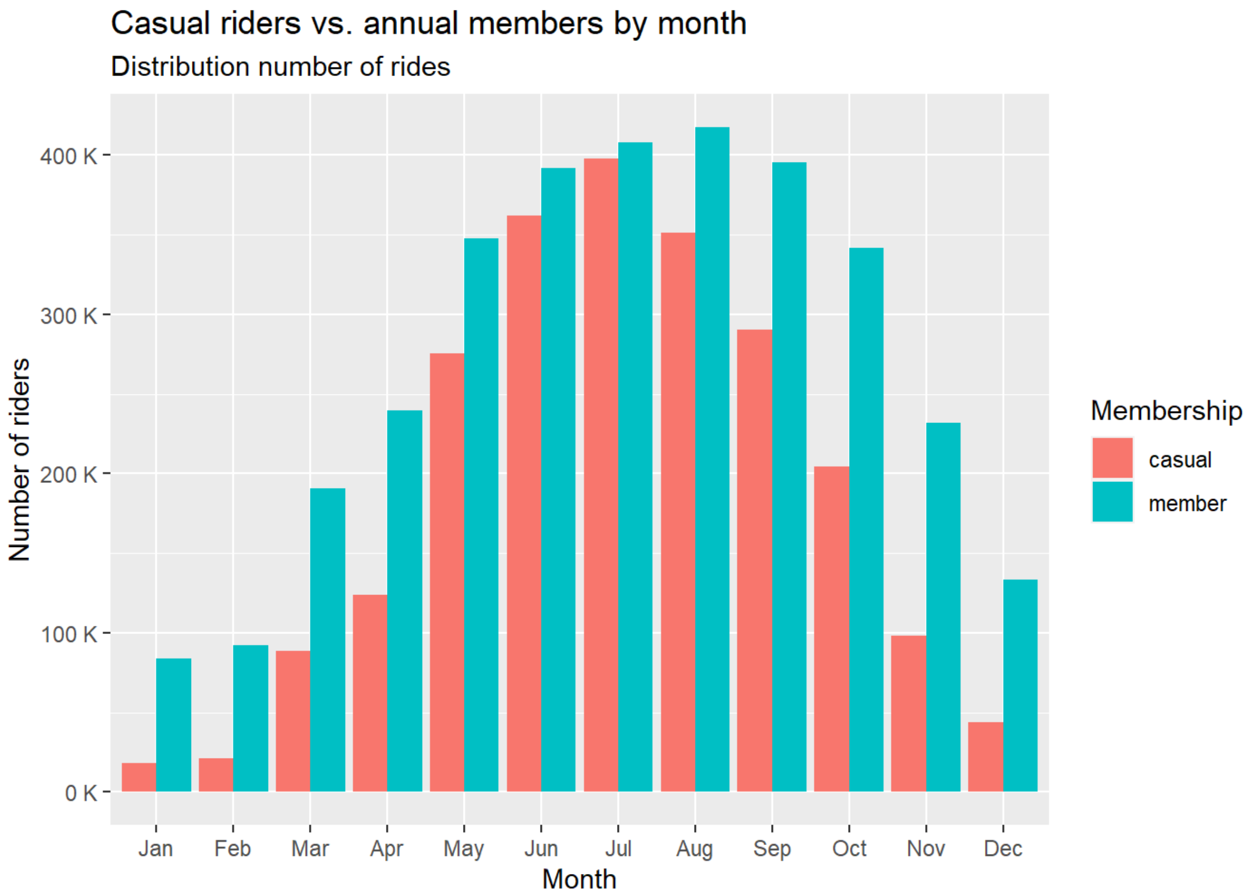
```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
```

```
#show the results
aggr_month %>% arrange(member_casual, month)
```

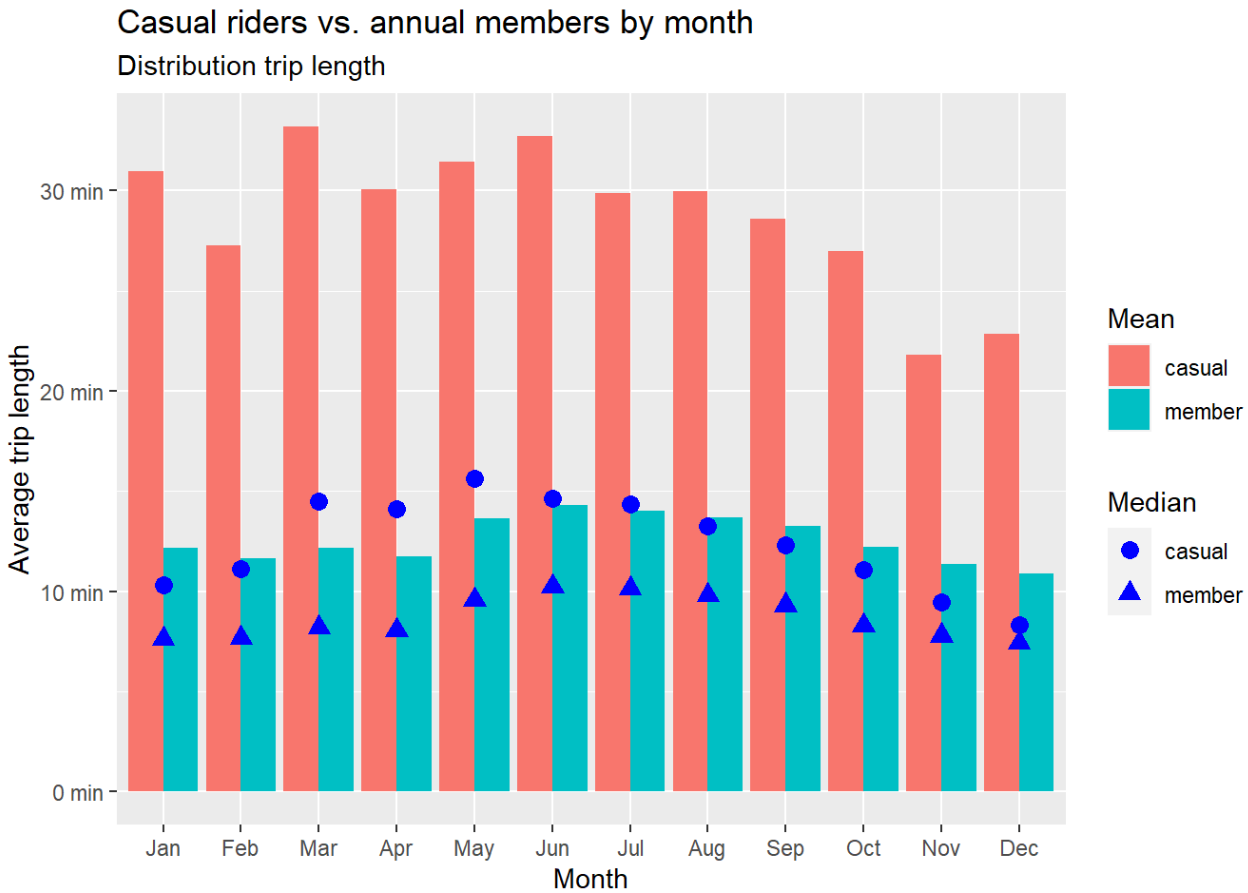
```
## # A tibble: 24 × 7
## # Groups:   member_casual [2]
##    member_casual month mean_trip median_trip max_trip min_trip count_trips
##    <chr>         <ord>     <dbl>       <dbl>    <dbl>    <dbl>       <int>
##  1 casual        Jan        31.0        10.3   29271.        1       18154
##  2 casual        Feb        27.3        11.1   10906.        1       20981
##  3 casual        Mar        33.2        14.5   34354.        1       88343
##  4 casual        Apr        30.1        14.1   21122.        1      124080
##  5 casual        May        31.5        15.6   36258.        1      275125
##  6 casual        Jun        32.7        14.6   35821.        1      361929
##  7 casual        Jul        29.9        14.4   34209.        1      397648
##  8 casual        Aug        29.9        13.3   28129.        1      351164
##  9 casual        Sep        28.6        12.3   27698.        1      290384
## 10 casual        Oct        27.0        11.1   41387.        1      204205
## # ℹ 14 more rows
```

Create visualizations for distribution by months:

```
#plot: number of rides by months
ggplot(data = aggr_month, aes(x = month, y = count_trips, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Month", y = "Number of riders", title = "Casual riders vs. annual members by month", s
ubtitle = "Distribution number of rides") +
  scale_y_continuous(labels = label_number(suffix = " K", scale = 1e-3)) +
  guides(fill = guide_legend(title = "Membership"))
```




```
#plot duration by months
ggplot(data = aggr_month) +
  geom_col(aes(x = month, y = mean_trip, fill = member_casual), position = "dodge") +
  geom_point(aes(x = month, y = median_trip, shape = member_casual), color = "blue", size = 3) +
  labs(x = "Month", y = "Average trip length", shape = "Median", fill = "Mean", title = "Casual ri
ders vs. annual members by month", subtitle = "Distribution trip length") +
  scale_y_continuous(labels = label_number(suffix = " min"))
```



- **Both types of riders have almost the same number of trips during summer month (June and July). And number of trips for casual riders is the lowest in winter months (less than 50 thousands a month).**

3. Aggregate rides by day of the week:

```
#aggr by weekday
aggr_weekday <- tripdata_2022_clean %>%
  group_by(member_casual, weekday = trip_weekday) %>%
  summarize(mean_trip = mean(trip_duration), median_trip = median(trip_duration), max_trip = max(t
rip_duration), min_trip = min(trip_duration), count_trips = n())
```

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

```
#show the results
aggr_weekday %>% arrange(member_casual, weekday)
```

```
## # A tibble: 14 × 7
## # Groups:   member_casual [2]
##    member_casual weekday mean_trip median_trip max_trip min_trip count_trips
##    <chr>         <ord>       <dbl>       <dbl>    <dbl>    <dbl>       <int>
##  1 casual        Sun          34.8       15.4    36258.        1      380940
##  2 casual        Mon          29.8       13.2    32035.        1      272051
##  3 casual        Tue          26.4       11.8    31086.        1      258282
##  4 casual        Wed          25.3       11.7    35821.        1      268784
##  5 casual        Thu          26.1       12      31024.        1      303014
##  6 casual        Fri          28.6       12.8    32403.        1      327770
##  7 casual        Sat          33.3       15.3    41387.        1      463301
##  8 member        Sun          14.4        9.78    1500.        1      378233
##  9 member        Mon          12.5        8.63    1500.        1      463268
## 10 member        Tue          12.4        8.65    1500.        1      507792
## 11 member        Wed          12.4        8.78    1500.        1      512627
## 12 member        Thu          12.6        8.85    1500.        1      520791
## 13 member        Fri          12.8        8.92    1500.        1      456920
## 14 member        Sat          14.5       10.0     1560.        1      432855
```

Create visualizations for distribution by day of the week:

```
#plot number of rides by weekdays
ggplot(data = aggr_weekday, aes(x = weekday, y = count_trips, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Weekday", y = "Number of riders", title = "Casual riders vs. annual members by weekday
s", subtitle = "Distribution number of rides") +
  scale_y_continuous(labels = label_number(suffix = " K", scale = 1e-3)) +
  guides(fill = guide_legend(title = "Membership"))
```

```
#plot trip duration by weekday
ggplot(data = aggr_weekday) +
  geom_col(aes(x = weekday, y = mean_trip, fill = member_casual), position = "dodge") +
  geom_point(aes(x = weekday, y = median_trip, shape = member_casual), color = "blue", size = 3) +
  labs(x = "Weekday", y = "Average trip length", shape = "Median", fill = "Mean", title = "Casual
riders vs. annual members by weekday", subtitle = "Distribution trip length") +
  scale_y_continuous(labels = label_number(suffix = " min"))
```

Show distribution by weekdays and hours:

```
#aggr by weekday and hour
aggr_weekday_hour <- tripdata_2022_clean %>%
  group_by(member_casual, weekday = trip_weekday, hour_start = hour(trip_start_time)) %>%
  summarize(count_trips = n())
```

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

```
#plot number of rides by weekdays
ggplot(data = aggr_weekday_hour, aes(x = hour_start, y = count_trips, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Hour", y = "Number of riders", title = "Casual riders vs. annual members by hours", su
btitle = "Distribution number of rides for every week day") +
  scale_y_continuous(labels = label_number(suffix = " K", scale = 1e-3)) +
  guides(fill = guide_legend(title = "Membership")) +
  facet_wrap(~weekday, nrow=4)
```

- **Annual members make more trips on workdays and casual riders rent bike more often on weekends. We can see that casual riders take even more rides on weekends than members. Also both types of riders make longer rides during weekends. And they behave almost the same way on Sunday and Saturday. But on work days we can see that annual members more often use bicycles about 8-9 a.m. and 17-18 p.m., so they use it to get to the workplace and to come back home.**

4. Aggregate rides by bike types (`rideable_type`).

```
#aggr by rideable_type
aggr_type <- tripdata_2022_clean %>%
  group_by(member_casual, rideable_type) %>%
  summarize(mean_trip = mean(trip_duration), median_trip = median(trip_duration), max_trip = max(t
rip_duration), min_trip = min(trip_duration), count_trips = n())
```

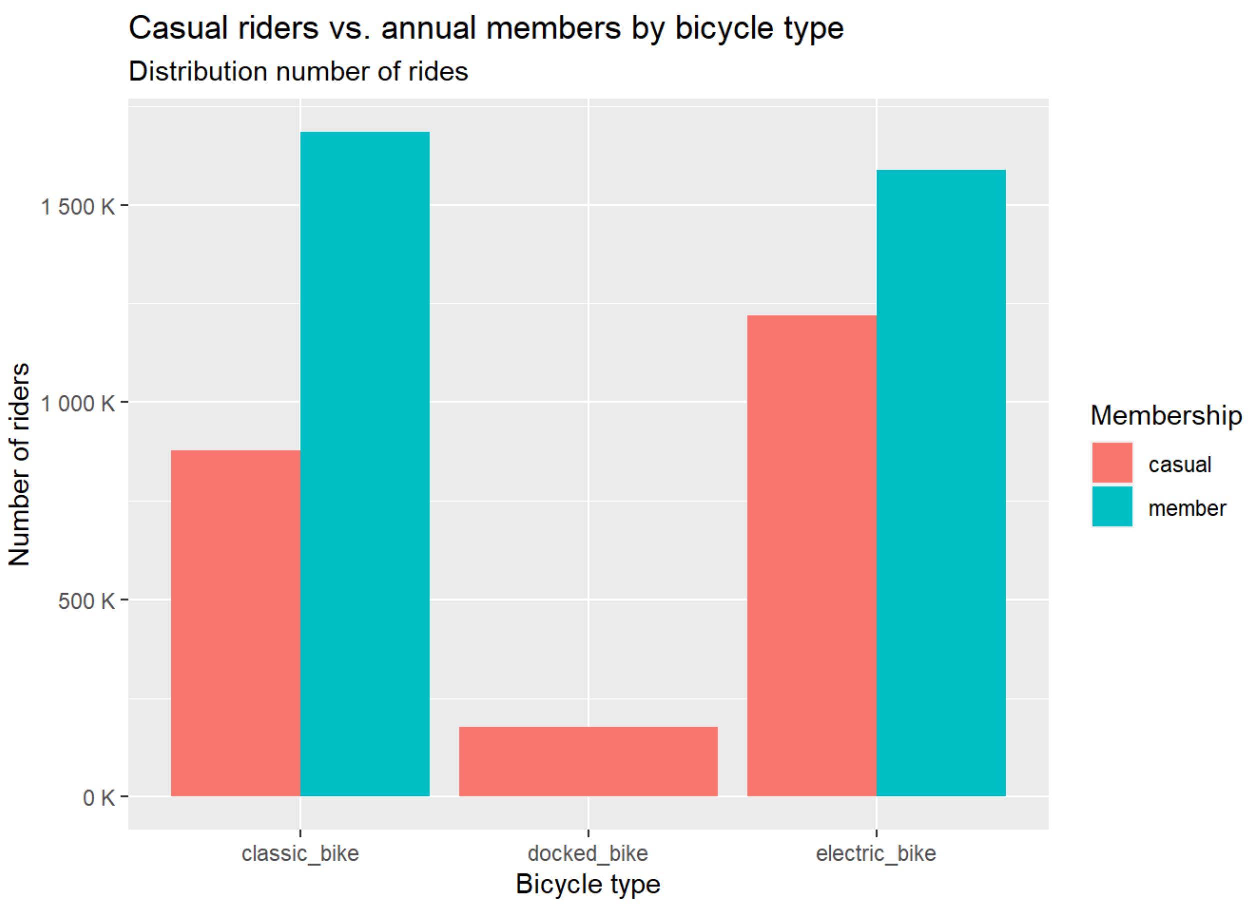
```
## `summarise()` has grouped output by 'member_casual'. You can override using the
## `.groups` argument.
```

```
#show the results
aggr_type %>% arrange(member_casual, rideable_type)
```

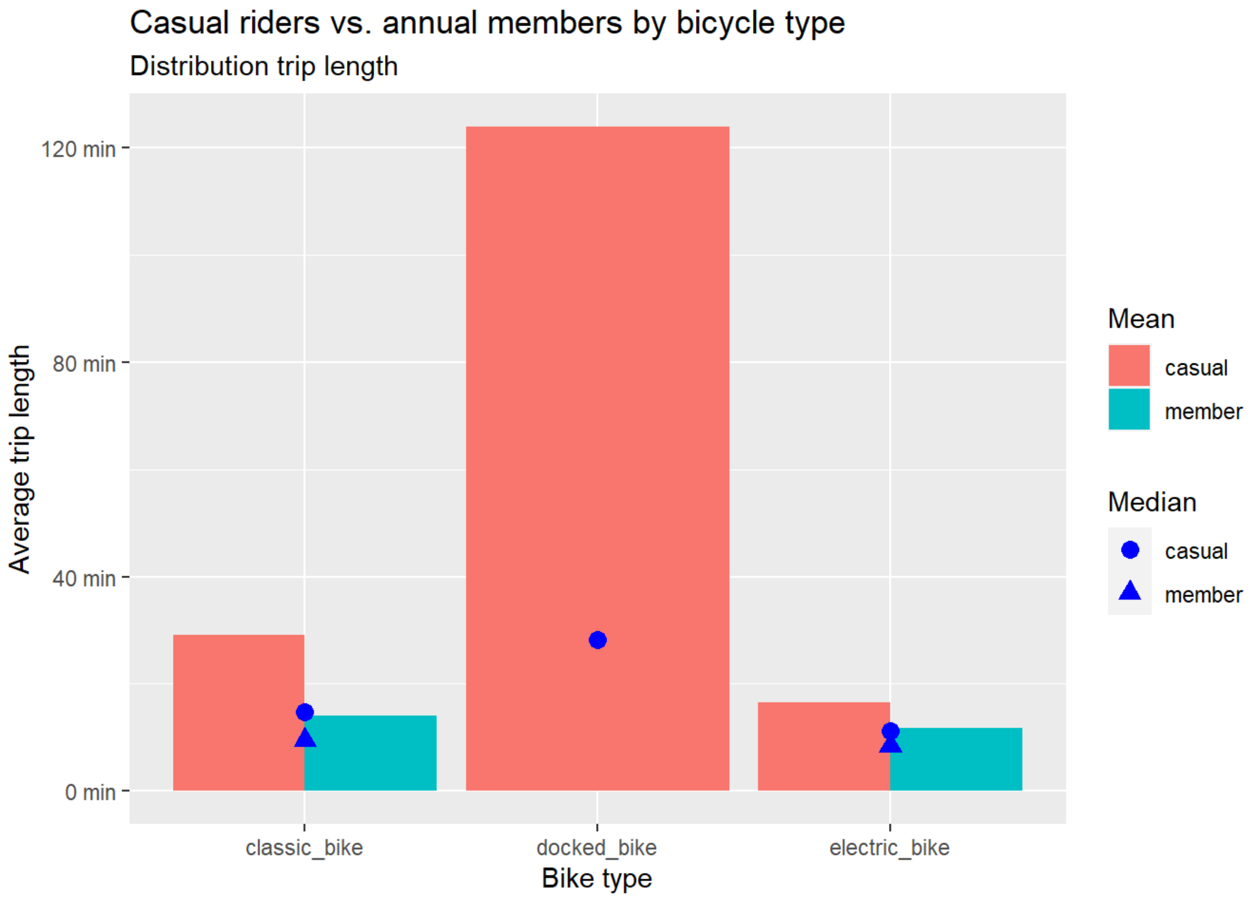
```
## # A tibble: 5 × 7
## # Groups:   member_casual [2]
##   member_casual rideable_type mean_trip median_trip max_trip min_trip
##   <chr>         <chr>             <dbl>       <dbl>    <dbl>    <dbl>
## 1 casual        classic_bike       29.2       14.8     1560.        1
## 2 casual        docked_bike       124.        28.3    41387.        1
## 3 casual        electric_bike      16.6       11.2      480.        1
## 4 member        classic_bike       14.1        9.53    1560.        1
## 5 member        electric_bike      11.8        8.53     614.        1
## # ℹ 1 more variable: count_trips <int>
```

Create visualizations for distribution by bicycle type:

```
#plot number of rides by rideable_type
ggplot(data = aggr_type, aes(x = rideable_type, y = count_trips, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Bicycle type", y = "Number of riders", title = "Casual riders vs. annual members by bi
cycle type", subtitle = "Distribution number of rides") +
  scale_y_continuous(labels = label_number(suffix = " K", scale = 1e-3)) +
  guides(fill = guide_legend(title = "Membership"))
```




```
#plot trip duration by rideable_type
ggplot(data = aggr_type) +
  geom_col(aes(x = rideable_type, y = mean_trip, fill = member_casual), position = "dodge") +
  geom_point(aes(x = rideable_type, y = median_trip, shape = member_casual), color = "blue", size
= 3) +
  labs(x = "Bike type", y = "Average trip length", shape = "Median", fill = "Mean", title = "Casua
l riders vs. annual members by bicycle type", subtitle = "Distribution trip length") +
  scale_y_continuous(labels = label_number(suffix = " min"))
```



- **Only casual riders use docked bikes in 2022 year. They used this type of bikes less often but trip duration is much longer then for classic and electric bikes: average trip length is more than 2 hours and median trip is 28 minutes.**

5. Let's look at the most popular starting points for casual riders and members on the map.

Leave only TOP 100 the most popular starting stations for casual riders and for annual members.

```
#top 100 popular casual stations
tripdata_2022_casual_top <- tripdata_2022_casual %>%
  #filter(start_lat != end_lat & start_lng != end_lng) %>%
  group_by(member_casual, start_lat, start_lng, end_lat, end_lng, rideable_type) %>%
  summarize(count = n(), .groups = "drop") %>%
  arrange(-count) %>%
  slice(1:100)
head(tripdata_2022_casual_top)
```

```
## # A tibble: 6 × 7
##   member_casual start_lat start_lng end_lat end_lng rideable_type count
##   <chr>             <dbl>     <dbl>   <dbl>   <dbl> <chr>         <int>
## 1 casual             41.9     -87.6    41.9   -87.6 classic_bike   5050
## 2 casual             41.9     -87.6    41.9   -87.6 classic_bike   2843
## 3 casual             41.9     -87.6    41.9   -87.6 classic_bike   2647
## 4 casual             41.9     -87.6    41.9   -87.6 docked_bike    2290
## 5 casual             41.9     -87.6    41.9   -87.6 classic_bike   2114
## 6 casual             41.9     -87.6    41.9   -87.6 docked_bike    1656
```

```
#top 100 popular members stations
tripdata_2022_member_top <- tripdata_2022_member %>%
  #filter(start_lat != end_lat & start_lng != end_lng) %>%
  group_by(member_casual, start_lat, start_lng, end_lat, end_lng, rideable_type) %>%
  summarize(count = n(), .groups = "drop") %>%
  arrange(-count) %>%
  slice(1:100)
head(tripdata_2022_member_top)
```

```
## # A tibble: 6 × 7
##   member_casual start_lat start_lng end_lat end_lng rideable_type count
##   <chr>             <dbl>     <dbl>   <dbl>   <dbl> <chr>         <int>
## 1 member             41.8     -87.6    41.8   -87.6 electric_bike  9968
## 2 member             41.8     -87.6    41.8   -87.6 classic_bike   5080
## 3 member             41.8     -87.6    41.8   -87.6 classic_bike   4794
## 4 member             41.8     -87.6    41.8   -87.6 classic_bike   4498
## 5 member             41.8     -87.6    41.8   -87.6 classic_bike   4088
## 6 member             41.8     -87.6    41.8   -87.6 electric_bike  2681
```

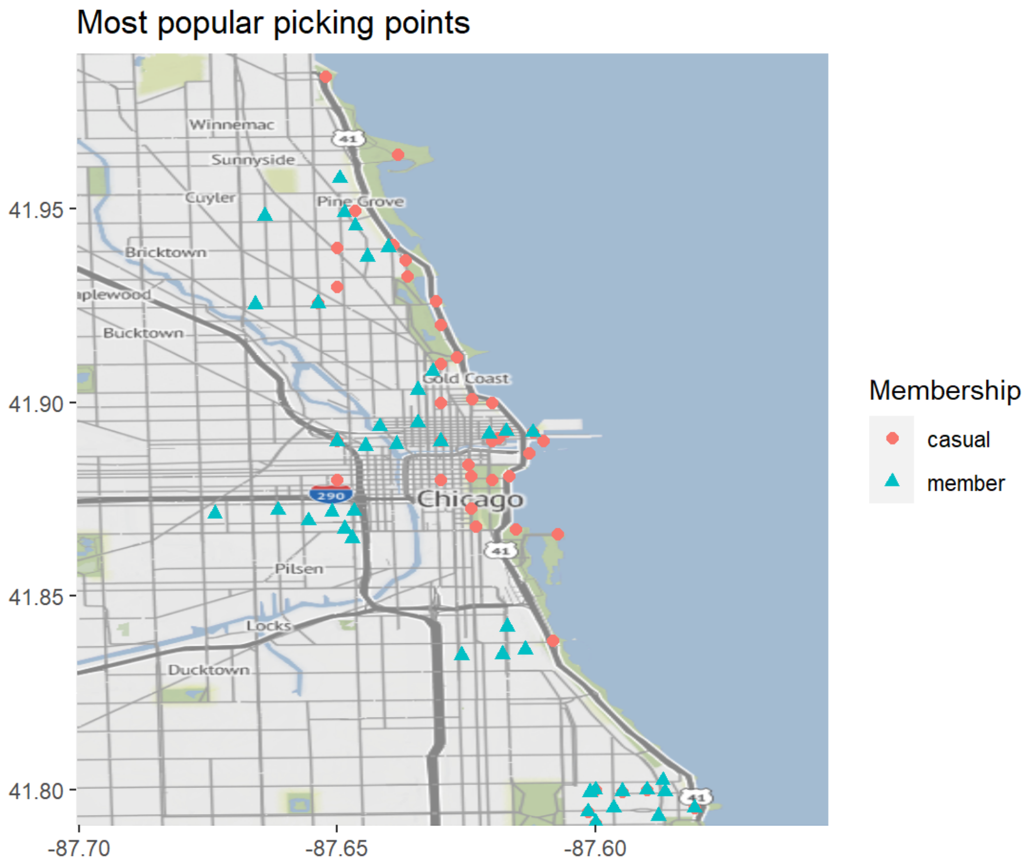
Unite data frames into one dataframe:

```
#union casual and members
tripdata_2022_top <- union(tripdata_2022_casual_top, tripdata_2022_member_top)
```

Show the most popular stations on the map:

```
# maps of Chicago
ggmap(chicago_map,darken = c(0.1, "white")) +
  #add points
  geom_point(tripdata_2022_top, mapping = aes(x = start_lng, y = start_lat, shape = member_casual,
color = member_casual), size = 2)  +
   coord_fixed(0.75) +
   labs(title = "Most popular picking points", x="", y="", shape = "Membership", color = "Membersh
ip")
```

```
## Coordinate system already present. Adding new coordinate system, which will
## replace the existing one.
```



- **We can notice that the stations close to the seaside is more popular for casual riders. Annual members use station all over the city.**

Deliverable:
[✓] *A summary of your analysis*

Share

Let's explore our visualizations and summarize the differences between members and casual riders:

1. Users with annual membership made more trips during the 2022 year than casual riders, but their trips are much shorter.
2. Casual riders make almost the same number of rides in summer months (June and July) and significantly less rides in other seasons.
3. Casual riders use Cyclistic almost the same way as annual members on weekends (Saturday and Sunday). But annual members prefer work days for their rides. Also during work days annual members more often take rides to get to work and back than casual riders.
4. Only casual riders used docked bikes in 2022 year. They used this type of bikes less often but trip duration was much longer then for classic and electric bikes.
5. We can notice that the stations close to the seaside is more popular for casual riders. Annual members use station all over the city.

Deliverable:
[✓] *Supporting visualizations and key findings*

Act

According to findings we have found and visualizations above, here are top three reccomendations for the stakeholders:

1. Find clients who behave like annual members (use bicycles for getting to work and back, make many shorts trips) and suggest them membership. We need additional data with clients IDs for this purpose.
2. Create promotion for special prices for annual members on weekends (Saturdays and Sundays). That can attract more casual riders that use bike-sharing on weekends to buy annual subscription.
3. Find places for new stations inside the city that also can attract new members. We need a survey to find out where people are more likely to use bicycles regularly.

Deliverable:
[✓] *Your top three recommendations based on your analysis*