BikeShare Case Study

26/06/2022

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

Scenario

The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, my team wants to understand **how casual riders and annual members use Cyclistic bikes differently**. My team will design a new marketing strategy to **convert casual riders into annual members from these insights.**

The analysis will follow the 6 phases of the Data Analysis Process: Ask, Prepare, Process, Analyze, Share, and Act (APPASA).

ASK

Key tasks

1. Business task: How can we convert casual riders into an annual members.
2. Key Stakeholders: Lily Moreno (director of marketing and my manager) and Cyclistic executive team.

A clear statement of the business task

The financial analysts have concluded that annual membership is much more profitable than single-ride and full-day passes from their analysis. So to make people pot for the yearly membership, our marketing campaign should urge the casual riders to convert to annual riders.

As a solution, we should understand why casual riders would convert to a yearly membership? Based on the insights from the above question, we can achieve the maximum required conversion rate from casual to annual riders.

PREPARE

Guiding Questions

1. **Where is your data Located?**

   Data is downloaded from https://divvy-tripdata.s3.amazonaws.com/index.html to local system and then uploaded in RStudio cloud where I could use the R programming language for the analysis.

2. **How is data organized?**

   Data is segregated into quarters from the year 2013 to 2020 till the first quarters of the latter year. Each year having its CSV file. (I will use the data from past 12 months)

3. **Are there any issues with bias or credibility in this data? Does your data ROCCC?**

   The data has been collected directly from the company's customers, that is, bike riders so there is no issue of bias and credibility for the same reason. It is also Reliable, Original, Comprehensive, Current, and Cited, which satisfies ROCCC.

4. **How are you accessing licensing, privacy, security, and accessibility?**

   The data was collected by Motivate International Inc. under the following license https://www.divvybikes.com/data-license-agreement. Also the data-set does not contain any personal information about its customers (or riders) to violate the privacy.

5. **How did you verify the data's integrity?**

   The qualities required to verify the data integrity are accuracy, completeness, consistency, and trustworthiness. The data is complete as it contains all the required components to measure the entity. The data is consistent across the years with year having its CSV file which is organized in an equal number of columns and same data types. As the credibility was proven before, it is also trustworthy.

6. **How does it help to answer your question?**

   By creating new features from existing ones like rideable_type, started_at, and ended_at(which are date-timestamp variables), we can deduce relationship between annual members and casual riders. The relationship analyzed will be useful to answer the question, that is, convert casual riders to annual members.

7. **Are there any problems with the data?**

   Yes, the data had a couple of problems. There are few rows with "N/A" values which needs to be removed. Also, there are duplicates which have to be eliminated.

Installed required packages.

```
#install.packages(c("tidyverse","ggplot2","lubridate","dplyr","readr","geosphere","scales", "janitor"))
```

Loading the installed packages above into the work-space.

```
library(tidyverse)
```

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
library(ggplot2)
library(lubridate)
```

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

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

```
library(dplyr)
library(readr)
library(geosphere)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Let's load the data stored in CSV file from (June-2021 to May-2022) i.e., 1 years of data

```
tripdata_202106 <- read.csv("./data/202106-divvy-tripdata.csv")
tripdata_202107 <- read.csv("./data/202107-divvy-tripdata.csv")
tripdata_202108 <- read.csv("./data/202108-divvy-tripdata.csv")
tripdata_202109 <- read.csv("./data/202109-divvy-tripdata.csv")
tripdata_202110 <- read.csv("./data/202110-divvy-tripdata.csv")
tripdata_202111 <- read.csv("./data/202111-divvy-tripdata.csv")
tripdata_202112 <- read.csv("./data/202112-divvy-tripdata.csv")
tripdata_202201 <- read.csv("./data/202201-divvy-tripdata.csv")
tripdata_202202 <- read.csv("./data/202202-divvy-tripdata.csv")
tripdata_202203 <- read.csv("./data/202203-divvy-tripdata.csv")
tripdata_202204 <- read.csv("./data/202204-divvy-tripdata.csv")
tripdata_202205 <- read.csv("./data/202205-divvy-tripdata.csv")
```

```
glimpse(tripdata_202106)
```

```
## Rows: 729,595
## Columns: 13
## $ ride_id            <chr> "99FEC93BA843FB20", "06048DCFC8520CAF", "9598066F68~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <chr> "2021-06-13 14:31:28", "2021-06-04 11:18:02", "2021~
## $ ended_at           <chr> "2021-06-13 14:34:11", "2021-06-04 11:24:19", "2021~
## $ start_station_name <chr> "", "", "", "", "", "", "", "", "", "", "", "", "",~
## $ start_station_id   <chr> "", "", "", "", "", "", "", "", "", "", "", "", "",~
## $ end_station_name   <chr> "", "", "", "", "", "", "", "", "", "Michigan Ave &~
## $ end_station_id     <chr> "", "", "", "", "", "", "", "", "", "13042", "", ""~
## $ start_lat          <dbl> 41.80, 41.79, 41.80, 41.78, 41.80, 41.78, 41.79, 41~
## $ start_lng          <dbl> -87.59, -87.59, -87.60, -87.58, -87.59, -87.58, -87~
## $ end_lat            <dbl> 41.80000, 41.80000, 41.79000, 41.80000, 41.79000, 4~
## $ end_lng            <dbl> -87.6000, -87.6000, -87.5900, -87.6000, -87.5900, -~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

Create a list with all the data sets and quickly inspect all the data frames

```
data_list = list(tripdata_202106, tripdata_202107, tripdata_202108, tripdata_202109, tripdata_202110, t
```

Print column names

```
column_names <- colnames(data_list[[1]])
print(column_names)
```

```
##  [1] "ride_id"            "rideable_type"      "started_at"        
##  [4] "ended_at"           "start_station_name" "start_station_id"  
##  [7] "end_station_name"   "end_station_id"     "start_lat"         
## [10] "start_lng"          "end_lat"            "end_lng"           
## [13] "member_casual"
```

Let's ensure data integrity by checking the column names of all CSV files for consistency. Check that all data frames include the same columns in the same order.

```
for (df in data_list) {
  df_colnames <- colnames(df)
  print(column_names %in% df_colnames)
}
```

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

Further Clean Up

Next step is to check the data for errors, inconsistencies, duplicate observations, empty rows or columns, NA values and outliers. In other words, we make sure the data provided is clean and ready for analysis.

Check that all data frames include the same data types for the the same variable across all data sets.

The compare_df_cols function will compare all data-frames and return the mismatched columns.

```
compare_df_cols(data_list, return = c("mismatch"), bind_method = c("bind_rows"))
```

```
##  [1] column_name  data_list_1  data_list_2  data_list_3  data_list_4 
##  [6] data_list_5  data_list_6  data_list_7  data_list_8  data_list_9 
## [11] data_list_10 data_list_11 data_list_12
## <0 rows> (or 0-length row.names)
```

From the above query, we can see that all variables have consistent data types.

Now, let's join the elements in the list into a one single data frame.

```
all_trips <- bind_rows(data_list)
head(all_trips)
```

```
##            ride_id rideable_type          started_at            ended_at
## 1 99FEC93BA843FB20 electric_bike 2021-06-13 14:31:28 2021-06-13 14:34:11
## 2 06048DCFC8520CAF electric_bike 2021-06-04 11:18:02 2021-06-04 11:24:19
## 3 9598066F68045DF2 electric_bike 2021-06-04 09:49:35 2021-06-04 09:55:34
## 4 B03C0FE48C412214 electric_bike 2021-06-03 19:56:05 2021-06-03 20:21:55
## 5 B9EEA89F8FEE73B7 electric_bike 2021-06-04 14:05:51 2021-06-04 14:09:59
## 6 62B943CEAAA420BA electric_bike 2021-06-03 19:32:01 2021-06-03 19:38:46
##   start_station_name start_station_id end_station_name end_station_id start_lat
## 1                                                                         41.80
```

```
## 2                                                                    41.79
## 3                                                                    41.80
## 4                                                                    41.78
## 5                                                                    41.80
## 6                                                                    41.78
##   start_lng end_lat end_lng member_casual
## 1    -87.59   41.80  -87.60        member
## 2    -87.59   41.80  -87.60        member
## 3    -87.60   41.79  -87.59        member
## 4    -87.58   41.80  -87.60        member
## 5    -87.59   41.79  -87.59        member
## 6    -87.58   41.78  -87.58        member
```

```r
glimpse(all_trips)
```

```
## Rows: 5,860,776
## Columns: 13
## $ ride_id            <chr> "99FEC93BA843FB20", "06048DCFC8520CAF", "9598066F68~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <chr> "2021-06-13 14:31:28", "2021-06-04 11:18:02", "2021~
## $ ended_at           <chr> "2021-06-13 14:34:11", "2021-06-04 11:24:19", "2021~
## $ start_station_name <chr> "", "", "", "", "", "", "", "", "", "", "", "", "",~
## $ start_station_id   <chr> "", "", "", "", "", "", "", "", "", "", "", "", "",~
## $ end_station_name   <chr> "", "", "", "", "", "", "", "", "", "Michigan Ave &~
## $ end_station_id     <chr> "", "", "", "", "", "", "", "", "", "13042", "", ""~
## $ start_lat          <dbl> 41.80, 41.79, 41.80, 41.78, 41.80, 41.78, 41.79, 41~
## $ start_lng          <dbl> -87.59, -87.59, -87.60, -87.58, -87.59, -87.58, -87~
## $ end_lat            <dbl> 41.80000, 41.80000, 41.79000, 41.80000, 41.79000, 4~
## $ end_lng            <dbl> -87.6000, -87.6000, -87.5900, -87.6000, -87.5900, -~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

I see few missing rows in the above glimpse, so lets remove those. First, let's replace blank values with NA.

```r
all_trips[all_trips==""] <- NA
glimpse(all_trips)
```

```
## Rows: 5,860,776
## Columns: 13
## $ ride_id            <chr> "99FEC93BA843FB20", "06048DCFC8520CAF", "9598066F68~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <chr> "2021-06-13 14:31:28", "2021-06-04 11:18:02", "2021~
## $ ended_at           <chr> "2021-06-13 14:34:11", "2021-06-04 11:24:19", "2021~
## $ start_station_name <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ start_station_id   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ end_station_name   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "Michigan Ave &~
## $ end_station_id     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, "13042", NA, NA~
## $ start_lat          <dbl> 41.80, 41.79, 41.80, 41.78, 41.80, 41.78, 41.79, 41~
## $ start_lng          <dbl> -87.59, -87.59, -87.60, -87.58, -87.59, -87.58, -87~
## $ end_lat            <dbl> 41.80000, 41.80000, 41.79000, 41.80000, 41.79000, 4~
## $ end_lng            <dbl> -87.6000, -87.6000, -87.5900, -87.6000, -87.5900, -~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

Let's check for missing values in the data frame.

```r
sapply(all_trips, 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
##            823167            823164            878338            878338
##         start_lat         start_lng           end_lat           end_lng
##                 0                 0              5036              5036
##      member_casual
##                 0
```

The datasets contain missing values as listed above. This may cause inaccuracies in the data analysis.

Let's drop all null values in the data frame.

```r
all_trips <- drop_na(all_trips)
glimpse(all_trips)
```

```
## Rows: 4,667,299
## Columns: 13
## $ ride_id            <chr> "0D904FEC5F84A538", "C4185F300D6B552B", "60F97090AC~
## $ rideable_type      <chr> "classic_bike", "classic_bike", "classic_bike", "cl~
## $ started_at         <chr> "2021-06-04 07:29:18", "2021-06-23 08:39:36", "2021~
## $ ended_at           <chr> "2021-06-04 07:45:34", "2021-06-23 08:41:37", "2021~
## $ start_station_name <chr> "Orleans St & Elm St", "Desplaines St & Kinzie St",~
## $ start_station_id   <chr> "TA1306000006", "TA1306000003", "TA1307000127", "KA~
## $ end_station_name   <chr> "Orleans St & Elm St", "Kingsbury St & Kinzie St", ~
## $ end_station_id     <chr> "TA1306000006", "KA1503000043", "TA1309000014", "TA~
## $ start_lat          <dbl> 41.90292, 41.88872, 41.95078, 41.88918, 41.88872, 4~
## $ start_lng          <dbl> -87.63772, -87.64445, -87.65917, -87.63851, -87.644~
## $ end_lat            <dbl> 41.90292, 41.88918, 41.96710, 41.88872, 41.88918, 4~
## $ end_lng            <dbl> -87.63772, -87.63851, -87.66743, -87.64445, -87.638~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

Are there any duplicated records?

```r
all_trips <- all_trips[!duplicated(all_trips$ride_id),]
```

```r
glimpse(all_trips)
```

```
## Rows: 4,667,299
## Columns: 13
## $ ride_id            <chr> "0D904FEC5F84A538", "C4185F300D6B552B", "60F97090AC~
## $ rideable_type      <chr> "classic_bike", "classic_bike", "classic_bike", "cl~
## $ started_at         <chr> "2021-06-04 07:29:18", "2021-06-23 08:39:36", "2021~
## $ ended_at           <chr> "2021-06-04 07:45:34", "2021-06-23 08:41:37", "2021~
## $ start_station_name <chr> "Orleans St & Elm St", "Desplaines St & Kinzie St",~
## $ start_station_id   <chr> "TA1306000006", "TA1306000003", "TA1307000127", "KA~
## $ end_station_name   <chr> "Orleans St & Elm St", "Kingsbury St & Kinzie St", ~
## $ end_station_id     <chr> "TA1306000006", "KA1503000043", "TA1309000014", "TA~
## $ start_lat          <dbl> 41.90292, 41.88872, 41.95078, 41.88918, 41.88872, 4~
## $ start_lng          <dbl> -87.63772, -87.64445, -87.65917, -87.63851, -87.644~
## $ end_lat            <dbl> 41.90292, 41.88918, 41.96710, 41.88872, 41.88918, 4~
## $ end_lng            <dbl> -87.63772, -87.63851, -87.66743, -87.64445, -87.638~
## $ member_casual      <chr> "member", "member", "member", "member", "member", "~
```

Seems like there isn't any duplicate values.

Let's get a list with the bike types.

```r
bike_types <- list(unique(all_trips$rideable_type))
print(bike_types)
```

```
## [[1]]
## [1] "classic_bike"  "docked_bike"   "electric_bike"
```

Get a list with the unique membership names

```r
member_type <- list(unique(all_trips$member_casual))
print(member_type)
```

```
## [[1]]
## [1] "member" "casual"
```

PROCESS

Guiding Questions

1. **What tools are you choosing and Why?**

   The entries in the trips tables are from the years 2013 to 2020, which is enormous. Since this is the case it is always easy and helpful to navigate through the data using either databases (like SQL) or R programming language. I'll be using R language to deal withe the data in this case study.

2. **What steps have you taken to ensure that your data is clean?**

   a. I have concatenated all the CSV files of each year into a single data frame

   b. Removed all the empty rows and columns from the concatenated data frame.

   c. Check the unique values in each variable using count() so that there is no miss spelling anywhere.

   d. Omitted N/A values from the entire data frame.

   e. Removed duplicates.

3. **How can you verify that your data is clean and ready to analyze?**

   After performing all the cleaning task mentioned above, I ran the below functions to verify:

   a. Used filter() to check if there were any missing values.

   b. Used count() to check the unique values of each variable.

   c. Used duplicated() to check for any duplicates present.

4. **Have you documented your cleaning process so you can review and share those results?**

   Yes, please find the below comments and snippets for the documentation.

Let's get started: Data Manipulation

First, let's convert started_at and ended_at columns from character to timestamp.

```r
all_trips$started_at = as.POSIXct(all_trips$started_at, format = "%Y-%m-%d %H:%M:%S")
all_trips$ended_at = as.POSIXct(all_trips$ended_at, format = "%Y-%m-%d %H:%M:%S")
```

Two new columns were added to compute the following.

- Time spent (in hours) on each trip (column name is labelled as time_difference_hours)
- Distance traveled (in Kilometers) on each trip (column name is labelled as distance_km)

The new dataframe is labelled as "all_trips_2".

```r
all_trips_2 <- mutate(all_trips, time_difference_hours = difftime(ended_at, started_at, units = "hours"
  mutate(all_trips, distance_km = distHaversine(cbind(start_lng, start_lat), cbind(end_lng, end_lat))*0
```

```r
head(all_trips_2)
```

```
##            ride_id rideable_type          started_at            ended_at
## 1 0D904FEC5F84A538  classic_bike 2021-06-04 07:29:18 2021-06-04 07:45:34
## 2 C4185F300D6B552B  classic_bike 2021-06-23 08:39:36 2021-06-23 08:41:37
## 3 60F97090AC85F55E  classic_bike 2021-06-27 12:26:58 2021-06-27 12:34:45
## 4 FBC7B1F0160AA304  classic_bike 2021-06-01 12:30:24 2021-06-01 12:33:02
## 5 37A52001AEEFA4E5  classic_bike 2021-06-01 11:32:17 2021-06-01 11:34:43
## 6 E49E5426F0B74023  classic_bike 2021-06-17 17:55:12 2021-06-17 17:58:50
##          start_station_name start_station_id          end_station_name
## 1       Orleans St & Elm St     TA1306000006       Orleans St & Elm St
## 2 Desplaines St & Kinzie St     TA1306000003  Kingsbury St & Kinzie St
## 3       Clark St & Grace St     TA1307000127     Clark St & Leland Ave
## 4  Kingsbury St & Kinzie St     KA1503000043 Desplaines St & Kinzie St
## 5 Desplaines St & Kinzie St     TA1306000003  Kingsbury St & Kinzie St
## 6  Kingsbury St & Kinzie St     KA1503000043 Desplaines St & Kinzie St
##   end_station_id start_lat start_lng  end_lat   end_lng member_casual
## 1   TA1306000006  41.90292 -87.63772 41.90292 -87.63772        member
## 2   KA1503000043  41.88872 -87.64445 41.88918 -87.63851        member
## 3   TA1309000014  41.95078 -87.65917 41.96710 -87.66743        member
## 4   TA1306000003  41.88918 -87.63851 41.88872 -87.64445        member
## 5   KA1503000043  41.88872 -87.64445 41.88918 -87.63851        member
## 6   TA1306000003  41.88918 -87.63851 41.88872 -87.64445        member
##   time_difference_hours distance_km
## 1      0.27111111 hours   0.0000000
## 2      0.03361111 hours   0.4950891
## 3      0.12972222 hours   1.9406431
## 4      0.04388889 hours   0.4950891
## 5      0.04055556 hours   0.4950891
## 6      0.06055556 hours   0.4950891
```

Perform some calculations to see if the data makes sense

```r
summary(all_trips_2$distance_km)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.000    0.898    1.601    2.114    2.781 1190.854
```

Median distance is 1.6km, maximum is way too far, at 1192.24 km. Let's investigate this outlier

```r
outlier_df <- filter(all_trips_2, (distance_km > 1000))
print(outlier_df$ride_id)
```

```
## [1] "3327172413547F64"
```

```r
print(outlier_df$time_difference_hours)
```

```
## Time difference of 0.04305556 hours
```

1192.24 km in ~3min? Doesn't look right

```r
print(paste("Distance (km): ", outlier_df$distance_km, " Duration: ", outlier_df$time_difference_hours,
```

```
## [1] "Distance (km):  1190.85454406115  Duration:  0.0430555555555556  Start Station:  Pawel Bialowas
```

Let's remove the outlier and show the summary again. This new dataframe is used for subsequent data analysis.

```
all_trips_cleaned <- filter(all_trips_2, !(distance_km > 1000 & time_difference_hours < 0.05))
summary(all_trips_cleaned$distance_km)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.898   1.601   2.113   2.781  32.247
```

```
summary(all_trips_cleaned)
```

```
##    ride_id          rideable_type        started_at
##  Length:4667298     Length:4667298     Min.   :2021-06-01 00:00:38.00
##  Class :character   Class :character   1st Qu.:2021-07-26 15:11:53.75
##  Mode  :character   Mode  :character   Median :2021-09-17 15:22:14.00
##                                        Mean   :2021-10-22 03:46:01.63
##                                        3rd Qu.:2021-12-30 16:09:50.25
##                                        Max.   :2022-05-31 23:59:56.00
##     ended_at                          start_station_name start_station_id
##  Min.   :2021-06-01 00:08:19.00   Length:4667298     Length:4667298
##  1st Qu.:2021-07-26 15:35:26.50   Class :character   Class :character
##  Median :2021-09-17 15:42:57.50   Mode  :character   Mode  :character
##  Mean   :2021-10-22 04:06:14.47
##  3rd Qu.:2021-12-30 16:24:49.25
##  Max.   :2022-06-01 16:45:51.00
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:4667298     Length:4667298     Min.   :41.65   Min.   :-87.83
##  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   :-87.64
##                                        3rd Qu.:41.93   3rd Qu.:-87.63
##                                        Max.   :42.06   Max.   :-87.53
##     end_lat         end_lng        member_casual      time_difference_hours
##  Min.   :41.65   Min.   :-87.83   Length:4667298     Length:4667298
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character   Class :difftime
##  Median :41.90   Median :-87.64   Mode  :character   Mode  :numeric
##  Mean   :41.90   Mean   :-87.64
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.17   Max.   :-87.53
##   distance_km
##  Min.   : 0.000
##  1st Qu.: 0.898
##  Median : 1.601
##  Mean   : 2.113
##  3rd Qu.: 2.781
##  Max.   :32.247
```

ANALYSIS

Scope of analysis

This report will analyse the user trends based on the historical data from June 2021 to May 2022.

Key Findings

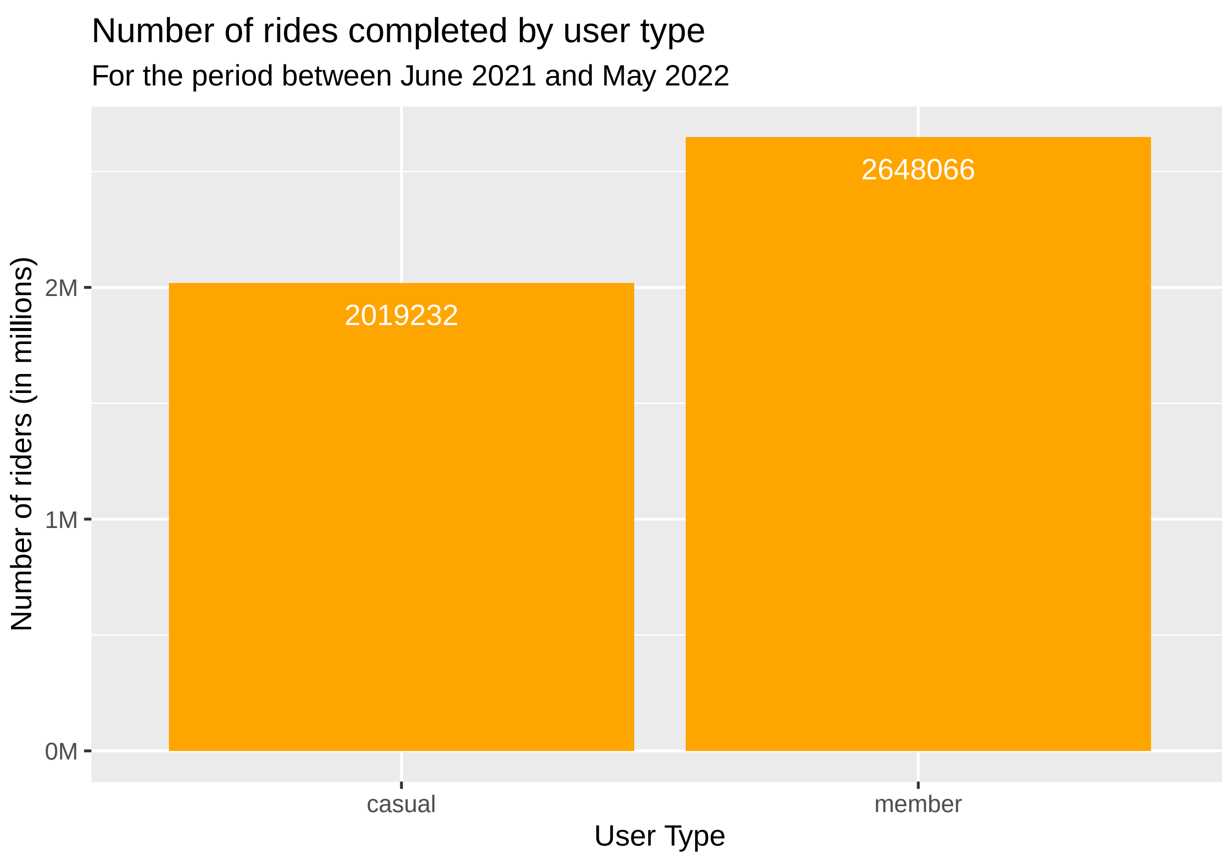
The data analysis revealed the following:

- Annual members made more trips compared to casual riders. This is expected because the annual members paid a fixed fee for unlimited 45-minutes rides, which may explain the propensity to maximize bike usages.
- The demand for bike rental increases during the summer period and decreases during the winter period. This observation is similar across casual riders and annual members. One plausible reason for this observation is that people are likely to go out during the summer and less likely to ride a bike during the winter.
- Among the casual riders, Saturdays and Sundays are the most popular days. However the same was not observed in the annual members; bike usage dropped by around 11.3% from weekday to weekend.
- For casual riders, the top 5 start and end bike stations are located near attractions. In contrast, for annual members, the top 5 start and end bike stations are located near residential areas.

Number of rides completed by user type

The annual members (referred to as members in the chart) made more trips than the casual riders.

```
ggplot(all_trips_cleaned, aes(x=member_casual)) +
  geom_bar(fill = "Orange") +
  labs(
    title = "Number of rides completed by user type",
    subtitle = "For the period between June 2021 and May 2022",
    x = "User Type",
    y = "Number of riders (in millions)") +
  scale_y_continuous(labels = label_number(suffix = "M", scale = 1e-6)) +
  geom_text(stat = 'count', aes(label=..count..), vjust=+2, color="white")
```

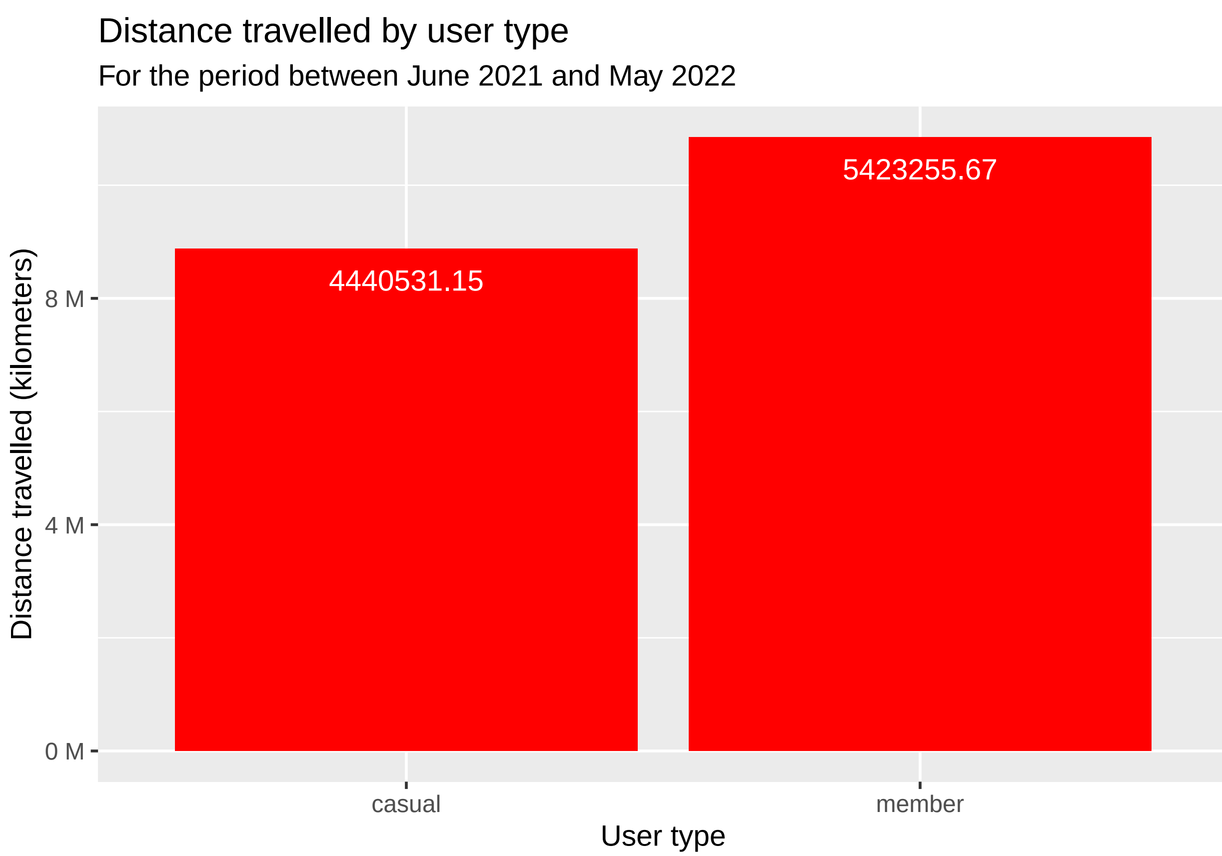


Total distance (in kilometers) traveled by user type (please note the assumption and limitation)

The annual members traveled more kilometers than the casual riders. When computing this distance between the starting and ending bike station (by long_lat coordinates), it is assumed that the riders traveled from one point to another without making any detours.

```
data_bar2 <- all_trips_cleaned %>%
  group_by(member_casual) %>%
  summarise(distance_km=sum(distance_km, na.rm=TRUE))

ggplot(data_bar2, aes(x=member_casual, y=distance_km)) +
  geom_bar(stat= "identity", fill= "red") +
  labs(
    title = "Distance travelled by user type",
    subtitle = "For the period between June 2021 and May 2022",
    x = "User type",
    y = "Distance travelled (kilometers)") +
  scale_y_continuous(labels = label_number(suffix = " M", scale = 2e-6)) +
  geom_text(aes(label=round(stat(y),2)), vjust=+2, color="white")
```



Although annual members traveled longer total distance, the average distance traveled between annual members and casual riders is roughly the same.

```
data_bar2.1 <- all_trips_cleaned %>%
  group_by(member_casual) %>%
  summarise(average_distance_km=mean(distance_km, na.rm=TRUE))
print(data_bar2.1)
```

```
## # A tibble: 2 x 2
##   member_casual average_distance_km
##   <chr>                       <dbl>
## 1 casual                       2.20
## 2 member                       2.05
```

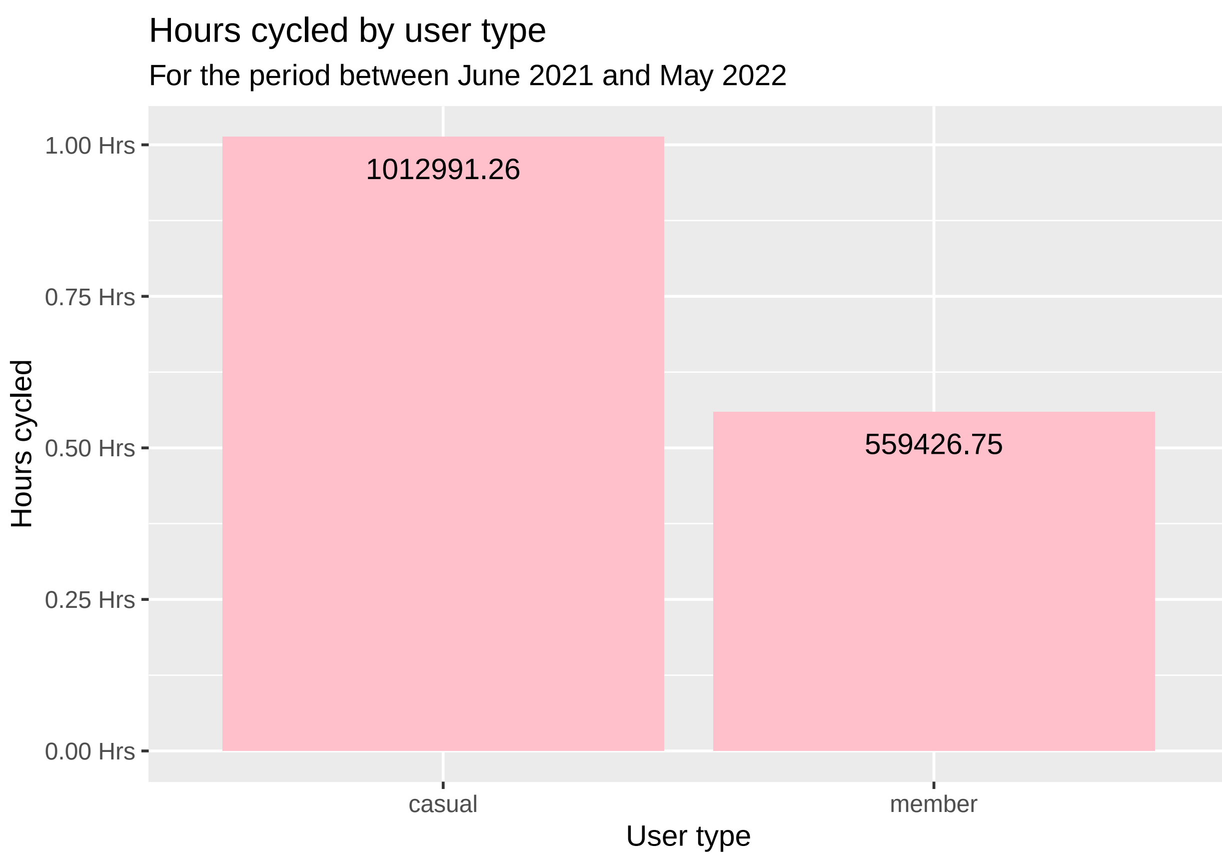
Hours cycled by user type

Annual members cycled less hours than casual riders.

```
data_bar3 <- all_trips_cleaned %>%
  group_by(member_casual) %>%
  summarise(time=sum(time_difference_hours, na.rm=TRUE))

ggplot(data_bar3, aes(x=member_casual, y=time)) +
  geom_bar(stat = "identity", fill = "pink") +
  labs(
    title = "Hours cycled by user type",
    subtitle = "For the period between June 2021 and May 2022",
    x = "User type",
    y = "Hours cycled") +
  scale_y_continuous(labels = label_number(suffix = " Hrs", scale = 1e-6)) +
  geom_text(aes(label=round(time, 2)), vjust=+2, color="black")
```

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



On average, annual members cycled 0.3.8 hours less than casual riders.

```r
data_bar3.1 <- all_trips_cleaned %>%
  group_by(member_casual) %>%
  summarise(average_time=mean(time_difference_hours, na.rm=TRUE))
print(data_bar3.1)
```

```
## # A tibble: 2 x 2
##   member_casual average_time
##   <chr>         <drtn>
## 1 casual        0.5016716 hours
## 2 member        0.2112586 hours
```

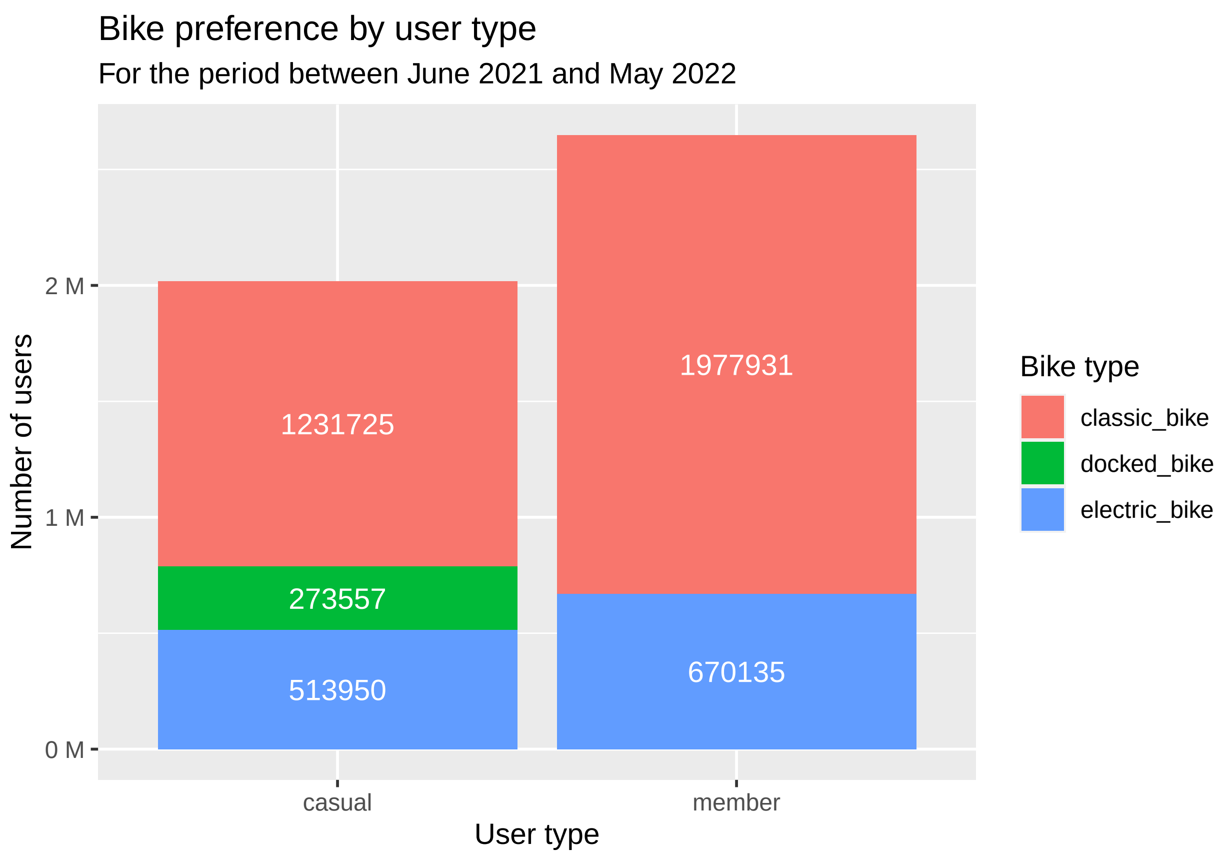
Bike preference by user type

Classic bike is the most preferred bike type among both casual riders and annual members.

```r
data_bar4 <- all_trips_cleaned %>%
  group_by(member_casual, rideable_type) %>%
  summarise(count_of = n())
```

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

```r
ggplot(data_bar4, aes(x=member_casual, y=count_of, fill = rideable_type)) +
   geom_bar(stat="identity") +
  labs(
    title = "Bike preference by user type",
    subtitle = "For the period between June 2021 and May 2022",
    fill = "Bike type",
    x = "User type",
    y = "Number of users") +
  geom_text(aes(label=count_of), position = position_stack(vjust = .5), color="white") +
  scale_y_continuous(labels = label_number(suffix = " M", scale = 1e-6))
```



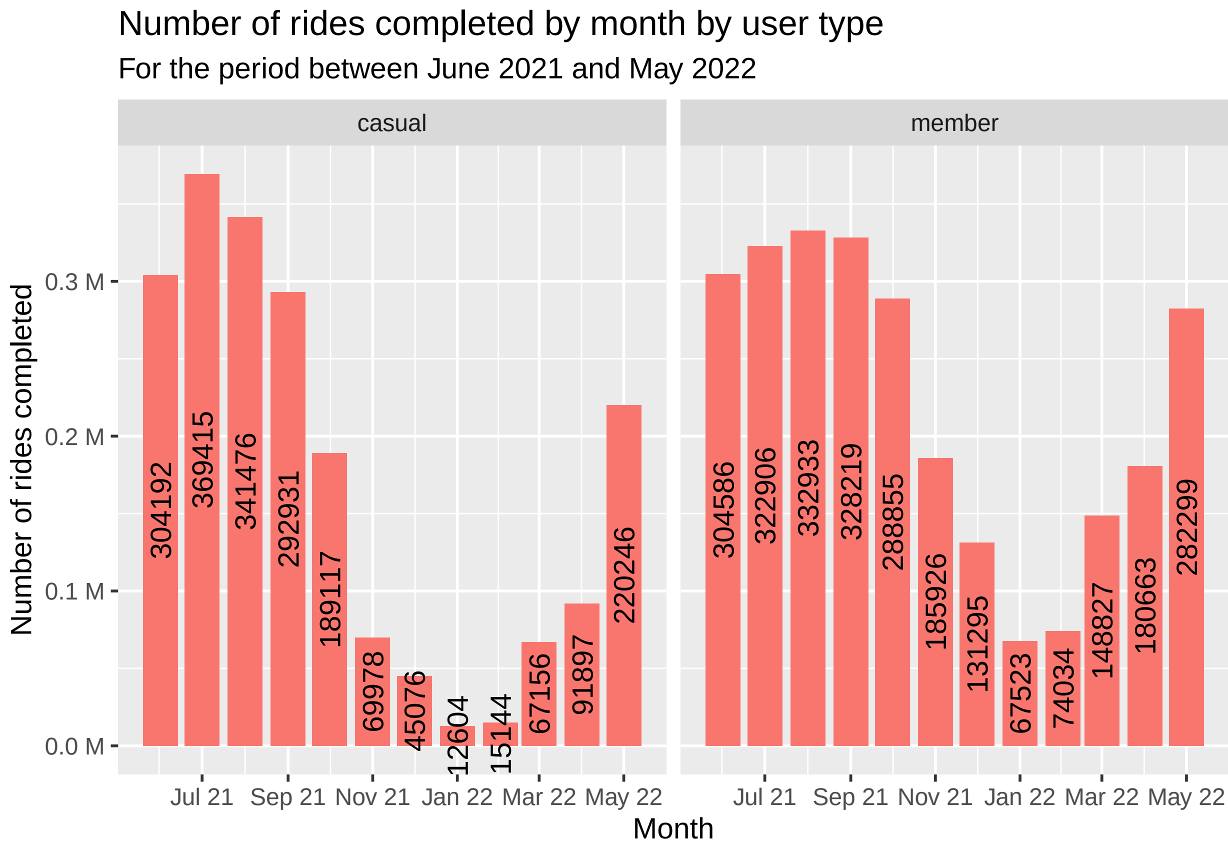
Number of rides completed by month by user type

The **summer period** (around June to September) saw **an increase** in rides completed, while the **winter period** (around Nov to February) saw a **marked reduction** in rides completed. This trend is similar in both user type, likely because people are less likely to go out in the winter.

```
data_bar5 <- mutate(all_trips_cleaned, start_month_year = floor_date(as_date(started_at), "month")) %>%
  group_by(start_month_year, member_casual) %>%
  summarise(count_of = n())
```

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

```
ggplot(data_bar5, aes(x=start_month_year, y=count_of, fill="orange"))+
  geom_bar(stat="identity") +
  facet_wrap(~member_casual)+
  labs(
    title = "Number of rides completed by month by user type",
    subtitle = "For the period between June 2021 and May 2022",
    x = "Month",
    y = "Number of rides completed") +
  geom_text(aes(label=count_of), position = position_stack(vjust = .5), color="black", angle = 90) +
  scale_y_continuous(labels = label_number(suffix = " M", scale = 1e-6)) +
  scale_x_date(date_labels = "%b %y", date_breaks = "2 month") +
  theme(legend.position = "none")
```



Number of rides completed by day by user type

Among the casual riders, there is a visible **increase of 73.7%** in bike rentals on **weekends** . This suggests that the casual riders used the bikes for leisure purposes predominantly. This finding corroborates the finding described in the next section.

On the contrary, there is a slight **decrease of 11.3%** in bike rentals on **weekends** among the annual members. This suggests that the annual members used the bikes for non-leisure or work purposes predominantly. This finding corroborates the finding described in the next section.

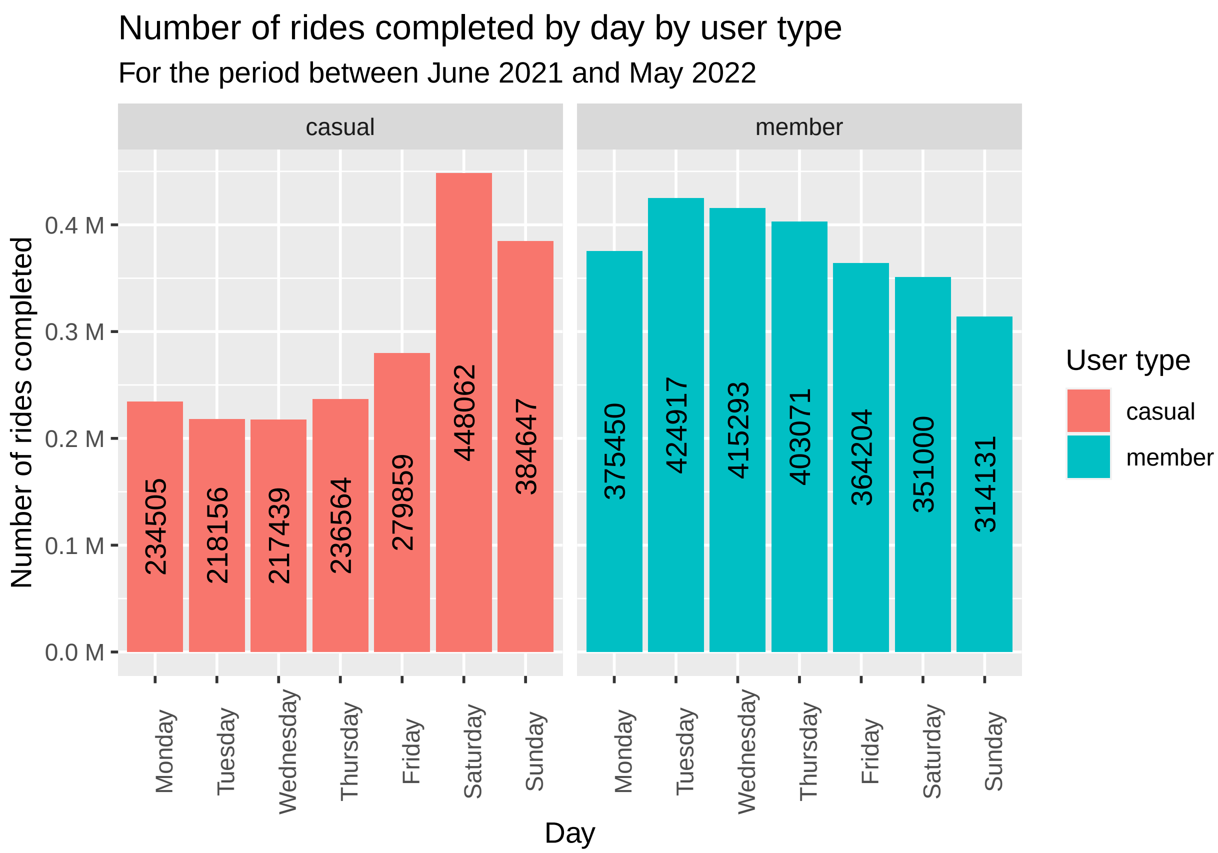
```
data_bar6 <- mutate(all_trips_cleaned, start_day = weekdays(started_at)) %>%
  group_by(start_day, member_casual) %>%
  summarise(count_of = n())
```

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

```
level_order <-c('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')

ggplot(data_bar6, aes(x=factor(start_day, level = level_order), y=count_of, fill=member_casual))+
  geom_bar(stat="identity") +
  facet_wrap(~member_casual) +
  labs(
    title = "Number of rides completed by day by user type",
    subtitle = "For the period between June 2021 and May 2022",
    x = "Day",
    y = "Number of rides completed",
    fill = "User type") +
```

```
  geom_text(aes(label=count_of), position = position_stack(vjust = .5), color="black", angle = 90) +
  scale_y_continuous(labels = label_number(suffix = " M", scale = 1e-6)) +
  theme(axis.text.x=element_text(angle = 90))
```



Top 5 start stations by user types

For the casual riders, the top five start stations are largely located near **places of attractions** . This suggests that the casual riders rented bikes to tour around the attractions.

For annual members, the top five start stations are largely located near **residential areas** . This suggests that the annual members rented bikes for non-leisure or work purposes.

```
table1 <- all_trips_cleaned %>%
  group_by(member_casual, start_station_name) %>%
  summarise(count_of=n()) %>%
  arrange(desc(count_of)) %>%
  na.omit(start_station_name)
```

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

```
  ## Table 1.1 - By casual riders ##
table1.1 <- filter(table1, member_casual =="casual") %>%
  rename(number_of_trips = count_of) %>%
  slice(1:5)

  #Table 1.2 - By members ##
table1.2 <- filter(table1, member_casual =="member") %>%
```

```r
  rename(number_of_trips = count_of) %>%
  slice(1:5)

print(table1.1)
```

```
## # A tibble: 5 x 3
## # Groups:   member_casual [1]
##   member_casual start_station_name                number_of_trips
##   <chr>         <chr>                                       <int>
## 1 casual        Streeter Dr & Grand Ave                     65322
## 2 casual        Millennium Park                             30416
## 3 casual        Michigan Ave & Oak St                       28035
## 4 casual        DuSable Lake Shore Dr & Monroe St           23294
## 5 casual        Shedd Aquarium                              22079
```

```r
print(table1.2)
```

```
## # A tibble: 5 x 3
## # Groups:   member_casual [1]
##   member_casual start_station_name       number_of_trips
##   <chr>         <chr>                              <int>
## 1 member        Kingsbury St & Kinzie St           25066
## 2 member        Clark St & Elm St                  23893
## 3 member        Wells St & Concord Ln              23148
## 4 member        Wells St & Elm St                  20239
## 5 member        Clinton St & Madison St            18573
```

Top 5 end stations by user types

The top five end stations are **similar** to the top five start stations as described earlier.

```r
table2 <- all_trips_cleaned %>%
  group_by(member_casual, end_station_name) %>%
  summarise(count_of=n()) %>%
  arrange(desc(count_of)) %>%
  na.omit(end_station_name)
```

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

```r
  ## Table 2.1 - By casual riders ##
table2.1 <- filter(table2, member_casual =="casual") %>%
  rename(number_of_trips = count_of) %>%
  slice(1:5)

  #Table 2.2 - By members ##
table2.2 <- filter(table2, member_casual =="member") %>%
  rename(number_of_trips = count_of) %>%
  slice(1:5)

print(table2.1)
```

```
## # A tibble: 5 x 3
## # Groups:   member_casual [1]
##   member_casual end_station_name                    number_of_trips
##   <chr>         <chr>                                         <int>
## 1 casual        Streeter Dr & Grand Ave                       68239
```

```
## 2 casual        Millennium Park                                 31877
## 3 casual        Michigan Ave & Oak St                           29790
## 4 casual        DuSable Lake Shore Dr & North Blvd              21907
## 5 casual        DuSable Lake Shore Dr & Monroe St               21699
```

```
print(table2.2)
```

```
## # A tibble: 5 x 3
## # Groups:   member_casual [1]
##   member_casual end_station_name         number_of_trips
##   <chr>         <chr>                              <int>
## 1 member        Kingsbury St & Kinzie St           25057
## 2 member        Clark St & Elm St                  23785
## 3 member        Wells St & Concord Ln              23658
## 4 member        Wells St & Elm St                  20728
## 5 member        Clinton St & Madison St            19005
```

SHARE

Recommendations

Three key recommendations are proposed to convert casual riders to annual members:

- Identify casual riders who typically start and end their ride sessions near residential areas and offer incentives (e.g. discounts, lucky draws, etc) to convert them into annual members.
- Offer an annual membership for weekday rental to increase number of annual members.
- Conduct a market survey with the casual riders, asking if they would purchase an annual package if there is a bike station near their residence. If the response is positive, consider identifying key residential areas to build new bike stations.

Two other recommendations are proposed to increase sales:

- Partner with tour operators to offer a combination package (e.g. attraction pass + bike pass) to increase number of casual riders.
- Increase marketing campaigns during the summer season.

Further exploration

- Collect data to assess whether there are sufficient bikes to cater to the demand in the top five bike stations
- Conduct marketing promotions and collect price vs demand data to determine the appropriate price point.

Back to the three questions:

1. How do annual members and casual riders use Cyclistic bikes differently?

Casual users prefer bike rides during the weekends, and mostly for leisure. They ride for a longer time and slightly longer distances than annual members. Annual members mostly use bikes for commutes. The use of bike rides peaks during summer months.

2. Why would casual riders buy Cyclistic annual memberships?

Casual riders would buy the annual memberships if there was a discount for longer rides or for weekends, or seasonal (summer) discount.

3. How can Cyclistic use digital media to influence casual riders to become members?

Cyclistic can create ads on social media in the areas surrounding the most touristic places (most popular destinations). It could also collaborate with touristic sites to provide discount or free entrance tickets to casual users converting into annual members.