Divvy Bikes Case Study

marty

2025-02-11

Introduction

The following analysis is part of a case study about a (fictional) bike-share company named Cyclistic. In line with their marketing strategy, this analysis aims to answer the question: **How do casual riders use Cyclistic's bikes differently from annual members?**

Ultimately, answering this question will help move the needle to the company's main problem: converting casual riders (also known as "Customers") to subscribers.

Throughout the study, I will be using data from Q1 of years 2019 and 2020. Both data sets contain over 300,000 observations of historical rider data (e.g. station names, ride length, coordinates) which, when manipulated, can hopefully reveal patterns that will help answer the question at hand. The data (https://divvy-tripdata.s3.amazonaws.com/index.html) is provided by Google under the following (license) [https://divvybikes.com/data-license-agreement (https://divvybikes.com/data-license-agreement)].

Setting Up the Environment

To begin, I installed the following packages to guide my analysis:

- **tidyverse**: a core collection of R packages used for data analysis
- **geosphere**: a package containing functions that can calculate the distance between two places given their longitude and latitude coordinates

```
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/marti/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\marti\AppData\Local\Temp\RtmpWqPiMN\downloaded_packages
```

```
install.packages("geosphere", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/marti/AppData/Local/R/win-library/4.4'
## (as 'lib' is unspecified)
```

```
## package 'geosphere' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\marti\AppData\Local\Temp\RtmpWqPiMN\downloaded_packages
```

```
library("tidyverse")
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.0.4
```

```
## ── 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 becom
e errors
```

```
library("geosphere")
```

Importing the Datasets

With the tools in hand, I then imported the datasets provided by Google. They are provided as csv files, so I used the **read_csv** function on R to store them in the environment.

P.S. I also did this analysis on RStudio Desktop, as you can see by the files' locations below.

```
# Q1 2019 Dataset
q1_2019 <- read_csv("C:\\Users\\marti\\Desktop\\CYCLISTIC CASE STUDY A\\data sources\\Divvy_Trip
s_2019_Q1.csv")
```

```
## Rows: 365069 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (4): from_station_name, to_station_name, usertype, gender
## dbl  (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num  (1): tripduration
## dttm (2): start_time, end_time
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Q1 2020 Dataset
q1_2020 <- read_csv("C:\\Users\\marti\\Desktop\\CYCLISTIC CASE STUDY A\\data sources\\Divvy_Trip
s_2020_Q1.csv")
```

```
## Rows: 426887 Columns: 13
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (5): ride_id, rideable_type, start_station_name, end_station_name, memb...
## dbl  (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, e...
## dttm (2): started_at, ended_at
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Cleaning the Data

Before conducting any analysis, I first wanted to check if the data sources were ready for use. (They were not) In cleaning the data sets, the primary issues were consistency, relevance, and completeness. Essentially, both data sets seemed foreign to one another, and most of the work I did was to ensure they weren't by the time I began the analysis.

The 2019 Dataset:

Standardizing the column names of stations

I thought "starting" and "end" was more intuitive to understand than "from" and "to". Not to mention, this was also the format for the 2020 data set, so I changed the column names here.

```
# Renaming the station column names
q1_2019 <- q1_2019 %>% rename(start_station_id = from_station_id, start_station_name = from_stat
ion_name, end_station_id = to_station_id, end_station_name = to_station_name)
```

Removing the "bikeid", "gender", and "birthyear" columns:

- **bikeid**: Since each trip had a unique ID already (the column "trip_id"), this was a redundant column.
- **gender**: Considering this wasn't in the 2020 data set either, it was a variable that I decided was not going to be useful in answering the main question.
- **birthyear**: This was not in the 2020 data set either and was similarly not too relevant to cycle use.

```
# Removing the columns
q1_2019 <- q1_2019 %>% select(-bikeid,-birthyear,-gender)
```

Cleaning the "tripduration" column:

The summary of this column shows a maximum value of 10,682,400. Despite being measured in seconds, that's still signs of an outlier to me. To account for other possible outliers, I calculated the interquartile range (IQR) below to determine the maximum and minimum thresholds for this column's values:

```
# Finding the maximum value, and Q1 and Q3 values of this column
summary(q1_2019)
```

```
##     trip_id           start_time
##  Min.   :21742443   Min.   :2019-01-01 00:04:37.00
##  1st Qu.:21848765   1st Qu.:2019-01-23 05:26:54.00
##  Median :21961829   Median :2019-02-25 07:52:56.00
##  Mean   :21960872   Mean   :2019-02-19 21:43:15.42
##  3rd Qu.:22071823   3rd Qu.:2019-03-17 16:52:47.00
##  Max.   :22178528   Max.   :2019-03-31 23:53:48.00
##     end_time                      tripduration      start_station_id
##  Min.   :2019-01-01 00:11:07.00   Min.   :      61   Min.   :  2.0
##  1st Qu.:2019-01-23 05:49:40.00   1st Qu.:     326   1st Qu.: 76.0
##  Median :2019-02-25 08:03:50.00   Median :     524   Median :170.0
##  Mean   :2019-02-19 22:00:11.91   Mean   :    1016   Mean   :198.1
##  3rd Qu.:2019-03-17 17:16:16.00   3rd Qu.:     866   3rd Qu.:287.0
##  Max.   :2019-06-17 16:04:35.00   Max.   :10628400   Max.   :665.0
##  start_station_name end_station_id  end_station_name     usertype
##  Length:365069      Min.   :  2.0   Length:365069      Length:365069
##  Class :character   1st Qu.: 76.0   Class :character   Class :character
##  Mode  :character   Median :168.0   Mode  :character   Mode  :character
##                     Mean   :198.6
##                     3rd Qu.:287.0
##                     Max.   :665.0
```

IQR = Q3-Q1 = 866-326 = 540 1.5*540 = 810

The IQR of this column is 810, so values that are greater than 1676 (Q3 + 810) and values less than 0 (because a trip duration cannot be negative in value) will be considered outliers and be removed from the dataset.

```
# Removing values greater than 1676 and values less than 0
q1_2019 <- q1_2019 %>% filter(tripduration < 1676 & tripduration > 0)
```

Creating a “day_of_week” column that identifies which day a specific trip occurred

I thought this was a useful variable to use to study the distribution of trips across both user types. Since the time columns specified a date as well, I used the “weekdays” function to extract the day of that date and stored the results in a separate column. (I also do this with the 2020 data set later on)

```
# Create a vector to store the days of the week a trip occurred
day_of_week_2019 <- c(weekdays(q1_2019$start_time))

# Store the vector in the data set
q1_2019$day_of_week <- day_of_week_2019
```

Adding latitude and longitude data to the 2019 dataset to allow

the analysis of trip distances

The 2020 data set contained latitude and longitude data about its stations. Since this was in the same city, I thought I could add those coordinates to the 2019 data set as well since distance could be a relevant metric in analyzing cycle use.

To do this, I first created a table of *all the stations* around the city along with their coordinates. Below, I combined a list of all unique values of both starting and ending stations (from the 2020 data) into one centralized data frame. Think of it as a directory.

```
# Creating a table of all unique stations

# Create a list of all starting stations
start_stations <- q1_2020 %>% group_by(id = start_station_id,lng = start_lng,lat = start_lat) %
>% summarize(names=unique(start_station_name))
```

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

```
# Create a list of all ending stations
end_stations <- q1_2020 %>% group_by(id = end_station_id,lng = end_lng,lat = end_lat) %>% summar
ize(names=unique(end_station_name))
```

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

```
# Combine both data frames above into a unique list of stations
all_stations <- rbind(start_stations,end_stations)
all_stations_unique <- all_stations %>% group_by(id, lng, lat) %>% summarize(names=unique(name
s))
```

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

With my directory of stations prepared, I had something to reference in the 2019 data set. Below, I created two copies of the directory (one for each starting and ending stations), because I couldn't have two primary keys in one table.

```
# Create two (starting and ending) tables that can be referenced for coordinates

# Create the starting station table of coordinates
start_station_coordinates <- all_stations_unique %>% rename(start_lng=lng, start_lat=lat, start_
station_id=id)

# Create the ending station table of coordinates
end_station_coordinates <- all_stations_unique %>% rename(end_lng=lng, end_lat=lat, end_station_
id=id)
```

Finally, to add the coordinates, I did a left join of the 2019 table and the coordinate tables above.

```
# Join the tables above to the existing 2019 dataset for coordinates
q1_2019 <- q1_2019 %>%
# Join the start_station coordinates
left_join(start_station_coordinates %>% select(start_station_id,start_lat,start_lng)) %>%
# Join the end_station coordinates
left_join(end_station_coordinates %>% select(end_station_id,end_lat,end_lng))
```

```
## Joining with `by = join_by(start_station_id)`
## Joining with `by = join_by(end_station_id)`
```

The 2020 Dataset:

Standardizing the “usertype” column

Both datasets use different language to distinguish between casual riders and subscribers. To create consistency, I decided to use the term “usertype” and categorized them as “Customers” and “Subscribers”. To reflect that, I made the following changes to the 2020 column:

```
# Change the column name from "member_casual" to "usertype"
q1_2020 <- q1_2020 %>% rename(usertype = member_casual)

# Replace the row name "members" with "Subscriber", and "casual" with "Customers"
q1_2020$usertype[q1_2020$usertype == "member"] <- "Subscriber"
q1_2020$usertype[q1_2020$usertype == "casual"] <- "Customer"
```

Removing the “rideable_type” column

Looking at all the values of this column, I saw that there was only one type of bike (“docked_bike”). I decided this wasn’t going to be useful in the analysis, so I dropped this variable entirely.

```
# Identify the unique ride types in the column
q1_2020 %>% count(rideable_type)
```

```
## # A tibble: 1 × 2
##   rideable_type      n
##   <chr>          <int>
## 1 docked_bike   426887
```

```
# Remove the column from the data frame
q1_2020 <- q1_2020 %>% select(-rideable_type)
```

Renaming the time columns

This goes back to the 2019 data set where it used “start_time” and “end_time” to indicate the time. To standardize formatting, I renamed the 2020 columns below.

```
q1_2020 <- q1_2020 %>% rename(start_time=started_at,end_time=ended_at)
```

Adding a “tripduration” column to determine the length of a trip in seconds

“tripduration” is a column in the 2019 data set calculated by subtracting both the end and starting times of a trip. This is relevant in understanding how different riders use the cycles, so I added it to the 2020 data set as well.

```
# Create a vector to store the calculated trip duration in the numeric data type
tripduration <- as.numeric(c(q1_2020$end_time-q1_2020$start_time))

# Store the vector to the 2020 dataset
q1_2020$tripduration <- tripduration
```

Cleaning the “tripduration” column

In plotting the distribution of data, I found that there were several points beyond the median value. To clean the dataset, I again determined outliers using the interquartile range (IQR) calculated below:

IQR = Q3-Q1 = 949-329 = 640 1.5*IQR = 930

With an IQR of 930, values that are greater than 1879 (Q3 + 930) and values less than 0 (because a trip duration cannot be negative in value), will be considered outliers and be removed from the dataset.

```
q1_2020 <- q1_2020 %>% filter(tripduration < 1879 & tripduration > 0)
```

Creating a “day of week” column that identifies which day a specific trip occurred

```
# Create a vector to store the days of the week a trip occurred
day_of_week_2020 <- c(weekdays(q1_2020$start_time))

# Store each vector to their respective data sets
q1_2020$day_of_week <- day_of_week_2020
```

Analyzing the Data

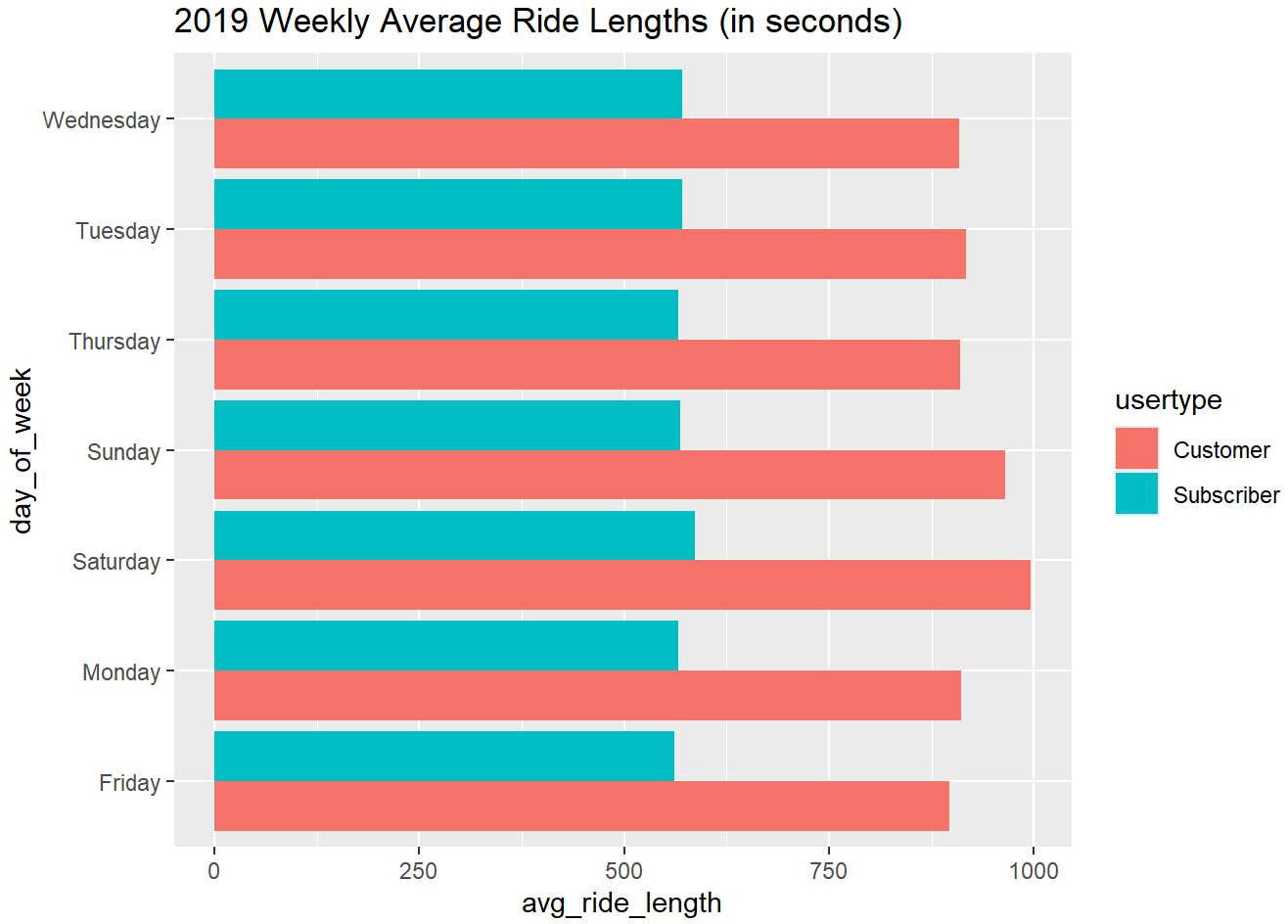
Comparing average trip duration

Starting off with trip duration, I took a quick glance at the average ride length between a customer and a subscriber. Surprisingly, **For both 2019 & 2020, the average ride length for customers was longer than subscribers.** Customers spent roughly 6.5 minutes (400 seconds) more on cycles than subscribers. This is surprising because I assume that customers are also traveling more. However, the average distances between both user types don’t vary too much either.

Customers are traveling the same distance on bikes, but are spending more time on it than subscribers. **This leads me to believe that the average customer likes to take their time with these cycles and are using it for more leisurely activities.**

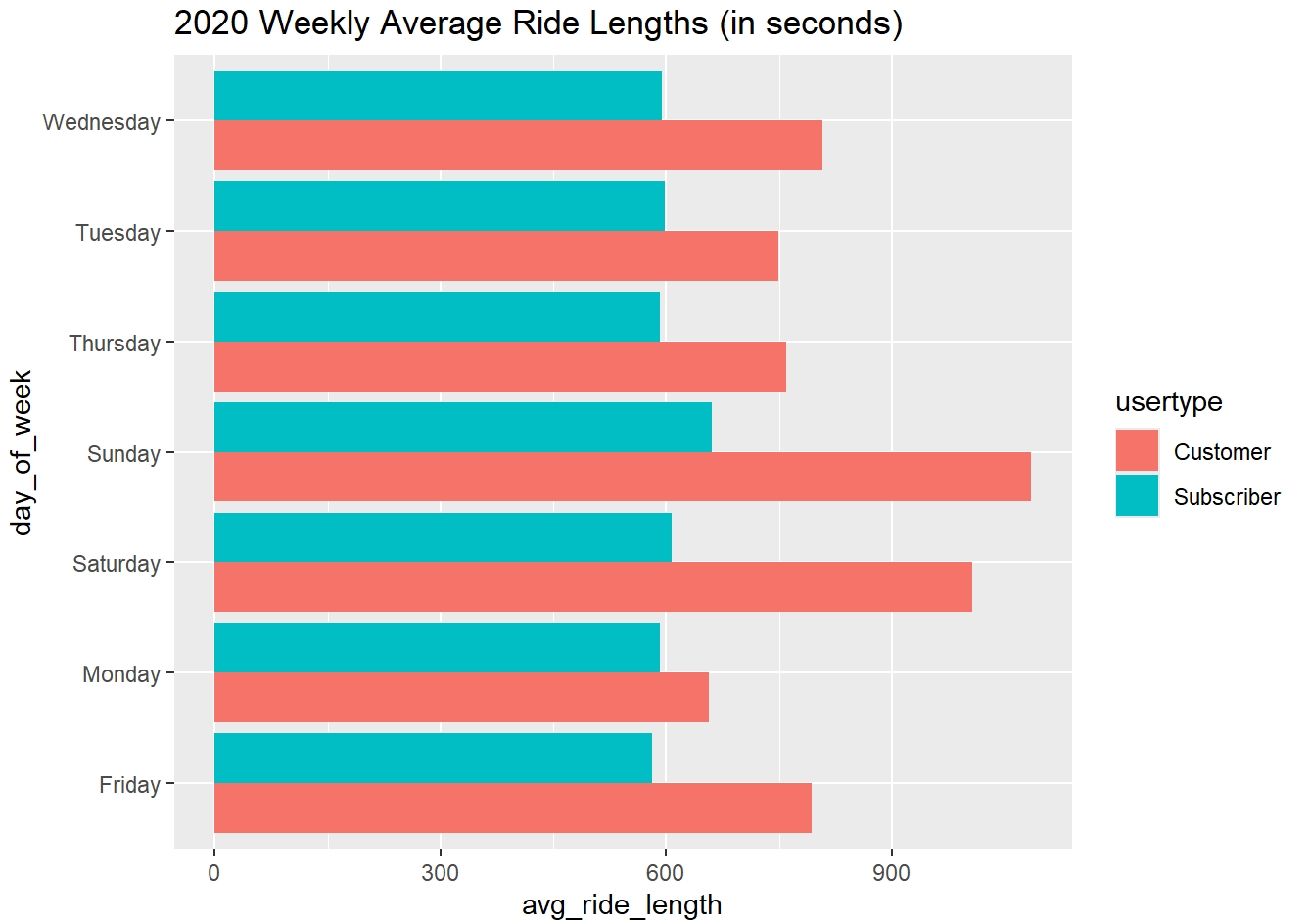
```
# 2019 Weekly Average Ride Lengths (in seconds)
q1_2019 %>% group_by(day_of_week, usertype) %>% summarize(avg_ride_length=mean(tripduration)) %
>% ggplot(aes(x=avg_ride_length,y=day_of_week,fill=usertype)) + geom_bar(stat="identity",positio
n="dodge") + labs(title="2019 Weekly Average Ride Lengths (in seconds)")
```

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




```
# 2020 Weekly Average Ride Lengths (in seconds)
q1_2020 %>% group_by(day_of_week, usertype) %>% summarize(avg_ride_length=mean(tripduration)) %
>% ggplot(aes(x=avg_ride_length,y=day_of_week,fill=usertype)) + geom_bar(stat="identity",positio
n="dodge") + labs(title="2020 Weekly Average Ride Lengths (in seconds)")
```

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



Comparing trip distances

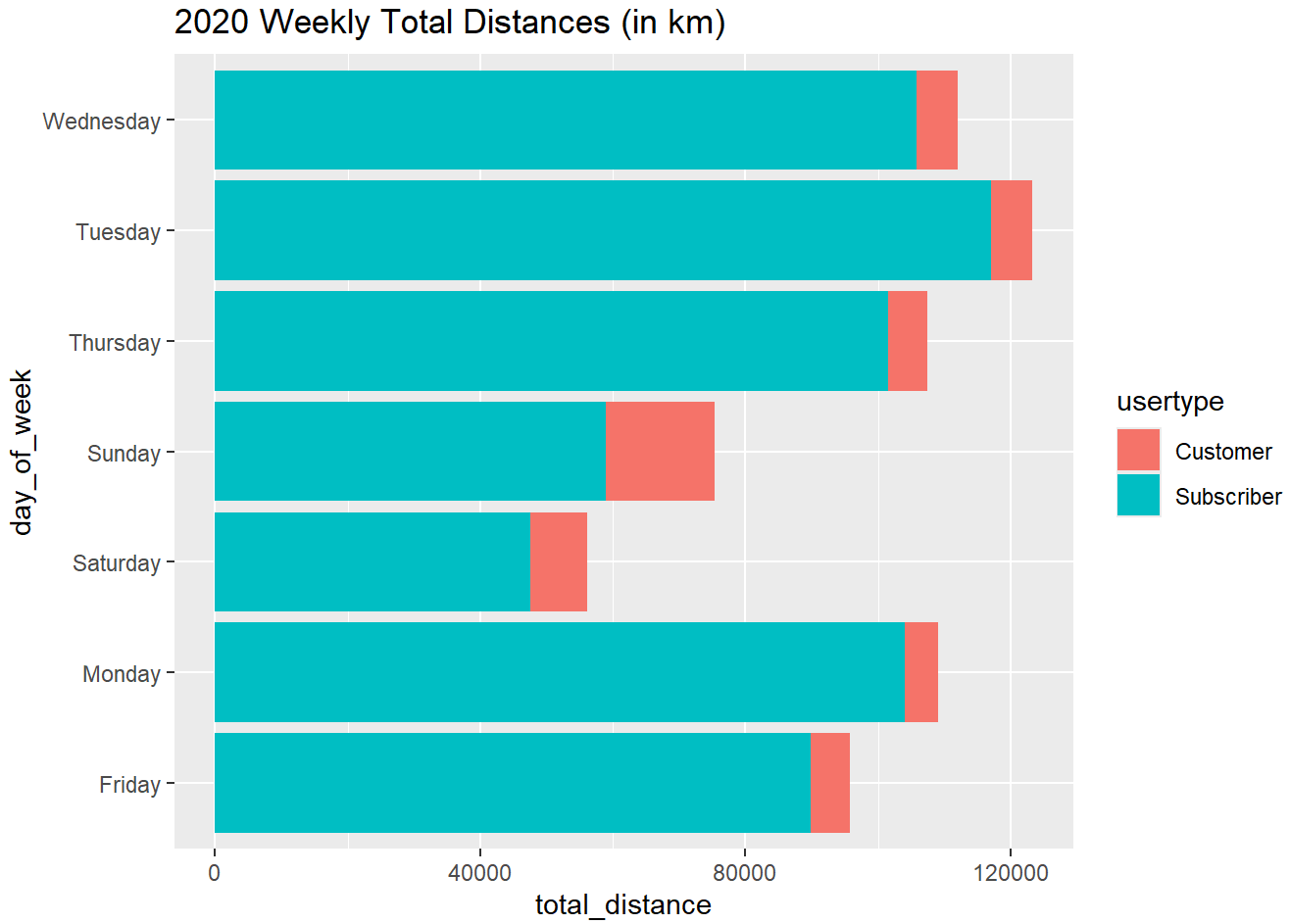
Given the longitude and latitude coordinates of both starting and ending stations, we can calculate the geospatial distances between them using the Haversine formula (http://www.movable-type.co.uk/scripts/latlong.html). With the “distHaversine” function in the “geosphere” library, we can create a new column to calculate the distance covered in any given trip:

```
# Adding the distance column for both datasets
q1_2020 <- q1_2020 %>% mutate(distance = distHaversine(cbind(start_lng,start_lat),cbind(end_lng,
end_lat))/1000)
q1_2019 <- q1_2019 %>% mutate(distance = distHaversine(cbind(start_lng,start_lat),cbind(end_lng,
end_lat))/1000)
```

If we look at the total distances covered by customers and subscribers, we can learn more about their *travel patterns.* As seen in the plots below, customers cover the most ground only during the weekends while subscribers constantly travel throughout the whole week. This goes to say that *customers are only using bikes on an as needed basis and subscribers are using it consistently.**

```
# Analyzing the weekly total distance covered between customers and subscribers (2020)
q1_2020 %>% group_by(day_of_week,usertype) %>% summarize(total_distance=sum(distance)) %>% ggplo
t(aes(x=total_distance,y=day_of_week,fill=usertype)) + geom_bar(stat="identity") + labs(title="2
020 Weekly Total Distances (in km)")
```

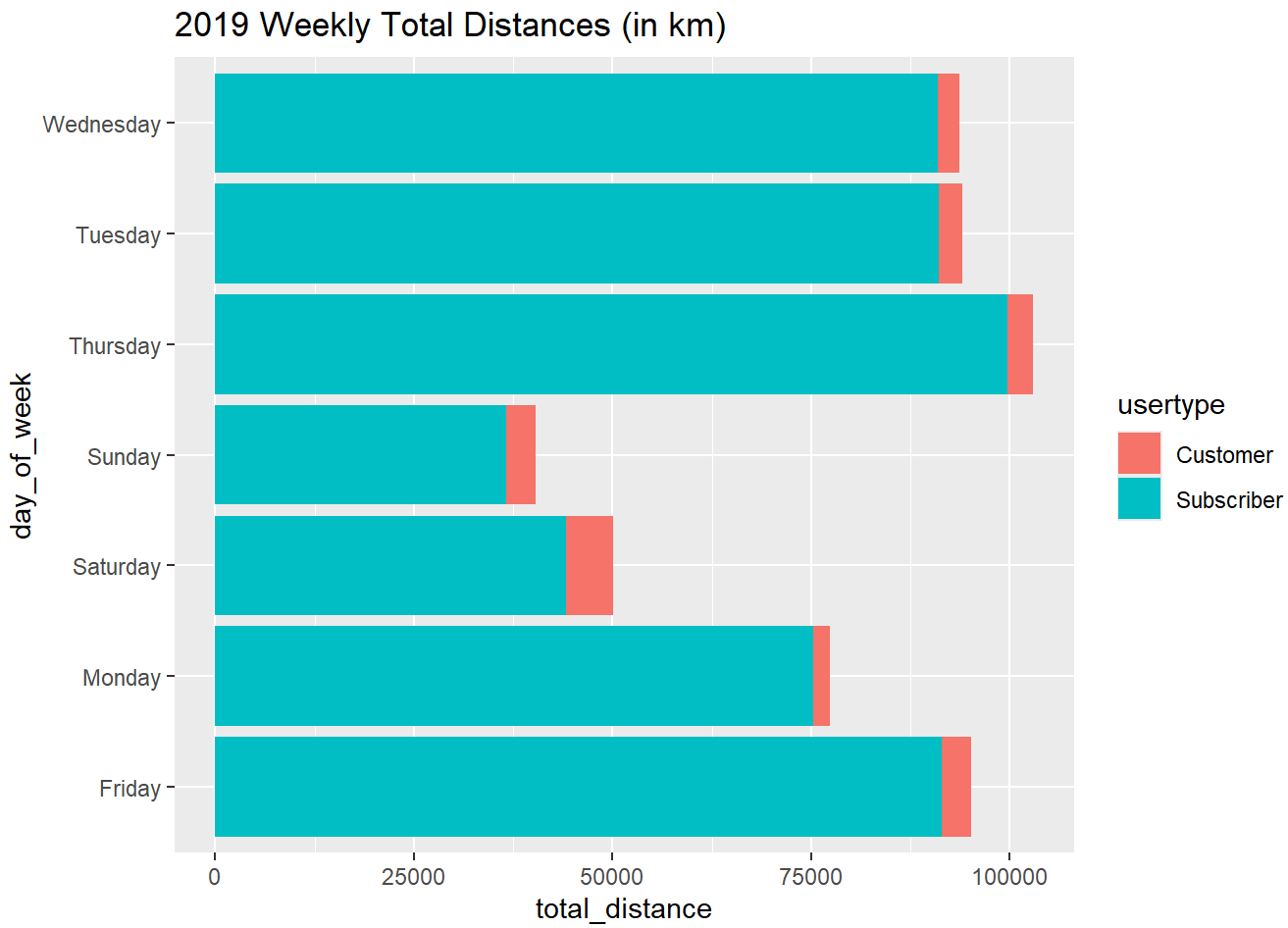
```
## `summarise()` has grouped output by 'day_of_week'. You can override using the
## `.groups` argument.
```




```
# 2019 Plot
q1_2019 <- q1_2019 %>% na.omit(distance)

q1_2019 %>% group_by(day_of_week,usertype) %>% summarize(total_distance=sum(distance)) %>% ggplo
t(aes(x=total_distance,y=day_of_week,fill=usertype)) + geom_bar(stat="identity") + labs(title="2
019 Weekly Total Distances (in km)")
```

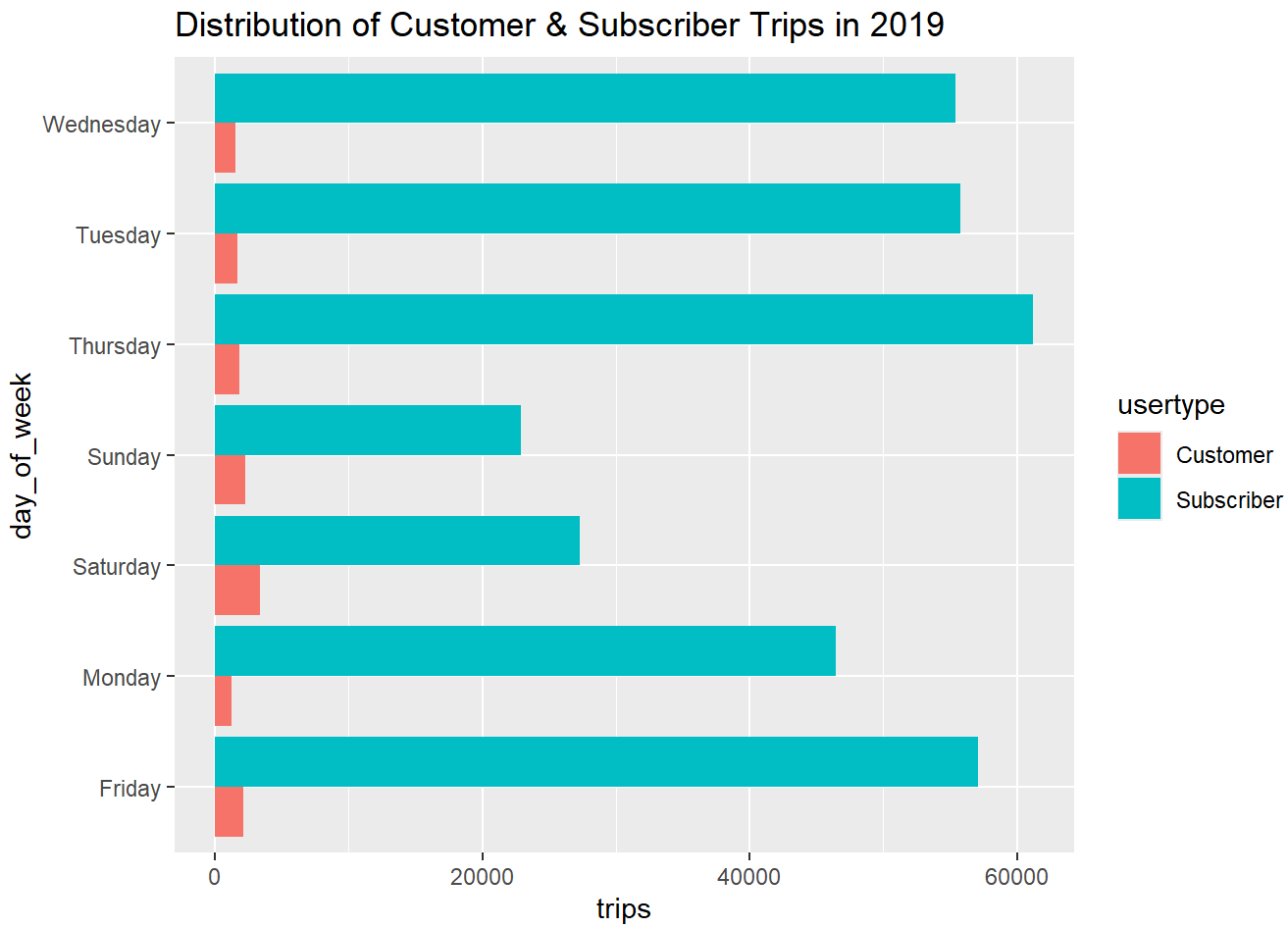
```
## `summarise()` has grouped output by 'day_of_week'. You can override using the
## `.groups` argument.
```



Comparing the distribution of trips throughout any given week

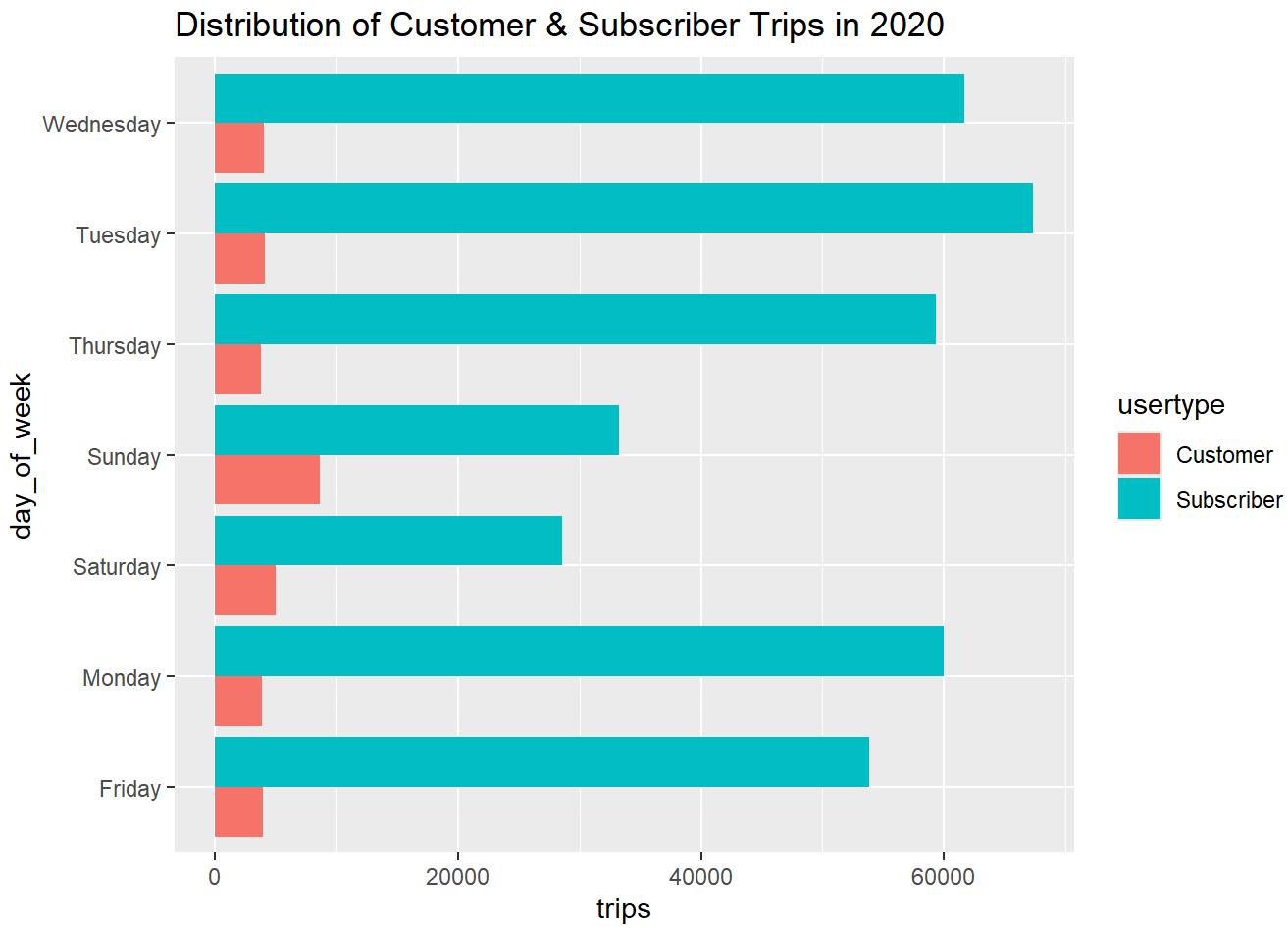
```
q1_2019 %>% group_by(day_of_week,usertype) %>% summarize(trips=n()) %>% ggplot(aes(x=trips,y=day
_of_week,fill=usertype)) + geom_bar(stat="identity",position="dodge") + labs(title="Distribution
of Customer & Subscriber Trips in 2019")
```

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




```
q1_2020 %>% group_by(day_of_week,usertype) %>% summarize(trips=n()) %>% ggplot(aes(x=trips,y=day
_of_week,fill=usertype)) + geom_bar(stat="identity",position="dodge") + labs(title="Distribution
of Customer & Subscriber Trips in 2020")
```

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



Looking at both plots, we can conclude that trips are not uniform between customers and subscribers. **Customers on one hand make more trips during the weekends. Meanwhile, subscribers tend to book more trips during the weekdays.**

Comparing the most booked stations

Looking at the most frequented stations between customers and subscribers can tell us more about *why* they use the bikes.

For customers at least, 6 of the top 10 most booked stations are public places (e.g. aquariums, theatres, museums, planetariums, etc.). This supports my hypothesis above that casual riders tend to book bikes for recreational use.

```
#The top 10 starting stations for customers in both years
custstation2020 <- q1_2020 %>% filter(usertype == "Customer") %>% group_by(start_station_name) %
>% summarize(trips=n()) %>% arrange(desc(trips)) %>% top_n(10)
```

```
## Selecting by trips
```

```
custstation2019 <- q1_2019 %>% filter(usertype == "Customer") %>% group_by(start_station_name) %
>% summarize(trips=n()) %>% arrange(desc(trips)) %>% top_n(10)
```

```
## Selecting by trips
```

```
custstations <- rbind(custstation2019,custstation2020) %>% group_by(start_station_name) %>% summ
arize_all(.,sum) %>% arrange(desc(trips))
head(custstations,n=10)
```

```
## # A tibble: 10 × 2
##    start_station_name           trips
##    <chr>                        <int>
##  1 HQ QR                         3556
##  2 Lake Shore Dr & Monroe St     1668
##  3 Streeter Dr & Grand Ave       1538
##  4 Shedd Aquarium                1253
##  5 Millennium Park                759
##  6 Dusable Harbor                 548
##  7 Adler Planetarium              521
##  8 Michigan Ave & Oak St          514
##  9 Michigan Ave & Washington St   434
## 10 Theater on the Lake            344
```

On the other hand, the top stations for subscribers all fall in between streets of the city. (i.g. Clinton & Madison, Canal & Adams) Typically, these are addresses for offices and residences. In other words, subscribers tend to book bikes for more essential uses.

This leads me to believe that **customers tend to use bikes for recreational purposes (touring, lounging, etc.), while subscribers mostly use them for essential activities (going to work, commuting, etc.).**

```
#The top 10 starting stations for subscribers in both years
substation2020 <- q1_2020 %>% filter(usertype == "Subscriber") %>% group_by(start_station_name)
%>% summarize(trips=n()) %>% arrange(desc(trips)) %>% top_n(10)
```

```
## Selecting by trips
```

```
substation2019 <- q1_2020 %>% filter(usertype == "Subscriber") %>% group_by(start_station_name)
%>% summarize(trips=n()) %>% arrange(desc(trips)) %>% top_n(10)
```

```
## Selecting by trips
```

```
substations <- rbind(substation2019,substation2020) %>% group_by(start_station_name) %>% summari
ze_all(.,sum) %>% arrange(desc(trips))
head(substations,n=10)
```

```
## # A tibble: 10 × 2
##    start_station_name           trips
##    <chr>                        <int>
##  1 Canal St & Adams St          14896
##  2 Clinton St & Madison St      12898
##  3 Clinton St & Washington Blvd 11508
##  4 Kingsbury St & Kinzie St      8884
##  5 Columbus Dr & Randolph St     8004
##  6 Canal St & Madison St         6912
##  7 Franklin St & Monroe St       6822
##  8 Clinton St & Lake St          6668
##  9 Larrabee St & Kingsbury St    6644
## 10 Michigan Ave & Washington St  5974
```