Case Study: Divvy Bike-Share

Travis Allen Parker

2021-07-05

Introduction

This case study considers a hypothetical business task for Divvy, a bicycle ride-share program in Chicago. Divvy maintains 5,824 GPS-tracked bicycles and 692 parking stations across the city, offering single-ride passes, full-day passes, and annual memberships.

In this scenario, prior financial analysis has found membership customers generate significantly more revenue than casual users. The company is therefore considering a shift in strategy from broad awareness towards encouraging existing casual users to become members. This would need to be approved by the executive team, who require convincing details to justify such a change.

Business Task

Using the most recent 12 months of available ride tracking data, identify how casual riders (single-ride pass and full-day pass buyers) and annual members differ in their use of the service. Present the results to the executive committee, and offer suggestions on how to encourage casual riders to become members.

Data Sourcing and Security

The data for this case study is provided by Motivate International Inc. under an open license.

- Divvy Data License Agreement (https://www.divvybikes.com/data-license-agreement)
- Data Files Directory (https://divvy-tripdata.s3.amazonaws.com/index.html)

The data consists of comma-separated value (CSV) files, each representing one month of ride data. We select most recent 12 months of data available (June 2020 through May 2021).

To the best of our knowledge this data is reliable, as it is being provided by the original collector of the data. It is also fairly recent, so our analysis should be meaningful. We will be on guard against potential bias in the data, but at this point there are no concerns.

Cleaning and Manipulation

Since these files are too large to open in a spreadsheet program, we are using R (R-Studio) to manipulate and clean the data.

To begin we install and load packages. The working directory is also set, to make file calls shorter. If running this analysis on your own machine, remove the comment marks in front of the installation commands for any packages you have not yet installed, and change your working directory path appropriately.

```
#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("geosphere")
library(tidyverse) # Many functions; includes dypler, stringr, and ggplot2
library(lubridate) # Useful date functions
library(geosphere) # Will allow us to examine GPS coordinate data

# Change the working directory
setwd("C:/Users/sheri/Documents/R")
```

Download the CSV files into a folder (here named `case-study-divvy-data`) then load the files into data frames for manipulation.

```
df_2020_06 <- read_csv("case-study-divvy-data/202006-divvy-tripdata.csv")
df_2020_07 <- read_csv("case-study-divvy-data/202007-divvy-tripdata.csv")
df_2020_08 <- read_csv("case-study-divvy-data/202008-divvy-tripdata.csv")
df_2020_09 <- read_csv("case-study-divvy-data/202009-divvy-tripdata.csv")
df_2020_10 <- read_csv("case-study-divvy-data/202010-divvy-tripdata.csv")
df_2020_11 <- read_csv("case-study-divvy-data/202011-divvy-tripdata.csv")
df_2020_12 <- read_csv("case-study-divvy-data/202012-divvy-tripdata.csv")
df_2021_01 <- read_csv("case-study-divvy-data/202101-divvy-tripdata.csv")
df_2021_02 <- read_csv("case-study-divvy-data/202102-divvy-tripdata.csv")
df_2021_03 <- read_csv("case-study-divvy-data/202103-divvy-tripdata.csv")
df_2021_04 <- read_csv("case-study-divvy-data/202104-divvy-tripdata.csv")
df_2021_05 <- read_csv("case-study-divvy-data/202105-divvy-tripdata.csv")
```

Before combining all of the data into a single data frame (incidentally, a UNION in SQL) first ensure the column names and data types all match. The `colnames()` and structure `str()` functions allow examination of all 12 data frames; for simplicity only representative results are displayed here (Nov 2020 and Dec 2020).

```
colnames(df_2020_11)
```

```
##  [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"
```

```
colnames(df_2020_12)
```

```
##  [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"
```

```
str(df_2020_11)
```

```
## spec_tbl_df [259,716 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:259716] "BD0A6FF6FFF9B921" "96A7A7A4BDE4F82D" "C61526D06582BDC
5" "E533E89C32080B9E" ...
##  $ rideable_type     : chr [1:259716] "electric_bike" "electric_bike" "electric_bike" "electr
ic_bike" ...
##  $ started_at        : POSIXct[1:259716], format: "2020-11-01 13:36:00" "2020-11-01 10:03:26"
...
##  $ ended_at          : POSIXct[1:259716], format: "2020-11-01 13:45:40" "2020-11-01 10:14:45"
...
##  $ start_station_name: chr [1:259716] "Dearborn St & Erie St" "Franklin St & Illinois St" "La
ke Shore Dr & Monroe St" "Leavitt St & Chicago Ave" ...
##  $ start_station_id  : num [1:259716] 110 672 76 659 2 72 76 NA 58 394 ...
##  $ end_station_name  : chr [1:259716] "St. Clair St & Erie St" "Noble St & Milwaukee Ave" "Fe
deral St & Polk St" "Stave St & Armitage Ave" ...
##  $ end_station_id    : num [1:259716] 211 29 41 185 2 76 72 NA 288 273 ...
##  $ start_lat         : num [1:259716] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num [1:259716] -87.6 -87.6 -87.6 -87.7 -87.6 ...
##  $ end_lat           : num [1:259716] 41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num [1:259716] -87.6 -87.7 -87.6 -87.7 -87.6 ...
##  $ member_casual     : chr [1:259716] "casual" "casual" "casual" "casual" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_double(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_double(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
```

```
str(df_2020_12)
```

```
## spec_tbl_df [131,573 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:131573] "70B6A9A437D4C30D" "158A465D4E74C54A" "5262016E0F1F2F9
A" "BE119628E44F871E" ...
##  $ rideable_type     : chr [1:131573] "classic_bike" "electric_bike" "electric_bike" "electri
c_bike" ...
##  $ started_at        : POSIXct[1:131573], format: "2020-12-27 12:44:29" "2020-12-18 17:37:15"
...
##  $ ended_at          : POSIXct[1:131573], format: "2020-12-27 12:55:06" "2020-12-18 17:44:19"
...
##  $ start_station_name: chr [1:131573] "Aberdeen St & Jackson Blvd" NA NA NA ...
##  $ start_station_id  : chr [1:131573] "13157" NA NA NA ...
##  $ end_station_name  : chr [1:131573] "Desplaines St & Kinzie St" NA NA NA ...
##  $ end_station_id    : chr [1:131573] "TA1306000003" NA NA NA ...
##  $ start_lat         : num [1:131573] 41.9 41.9 41.9 41.9 41.8 ...
##  $ start_lng         : num [1:131573] -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num [1:131573] 41.9 41.9 41.9 41.9 41.8 ...
##  $ end_lng           : num [1:131573] -87.6 -87.7 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr [1:131573] "member" "member" "member" "member" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   started_at = col_datetime(format = ""),
##   ..   ended_at = col_datetime(format = ""),
##   ..   start_station_name = col_character(),
##   ..   start_station_id = col_character(),
##   ..   end_station_name = col_character(),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_casual = col_character()
##   .. )
```

Carefully looking through the results shows perfect matches in column names across all twelve data frames.

If some inconsistencies had appeared we could use the `rename()` function as appropriate. For example:

```
df_2020_06 <- rename(df_2020_06, ride_id = trip_id)
```

Although the column names match, the `str()` function revealed a problem with the data types for the columns `start_station_id` and `end_station_id`. These columns are type `double` in months June 2020 to November 2020, but from December 2020 onward become type `character`. It seems the company decided to change the naming convention for its station IDs, adding some letters to the beginning of each.

Before combining the data frames we should change the datatypes in these columns to match the new convention. For this task the `mutate()` function from the `dplyr` package can be used. Only the first six data frames need to be changed.

```
df_2020_06 <- df_2020_06 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
df_2020_07 <- df_2020_07 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
df_2020_08 <- df_2020_08 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
df_2020_09 <- df_2020_09 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
df_2020_10 <- df_2020_10 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
df_2020_11 <- df_2020_11 %>%
              mutate(start_station_id = as.character(start_station_id),
              end_station_id = as.character(end_station_id))
```

Checking with `str()` confirms the changes were made.

Combining to a single data frame

The data is now ready to be combined into a single data frame.

```
all_rides <- bind_rows(df_2020_06, df_2020_07, df_2020_08, df_2020_09,
                df_2020_10, df_2020_11, df_2020_12, df_2021_01, df_2021_02,
                df_2021_03, df_2021_04, df_2021_05)
```

Now that the data is together, it can be inspected more closely to see what cleaning may be required before analysis. Several general functions allow a scan for anything unusual or unexpected:

```
colnames(all_rides) # Not expecting anything new here
dim(all_rides) # Finds the dimensions (rows and columns) of the data frame
head(all_rides) # Gives a tibble of the first six rows
str(all_rides) # Not expecting to see anything new here
```

```
summary(all_rides) # Some statistical data, might alert us to some problems
```

```
##    ride_id          rideable_type        started_at
##  Length:4073561     Length:4073561     Min.   :2020-06-03 05:59:59
##  Class :character   Class :character   1st Qu.:2020-08-07 19:09:29
##  Mode  :character   Mode  :character   Median :2020-09-30 07:36:28
##                                        Mean   :2020-11-08 09:07:38
##                                        3rd Qu.:2021-03-13 10:03:09
##                                        Max.   :2021-05-31 23:59:16
##
##     ended_at                   start_station_name start_station_id
##  Min.   :2020-06-03 06:03:37   Length:4073561     Length:4073561
##  1st Qu.:2020-08-07 19:39:10   Class :character   Class :character
##  Median :2020-09-30 07:51:42   Mode  :character   Mode  :character
##  Mean   :2020-11-08 09:31:51
##  3rd Qu.:2021-03-13 10:22:00
##  Max.   :2021-06-10 22:17:11
##
##  end_station_name   end_station_id       start_lat       start_lng
##  Length:4073561     Length:4073561     Min.   :41.64   Min.   :-87.87
##  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.   :42.08   Max.   :-87.52
##
##     end_lat         end_lng       member_casual
##  Min.   :41.54   Min.   :-88.07   Length:4073561
##  1st Qu.:41.88   1st Qu.:-87.66   Class :character
##  Median :41.90   Median :-87.64   Mode  :character
##  Mean   :41.90   Mean   :-87.64
##  3rd Qu.:41.93   3rd Qu.:-87.63
##  Max.   :42.16   Max.   :-87.44
##  NA's   :5037    NA's   :5037
```

Most of the output for these functions raised no concerns, but the `summary()` function tells us columns `end_lat` and `end_lng` both have 5,037 entries listing “NA”. Curiously, the `start_lat` and `start_lng` columns have no such lacking information. Speaking with the company supplying the data, we are informed these records represent bicycles being removed or added from service, or those that were stolen or otherwise lost. We can thus remove these records.

```
all_rides <- subset(all_rides, end_lat != 'NA' & end_lng != 'NA')
```

Adding calculated values

Knowing the duration of each ride is likely to be a helpful piece of information, so we will add a new column calculating those values. We can also calculate distance traveled “as the crow flies” by comparing the start and end latitude and longitude GPS coordinates with `distGeo()` . A quick check to ensure the values are numeric reveals the length not, so their type is converted.

```
# Using the base function `difftime()` we calculate trip duration into a new column.
all_rides$ride_duration <- difftime(all_rides$ended_at, all_rides$started_at)

# Using the `distGeo()` function from `geosphere` library to calculate trip distance into a new
 column.
all_rides$ride_distance <- distGeo(matrix(c(all_rides$start_lng, all_rides$start_lat), ncol=2),
                                   matrix(c(all_rides$end_lng, all_rides$end_lat), ncol=2))

# A quick check to ensure these new values are numeric.
is.numeric(all_rides$ride_duration)
```

```
## [1] FALSE
```

```
is.numeric(all_rides$ride_distance)
```

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

```
# For some reason the length value isn't numeric, so we convert.
all_rides$ride_duration <- as.numeric(all_rides$ride_duration)
```

We also add separate columns for the day, month, year, and day of the week that each ride took place, allowing us to examine each of those factors separately.

```
# We convert the starting date/time to just the date, then use that to
# calculate separate columns for year/month/day.
all_rides$date <- as.Date(all_rides$started_at)
all_rides$year <- format(as.Date(all_rides$date), "%Y")
all_rides$month <- format(as.Date(all_rides$date), "%m")
all_rides$day <- format(as.Date(all_rides$date), "%d")
all_rides$day_of_week <- format(as.Date(all_rides$date), "%A")
```

Removing Irrelevant / Bad Data

Making another call to `summary(all_rides)` to check our progress, the minimum ride duration is negative. Our data source tells us trips with negative duration represent either errant or corrupted data, or situations where bikes were removed for maintenance. Maintenance rides, the company states, all have start stations “HQ QR”. We can therefore delete all rides with negative duration and/or which started at HQ.

We will put the data of interest into a new data frame, preserving the “deleted” data just in case.

```
# This logic checks if the start station is HQ or if the ride length is negative,
# and only returns rows where that is not the case.

rides_v2 <- subset(all_rides, start_station_name != "HQ QR" & ride_duration > 0)
```

We do a few final checks to ensure there are no hidden problems.

The first column, `ride_id` , should be unique. We check that by counting the unique entries in that column and comparing it to the number of rows overall. We also check the `rideable_type` and `member_casual` columns to ensure they only contain valid entries.

```
nrow(rides_v2)
```

```
## [1] 3855871
```

```
length(unique(rides_v2$ride_id))
```

```
## [1] 3855871
```

```
table(rides_v2$rideable_type)
```

```
##
##  classic_bike   docked_bike electric_bike
##        842250       2327390        686231
```

```
table(rides_v2$member_casual)
```

```
##
##  casual  member
## 1624836 2231035
```

All appears well, with each row having a unique ride ID, and the `ridable_type` and `member_casual` columns have only valid entries (i.e., no misspellings or invalid options).

We also check the station IDs and station names in a similar manner:

```
length(unique(rides_v2$start_station_id))
```

```
## [1] 1270
```

```
length(unique(rides_v2$start_station_name))
```

```
## [1] 715
```

```
length(unique(rides_v2$end_station_id))
```

```
## [1] 1268
```

```
length(unique(rides_v2$end_station_name))
```

```
## [1] 714
```

We see there are many more station IDs than there are unique station names. However, earlier it was discovered the station ID convention had been changed to include some letters. We can hypothesize these “extra” station IDs are actually the same stations, just their old numeric-only IDs.

Doing another search for only rides that occurred in 2021 (after the convention was changed) lends credence to this.

```
later_rides <- rides_v2[rides_v2$year == "2021", ]
length(unique(later_rides$start_station_id))
```

```
## [1] 689
```

```
length(unique(later_rides$start_station_name))
```

```
## [1] 694
```

```
length(unique(later_rides$end_station_id))
```

```
## [1] 688
```

```
length(unique(later_rides$end_station_name))
```

```
## [1] 693
```

These numbers are much more in line with each other. The slight differences could be explained by some stations being renamed, closed, or opened in the past few months. We could confirm this with the company if needed.

To fully clean the station ID column we would need to replace the old IDs with their corresponding updated ones. Depending on how the ID convention was altered it might be a relatively simple process involving the `paste()` function to add a few characters to the start of each string, or it could be quite tedious. It would depend on the precise details of the new naming convention. Such cleaning would be necessary should we wish to do meaningful analysis involving specific stations, for example analyzing the relative popularity of stations. Unfortunately, this data is not available to us.

The remaining columns in the data frame were either calculated or used in calculations, so assuming no errors occurred during this process we can be reasonably confident the data in these columns are valid.

Analysis

Now that our data is cleaned and organized, and a few calculated values have been helpfully added, we can begin some analysis to answer the business task. We begin with a summary of ride duration, and some statistical comparisons between casual riders and members.

```
summary(rides_v2$ride_duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     470     854    1619    1570 3257001
```

```
aggregate(rides_v2$ride_duration ~ rides_v2$member_casual, FUN = mean)
```

```
##   rides_v2$member_casual rides_v2$ride_duration
## 1                 casual              2577.0568
## 2                 member               921.6011
```

```
aggregate(rides_v2$ride_duration ~ rides_v2$member_casual, FUN = median)
```

```
##   rides_v2$member_casual rides_v2$ride_duration
## 1                 casual                  1239
## 2                 member                   670
```

```
aggregate(rides_v2$ride_duration ~ rides_v2$member_casual, FUN = max)
```

```
##   rides_v2$member_casual rides_v2$ride_duration
## 1                 casual                3257001
## 2                 member                2476260
```

```
aggregate(rides_v2$ride_duration ~ rides_v2$member_casual, FUN = min)
```

```
##   rides_v2$member_casual rides_v2$ride_duration
## 1                 casual                     1
## 2                 member                     1
```

The results indicate a clear tendency for casual users to ride for longer duration than members. But let's see how that compares to actual distance ridden. Dividing distance by duration also lets us compare travel speed.

```
summary(rides_v2$ride_distance)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   870.7  1686.0  2219.3  3016.4 48331.9
```

```
aggregate(rides_v2$ride_distance ~ rides_v2$member_casual, FUN = mean)
```

```
##   rides_v2$member_casual rides_v2$ride_distance
## 1                 casual               2174.571
## 2                 member               2251.837
```

```
aggregate(rides_v2$ride_distance ~ rides_v2$member_casual, FUN = median)
```

```
##   rides_v2$member_casual rides_v2$ride_distance
## 1                 casual              1660.949
## 2                 member              1700.374
```

```
aggregate(rides_v2$ride_distance ~ rides_v2$member_casual, FUN = max)
```

```
##   rides_v2$member_casual rides_v2$ride_distance
## 1                 casual              33762.61
## 2                 member              48331.86
```

```
aggregate(rides_v2$ride_distance ~ rides_v2$member_casual, FUN = min)
```

```
##   rides_v2$member_casual rides_v2$ride_distance
## 1                 casual                     0
## 2                 member                     0
```

Interestingly, the average (mean) and median distances traveled don't meaningfully differ between casual riders and members. This is expected to result in a slower average riding speed for casual users (below).

Despite the average distances being similar, it is worth nothing that members do tend to have longer maximum distances; these are likely outliers however, as the higher maximum didn't have enough weight to shift the mean.

```
summary(rides_v2$ride_distance / rides_v2$ride_duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.527   2.641   2.484   3.398 969.956
```

```
aggregate((rides_v2$ride_distance / rides_v2$ride_duration) ~ rides_v2$member_casual, FUN = mea
n)
```

```
##   rides_v2$member_casual (rides_v2$ride_distance/rides_v2$ride_duration)
## 1                 casual                                       1.920461
## 2                 member                                       2.894863
```

```
aggregate((rides_v2$ride_distance / rides_v2$ride_duration) ~ rides_v2$member_casual, FUN = medi
an)
```

```
##   rides_v2$member_casual (rides_v2$ride_distance/rides_v2$ride_duration)
## 1                 casual                                       2.006028
## 2                 member                                       2.940627
```

```
aggregate((rides_v2$ride_distance / rides_v2$ride_duration) ~ rides_v2$member_casual, FUN = max)
```

```
##   rides_v2$member_casual (rides_v2$ride_distance/rides_v2$ride_duration)
## 1                 casual                                        554.4661
## 2                 member                                        969.9559
```

```
aggregate((rides_v2$ride_distance / rides_v2$ride_duration) ~ rides_v2$member_casual, FUN = min)
```

```
##   rides_v2$member_casual (rides_v2$ride_distance/rides_v2$ride_duration)
## 1                 casual                                              0
## 2                 member                                              0
```

As we expected for having shorter average travel duration yet similar travel distances, members tend to ride faster than casual users: Approximately 50% faster on average, and in some cases up to 75% faster.

Taking all of this information together, we can hypothesize that members may be more intentional riders that stick to consistent routes, such as commuting to work or running regular errands. Alternatively, casual riders are more likely to “go on a ride”, traveling at a more leisurely rate and for a longer time.

We might see a difference in behavior depending on the day of the week. We first reorder the days of the week to be sequential rather than alphabetical, then compare ride duration.

```
rides_v2$day_of_week <- ordered(rides_v2$day_of_week, levels = c("Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday"))

aggregate(rides_v2$ride_duration ~ rides_v2$member_casual + rides_v2$day_of_week, FUN = mean)
```

```
##    rides_v2$member_casual rides_v2$day_of_week rides_v2$ride_duration
## 1                  casual               Sunday              2950.4738
## 2                  member               Sunday              1043.4446
## 3                  casual               Monday              2545.5411
## 4                  member               Monday               883.5050
## 5                  casual              Tuesday              2291.9208
## 6                  member              Tuesday               867.6516
## 7                  casual            Wednesday              2319.7532
## 8                  member            Wednesday               876.4063
## 9                  casual             Thursday              2413.8835
## 10                 member             Thursday               869.0728
## 11                 casual               Friday              2446.0217
## 12                 member               Friday               901.5859
## 13                 casual             Saturday              2680.8808
## 14                 member             Saturday              1013.5092
```

Here we can see ride duration is a bit higher on the weekends, but no matter the day the casual riders tend to ride at least twice as long as members, sometimes nearly three times as long.

To clean things up, let’s create and view a table with shortened day names using the `wday()` function, as well as a calculated average duration, and with results grouped by member status and sorted by day of the week.

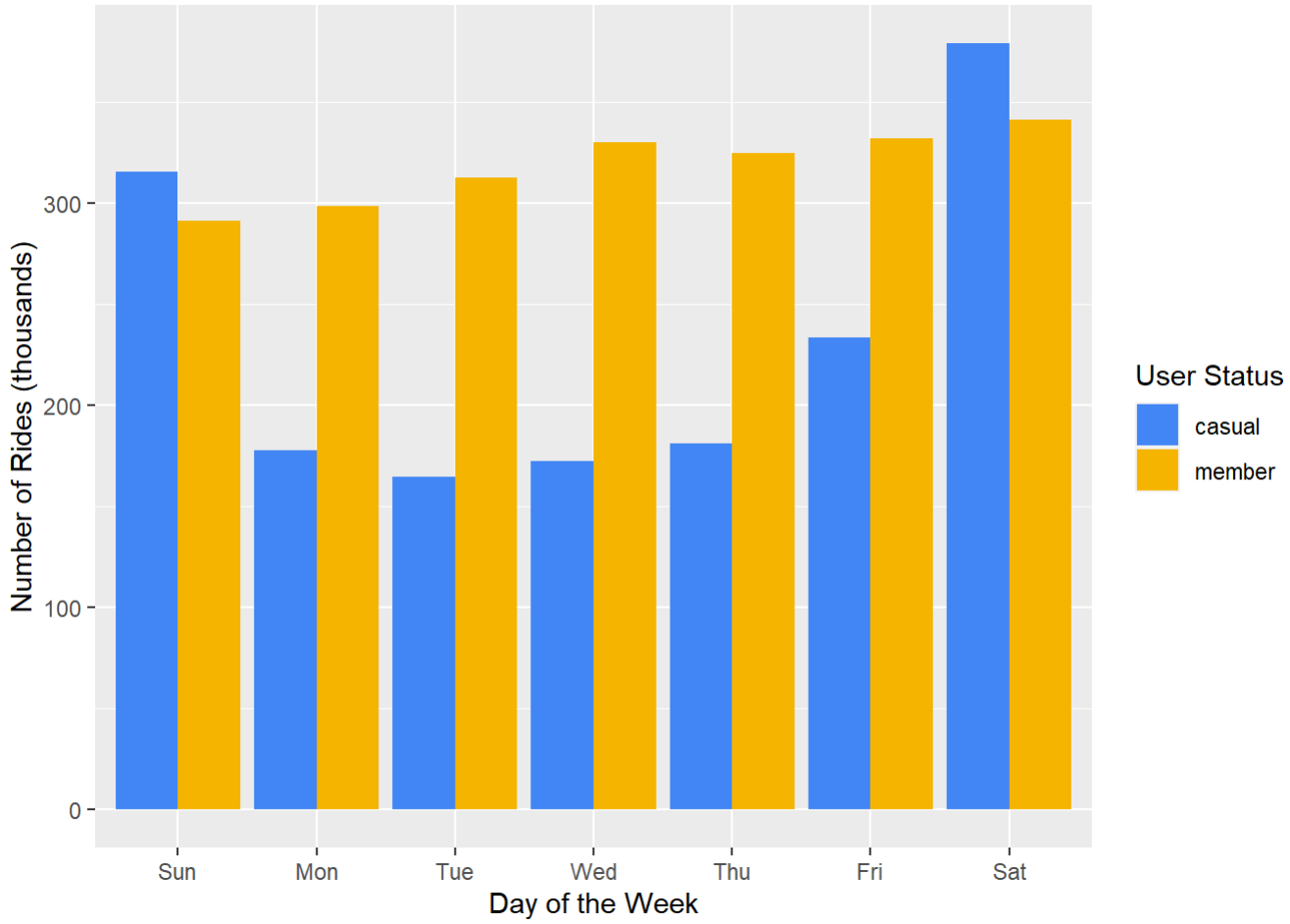
```
rides_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%  # creates weekday field wday()
  group_by(member_casual, weekday) %>%  # groups by user type and then weekday status
  summarise(number_of_rides = n(),  # calculates number of rides...
  average_duration = mean(ride_duration)) %>%       # and the average duration
  arrange(member_casual, weekday)   # sorts for presentation; overrides weekday grouping
```

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun              315524            2950.
##  2 casual        Mon              177887            2546.
##  3 casual        Tue              164756            2292.
##  4 casual        Wed              172483            2320.
##  5 casual        Thu              181383            2414.
##  6 casual        Fri              233690            2446.
##  7 casual        Sat              379113            2681.
##  8 member        Sun              291513            1043.
##  9 member        Mon              298690             884.
## 10 member        Tue              312666             868.
## 11 member        Wed              330064             876.
## 12 member        Thu              324847             869.
## 13 member        Fri              332090             902.
## 14 member        Sat              341165            1014.
```

Visualizations

To begin, let's see the number of rides by rider type. And we can export a .jpeg image of the result.

```
rides_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n() / 1000,
            average_duration = mean(ride_duration)) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Day of the Week", y = "Number of Rides (thousands)", fill = "User Status") +
  scale_fill_manual(values = c("#4285f4", "#f4b400"))
```




```
# Uncomment to save an image of the result.
# jpeg(file = "number_of_rides.jpeg")
```

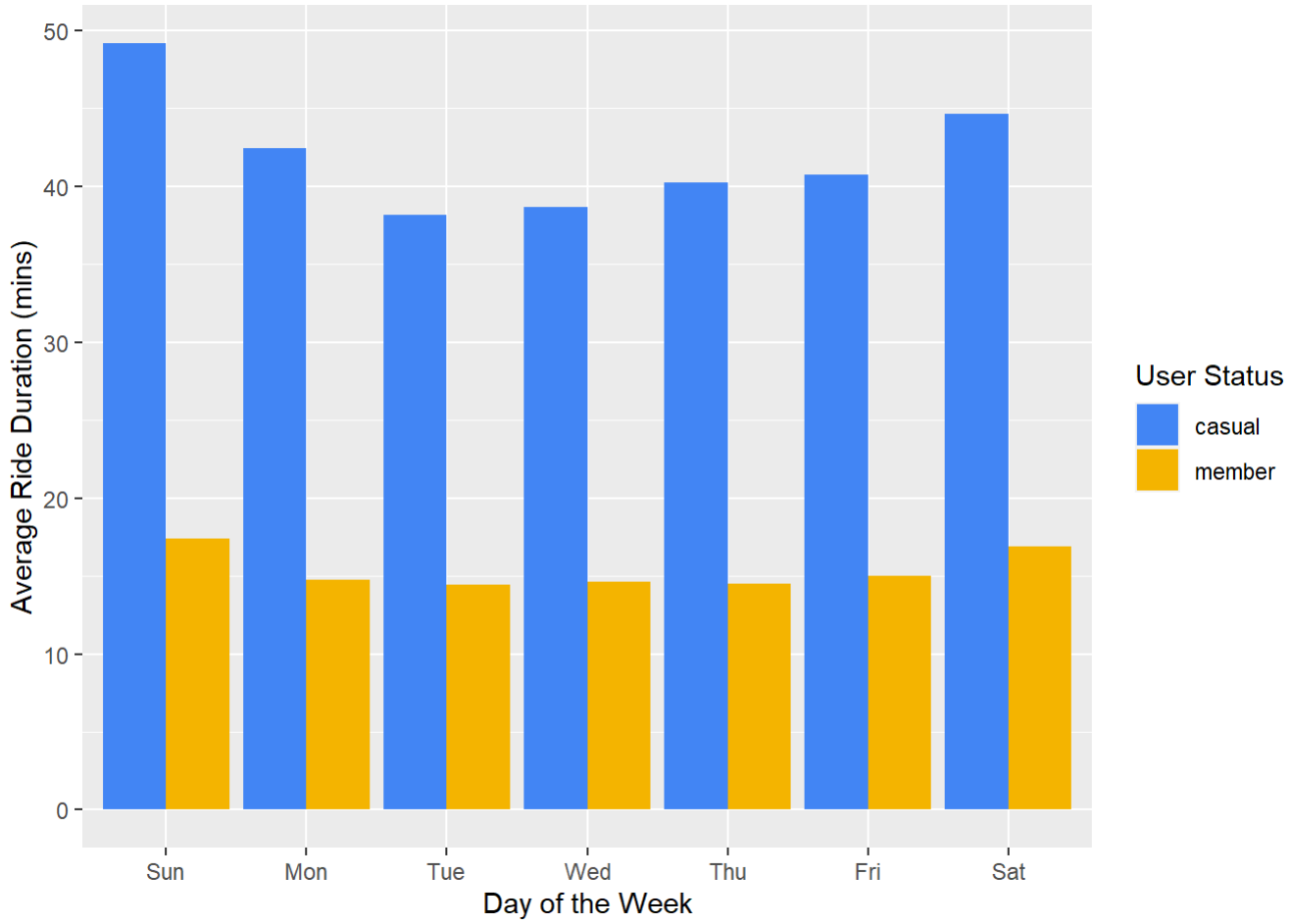
We see here that members are fairly consistent in the number of rides taken, gradually increasing through the week but overall with little change. This is not quite what we would have expected if we assumed members were work commuters; it seems a stretch to assume bicycle commuters would largely ride their bikes just as much on the weekends.

So perhaps members represent those with more consistent needs, such as for personal obligations and errands. They could also represent exercise enthusiasts, who would be more likely to cycle consistently whether or not they happen to be commuting as well. Or they could indeed largely consist of commuters, if Chicago's bike share user population tends to work throughout the week.

By comparison, the casual users have such a clear preference for the weekends they even take more rides than members on those days. This would be expected if one assumes casual riders are more likely to cycle for entertainment, exercise, or other casual reasons. However, there is a fairly large chunk of the casual user base who consistently ride during the week as well. This group may be the best to target for conversion to membership.

We can also compare average trip duration:

```
rides_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_duration) / 60) %>%
  arrange(member_casual, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(x = "Day of the Week", y = "Average Ride Duration (mins)", fill = "User Status") +
  scale_fill_manual(values = c("#4285f4", "#f4b400"))
```




```
# Export an image
# jpeg(file = "ride_duration.jpeg")
```

For members, once again we see consistent behavior. Not only do members tend to make just as many rides every day of the week, they also tend to ride for a similar length of time every day (although, there is a slight increase in duration on the weekends). This is consistent with the hypothesis that members have a utilitarian purpose in mind when using the bikes, however it would suggest that not many of them may be work commuters (unless workers in the area tend to work without regard to a standard Monday through Friday schedule).

Casual user ride duration is also remarkable, showing much longer lengths of time than members on average. Casual rides average at least twice as long, and often three times as long. In addition, casual use doesn't change hugely during the week compared to the weekend, though weekend rides do have slightly higher ride times.

Conclusion and recommendations

Member ride patterns are very consistent regardless of the day of the week. Whether during the work week or the weekend, members show little variation in the number of rides taken or the length of time spent riding (approximately 15 minutes per ride). In addition, members tend to ride more quickly, riding on average 150% the speeds of casual users.

This suggests members are more likely to be riding for the sake of commuting, or a regular exercise program, or to travel for personal obligations. Commuting assumes, however, that the population of users for the Divvy service tend to work throughout the week, rather than being heavily focused on traditional Monday-Friday scheduling.

In contrast, casual riders tend to take fewer rides during the weekdays, but then nearly double their number of rides on the weekends. However, regardless of the day of the week, casual riders consistently travel 2-3 times as long as members, and travel at a more leisurely pace.

This suggests many casual riders are more likely riding for personal enjoyment or casual needs, and not out of a necessity to travel. However, a significant portion of casual users consistently ride during the traditional work week. This fraction of the casual user base may be following a similar pattern of use as member users, and so may be prime targets for conversion to membership.

Based on this information, here are our top recommendations to convince casual riders to purchase annual memberships:

- Add features to bikes that expand their utility for personal errands and obligations. For example, add drink holders, baskets, and perhaps baby carriers to some bikes. The data suggests annual members are people who have a consistent need for transportation to fulfill their personal obligations, even outside of work. They may be running to the grocery store or grabbing a coffee. In other words, members use bike-sharing for their everyday miscellaneous transportation needs (though not necessarily for work commuting). Therefore, making the bikes more amenable to errand-like activities may increase the value to casual users, encouraging them to become members.
- Perform further analysis comparing user type and station use. If a particular type of customer is more likely to use a certain station, for example, this may provide more insight into the needs and thoughts of the customers. Or at the very least, it can allow for more targeted advertising; for instance, one may want to spend less time and money advertising for memberships at stations that already serve a high proportion of members. Station popularity data can also help when determining how to best distribute bikes, or where to add more stations to meet heavy demand and where to possibly remove underused stations.
- Do not focus heavily on marketing methods that encourage bike-share as an alternative form of commute. For example, heavy advertising within cabs, buses, or subways is not likely to be very effective. Casual users are already familiar with the bike-share service, and if using it for commuting were a good choice for them they most likely would have already signed up for membership. Instead, marketing that features the utility of the bikes for quick, easy, and inexpensive utility may be more encouraging.

Although we performed all of our analysis within this R markdown file, we can take our final data frame and export it for potential use in other software, such as Excel and Tableau.

```
counts <-aggregate(rides_v2$ride_duration ~ rides_v2$member_casual + rides_v2$day_of_week, FUN = mean)
write.csv(counts, file = "avg_ride_length.csv")
```

End of document.