Google-Capstone-Project.R

user

2022-10-29

```
#1.ASK
#1.0 Business task
#(a).How do annual members and casual riders use Cyclistic bikes differently?
#(b).Why would casual riders buy Cyclistic annual memberships?
#(c).How can Cyclistic use digital media to influence casual riders to become
members?
#1.1 Stakeholders
#stakeholders include the following
#director of marketing
#cyclistic marketing analytics team
#cyclistic exectuive team
#2.PREPARE
#The data is public and has been made available by motivate international inc
#The link.https://divvy-tripdata.s3.amazonaws.com/index.html
#.Data is organized in csv files
#.Credibility of data not in question
#.This data has been stripped of all identifying information ensuring its
privacy

#3.PROCESS
#.For this project I choose RStudio Desktop in order to prepare, process,
clean, analyze and create the visualizations.
#Data review involved the following:
#.Checking column names across all the 12 original files.
#.Checking for missing values.
#.Checking of white spaces.
#.Checking of duplicate records
#3.1 COLLECT & DATA WRANGLING
#load readr for reading rectangular data
#load lubridate for data wrangling
#load ggplot2 for data visualization
#install packages
#tidyverse for data wrangling
#lubridate for dealing with dates
#ggplot2 for data visualisation

library(tidyverse)

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

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

library(lubridate)

##
## Attaching package: 'lubridate'

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

library(ggplot2)
library(dplyr)
#3.1.1.COLLECT  AND READ DATA
#Upload data here
b_Trips_2019_Q1<- read_csv("C:/Users/user/Desktop/DATA SETS/BICYCLE
DATA/Divvy_Trips_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
## 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.

b_Trips_2019_Q2<- read_csv("C:/Users/user/Desktop/DATA SETS/BICYCLE
DATA/Divvy_Trips_2019_Q2.csv")

## Rows: 1108163 Columns: 12

## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr  (4): 03 - Rental Start Station Name, 02 - Rental End Station Name,
User...
## dbl  (5): 01 - Rental Details Rental ID, 01 - Rental Details Bike ID, 03 -
R...
```

```
## dttm (2): 01 - Rental Details Local Start Time, 01 - Rental Details Local
En...

##
## 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.
```

```
b_Trips_2019_Q3<- read_csv("C:/Users/user/Desktop/DATA SETS/BICYCLE
DATA/Divvy_Trips_2019_Q3.csv")
```

```
## Rows: 1640718 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
## 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.
```

```
b_Trips_2019_Q4<- read_csv("C:/Users/user/Desktop/DATA SETS/BICYCLE
DATA/Divvy_Trips_2019_Q4.csv")
```

```
## Rows: 704054 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
## 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.
```

```
head(b_Trips_2019_Q1)
```

```
## # A tibble: 6 x 12
##    trip_id start_time          end_time            bikeid tripduration
##      <dbl> <dttm>              <dttm>               <dbl>        <dbl>
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07   2167          390
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34   4386          441
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12   1524          829
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28    252         1783
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56   1170          364
```

```
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09   2437          216
## # ... with 7 more variables: from_station_id <dbl>, from_station_name
<chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender
<chr>,
## #   birthyear <dbl>

#3.2.2.WRANGLE DATA AND COMBINE INTO A SINGLE FILE
#Compare column names
colnames(b_Trips_2019_Q1)

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"

colnames(b_Trips_2019_Q2)

##  [1] "01 - Rental Details Rental ID"
##  [2] "01 - Rental Details Local Start Time"
##  [3] "01 - Rental Details Local End Time"
##  [4] "01 - Rental Details Bike ID"
##  [5] "01 - Rental Details Duration In Seconds Uncapped"
##  [6] "03 - Rental Start Station ID"
##  [7] "03 - Rental Start Station Name"
##  [8] "02 - Rental End Station ID"
##  [9] "02 - Rental End Station Name"
## [10] "User Type"
## [11] "Member Gender"
## [12] "05 - Member Details Member Birthday Year"

colnames(b_Trips_2019_Q3)

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"

colnames(b_Trips_2019_Q4)

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"

#colnames for dataset Divvy_Trips_2019_Q1,Divvy_Trips_2019_Q3 &
Divvy_Trips_2019_Q4 match
# While the names don't have to be in the same order, they DO need to match
perfectly before we can use a command to join them into one file
#Rename columns  to make them consisent
b_Trips_2019_Q2 <- b_Trips_2019_Q2%>%rename(trip_id= "01 - Rental Details
Rental ID"
```

```
         ,bikeid = "01 - Rental Details Bike ID"
         ,start_time ="01 - Rental Details Local Start Time"
         ,end_time = "01 - Rental Details Local End Time"
         ,from_station_name ="03 - Rental Start Station Name"
         ,from_station_id = "03 - Rental Start Station ID"
         ,to_station_name = "02 - Rental End Station Name"
         ,to_station_id = "02 - Rental End Station ID"
         ,usertype="User Type"
         ,gender="Member Gender"
         ,birthyear="05 - Member Details Member Birthday Year"
         ,tripduration="01 - Rental Details Duration In Seconds Uncapped")

#confirm the colnames have changed
colnames(b_Trips_2019_Q2)

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"

table(b_Trips_2019_Q2$Rental.Details.Rental.ID)

## Warning: Unknown or uninitialised column: `Rental.Details.Rental.ID`.

## < table of extent 0 >

# Inspect the dataframes and look for incongruencies

# Convert ride_id and rideable_type to character so that they can stack
correctly
b_Trips_2019_Q1<- b_Trips_2019_Q1%>%
                  mutate(trip_id = as.character(trip_id)
                  ,bikeid = as.character(bikeid))

b_Trips_2019_Q2 <- b_Trips_2019_Q2%>%
                   mutate(trip_id = as.character(trip_id)
                  ,bikeid = as.character(bikeid))

b_Trips_2019_Q3 <-  b_Trips_2019_Q3%>%
                    mutate(trip_id = as.character(trip_id)
                   ,bikeid = as.character(bikeid))

b_Trips_2019_Q4<- b_Trips_2019_Q4%>%
                  mutate(trip_id = as.character(trip_id)
                  ,bikeid = as.character(bikeid))

#rbind combines columns that match.
#use bind_rows instead
all_b_trips<-
bind_rows(b_Trips_2019_Q1,b_Trips_2019_Q2,b_Trips_2019_Q3,b_Trips_2019_Q4)
```

```
#3.2 Data Validation
# Inspect the new table that has been created
colnames(all_b_trips)  #List of column names

##  [1] "trip_id"           "start_time"        "end_time"
##  [4] "bikeid"            "tripduration"      "from_station_id"
##  [7] "from_station_name" "to_station_id"     "to_station_name"
## [10] "usertype"          "gender"            "birthyear"

nrow(all_b_trips)  #How many rows are in data frame?

## [1] 3818004

dim(all_b_trips)  #Dimensions of the data frame?

## [1] 3818004      12

head(all_b_trips)  #See the first 6 rows of data frame.  Also tail(all_trips)

## # A tibble: 6 x 12
##   trip_id  start_time          end_time            bikeid tripduration
##   <chr>    <dttm>              <dttm>              <chr>         <dbl>
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07 2167            390
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34 4386            441
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12 1524            829
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28 252            1783
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56 1170            364
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09 2437            216
## # ... with 7 more variables: from_station_id <dbl>, from_station_name
<chr>,
## #   to_station_id <dbl>, to_station_name <chr>, usertype <chr>, gender
<chr>,
## #   birthyear <dbl>

str(all_b_trips)  #See list of columns and data types (numeric, character,
etc)

## spec_tbl_df [3,818,004 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ trip_id          : chr [1:3818004] "21742443" "21742444" "21742445"
"21742446" ...
##  $ start_time       : POSIXct[1:3818004], format: "2019-01-01 00:04:37"
"2019-01-01 00:08:13" ...
##  $ end_time         : POSIXct[1:3818004], format: "2019-01-01 00:11:07"
"2019-01-01 00:15:34" ...
##  $ bikeid           : chr [1:3818004] "2167" "4386" "1524" "252" ...
##  $ tripduration     : num [1:3818004] 390 441 829 1783 364 ...
##  $ from_station_id  : num [1:3818004] 199 44 15 123 173 98 98 211 150 268
...
##  $ from_station_name: chr [1:3818004] "Wabash Ave & Grand Ave" "State St &
Randolph St" "Racine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id    : num [1:3818004] 84 624 644 176 35 49 49 142 148 141
...
```

```
##  $ to_station_name  : chr [1:3818004] "Milwaukee Ave & Grand Ave"
"Dearborn St & Van Buren St (*)" "Western Ave & Fillmore St (*)" "Clark St &
Elm St" ...
##  $ usertype         : chr [1:3818004] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ gender           : chr [1:3818004] "Male" "Female" "Female" "Male" ...
##  $ birthyear        : num [1:3818004] 1989 1990 1994 1993 1994 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>

summary(all_b_trips)  #Statistical summary of data. Mainly for numerics

##    trip_id           start_time                     end_time
##  Length:3818004     Min.   :2019-01-01 00:04:37   Min.   :2019-01-01
00:11:07
##  Class :character   1st Qu.:2019-05-29 15:49:26   1st Qu.:2019-05-29
16:09:28
##  Mode  :character   Median :2019-07-25 17:50:54   Median :2019-07-25
18:12:23
##                     Mean   :2019-07-19 21:47:37   Mean   :2019-07-19
22:11:47
##                     3rd Qu.:2019-09-15 06:48:05   3rd Qu.:2019-09-15
08:30:13
##                     Max.   :2019-12-31 23:57:17   Max.   :2020-01-21
13:54:35
##
##     bikeid           tripduration      from_station_id from_station_name
##  Length:3818004     Min.   :      61   Min.   :  1.0   Length:3818004
##  Class :character   1st Qu.:     411   1st Qu.: 77.0   Class :character
##  Mode  :character   Median :     709   Median :174.0   Mode  :character
##                     Mean   :    1450   Mean   :201.7
##                     3rd Qu.:    1283   3rd Qu.:289.0
##                     Max.   :10628400   Max.   :673.0
##
##  to_station_id   to_station_name      usertype           gender
##  Min.   :  1.0   Length:3818004     Length:3818004     Length:3818004
```

```
##  1st Qu.: 77.0   Class :character   Class :character   Class :character
##  Median :174.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   :202.6
##  3rd Qu.:291.0
##  Max.   :673.0
##
##    birthyear
##  Min.   :1759
##  1st Qu.:1979
##  Median :1987
##  Mean   :1984
##  3rd Qu.:1992
##  Max.   :2014
##  NA's   :538751

# STEP 4: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS
# Inspect the new table that has been created

# There are a few problems we will need to fix:
# (1) In the "member_casual" column, there are two names for members
("member" and "Subscriber") and two names for casual riders ("Customer" and
"casual"). We will need to consolidate that from four to two labels.
# (2) The data can only be aggregated at the ride-level, which is too
granular. We will want to add some additional columns of data -- such as day,
month, year -- that provide additional opportunities to aggregate the data.
# (3) We will want to add a calculated field for length of ride since the
2020Q1 data did not have the "tripduration" column. We will add "ride_length"
to the entire dataframe for consistency.
# (4) There are some rides where tripduration shows up as negative, including
several hundred rides where Divvy took bikes out of circulation for Quality
Control reasons. We will want to delete these rides.

#Converting the start_time and end_time columns to
all_b_trips<- all_b_trips%>%
              mutate(start_time=as.POSIXlt(start_time))
all_b_trips<- all_b_trips%>%
              mutate(end_time=as.POSIXlt(end_time))

# Add columns that list the date, month, day, and year of each ride
# This will allow us to aggregate ride data for each month, day, or year ...
before completing these operations we could only aggregate at the ride level
# more on date formats in R found at that link
all_b_trips$date <- as.Date(all_b_trips$start_time) #The default format is
yyyy-mm-dd
all_b_trips$month <- format(as.Date(all_b_trips$date), "%m")
all_b_trips$day <- format(as.Date(all_b_trips$date), "%d")
all_b_trips$year <- format(as.Date(all_b_trips$date), "%Y")
all_b_trips$day_of_week <- format(as.Date(all_b_trips$date), "%A")
# Add a "ride_length" calculation to all_trips (in seconds)
# https://stat.ethz.ch/R-manual/R-devel/library/base/html/difftime.html
```

```
all_b_trips$ride_length <-
difftime(all_b_trips$end_time,all_b_trips$start_time)

# Remove "bad" data
# The dataframe includes a few hundred entries when bikes were taken out of
docks and checked for quality by Divvy or ride_length was negative
#Check for missing values
colSums(is.na(all_b_trips))
```

```
##           trip_id        start_time          end_time            bikeid
##                 0                 0                 0                 0
##      tripduration   from_station_id from_station_name     to_station_id
##                 0                 0                 0                 0
##   to_station_name          usertype            gender         birthyear
##                 0                 0            559206            538751
##              date             month               day              year
##                 0                 0                 0                 0
##       day_of_week       ride_length
##                 0                 0
```

```
#check for duplicates
distinct(all_b_trips)
```

```
## # A tibble: 3,818,004 x 18
##    trip_id  start_time          end_time            bikeid tripduration
##    <chr>    <dttm>              <dttm>              <chr>         <dbl>
##  1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07 2167            390
##  2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34 4386            441
##  3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12 1524            829
##  4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28 252            1783
##  5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56 1170            364
##  6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09 2437            216
##  7 21742449 2019-01-01 00:16:06 2019-01-01 00:19:03 2708            177
##  8 21742450 2019-01-01 00:18:41 2019-01-01 00:20:21 2796            100
##  9 21742451 2019-01-01 00:18:43 2019-01-01 00:47:30 6205           1727
## 10 21742452 2019-01-01 00:19:18 2019-01-01 00:24:54 3939            336
## # ... with 3,817,994 more rows, and 13 more variables: from_station_id
<dbl>,
## #   from_station_name <chr>, to_station_id <dbl>, to_station_name <chr>,
## #   usertype <chr>, gender <chr>, birthyear <dbl>, date <date>, month
<chr>,
## #   day <chr>, year <chr>, day_of_week <chr>, ride_length <drtn>
```

```
#View column data types
str(all_b_trips)
```

```
## spec_tbl_df [3,818,004 x 18] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ trip_id          : chr [1:3818004] "21742443" "21742444" "21742445"
"21742446" ...
##  $ start_time       : POSIXlt[1:3818004], format: "2019-01-01 00:04:37"
"2019-01-01 00:08:13" ...
```

```
##  $ end_time          : POSIXlt[1:3818004], format: "2019-01-01 00:11:07"
"2019-01-01 00:15:34" ...
##  $ bikeid            : chr [1:3818004] "2167" "4386" "1524" "252" ...
##  $ tripduration      : num [1:3818004] 390 441 829 1783 364 ...
##  $ from_station_id   : num [1:3818004] 199 44 15 123 173 98 98 211 150 268
...
##  $ from_station_name: chr [1:3818004] "Wabash Ave & Grand Ave" "State St &
Randolph St" "Racine Ave & 18th St" "California Ave & Milwaukee Ave" ...
##  $ to_station_id     : num [1:3818004] 84 624 644 176 35 49 49 142 148 141
...
##  $ to_station_name   : chr [1:3818004] "Milwaukee Ave & Grand Ave"
"Dearborn St & Van Buren St (*)" "Western Ave & Fillmore St (*)" "Clark St &
Elm St" ...
##  $ usertype          : chr [1:3818004] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" ...
##  $ gender            : chr [1:3818004] "Male" "Female" "Female" "Male" ...
##  $ birthyear         : num [1:3818004] 1989 1990 1994 1993 1994 ...
##  $ date              : Date[1:3818004], format: "2019-01-01" "2019-01-01"
...
##  $ month             : chr [1:3818004] "01" "01" "01" "01" ...
##  $ day               : chr [1:3818004] "01" "01" "01" "01" ...
##  $ year              : chr [1:3818004] "2019" "2019" "2019" "2019" ...
##  $ day_of_week       : chr [1:3818004] "Tuesday" "Tuesday" "Tuesday"
"Tuesday" ...
##  $ ride_length       : 'difftime' num [1:3818004] 6.5 7.35 13.8166666666667
29.7166666666667 ...
##   ..- attr(*, "units")= chr "mins"
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_datetime(format = ""),
##   ..   end_time = col_datetime(format = ""),
##   ..   bikeid = col_double(),
##   ..   tripduration = col_number(),
##   ..   from_station_id = col_double(),
##   ..   from_station_name = col_character(),
##   ..   to_station_id = col_double(),
##   ..   to_station_name = col_character(),
##   ..   usertype = col_character(),
##   ..   gender = col_character(),
##   ..   birthyear = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# 4: CONDUCT DESCRIPTIVE ANALYSIS
#====================================
# Descriptive analysis on ride_length (all figures in seconds)
mean(all_b_trips$ride_length) #straight average (total ride length / rides)
```

```
## Time difference of 24.17419 mins
```

```r
median(all_b_trips$ride_length) #midpoint number in the ascending array of
ride lengths

## Time difference of 11.81667 mins

max(all_b_trips$ride_length) #Longest ride

## Time difference of 177200.4 mins

min(all_b_trips$ride_length) #shortest ride

## Time difference of -56.36667 mins

# You can condense the four lines above to one line using summary() on the
specific attribute
summary(all_b_trips$ride_length)

##   Length    Class     Mode
##  3818004 difftime  numeric

# Compare members and casual users
aggregate(all_b_trips$ride_length ~ all_b_trips$usertype, FUN = mean)

##   all_b_trips$usertype all_b_trips$ride_length
## 1             Customer           57.01734 mins
## 2           Subscriber           14.32765 mins

aggregate(all_b_trips$ride_length ~ all_b_trips$usertype, FUN = median)

##   all_b_trips$usertype all_b_trips$ride_length
## 1             Customer           25.83333 mins
## 2           Subscriber            9.80000 mins

aggregate(all_b_trips$ride_length ~ all_b_trips$usertype, FUN = max)

##   all_b_trips$usertype all_b_trips$ride_length
## 1             Customer           177200.4 mins
## 2           Subscriber           150943.9 mins

aggregate(all_b_trips$ride_length ~ all_b_trips$usertype, FUN = min)

##   all_b_trips$usertype all_b_trips$ride_length
## 1             Customer          -48.28333 mins
## 2           Subscriber          -56.36667 mins

# See the average ride time by each day for members vs casual users
aggregate(all_b_trips$ride_length ~ all_b_trips$usertype +
all_b_trips$day_of_week, FUN = mean)

##    all_b_trips$usertype all_b_trips$day_of_week all_b_trips$ride_length
## 1              Customer                  Friday           60.17561 mins
## 2            Subscriber                  Friday           13.89748 mins
## 3              Customer                  Monday           54.49989 mins
## 4            Subscriber                  Monday           14.24928 mins
```

```
## 5              Customer                Saturday           54.06111 mins
## 6            Subscriber                Saturday           16.30271 mins
## 7              Customer                  Sunday           56.18169 mins
## 8            Subscriber                  Sunday           15.40119 mins
## 9              Customer                Thursday           59.95112 mins
## 10           Subscriber                Thursday           13.77979 mins
## 11             Customer                 Tuesday           57.41328 mins
## 12           Subscriber                 Tuesday           14.15259 mins
## 13             Customer               Wednesday           60.33407 mins
## 14           Subscriber               Wednesday           13.80984 mins
```

```
# Notice that the days of the week are out of order. Let's fix that.
all_b_trips$day_of_week <- ordered(all_b_trips$day_of_week,
levels=c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturda
y"))

# Now, let's run the average ride time by each day for members vs casual
users
aggregate(all_b_trips$ride_length ~ all_b_trips$usertype +
all_b_trips$day_of_week, FUN = mean)
```

```
##    all_b_trips$usertype all_b_trips$day_of_week all_b_trips$ride_length
## 1              Customer                  Sunday           56.18169 mins
## 2            Subscriber                  Sunday           15.40119 mins
## 3              Customer                  Monday           54.49989 mins
## 4            Subscriber                  Monday           14.24928 mins
## 5              Customer                 Tuesday           57.41328 mins
## 6            Subscriber                 Tuesday           14.15259 mins
## 7              Customer               Wednesday           60.33407 mins
## 8            Subscriber               Wednesday           13.80984 mins
## 9              Customer                Thursday           59.95112 mins
## 10           Subscriber                Thursday           13.77979 mins
## 11             Customer                  Friday           60.17561 mins
## 12           Subscriber                  Friday           13.89748 mins
## 13             Customer                Saturday           54.06111 mins
## 14           Subscriber                Saturday           16.30271 mins
```

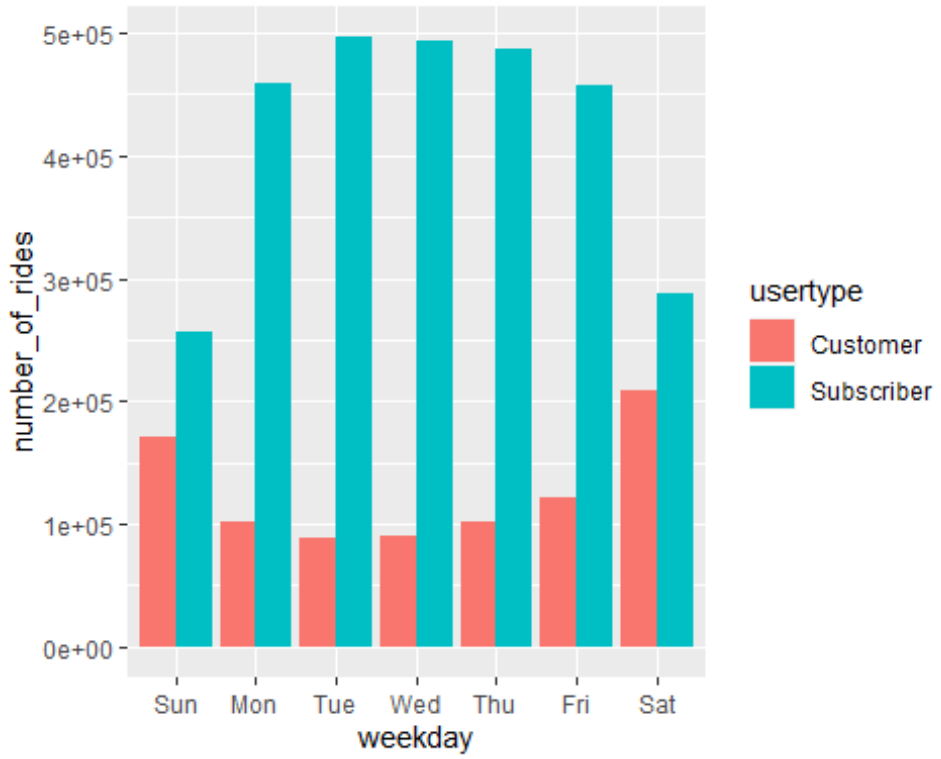
```
# analyze ridership data by type and weekday
all_b_trips %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%  #creates weekday
field using wday()
  group_by(usertype, weekday) %>%  #groups by usertype and weekday
  summarise(number_of_rides = n(),                           #calculates the
number of rides and average duration
  average_duration = mean(ride_length)) %>%         # calculates the average
duration
  arrange(usertype, weekday)                                 # sorts
```

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

```
## # A tibble: 14 x 4
## # Groups:   usertype [2]
##    usertype   weekday number_of_rides average_duration
##    <chr>      <ord>             <int> <drtn>
##  1 Customer   Sun              170179 56.18169 mins
##  2 Customer   Mon              101489 54.49989 mins
##  3 Customer   Tue               88655 57.41328 mins
##  4 Customer   Wed               89745 60.33407 mins
##  5 Customer   Thu              101372 59.95112 mins
##  6 Customer   Fri              121141 60.17561 mins
##  7 Customer   Sat              208056 54.06111 mins
##  8 Subscriber Sun              256241 15.40119 mins
##  9 Subscriber Mon              458780 14.24928 mins
## 10 Subscriber Tue              497025 14.15259 mins
## 11 Subscriber Wed              494277 13.80984 mins
## 12 Subscriber Thu              486915 13.77979 mins
## 13 Subscriber Fri              456966 13.89748 mins
## 14 Subscriber Sat              287163 16.30271 mins
```

```
# Let's visualize the number of rides by rider type
all_b_trips %>%
  mutate(weekday = wday(start_time, label = TRUE))%>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n()
 ,average_duration = mean(ride_length))%>%
  arrange(usertype, weekday)  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = usertype)) +
  geom_col(position = "dodge")
```

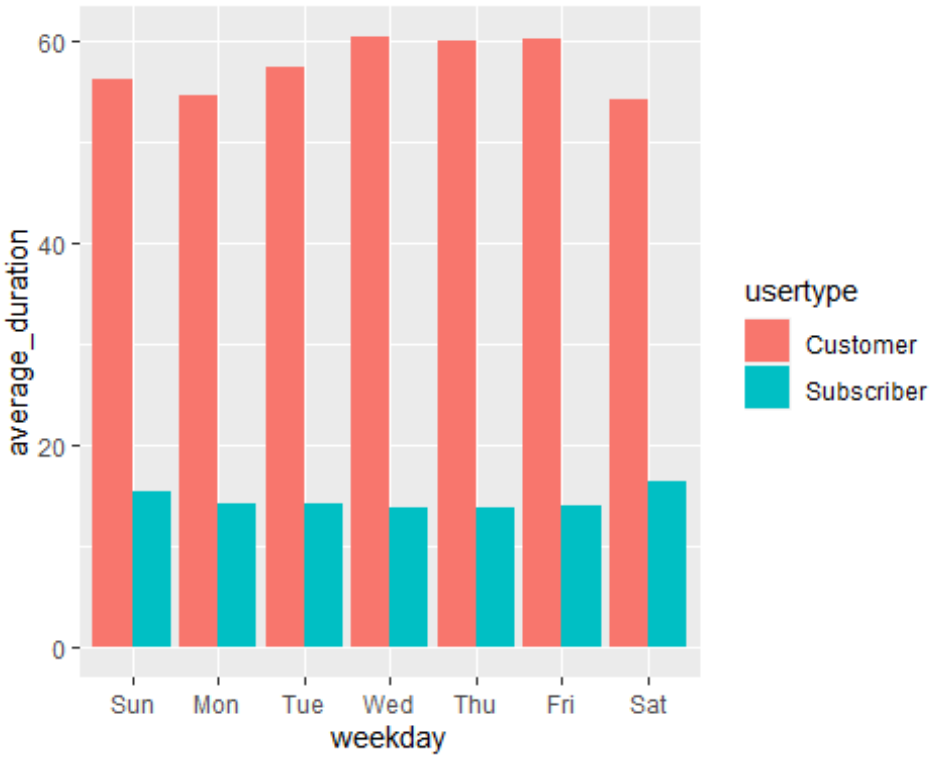
```
## `summarise()` has grouped output by 'usertype'. You can override using the
`.groups` argument.
```




```
# Let's create a visualization for average duration
all_b_trips %>%
  mutate(weekday = wday(start_time, label = TRUE)) %>%
  group_by(usertype, weekday) %>%
  summarise(number_of_rides = n()
  ,average_duration = mean(ride_length)) %>%
  arrange(usertype, weekday)  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = usertype)) +
  geom_col(position = "dodge")
```

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

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




```
#=============================================
# 5: EXPORT SUMMARY FILE FOR FURTHER ANALYSIS
#=============================================
# Create a csv file that we will visualize in Excel, Tableau, or my
presentation software
# N.B.: This file location is for a Mac. If you are working on a PC, change
the file location accordingly (most likely
"C:\Users\YOUR_USERNAME\Desktop\...") to export the data. You can read more
here: https://datatofish.com/export-dataframe-to-csv-in-r/
counts <- aggregate(all_b_trips$ride_length ~ all_b_trips$usertype +
all_b_trips$day_of_week, FUN = mean)
write.csv(counts,'C:\\Users\\user\\Desktop\\RSTUDIO\\avg_ride_length.csv')
```