Cyclistic bike-share analysis case study

Ramya Bandi

2024-11-30

library(tidyverse) #helps wrangle data

Use the conflicted package to manage conflicts

library(conflicted)

Set dplyr::filter and dplyr::lag as the default choices

conflict_prefer("filter", "dplyr")

conflict_prefer("lag", "dplyr")

#=====================

STEP 1: COLLECT DATA

#=====================

Upload Divvy datasets (csv files) here

```
q1_2019 <- read_csv("Divvy_Trips_2019_Q1.csv")
```

```
## Rows: 365069 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): start_time, end_time, from_station_name, to_station_name, usertype,...
## dbl (5): trip_id, bikeid, from_station_id, to_station_id, birthyear
## num (1): tripduration
##
## 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 <- read_csv("Divvy_Trips_2020_Q1.csv")
```

```
## Rows: 426887 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): ride_id, rideable_type, started_at, ended_at, start_station_name, e...
## dbl (6): start_station_id, end_station_id, start_lat, start_lng, end_lat, 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.
```

#===

STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE

#===

Compare column names each of the files

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

```
colnames(q1_2019)
```

```
##  [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(q1_2020)
```

```
##  [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"
```

Rename columns to make them consistent with q1_2020 (as this will be the supposed

going-forward table design for Divvy)

```
(q1_2019 <- rename(q1_2019
,ride_id = trip_id
,rideable_type = bikeid
,started_at = start_time
,ended_at = end_time
,start_station_name = from_station_name
,start_station_id = from_station_id
,end_station_name = to_station_name
,end_station_id = to_station_id
,member_casual = usertype
))
```

```
## # A tibble: 365,069 x 12
##     ride_id started_at      ended_at rideable_type tripduration start_station_id
##       <dbl> <chr>           <chr>            <dbl>        <dbl>            <dbl>
##  1 21742443 01-01-2019 00:~ 01-01-2~          2167          390              199
##  2 21742444 01-01-2019 00:~ 01-01-2~          4386          441               44
##  3 21742445 01-01-2019 00:~ 01-01-2~          1524          829               15
##  4 21742446 01-01-2019 00:~ 01-01-2~           252         1783              123
##  5 21742447 01-01-2019 00:~ 01-01-2~          1170          364                173
##  6 21742448 01-01-2019 00:~ 01-01-2~          2437          216                 98
##  7 21742449 01-01-2019 00:~ 01-01-2~          2708          177                 98
##  8 21742450 01-01-2019 00:~ 01-01-2~          2796          100                211
##  9 21742451 01-01-2019 00:~ 01-01-2~          6205         1727                150
## 10 21742452 01-01-2019 00:~ 01-01-2~          3939          336                268
## # i 365,059 more rows
## # i 6 more variables: start_station_name <chr>, end_station_id <dbl>,
## #   end_station_name <chr>, member_casual <chr>, gender <chr>, birthyear <dbl>
```

Inspect the dataframes and look for incongruencies

```
str(q1_2019)
```

```
## spc_tbl_ [365,069 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : num [1:365069] 21742443 21742444 21742445 21742446 21742447 ...
##  $ started_at        : chr [1:365069] "01-01-2019 00:04" "01-01-2019 00:08" "01-01-2019 00:13" "01-0
##  $ ended_at          : chr [1:365069] "01-01-2019 00:11" "01-01-2019 00:15" "01-01-2019 00:27" "01-0
##  $ rideable_type     : num [1:365069] 2167 4386 1524 252 1170 ...
##  $ tripduration      : num [1:365069] 390 441 829 1783 364 ...
##  $ start_station_id  : num [1:365069] 199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr [1:365069] "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave &
##  $ end_station_id    : num [1:365069] 84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr [1:365069] "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "
##  $ member_casual     : chr [1:365069] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
##  $ gender            : chr [1:365069] "Male" "Female" "Female" "Male" ...
##  $ birthyear         : num [1:365069] 1989 1990 1994 1993 1994 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   trip_id = col_double(),
##   ..   start_time = col_character(),
##   ..   end_time = col_character(),
##   ..   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>
```

```
str(q1_2020)
```

```
## spc_tbl_ [426,887 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:426887] "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A3
##  $ rideable_type     : chr [1:426887] "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
##  $ started_at        : chr [1:426887] "21-01-2020 20:06" "30-01-2020 14:22" "09-01-2020 19:29" "06-0
##  $ ended_at          : chr [1:426887] "21-01-2020 20:14" "30-01-2020 14:26" "09-01-2020 19:32" "06-0
##  $ start_station_name: chr [1:426887] "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway
##  $ start_station_id  : num [1:426887] 239 234 296 51 66 212 96 96 212 38 ...
##  $ end_station_name  : chr [1:426887] "Clark St & Leland Ave" "Southport Ave & Irving Park Rd" "Wilt
## $ end_station_id    : num [1:426887] 326 318 117 24 212 96 212 212 96 100 ...
## $ start_lat         : num [1:426887] 42 42 41.9 41.9 41.9 ...
## $ start_lng         : num [1:426887] -87.7 -87.7 -87.6 -87.6 -87.6 ...
## $ end_lat           : num [1:426887] 42 42 41.9 41.9 41.9 ...
## $ end_lng           : num [1:426887] -87.7 -87.7 -87.7 -87.6 -87.6 ...
## $ member_casual     : chr [1:426887] "member" "member" "member" "member" ...
## - attr(*, "spec")=
##  .. cols(
##  ..   ride_id = col_character(),
##  ..   rideable_type = col_character(),
##  ..   started_at = col_character(),
##  ..   ended_at = col_character(),
##  ..   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()
##  .. )
## - attr(*, "problems")=<externalptr>
```

Convert ride_id and rideable_type to character so that they can stack correctly

```
q1_2019 <- mutate(q1_2019, ride_id = as.character(ride_id)
,rideable_type = as.character(rideable_type))
```

Stack individual quarter's data frames into one big data frame

```
all_trips <- bind_rows(q1_2019, q1_2020)#, q3_2019)#, q4_2019, q1_2020)
```

Remove lat, long, birthyear, and gender fields as this data was dropped beginning in 2020

```
all_trips <- all_trips %>%
select(-c(start_lat, start_lng, end_lat, end_lng, birthyear, gender, "tripduration"))
```

#===

STEP 3: CLEAN UP AND ADD DATA TO PREPARE FOR ANALYSIS

#===

Inspect the new table that has been created

```r
colnames(all_trips) #List of column names
```

```
## [1] "ride_id"            "started_at"         "ended_at"
## [4] "rideable_type"      "start_station_id"   "start_station_name"
## [7] "end_station_id"     "end_station_name"   "member_casual"
```

```r
nrow(all_trips) #How many rows are in data frame?
```

```
## [1] 791956
```

```r
dim(all_trips) #Dimensions of the data frame?
```

```
## [1] 791956      9
```

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

```
## # A tibble: 6 x 9
##   ride_id  started_at ended_at rideable_type start_station_id start_station_name
##   <chr>    <chr>      <chr>    <chr>                    <dbl> <chr>
## 1 21742443 01-01-201~ 01-01-2~ 2167                       199 Wabash Ave & Gran~
## 2 21742444 01-01-201~ 01-01-2~ 4386                        44 State St & Randol~
## 3 21742445 01-01-201~ 01-01-2~ 1524                        15 Racine Ave & 18th~
## 4 21742446 01-01-201~ 01-01-2~ 252                        123 California Ave & ~
## 5 21742447 01-01-201~ 01-01-2~ 1170                       173 Mies van der Rohe~
## 6 21742448 01-01-201~ 01-01-2~ 2437                        98 LaSalle St & Wash~
## # i 3 more variables: end_station_id <dbl>, end_station_name <chr>,
## #   member_casual <chr>
```

```r
str(all_trips) #See list of columns and data types (numeric, character, etc)
```

```
## tibble [791,956 x 9] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:791956] "21742443" "21742444" "21742445" "21742446" ...
##  $ started_at        : chr [1:791956] "01-01-2019 00:04" "01-01-2019 00:08" "01-01-2019 00:13" "01-0
##  $ ended_at          : chr [1:791956] "01-01-2019 00:11" "01-01-2019 00:15" "01-01-2019 00:27" "01-0
##  $ rideable_type     : chr [1:791956] "2167" "4386" "1524" "252" ...
##  $ start_station_id  : num [1:791956] 199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr [1:791956] "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave
##  $ end_station_id    : num [1:791956] 84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr [1:791956] "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "
##  $ member_casual     : chr [1:791956] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```r
summary(all_trips) #Statistical summary of data. Mainly for numerics
```

```
##    ride_id           started_at          ended_at         rideable_type
##  Length:791956      Length:791956      Length:791956      Length:791956
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  start_station_id start_station_name end_station_id  end_station_name
##  Min.   :  2.0    Length:791956      Min.   :  2.0   Length:791956
##  1st Qu.: 77.0    Class :character   1st Qu.: 77.0   Class :character
##  Median :174.0    Mode  :character   Median :174.0   Mode  :character
##  Mean   :204.4                    Mean   :204.4
##  3rd Qu.:291.0                    3rd Qu.:291.0
##  Max.   :675.0                    Max.   :675.0
##                                   NA's   :1
##  member_casual
##  Length:791956
##  Class :character
##  Mode  :character
##
##
##
##
```

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. # In the "member_casual" column, replace "Subscriber" with "member" and "Customer" with "casual" # Before 2020, Divvy used different labels for these two types of riders … we will want to make our dataframe consistent with their current nomenclature

N.B.: "Level" is a special property of a column that is retained even if a subset does not

contain any values from a specific level # Begin by seeing how many observations fall under each usertype

```
table(all_trips$member_casual)
```

```
##
##     casual   Customer     member Subscriber
##      48480      23163     378407     341906
```

Reassign to the desired values (we will go with the current 2020 labels)

```
all_trips <- all_trips %>%
mutate(member_casual = recode(member_casual
,"Subscriber" = "member"
,"Customer" = "casual"))
```

Check to make sure the proper number of observations were reassigned

```
table(all_trips$member_casual)
```

```
##
## casual member
##  71643 720313
```

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 # https://www.statmethods.net/input/dates.html more on date formats in R found at that link

```
all_trips$date <- as.Date(all_trips$started_at) #The default format is yyyy-mm-dd
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$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_trips$ride_length <- difftime(all_trips$ended_at,all_trips$started_at)
```

Inspect the structure of the columns

```
str(all_trips)
```

```
## tibble [791,956 x 15] (S3: tbl_df/tbl/data.frame)
##  $ ride_id           : chr [1:791956] "21742443" "21742444" "21742445" "21742446" ...
##  $ started_at        : chr [1:791956] "01-01-2019 00:04" "01-01-2019 00:08" "01-01-2019 00:13" "01-0
##  $ ended_at          : chr [1:791956] "01-01-2019 00:11" "01-01-2019 00:15" "01-01-2019 00:27" "01-0
##  $ rideable_type     : chr [1:791956] "2167" "4386" "1524" "252" ...
##  $ start_station_id  : num [1:791956] 199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr [1:791956] "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave &
##  $ end_station_id    : num [1:791956] 84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr [1:791956] "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "
##  $ member_casual     : chr [1:791956] "member" "member" "member" "member" ...
##  $ date              : Date[1:791956], format: "1-01-20" "1-01-20" ...
##  $ month             : chr [1:791956] "01" "01" "01" "01" ...
##  $ day               : chr [1:791956] "20" "20" "20" "20" ...
##  $ year              : chr [1:791956] "1" "1" "1" "1" ...
##  $ day_of_week       : chr [1:791956] "Saturday" "Saturday" "Saturday" "Saturday" ...
##  $ ride_length       : 'difftime' num [1:791956] 0 0 0 0 ...
##   ..- attr(*, "units")= chr "secs"
```

Convert "ride_length" from Factor to numeric so we can run calculations on the data

```
is.factor(all_trips$ride_length)
```

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

```
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
is.numeric(all_trips$ride_length)
```

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

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 # We will create a new version of the dataframe (v2) since data is being removed # https://www.datasciencemadesimple.com/delete-or-drop-rows-in-r-with-conditions-2/

```
all_trips_v2 <- all_trips[!(all_trips$start_station_name == "HQ QR" | all_trips$ride_length<0),]
```

#=====================================

STEP 4: CONDUCT DESCRIPTIVE ANALYSIS

#=====================================

Descriptive analysis on ride_length (all figures in seconds)

```
mean(all_trips_v2$ride_length) #straight average (total ride length / rides)
```

```
## [1] 114170.8
```

```
median(all_trips_v2$ride_length) #midpoint number in the ascending array of ride lengths
```

```
## [1] 0
```

```
max(all_trips_v2$ride_length) #longest ride
```

```
## [1] 946684800
```

```
min(all_trips_v2$ride_length) #shortest ride
```

```
## [1] 0
```

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

```r
summary(all_trips_v2$ride_length)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
##         0         0         0    114171         0 946684800
```

Compare members and casual users

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                696019.11
## 2                     member                 59395.69
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        0
## 2                     member                        0
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                946684800
## 2                     member                820454400
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        0
## 2                     member                        0
```

See the average ride time by each day for members vs casual users

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week,
FUN = mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Friday                742739.29
## 2                      member                   Friday                 68795.18
## 3                      casual                   Monday                485311.28
## 4                      member                   Monday                 53084.27
## 5                      casual                 Saturday                849717.58
## 6                      member                 Saturday                 55277.17
## 7                      casual                   Sunday                983708.54
## 8                      member                   Sunday                 45735.78
## 9                      casual                 Thursday                554798.24
## 10                     member                 Thursday                 56110.19
## 11                     casual                  Tuesday                758923.40
## 12                     member                  Tuesday                 56182.94
## 13                     casual                Wednesday                680982.83
## 14                     member                Wednesday                 83657.82
```

Notice that the days of the week are out of order. Let's fix that.

```
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday",
"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))
```

Now, let's run the average ride time by each day for members vs casual users

```
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week,
FUN = mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Sunday                983708.54
## 2                      member                   Sunday                 45735.78
## 3                      casual                   Monday                485311.28
## 4                      member                   Monday                 53084.27
## 5                      casual                  Tuesday                758923.40
## 6                      member                  Tuesday                 56182.94
## 7                      casual                Wednesday                680982.83
## 8                      member                Wednesday                 83657.82
## 9                      casual                 Thursday                554798.24
## 10                     member                 Thursday                 56110.19
## 11                     casual                   Friday                742739.29
## 12                     member                   Friday                 68795.18
## 13                     casual                 Saturday                849717.58
## 14                     member                 Saturday                 55277.17
```

install.packages("lubridate") library(lubridate) install.packages("dplyr") # For only dplyr library(dplyr)

OR (for tidyverse, which includes dplyr and other useful packages)

install.packages("tidyverse") library(tidyverse) all_trips_v2 <- all_trips_v2 %>% mutate(started_at = ymd_hms(started_at)) # Convert started_at to POSIXct format

analyze ridership data by type and weekday

```
library(lubridate)
library(dplyr)

all_trips_v2 %>%
  mutate(
    started_at = ymd_hms(started_at), # Ensure 'started_at' is in proper date-time format
    weekday = wday(started_at, label = TRUE) # Create weekday field
  ) %>%
  group_by(member_casual, weekday) %>% # Group by user type and weekday
  summarise(
    number_of_rides = n(), # Calculate number of rides
    average_duration = mean(ride_length, na.rm = TRUE) # Calculate average ride length
  ) %>%
  arrange(member_casual, weekday) # Sort by user type and weekday
```

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

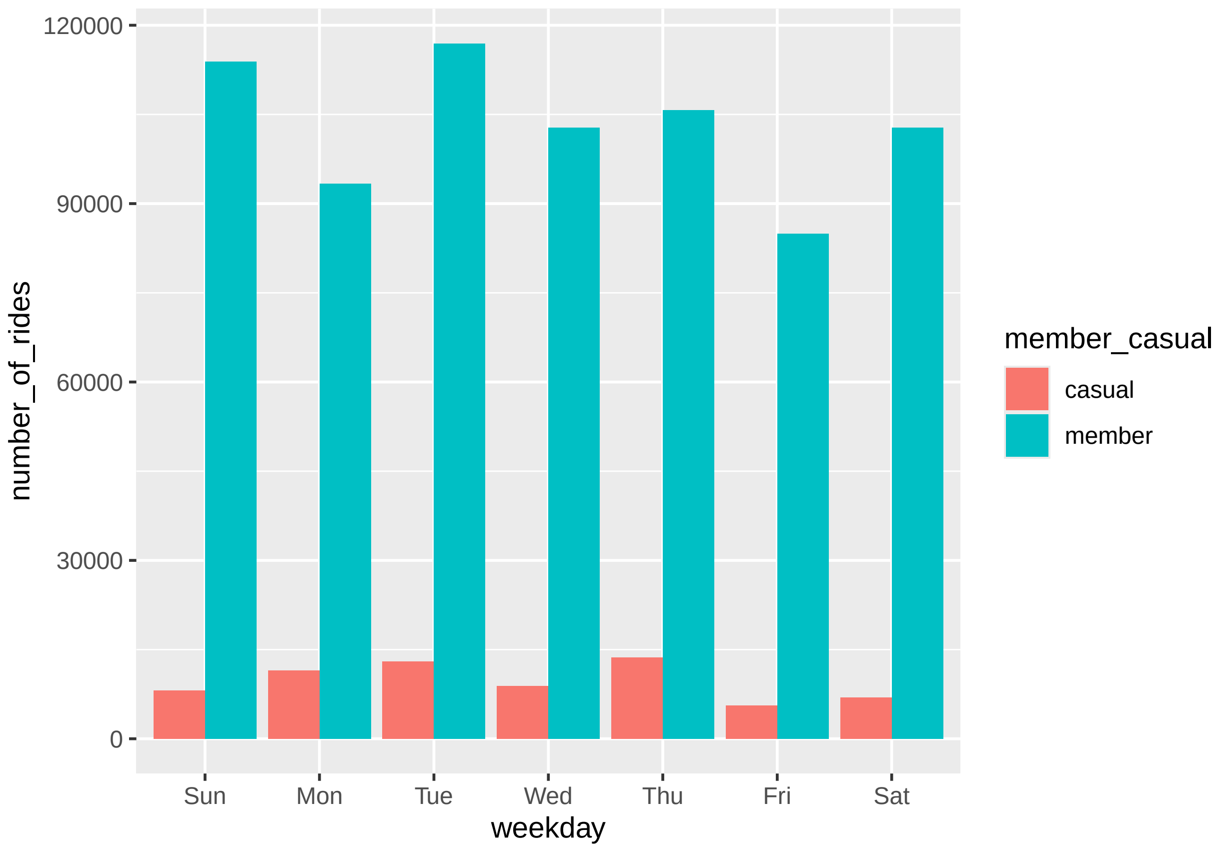
```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun                8119          983709.
##  2 casual        Mon               11510          485311.
##  3 casual        Tue               13041          758923.
##  4 casual        Wed                8913          680983.
##  5 casual        Thu               13668          554798.
##  6 casual        Fri                5574          742739.
##  7 casual        Sat                6981          849718.
##  8 member        Sun              113887           45736.
##  9 member        Mon               93325           53084.
## 10 member        Tue              116923           56183.
## 11 member        Wed              102747           83658.
## 12 member        Thu              105723           56110.
## 13 member        Fri               84904           68795.
## 14 member        Sat              102760           55277.
```

Let's visualize the number of rides by rider type

```
all_trips_v2 %>%
mutate(weekday = wday(started_at, label = TRUE)) %>%
group_by(member_casual, weekday) %>%

summarise(number_of_rides = n()
,average_duration = mean(ride_length)) %>%
arrange(member_casual, weekday) %>%
ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
geom_col(position = "dodge")
```

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

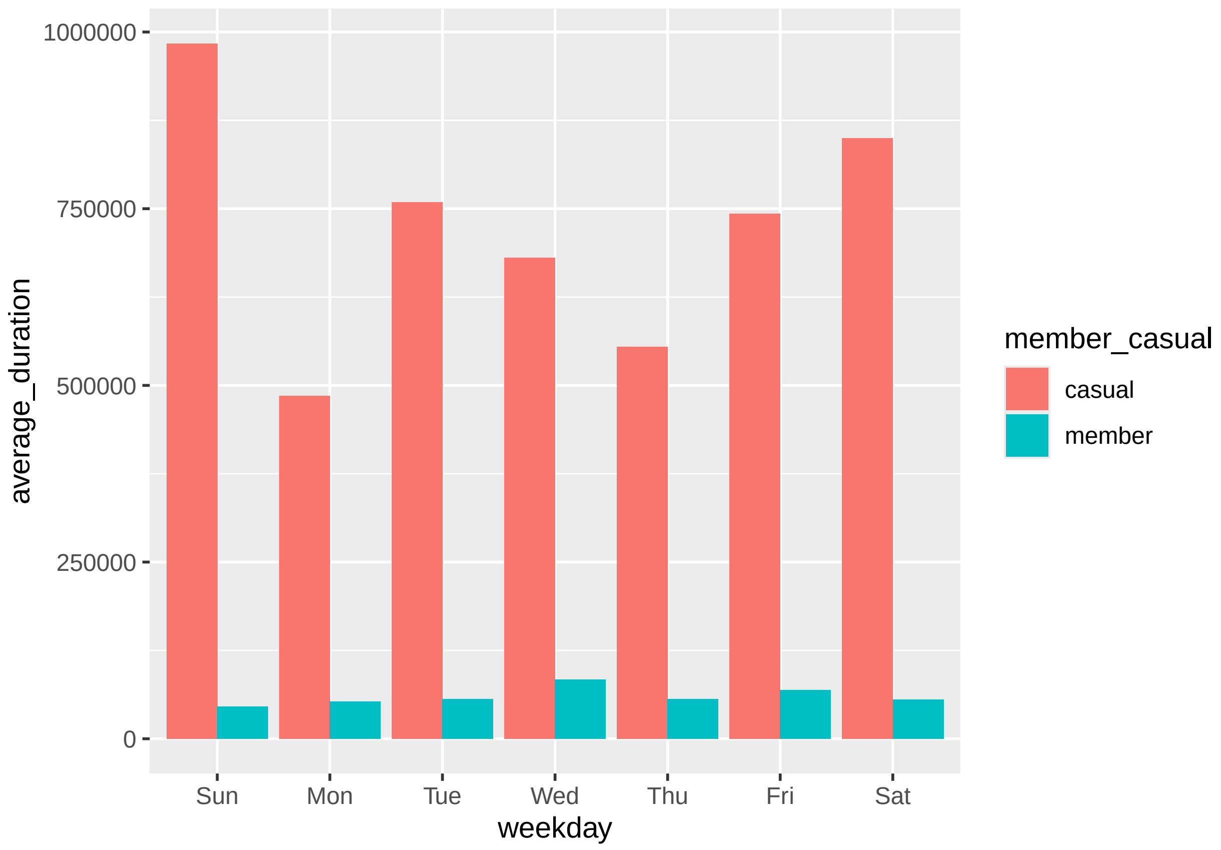


#

Let's create a visualization for average duration

```
all_trips_v2 %>%
mutate(weekday = wday(started_at, label = TRUE)) %>%
group_by(member_casual, weekday) %>%
summarise(number_of_rides = n()
,average_duration = mean(ride_length)) %>%
arrange(member_casual, weekday) %>%
ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
geom_col(position = "dodge")
```

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



1000000
750000
500000
250000
0
average_duration
Sun
Mon
Tue
Wed
Thu
Fri
Sat
weekday
member_casual
casual
member