Case Study Bikes

Mats Chattin

2025-06-28

library(tidyverse) #helps wrangle data

## ── 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 3.5.2 ✔ tibble 3.3.0  
## ✔ 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 become errors

# Use the conflicted package to manage conflicts  
library(conflicted)  
library(lubridate)  
  
# Set dplyr::filter and dplyr::lag as the default choices  
conflict\_prefer("filter", "dplyr")

## [conflicted] Will prefer dplyr::filter over any other package.

conflict\_prefer("lag", "dplyr")

## [conflicted] Will prefer dplyr::lag over any other package.

#===================== # STEP 1: COLLECT DATA #=====================

# # Upload Divvy datasets (csv files) here  
q1\_2019 <- read\_csv("Divvy\_Trips\_2019\_Q1.csv")

## Rows: 365069 Columns: 14  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (6): start\_time, end\_time, from\_station\_name, to\_station\_name, usertype...  
## dbl (6): trip\_id, bikeid, from\_station\_id, to\_station\_id, birthyear, day\_of...  
## num (1): tripduration  
## time (1): ride\_length  
##   
## 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")

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

## Rows: 426887 Columns: 15  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (7): ride\_id, rideable\_type, started\_at, ended\_at, start\_station\_name, ...  
## dbl (7): start\_station\_id, end\_station\_id, start\_lat, start\_lng, end\_lat, e...  
## time (1): ride\_length  
##   
## 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"   
## [13] "ride\_length" "day\_of\_week"

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" "ride\_length" "day\_of\_week"

# 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 × 14  
## ride\_id started\_at ended\_at rideable\_type tripduration start\_station\_id  
## <dbl> <chr> <chr> <dbl> <dbl> <dbl>  
## 1 21742443 1/1/2019 0:04 1/1/2019 … 2167 390 199  
## 2 21742444 1/1/2019 0:08 1/1/2019 … 4386 441 44  
## 3 21742445 1/1/2019 0:13 1/1/2019 … 1524 829 15  
## 4 21742446 1/1/2019 0:13 1/1/2019 … 252 1783 123  
## 5 21742447 1/1/2019 0:14 1/1/2019 … 1170 364 173  
## 6 21742448 1/1/2019 0:15 1/1/2019 … 2437 216 98  
## 7 21742449 1/1/2019 0:16 1/1/2019 … 2708 177 98  
## 8 21742450 1/1/2019 0:18 1/1/2019 … 2796 100 211  
## 9 21742451 1/1/2019 0:18 1/1/2019 … 6205 1727 150  
## 10 21742452 1/1/2019 0:19 1/1/2019 … 3939 336 268  
## # ℹ 365,059 more rows  
## # ℹ 8 more variables: start\_station\_name <chr>, end\_station\_id <dbl>,  
## # end\_station\_name <chr>, member\_casual <chr>, gender <chr>, birthyear <dbl>,  
## # ride\_length <time>, day\_of\_week <dbl>

# Inspect the dataframes and look for incongruencies

str(q1\_2019)

## spc\_tbl\_ [365,069 × 14] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ ride\_id : num [1:365069] 21742443 21742444 21742445 21742446 21742447 ...  
## $ started\_at : chr [1:365069] "1/1/2019 0:04" "1/1/2019 0:08" "1/1/2019 0:13" "1/1/2019 0:13" ...  
## $ ended\_at : chr [1:365069] "1/1/2019 0:11" "1/1/2019 0:15" "1/1/2019 0:27" "1/1/2019 0:43" ...  
## $ 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 & 18th St" "California Ave & Milwaukee 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 (\*)" "Western Ave & Fillmore St (\*)" "Clark St & Elm 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 ...  
## $ ride\_length : 'hms' num [1:365069] 00:06:30 00:07:21 00:13:49 00:29:43 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ day\_of\_week : num [1:365069] 3 3 3 3 3 3 3 3 3 3 ...  
## - 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(),  
## .. ride\_length = col\_time(format = ""),  
## .. day\_of\_week = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

str(q1\_2020)

## spc\_tbl\_ [426,887 × 15] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ ride\_id : chr [1:426887] "EACB19130B0CDA4A" "8FED874C809DC021" "789F3C21E472CA96" "C9A388DAC6ABF313" ...  
## $ rideable\_type : chr [1:426887] "docked\_bike" "docked\_bike" "docked\_bike" "docked\_bike" ...  
## $ started\_at : chr [1:426887] "1/21/2020 20:06" "1/30/2020 14:22" "1/9/2020 19:29" "1/6/2020 16:17" ...  
## $ ended\_at : chr [1:426887] "1/21/2020 20:14" "1/30/2020 14:26" "1/9/2020 19:32" "1/6/2020 16:25" ...  
## $ start\_station\_name: chr [1:426887] "Western Ave & Leland Ave" "Clark St & Montrose Ave" "Broadway & Belmont Ave" "Clark St & Randolph St" ...  
## $ 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" "Wilton Ave & Belmont Ave" "Fairbanks Ct & Grand Ave" ...  
## $ 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" ...  
## $ ride\_length : 'hms' num [1:426887] 00:07:31 00:03:43 00:02:51 00:08:49 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ day\_of\_week : num [1:426887] 3 5 5 2 5 6 6 6 6 6 ...  
## - 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(),  
## .. ride\_length = col\_time(format = ""),  
## .. day\_of\_week = col\_double()  
## .. )  
## - 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)

# 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  
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"   
## [10] "ride\_length" "day\_of\_week"

nrow(all\_trips) #How many rows are in data frame?

## [1] 791956

dim(all\_trips) #Dimensions of the data frame?

## [1] 791956 11

head(all\_trips) #See the first 6 rows of data frame. Also tail(all\_trips)

## # A tibble: 6 × 11  
## ride\_id started\_at ended\_at rideable\_type start\_station\_id start\_station\_name  
## <chr> <chr> <chr> <chr> <dbl> <chr>   
## 1 21742443 1/1/2019 … 1/1/201… 2167 199 Wabash Ave & Gran…  
## 2 21742444 1/1/2019 … 1/1/201… 4386 44 State St & Randol…  
## 3 21742445 1/1/2019 … 1/1/201… 1524 15 Racine Ave & 18th…  
## 4 21742446 1/1/2019 … 1/1/201… 252 123 California Ave & …  
## 5 21742447 1/1/2019 … 1/1/201… 1170 173 Mies van der Rohe…  
## 6 21742448 1/1/2019 … 1/1/201… 2437 98 LaSalle St & Wash…  
## # ℹ 5 more variables: end\_station\_id <dbl>, end\_station\_name <chr>,  
## # member\_casual <chr>, ride\_length <time>, day\_of\_week <dbl>

str(all\_trips) #See list of columns and data types (numeric, character, etc)

## tibble [791,956 × 11] (S3: tbl\_df/tbl/data.frame)  
## $ ride\_id : chr [1:791956] "21742443" "21742444" "21742445" "21742446" ...  
## $ started\_at : chr [1:791956] "1/1/2019 0:04" "1/1/2019 0:08" "1/1/2019 0:13" "1/1/2019 0:13" ...  
## $ ended\_at : chr [1:791956] "1/1/2019 0:11" "1/1/2019 0:15" "1/1/2019 0:27" "1/1/2019 0:43" ...  
## $ 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 & 18th St" "California Ave & Milwaukee 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 (\*)" "Western Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ member\_casual : chr [1:791956] "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...  
## $ ride\_length : 'hms' num [1:791956] 00:06:30 00:07:21 00:13:49 00:29:43 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ day\_of\_week : num [1:791956] 3 3 3 3 3 3 3 3 3 3 ...

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 ride\_length day\_of\_week   
## Length:791956 Min. :00:00:00.000000 Min. :1.00   
## Class :character 1st Qu.:00:05:28.000000 1st Qu.:3.00   
## Mode :character Median :00:08:57.000000 Median :4.00   
## Mean :00:14:01.812206 Mean :3.99   
## 3rd Qu.:00:15:10.000000 3rd Qu.:5.00   
## Max. :23:57:54.000000 Max. :7.00   
## NA's :117

# 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)

# 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")

all\_trips <- all\_trips %>%  
 mutate(  
 # Clean and convert both columns (assuming format like "1/1/2019 0:04")  
 started\_at = as.POSIXct(trimws(started\_at), format = "%m/%d/%Y %H:%M"),  
 ended\_at = as.POSIXct(trimws(ended\_at), format = "%m/%d/%Y %H:%M"),  
   
 # Now calculate ride\_length  
 ride\_length = difftime(ended\_at, started\_at, units = "mins")  
 )  
# 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 × 15] (S3: tbl\_df/tbl/data.frame)  
## $ ride\_id : chr [1:791956] "21742443" "21742444" "21742445" "21742446" ...  
## $ started\_at : POSIXct[1:791956], format: "2019-01-01 00:04:00" "2019-01-01 00:08:00" ...  
## $ ended\_at : POSIXct[1:791956], format: "2019-01-01 00:11:00" "2019-01-01 00:15:00" ...  
## $ 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 & 18th St" "California Ave & Milwaukee 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 (\*)" "Western Ave & Fillmore St (\*)" "Clark St & Elm St" ...  
## $ member\_casual : chr [1:791956] "member" "member" "member" "member" ...  
## $ ride\_length : 'difftime' num [1:791956] 420 420 840 1800 ...  
## ..- attr(\*, "units")= chr "secs"  
## $ day\_of\_week : chr [1:791956] "Saturday" "Saturday" "Saturday" "Saturday" ...  
## $ date : Date[1:791956], format: "0001-01-20" "0001-01-20" ...  
## $ month : chr [1:791956] "01" "01" "01" "01" ...  
## $ day : chr [1:791956] "20" "20" "20" "20" ...  
## $ year : chr [1:791956] "0001" "0001" "0001" "0001" ...

# 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] 1189.154

median(all\_trips\_v2$ride\_length) #midpoint number in the ascending array of ride lengths

## [1] 540

max(all\_trips\_v2$ride\_length) #longest ride

## [1] 10628400

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  
summary(all\_trips\_v2$ride\_length)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0 360 540 1189 900 10628400

# Compare members and casual users  
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 5370.7474  
## 2 member 795.1109

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 1380  
## 2 member 480

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 10628400  
## 2 member 6096420

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  
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 6811.7619  
## 2 member Friday 737.7728  
## 3 casual Monday 3606.5197  
## 4 member Monday 707.6045  
## 5 casual Saturday 5788.8659  
## 6 member Saturday 904.0163  
## 7 casual Sunday 5058.1153  
## 8 member Sunday 786.9422  
## 9 casual Thursday 6547.4296  
## 10 member Thursday 825.5676  
## 11 casual Tuesday 3464.3096  
## 12 member Tuesday 859.8252  
## 13 casual Wednesday 5491.4342  
## 14 member Wednesday 851.4219

# 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 5058.1153  
## 2 member Sunday 786.9422  
## 3 casual Monday 3606.5197  
## 4 member Monday 707.6045  
## 5 casual Tuesday 3464.3096  
## 6 member Tuesday 859.8252  
## 7 casual Wednesday 5491.4342  
## 8 member Wednesday 851.4219  
## 9 casual Thursday 6547.4296  
## 10 member Thursday 825.5676  
## 11 casual Friday 6811.7619  
## 12 member Friday 737.7728  
## 13 casual Saturday 5788.8659  
## 14 member Saturday 904.0163

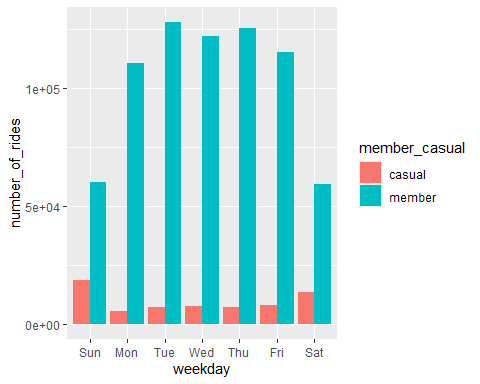
# analyze ridership data by type and weekday  
all\_trips\_v2 %>%   
 mutate(weekday = wday(started\_at, label = TRUE)) %>% #creates weekday field using wday()  
 group\_by(member\_casual, weekday) %>% #groups by usertype and weekday  
 summarise(number\_of\_rides = n() #calculates the number of rides and average duration   
 ,average\_duration = mean(ride\_length)) %>% # calculates the average duration  
 arrange(member\_casual, weekday) # sorts

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

## # A tibble: 14 × 4  
## # Groups: member\_casual [2]  
## member\_casual weekday number\_of\_rides average\_duration  
## <chr> <ord> <int> <dbl>  
## 1 casual Sun 18652 5059.  
## 2 casual Mon 5591 4751.  
## 3 casual Tue 7311 4561.  
## 4 casual Wed 7690 4480.  
## 5 casual Thu 7147 8450.  
## 6 casual Fri 8013 6089.  
## 7 casual Sat 13473 4946.  
## 8 member Sun 60197 972.  
## 9 member Mon 110430 822.  
## 10 member Tue 127974 769.  
## 11 member Wed 121902 712.  
## 12 member Thu 125228 707.  
## 13 member Fri 115168 797.  
## 14 member Sat 59413 974.

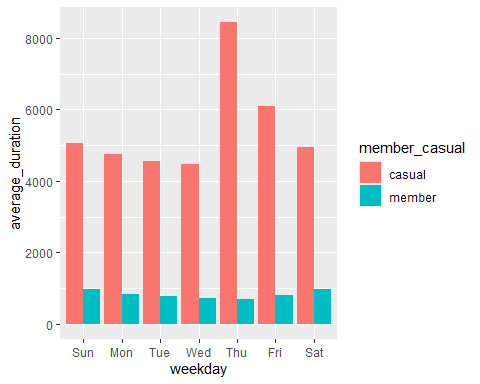
# Let's visualize the number of rides by rider type  
all\_trips\_v2 %>%   
 mutate(weekday = wday(started\_at, label = TRUE)) %>%   
 group\_by(member\_casual, weekday) %>%   
 summarise(number\_of\_rides = n()  
 ,average\_duration = mean(ride\_length)) %>%   
 arrange(member\_casual, weekday) %>%   
 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.

 #================================================= # STEP 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\_trips\_v2$ride\_length ~ all\_trips\_v2$member\_casual + all\_trips\_v2$day\_of\_week, FUN = mean)  
#write.csv(counts, file = 'avg\_ride\_length.csv')

# Prepare the summary data  
# mutate(weekday = wday(started\_at, label = TRUE, week\_start = 1)) %>% # week\_start=1 makes Monday first day  
# group\_by(member\_casual, weekday) %>%  
# summarise(number\_of\_rides = n(), .groups = "drop") %>%  
# arrange(member\_casual, weekday)  
  
# Export to CSV  
#write.csv(rides\_count\_summary, file = "rides\_count\_by\_member\_weekday.csv", row.names = FALSE)