Coursera Capstone project

Igor Sorochan

2023-03-06

Contents

Coursera is the global online learning platform that offers anyone, anywhere access to online courses and degrees from world-class universities and companies.

Google is a multinational corporation specializing in internet-related services and products.

Case Study: How Does a Bike-Share navigate speedy Success?

1. ASK

Scenario

I'm a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago.

Lily Moreno, the director of marketing, believes the company's future success depends on maximizing the number of annual memberships. Therefore, my team wants to understand

How casual riders and annual members use Cyclistic bikes differently?

Settings

About the company

Cyclistic is bike share system across Chicago and Evanston. Cyclistic provides residents and visitors with a convenient, fun and affordable transportation option for getting around and exploring Chicago.

Cyclistic, like other bike share systems, consists of a fleet of specially-designed, sturdy and durable bikes that are locked into a network of docking stations throughout the region. The bikes can be unlocked from one

station and returned to any other station in the system. People use bike share to explore Chicago, commute to work or school, run errands, get to appointments or social engagements, and more.

Cyclistic is available for use 24 hours/day, 7 days/week, 365 days/year, and riders have access to all bikes and stations across the system.

Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as casual riders. Customers who purchase annual memberships are Cyclistic members.

Cyclistic's finance analysts have concluded that annual members are much more profitable than casual riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno believes that **maximizing the number of annual members** will be **key to future growth**. Rather than creating a marketing campaign that targets all-new customers, Moreno believes there is a very good chance **to convert casual riders into members**. She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs. Moreno has set a clear goal: **Design marketing strategies aimed at converting casual riders into annual members**. In order to do that, however, the marketing analyst team needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Project stakeholders

Primary stakeholders:

- Cyclistic executive team
- Lily Moreno, the director of marketing

Secondary stakeholders:

- Cyclistic marketing analytics team

From these insights, my team will design a new marketing strategy to convert casual riders into annual members.

Questions my team has to answer:

1. How casual riders and annual members use Cyclistic bikes differently?
2. Why would casual riders buy Cyclistic annual memberships?
3. How can Cyclistic use digital media to influence casual riders to become members?

2. PREPARE

Data location

Lyft Bikes and Scooters, LLC ("Bikeshare") operates the City of Chicago's ("City") Divvy bicycle sharing service. Bikeshare and the City are committed to supporting bicycling as an alternative transportation option. As part of that commitment, the City permits Bikeshare to make certain Divvy system data owned by the City ("Data") available to the publicData organization.

The data has been made available by Motivate International Inc. under this license. It is a **First-party data.**

We'll use that Data in Case study as Cyclistic's historical trip data.

Data is reliable, original, comprehensive, current and cited.

Data credibility and data bias

The Data itself is a First-party data and it is credible and has no evidence of bias of any kind.

Data ethics

There is no any personal information that we can associate with real customers.

Each trip is anonymized.

We accept all limitations on Data usage noted in "Prohibited conduct" in **Data License Agreement**.

Data tools

At first glance, the overall dataset would be **Large** enough to process (mlns of rows) and will force any available spreadsheet software to struggle, so our team decided to use R to handle it.

Let's do that.

Setting the environment.

```
library(tidyverse)
library(dplyr)
library(tidyr)
library(janitor)
library(lubridate)
library(ggplot2)
library(plotly)
library(scales)
library(skimr)
library(DT)
library(crosstable)
library(flextable)
library(sf)
options(dplyr.summarise.inform = FALSE)
options(max.print=100)
# options(ggplot2.discrete.colour= c("#FFDB6D", "#00AFBB"))
```

Take a mention on the current working folder in output of `getwd()` and if redefine it if needed:

```
getwd()
```

```
## [1] "/Users/velo1/Documents/Portfolio/Case_study"
```

```
# uncomment and redefine it if needed (use your actual folder)
# setwd("../Coursera/Case_study/")
```

Original data lives here.

We've selected appropriate .zip files from 01-Jan-2022 till 30-Jan2023 (13 months of data were available as of the date of this report) and store them locally at `zip_dir` folder.

Defining the directory where all original zip files are placed and defining `report_caption` :

```
zip_dir<- paste0(getwd(),"/Divvy_tripdata/")
report_caption <- "Jan 2022 - Jan 2023"
```

Defining the directory csv files to extract:

```
csv_Dir<- paste0(getwd(),"/Divvy_tripdata/csv/")
```

Unzipping all files and put them to `csv_dir`:

```
files <- list.files(path = zip_dir, pattern = "*.zip")
for (i in files) {
  unzip(paste0(zip_dir,i), exdir=csv_Dir)
}
```

Reading csv files and nesting them into Large list (almost 2Gb).

Wait a little bit, please. Need a minute to execute:

```
temp <- list.files(path = csv_Dir, pattern = "*.csv")
myfiles <- lapply(paste0(csv_Dir,temp), read.csv)
```

Thus, all the data we need is collected in one place.

We haven't performed any data manipulations so far.

Let's go further.

3. PROCESS

Data transformations

Do the data frames have the same columns and types? Let's check it out:

```
janitor::compare_df_cols_same(myfiles)
```

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

TRUE - means that all columns in all data frames have appropriate names and types of data.

Finally forming united table. Binding data frames by row, making a longer result (few seconds to execute, don't panic):

```
raw_df <- dplyr::bind_rows(myfiles)
```

Let's look in:

```
head(raw_df,5)
```

```
##            ride_id rideable_type          started_at            ended_at
## 1 C2F7DD78E82EC875 electric_bike 2022-01-13 11:59:47 2022-01-13 12:02:44
## 2 A6CF8980A652D272 electric_bike 2022-01-10 08:41:56 2022-01-10 08:46:17
## 3 BD0F91DFF741C66D  classic_bike 2022-01-25 04:53:40 2022-01-25 04:58:01
## 4 CBB80ED419105406  classic_bike 2022-01-04 00:18:04 2022-01-04 00:33:00
## 5 DDC963BFDDA51EEA  classic_bike 2022-01-20 01:31:10 2022-01-20 01:37:12
##              start_station_name start_station_id             end_station_name
## 1      Glenwood Ave & Touhy Ave              525          Clark St & Touhy Ave
## 2      Glenwood Ave & Touhy Ave              525          Clark St & Touhy Ave
## 3 Sheffield Ave & Fullerton Ave     TA1306000016 Greenview Ave & Fullerton Ave
## 4      Clark St & Bryn Mawr Ave     KA1504000151     Paulina St & Montrose Ave
## 5   Michigan Ave & Jackson Blvd     TA1309000002        State St & Randolph St
##   end_station_id start_lat start_lng  end_lat   end_lng member_casual
## 1         RP-007  42.01280 -87.66591 42.01256 -87.67437        casual
## 2         RP-007  42.01276 -87.66597 42.01256 -87.67437        casual
## 3   TA1307000001  41.92560 -87.65371 41.92533 -87.66580        member
## 4   TA1309000021  41.98359 -87.66915 41.96151 -87.67139        casual
## 5   TA1305000029  41.87785 -87.62408 41.88462 -87.62783        member
```

```
raw_df$started_at = ymd_hms(raw_df$started_at)
raw_df$ended_at = ymd_hms(raw_df$ended_at)
```

We need to convert date related columns to appropriate type:

Adding a calculated columns It is obvious that we'll need a duration information in our analysis.

Let's add a calculated column `trip_duration` that counts trip duration in **minutes**.

```
raw_df[,"trip_duration"] <- as.numeric(as.duration(raw_df$ended_at - raw_df$started_at), "minutes")
```

Data integrity

Tables naming

Tables used in the project	Table purpose	Table dimensions
raw_df	untouched imported data	5,858,018 x 13
trip_df	store filtered valid data, add calculated columns	5,548,446 x 15

Tables used in the project	Table purpose	Table dimensions
stations_df	station dictionary	1,716 x 6
stations_df2	temporary table	
df	tibble, cleaned data	5,548,446 x 16

1. **Domain integrity:** Domain integrity ensures that each value in a column falls within the permissible range of the domain of that column. Moreover, the conditions for default and null values must also be met.
2. **Entity integrity:** Entity integrity ensures that each row of the database has a non-null unique primary key.
3. **Referential integrity:** Referential integrity ensures a valid relationship between two tables by checking the relationship between the foreign key and primary key in those tables.

Let's evaluate values domain for integrity:

```
skim_without_charts(raw_df)
```

Table 3: Data summary

Name	raw_df
Number of rows	5858018
Number of columns	14
Column type frequency:	
character	7
numeric	5
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1	16	16	0	5858018	0
rideable_type	0	1	11	13	0	3	0
start_station_name	0	1	0	64	859785	1682	0
start_station_id	0	1	0	44	859785	1314	0
end_station_name	0	1	0	64	920582	1700	0
end_station_id	0	1	0	44	920582	1319	0
member_casual	0	1	6	6	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_lat	0	1	41.90	0.05	41.64	41.88	41.90	41.93	45.64
start_lng	0	1	-87.65	0.03	-87.84	-87.66	-87.64	-87.63	-73.80
end_lat	5985	1	41.90	0.07	0.00	41.88	41.90	41.93	42.37

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
end_lng	5985	1	-87.65	0.11	-88.14	-87.66	-87.64	-87.63	0.00
trip_duration	0	1	19.23	175.38	-10353.35	5.75	10.15	18.25	41387.25

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2022-01-01 00:00:05	2023-01-31 23:56:09	2022-07-25 21:18:11	4924734
ended_at	0	1	2022-01-01 00:01:48	2023-02-04 04:27:03	2022-07-25 21:40:04	4937657

Data set consists of 5.858.018 observations with 14 characteristics (columns).

Time scope of all trips is relevant to the scope of business problem.

Quantitative (numeric and POSIXct) data:

Table 7: Qualitative(character)data:

Attribute	min value	max value	number of NA	Domain integrity	Entity integrity	Notes
started_at	2022-01-01 00:00:05	2023-01-31 23:56:09	0	+	+	
ended_at	2022-01-01 00:01:48	2023-02-04 04:27:03	0	+	+	
trip_duration	-10353.35	41387.25	0	fault	+	negatives, very high std dev
start_lat	41.64	45.63503	0	fault	+	too big range for a city
start_lng	-87.8	-73.796	0	fault	+	too big range for a city
end_lat	0.00	42.37000	5985	fault	fault	zeros
end_lng	-88.1	0.00000	5985	fault	fault	zeros

Attribute	Number of empty entries	Number of unique	Domain integrity	**Entity integrity**	Notes
ride_id	0	5858018	+	+	+
rideable_type	0	3	+	+	+
start_station_name	859785	1682	+	fault	number of stations is greater than station IDs
start_station_id	859785	1314	+	+	+
end_station_name	920582	1700	+	fault	number of stations is greater than station IDs
end_station_id	920582	1319	+	+	+
member_casual	0	2	+	+	+

How many observations are affected with empty start and finish points?

```
raw_df %>%
  filter(start_station_name == "" | end_station_name == ""  ) %>%
  nrow()
```

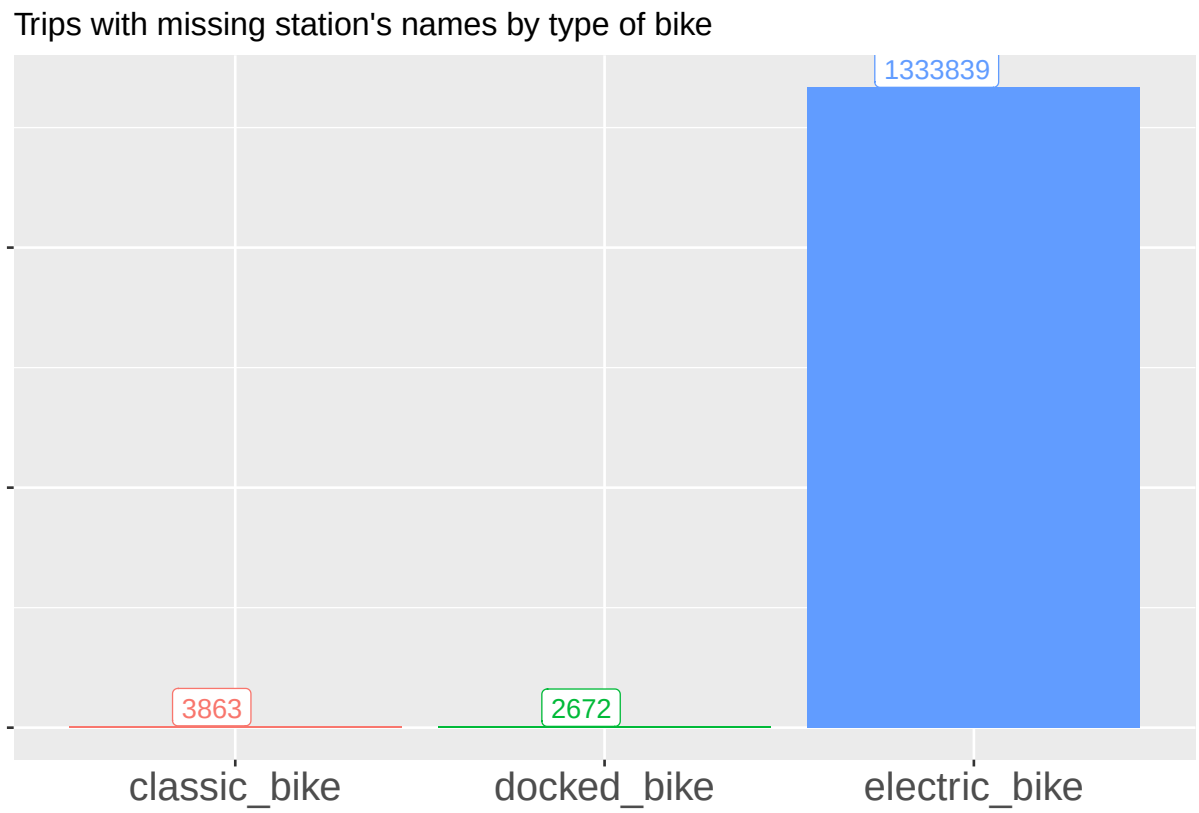
```
## [1] 1340374
```

A lot of data affected. We might manage to restore some observations station names by geo data if it possible. This will potentially lead us to more comprehensive reports with geo data.

Data issues

Most of the lost of station names fell on electric bikes:

```
raw_df %>%
  filter(start_station_name == "" |
           end_station_name == "" ) %>%
  group_by(rideable_type) %>%
  summarise(sum= n()) %>%
  ggplot(aes(rideable_type, y= sum )) +
  geom_col(aes(fill= rideable_type), show.legend = FALSE) +
  scale_y_continuous(labels = label_comma()) +
  geom_label(aes(y = sum, x = rideable_type, label = sum,
                 color = rideable_type), hjust =0.8, vjust = 0, show.legend = FALSE)+
  labs(title = "Trips with missing station's names by type of bike",
       caption = report_caption,
       x ="", y= "",
       fill='Type of rider')+
  theme(axis.text.y=element_blank(),
        axis.text = element_text(size = 16) )
```



The reason could be a complete discharge of the batteries.

100 observations have negative trip duration.

```
raw_df %>%
  filter(trip_duration < 0) %>%
  group_by(rideable_type) %>%
  summarise(sum= n()) %>%
  as_flextable()
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

rideable_type	sum
character	integer
classic_bike	28
electric_bike	72

The data is not clean

1. **Domain integrity issue.** Standard deviation of `trip_duration` is unreasonably high: (175 min, while mean =19 min). This clearly indicates the presence of extreme outliers.

   Some `started_at` is greater than `ended_at`. It means negative `trip_duration` .

2. **Entity integrity issue.** 5985 trips (0.1% of all trips) have no gps data at all. This concern we might take into account if we'll plan to investigate routes. Some observations at end stations include zeros.

   1.340.374 (23 % of all trips) of data concerning station's names is empty. 99,5 % among them are electric bikes.

3. **Referential integrity issue.** Station's naming is not consistent.

   The number of station identifiers is less than their names.

Solutions:

1. Exclude data with negative `trip_duration` (100 observations)
2. Exclude data with too big `trip_duration` (greater than 1499 minutes, ~25 hours) and with simultaneously empty `end_station_name`
3. Exclude classic bike's trips with missing station's names (3863 observations)
4. Restore stations ID by geo data where possible. Maybe this won't screw the overall patterns but it's better to restore the missing data.

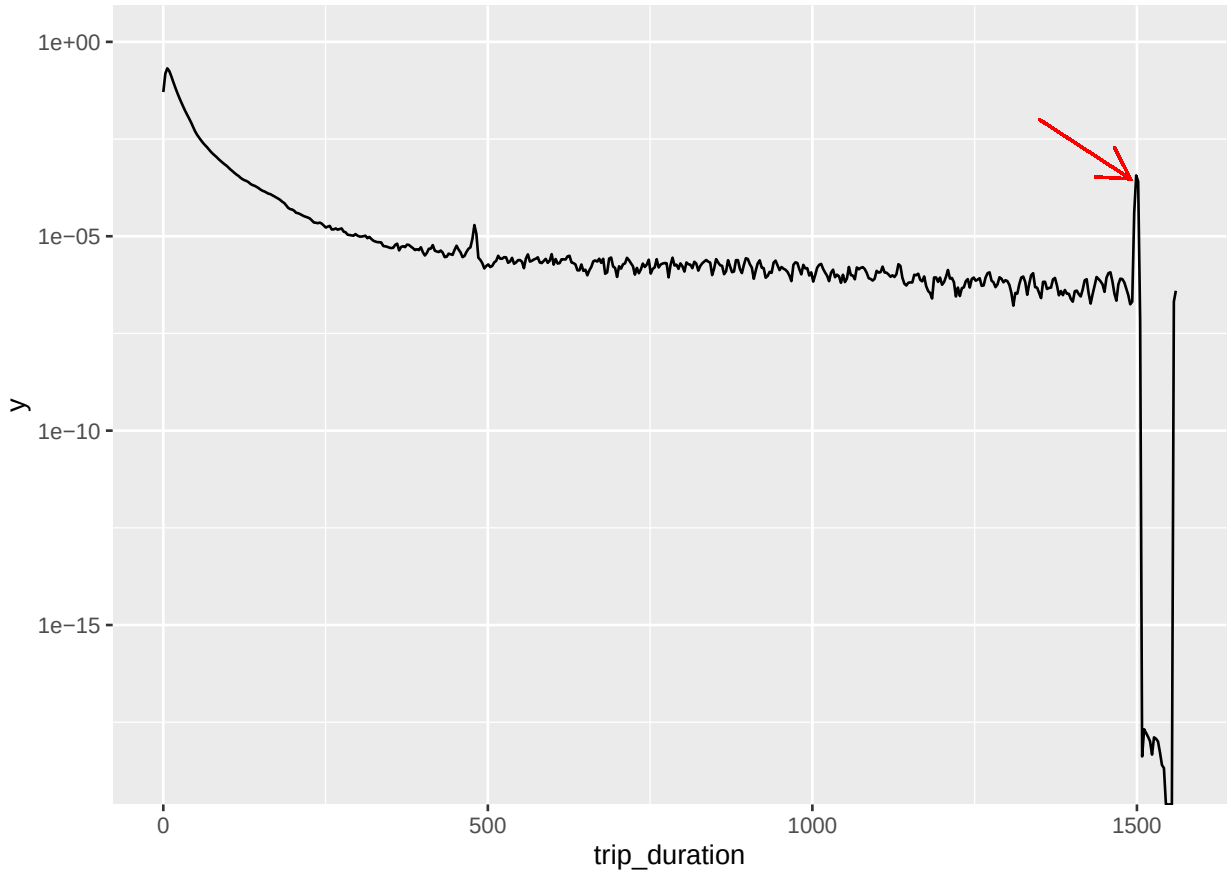
Dropping irrelevant data The data has to be processed.

Let's take a look on distribution of "strange" observations
where `trip_duration > 1499 & rideable_type != "docked_bike"`

```
raw_df %>%

  filter(rideable_type != "docked_bike") %>%
  filter(trip_duration > 0) %>%


  ggplot() +
  geom_density(aes(x=trip_duration)) +
  scale_y_log10() +
  geom_segment(aes(x = 1.35e3, y = 1e-2, xend = 1.49e3, yend = 3e-4),color = "red",
                lineend = "round", linejoin = "mitre",
                 arrow = arrow(length = unit(0.5, "cm")))
```



The surge at mark 25h(1500 minutes) **might be a service notation**, e.g. bikes that had been left out of parking stations, fully discharged, defected or stolen bikes.

Finally, we will **consider a trip valid** if it satisfies the following conditions:

1. `rideable_type != "docked_bike"` to exclude service observations.
2. `trip_duration` $>$ 1 minute (exclude any trips that were below 60 seconds in length (potentially false starts or users trying to re-dock a bike to ensure it was secure).
3. `trip_duration` $<$ 1499 minutes(25 hours). (NOTE: If you do not return a bike within a 24-hour period, you may be charged a lost or stolen bike fee of $250 (plus tax).
4. `end_station_name` is not empty

```
raw_df %>%
  filter(rideable_type == "docked_bike" | is.na(end_lat) |
           trip_duration < 1 | trip_duration > 1499 ) %>%
  nrow()
```

```
## [1] 308332
```

This assumption will exclude 308,332 (5.2 %) observations from "dirty" data. This is acceptable.

We keep the source untouched and put valid data to `trip_df` :

```
trip_df <- raw_df %>%
  filter(rideable_type != "docked_bike" & !is.na(end_lat)) %>%
  filter(trip_duration > 1 &
           trip_duration < 1499 )
```

Restoring missed station names

The logic:
1. Build a data frame (`stations_df`) with all station's names and appropriate geo data
2. Restore station's names according to the dictionary geo data.

Let's try.

Station's names To avoid duplicates we round the geo data to 4 decimals places (11 m accuracy).

```
# setting geo data accuracy (decimal places)
geo_acc <- 4

# parcing end stations
stations_df <- trip_df %>%
    filter(end_station_name != "" &
             (end_lat != 0 | !is.na(end_lat) ) ) %>%
  group_by(end_station_name) %>%

  # we use means here to increase geo data accuracy of stations
  summarise(latit = mean(end_lat),
            lngit = mean(end_lng)) %>%
  unique()

# all columns with postfix '2' at the end will serve later as joining instances
stations_df[,"end_lat2"] = round(stations_df$latit,geo_acc)
stations_df[,"end_lng2"] = round(stations_df$lngit,geo_acc)

# adding station IDs
stations_df <-
  left_join(stations_df, trip_df, by = c("end_station_name"), multiple = "first") %>%
  select("end_station_id",
         "end_station_name",
         "latit",
         "lngit",
         "end_lat2",
         "end_lng2")
# renaming
stations_df <-
  rename(stations_df, all_of( c(station_name = "end_station_name",
                                station_id =  "end_station_id")) )


# parcing start stations
# stations_df2 - start_station data frame
stations_df2 <- trip_df %>%
    filter(start_station_name != "" ) %>%
```

```
  group_by(start_station_name) %>%
  # all columns with '2' at the end will serve later as joining instances
  # we use means here to increase geo data accuracy
  summarise(latit = mean(start_lat),
            lngit = mean(start_lng) ) %>%
  unique()

# all columns with postfix '2' at the end will serve later as joining instances
stations_df2[,"end_lat2"] = round(stations_df2$latit,geo_acc)
stations_df2[,"end_lng2"] = round(stations_df2$lngit,geo_acc)


# adding station IDs
stations_df2 <-
  left_join(stations_df2, trip_df, by = c("start_station_name"), multiple = "first") %>%
  select("start_station_id",
         "start_station_name",
         "latit",
         "lngit",
         "end_lat2",
         "end_lng2")
# renaming
stations_df2 <-
  rename(stations_df2, all_of( c(station_name = "start_station_name",
                                 station_id =  "start_station_id")) )

stations_df <-
  bind_rows(stations_df, stations_df2) %>%
  dplyr::distinct(station_name, .keep_all = TRUE)
```

Restoring station names

```
# service columns with rounded geo data
trip_df[,"end_lat2"] <- round(trip_df$end_lat,geo_acc)
trip_df[,"end_lng2"] <- round(trip_df$end_lng,geo_acc)
trip_df[,"restored"] <- NA

trip_df <-
  left_join(trip_df, stations_df, by = c("end_lat2","end_lng2"), multiple = 'first')

# logging restoration
trip_df$restored =
  ifelse(trip_df$end_station_name == "" & !is.na(trip_df$station_name),
         "end_station_name",
         ifelse(trip_df$end_station_id == "" & !is.na(trip_df$station_id),
                "end_station_id", NA)
  )
# adding end_station_name
trip_df$end_station_name =
  ifelse(trip_df$end_station_name == ""& !is.na(trip_df$station_name),
```

```
         trip_df$station_name,
         trip_df$end_station_name)
# adding end_station_id
trip_df$end_station_id =
  ifelse(trip_df$end_station_id == ""& !is.na(trip_df$station_id),
         trip_df$station_id,
         trip_df$end_station_id)

# dropping service joining columns
trip_df <- within(trip_df, rm("end_lat2",
                                      "end_lng2",
                                      "station_id",
                                      "station_name" ))
```

Restoring `end_station_name` and `end_station_id`.

```
trip_df[,"start_lat2"] <- round(trip_df$start_lat,geo_acc)
trip_df[,"start_lng2"] <- round(trip_df$start_lng,geo_acc)

stations_df <-
  rename(stations_df, all_of( c(start_lat2 = "end_lat2",
                                start_lng2 =  "end_lng2")) )

trip_df <-
  left_join(trip_df, stations_df, by = c("start_lat2","start_lng2"), multiple = 'first')
# logging restoration
trip_df$restored =
  ifelse(trip_df$start_station_name == "" & !is.na(trip_df$station_name),
         "start_station_name",
         ifelse(trip_df$start_station_id == "" & !is.na(trip_df$station_id),
                "start_station_id", NA)
  )

# adding start_station_name
trip_df$start_station_name =
  ifelse(trip_df$start_station_name == "" & !is.na(trip_df$station_name),
         trip_df$station_name,
         trip_df$start_station_name)
# adding start_station_id
trip_df$start_station_id =
  ifelse(trip_df$start_station_id == "" & !is.na(trip_df$station_id),
         trip_df$station_id,
         trip_df$start_station_id)

# dropping joining columns
trip_df <- within(trip_df, rm(
                                  "start_lat2",
                                  "start_lng2",
                                  "station_id",
                                  "station_name",
```

```
                                    "latit.x",
                                    "lngit.x",
                                    "latit.y",
                                    "lngit.y"))
nrow(filter(trip_df, !is.na(restored) ) )
```

Restoring `start_station_name` and `start_station_id`.

```
## [1] 312438
```

We've managed to restore station's names within 312,438 observations.

Converting cleaned data frame to tibble:

```
df <- as_tibble(trip_df)
```

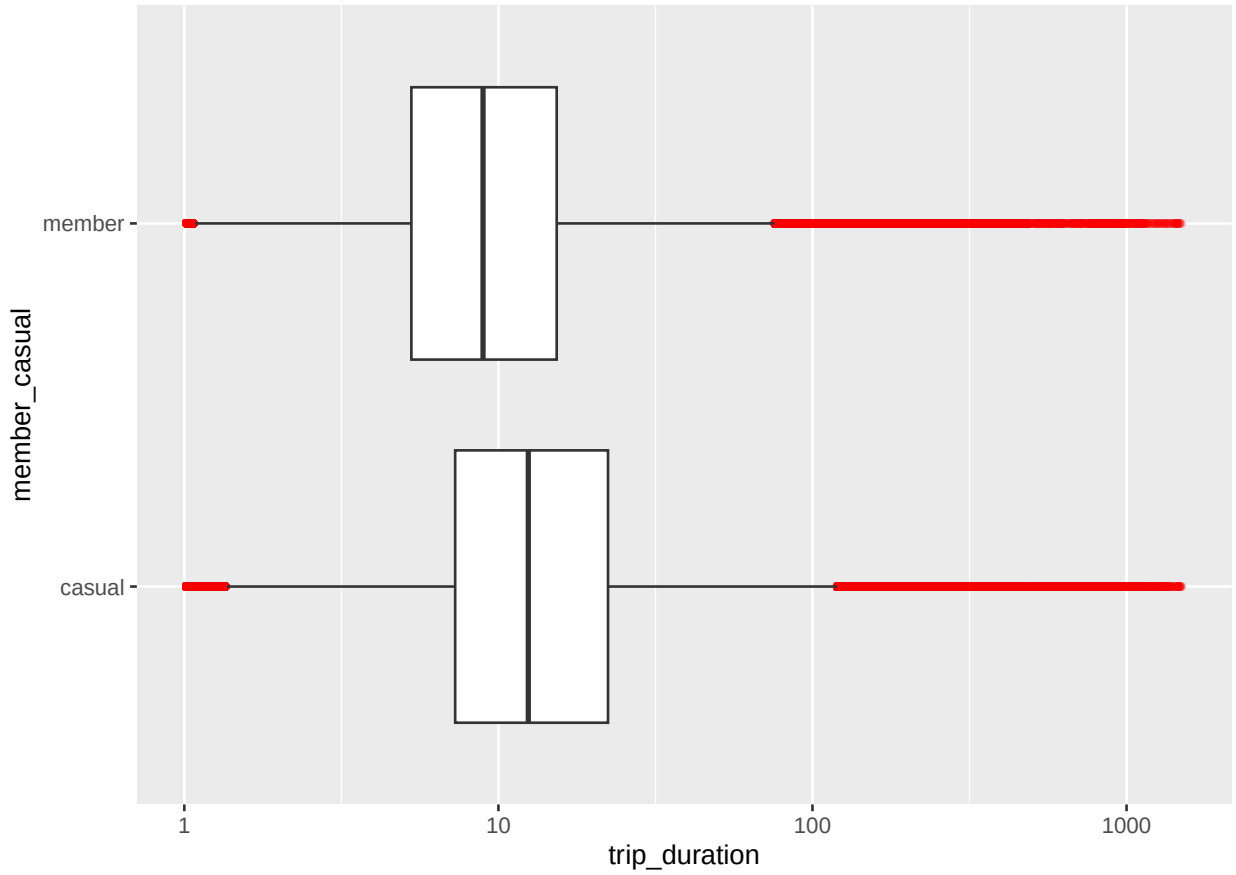
Adding a trip the day of the week :

```
df[, "weekday"] <- wday(df$started_at, label = TRUE)
```

4. ANALYSE

Assumptions and constraints Let's take a look at distribution of trips over data set:

```
df %>%
  ggplot(aes(x=trip_duration, y= member_casual)) +
  geom_boxplot( outlier.colour = "red",
                outlier.stroke = 0,
                outlier.alpha = 0.1,
                varwidth = FALSE) +
  scale_x_log10()
```



There are too many outliers that might skew overall statistics.

We constrain data with :

- the upper limit to `mean + 5sigma` = 142 minutes

```
(trip_limit <- mean(df$trip_duration) + 5* sd(df$trip_duration) )
```

```
## [1] 142.4059
```

It removes only 17.837 rows (0.3 % of all trips). So our conclusions will be based on 99.7 % of "clean" data.

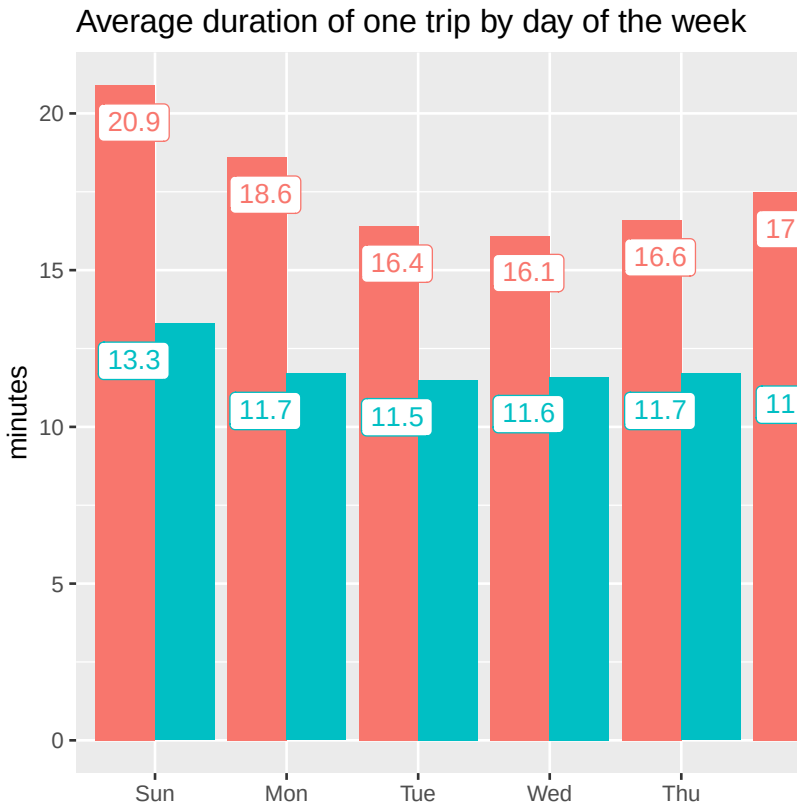
```
count(filter(df, trip_duration > trip_limit))
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1 17837
```

```
filter(df, trip_duration < trip_limit) -> df
```

Descriptive statistics. Trip durations

```
df %>%
  group_by(member_casual, weekday) %>%
  summarise(ride_mean = round(mean(trip_duration),1))  %>%
  # summarise(ride_mean = round(mean(trip_duration),1))  %>%
  ggplot() +
  geom_col(aes(y = ride_mean, x = weekday, fill = member_casual ),
           position = "dodge") + #, show.legend = FALSE, , alpha = 0.8
  labs(title = "Average duration of one trip by day of the week",
       caption = report_caption,
       x ="", y= "minutes",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma()) +
geom_label(aes(y = ride_mean, x = weekday, label = ride_mean,
               color = member_casual), hjust =0.8, vjust = 1.5, show.legend = FALSE)
```



Average duration of one trip throughout a year

```
df %>%
  group_by(member_casual) %>%
  summarise(ride_mean = round(mean(trip_duration),1) ) %>%
  as_flextable()  %>%
set_caption(paste("Trip average duration.", report_caption)) %>% delete_part("header")
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

Table 10: Trip average duration. Jan 2022 - Jan 2023

casual	18.4
member	12.1

```
# df %>%
#   crosstable(c(" "= trip_duration), by=c(rider=member_casual), funs=c("Average duration"= mean),  sho
#   as_flextable(compact=TRUE, keep_id=FALSE)  %>%
#   set_caption(paste("Trip average duration.", report_caption))  #%>% delete_part("header")
```

Insights:

- Average duration of **casual riders is significantly higher** (+ 51 %)
- Trips on **Wednesdays** are 17 % shorter than on weekends among all customers.

The maximum ride duration Finding the maximum ride duration we assume:

1. Bike is not docked
2. Start and end stations are defined
3. We'll look up through the source data

```
raw_df  %>%
  filter(rideable_type != "docked_bike") %>%
  filter(start_station_name !="" & end_station_name != "") %>%
  # group_by(rideable_type) %>%
  # summarise(max=as.duration(max(ended_at - started_at))) %>%
  crosstable(c(" "= trip_duration), by=c(bycicle = rideable_type), funs=c("Max trip in minutes"= max),
  as_flextable(compact=TRUE, keep_id=FALSE)  %>%
  set_caption(paste("The maximum ride duration.", report_caption))
```

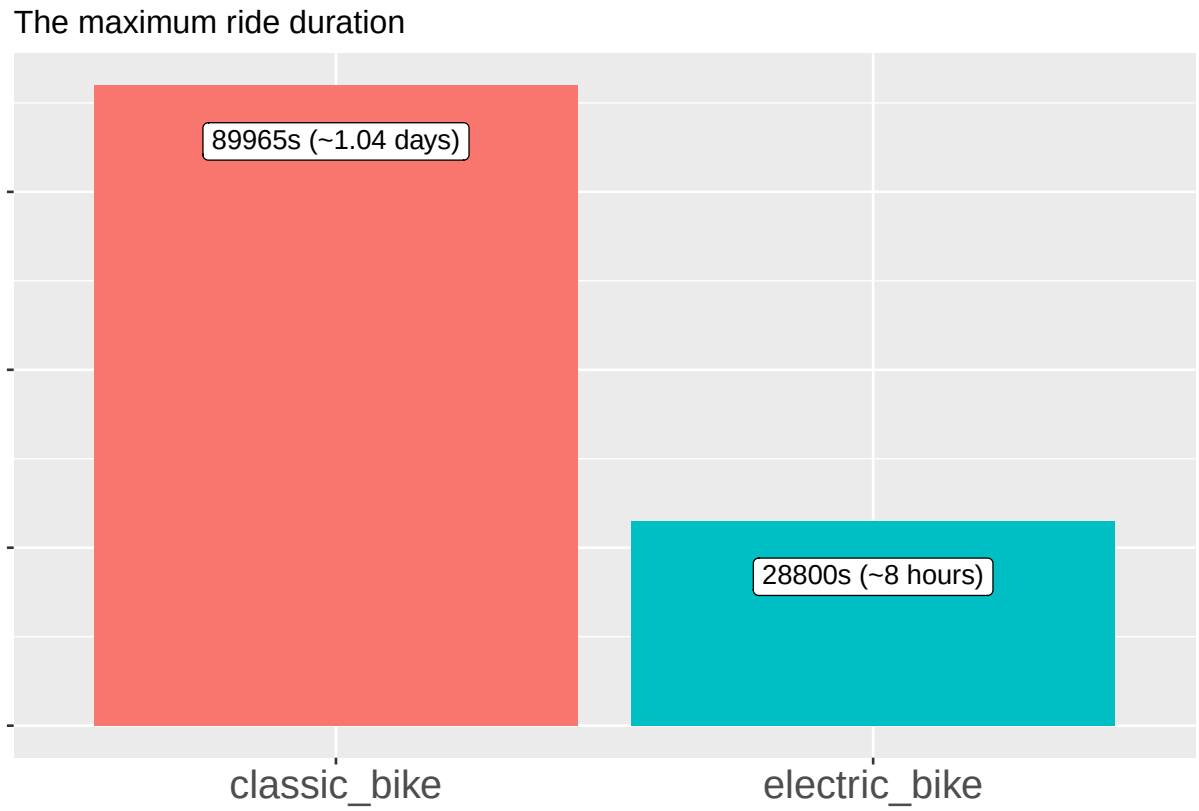
```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

Table 12: The maximum ride duration. Jan 2022 - Jan 2023

	bycicle	
	classic_bike	**electric_bike**
Max trip in minutes	1499.4	480.0

```
raw_df  %>%
  filter(rideable_type != "docked_bike") %>%
  filter(start_station_name !="" & end_station_name != "" &
           !is.na(end_lat)) %>%
  group_by(rideable_type) %>%
  summarise(max=as.duration(max(ended_at - started_at))) %>%

  ggplot() +
  geom_col(aes(y = max, x = rideable_type, fill = rideable_type ),
           show.legend = FALSE) +
  geom_label(aes(y = max, x = rideable_type,
                 label = format(max,big.mark=",") )
             , vjust = 2) +
  labs(title = "The maximum ride duration",
       caption = report_caption,
       x ="", y= "") +
  scale_y_continuous(labels = label_comma()) +
  theme(axis.text.y=element_blank(),
        axis.text = element_text(size = 16)
        )
```



Jan 2022 − Jan 2023

```
raw_df %>%
  filter(rideable_type == "electric_bike") %>%
  filter(start_station_name !="" & end_station_name != "" &
           !is.na(end_lat)) %>%
  select("rideable_type", "member_casual", "trip_duration") %>%
  arrange(desc(trip_duration)) %>%
  head(30)
```

```
##    rideable_type member_casual trip_duration
## 1  electric_bike        member      480.0000
## 2  electric_bike        member      479.9833
## 3  electric_bike        casual      479.9167
## 4  electric_bike        member      478.6000
## 5  electric_bike        member      478.5333
## 6  electric_bike        member      477.3000
## 7  electric_bike        member      475.2833
## 8  electric_bike        member      473.2000
## 9  electric_bike        member      471.2667
## 10 electric_bike        member      470.6667
## 11 electric_bike        member      470.2167
## 12 electric_bike        member      468.6500
## 13 electric_bike        member      468.0000
## 14 electric_bike        member      465.8500
## 15 electric_bike        member      465.3000
## 16 electric_bike        member      464.6500
```

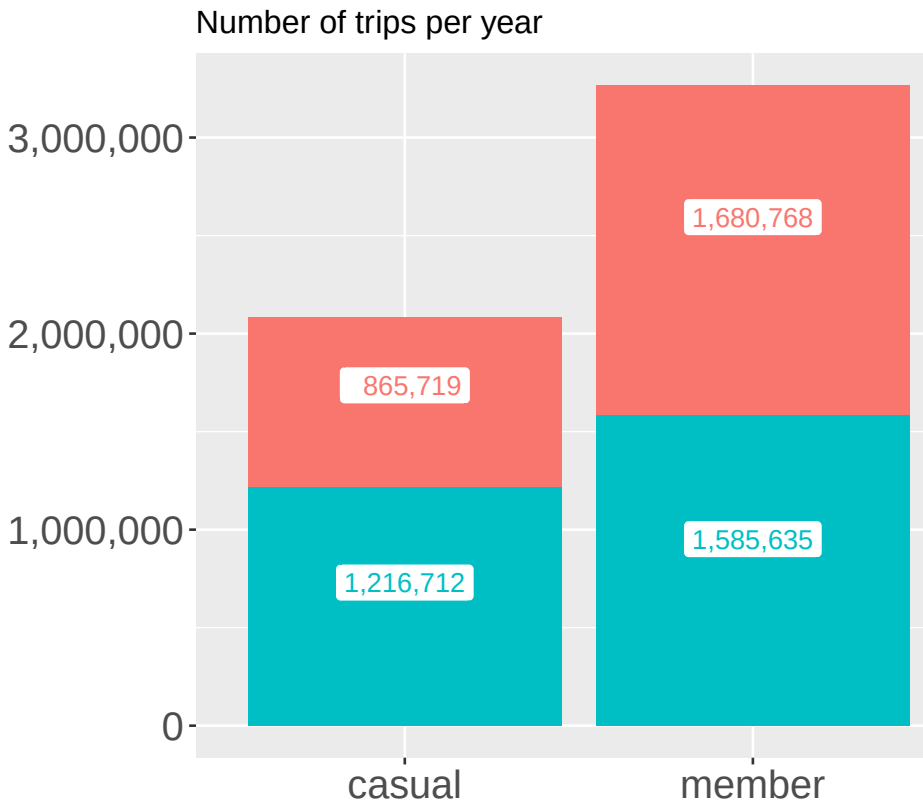
```
## 17 electric_bike        member      463.9333
## 18 electric_bike        casual      454.6167
## 19 electric_bike        member      449.3333
## 20 electric_bike        member      448.3500
## 21 electric_bike        member      442.0167
## 22 electric_bike        member      441.9667
## 23 electric_bike        member      440.7667
## 24 electric_bike        member      438.8000
## 25 electric_bike        member      430.8000
## 26 electric_bike        member      429.5000
## 27 electric_bike        member      427.7667
## 28 electric_bike        member      425.0500
## 29 electric_bike        member      423.5667
## 30 electric_bike        member      422.6833
```

Insights:

- Classic bike still leads the way in 1 full day trips
- The electric bike was able to last for 8 hours)
- The absolute majority of long rides on e-bikes were taken by members. (Pricing starts at $1 to unlock plus $0.39/minute for casual riders ($0 to unlock plus **$0.16/minute for members)**.)

```
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  group_by(member_casual, rideable_type) %>%
  summarise(ride_count = n()) %>%

  ggplot() +
  geom_col(aes(x = member_casual, y = ride_count,  fill = rideable_type ),
           show.legend = TRUE) +
  geom_label(aes(x = member_casual, y = ride_count,
                 label = format(ride_count,big.mark=",") , color = rideable_type),
             position = position_stack(vjust = 0.6) , show.legend = FALSE) +
  labs(title = "Number of trips per year",
       caption = "2022 calendar year data",
       x ="", y= "") +
  scale_y_continuous(labels = label_comma()) +
  theme(axis.text = element_text(size = 16) )
```



2022 calendar year

Number of trips throughout a year

```
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  crosstable(c(" "=rideable_type), by=member_casual, total="both", showNA="no",
        percent_digits=0, percent_pattern="{n} ({p_col})") %>%
  as_flextable(compact=TRUE, keep_id=FALSE) %>%
  set_caption(paste("Number of trips. 2022 calendar year"))
```

Table 14: Number of trips. 2022 calendar year

	member_casual		Total
	casual	member	
classic_bike	865719 (42%)	1680768 (51%)	2546487 (48%)
electric_bike	1216712 (58%)	1585635 (49%)	2802347 (52%)
Total	2082431 (39%)	3266403 (61%)	5348834 (100%)

```
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  mutate(Month=  month(started_at, label = TRUE)) %>%
  crosstable(c(" "= Month), by=member_casual, total="both", showNA="no",
```

```
      percent_digits=0, percent_pattern="{n} ({p_col})") %>%
  as_flextable(compact=TRUE, keep_id=FALSE) %>%
  set_caption(paste("Number of trips. 2022 calendar year"))
```

Table 15: Number of trips. 2022 calendar year

	member_casual		Total
	casual	member	
Jan	16997 (1%)	83482 (3%)	100479 (2%)
Feb	19376 (1%)	91871 (3%)	111247 (2%)
Mar	79311 (4%)	190241 (6%)	269552 (5%)
Apr	111128 (5%)	239523 (7%)	350651 (7%)
May	246537 (12%)	346854 (11%)	593391 (11%)
Jun	328588 (16%)	391205 (12%)	719793 (13%)
Jul	363992 (17%)	407415 (12%)	771407 (14%)
Aug	322906 (16%)	416457 (13%)	739363 (14%)
Sep	269081 (13%)	394612 (12%)	663693 (12%)
Oct	190741 (9%)	340655 (10%)	531396 (10%)
Nov	92165 (4%)	231179 (7%)	323344 (6%)
Dec	41609 (2%)	132909 (4%)	174518 (3%)
Total	2082431 (39%)	3266403 (61%)	5348834 (100%)

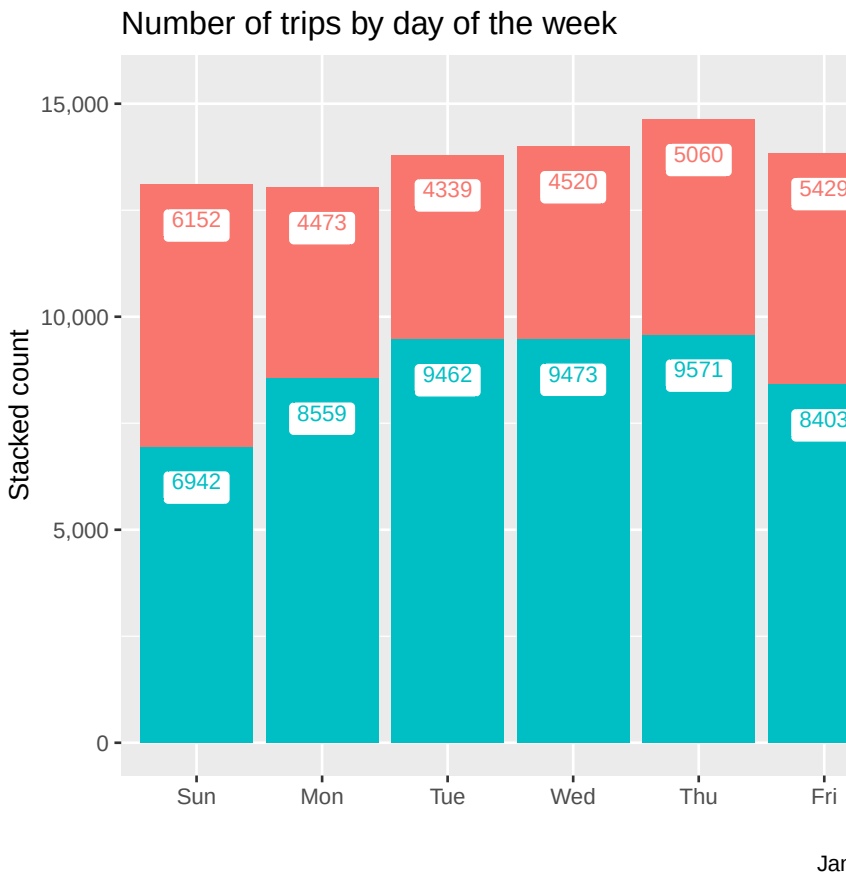
Number of trips

- **Casual** riders **prefer e-bikes**
- while Cyclistic's **members** choose e-bikes and classic bikes roughly equally
- **Members** use Cyclistics's services **much more often** than casual riders (+50 %).

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

num_weeks <- as.numeric(max(df$ended_at) - min(df$ended_at),"weeks")
df %>%
  group_by(member_casual, weekday) %>%
  summarise(mode = Mode(weekday),
            count = n() / num_weeks )  %>%
  # arrange(desc(count))
```

```
  ggplot() +
  geom_col(aes(y = count, x = weekday, fill = member_casual ),
           position = "stack") + #, show.legend = FALSE, , alpha = 0.8
  labs(title = "Number of trips by day of the week",
       caption = report_caption,
       x ="", y= "Stacked count",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma()) +
  geom_label(aes(x = weekday, y = count,  label = round(count), color = member_casual),
      hjust =0.5, vjust = 1.7, show.legend = FALSE, size = 3,
      position = "stack") # position_dodge(width = 1)
```



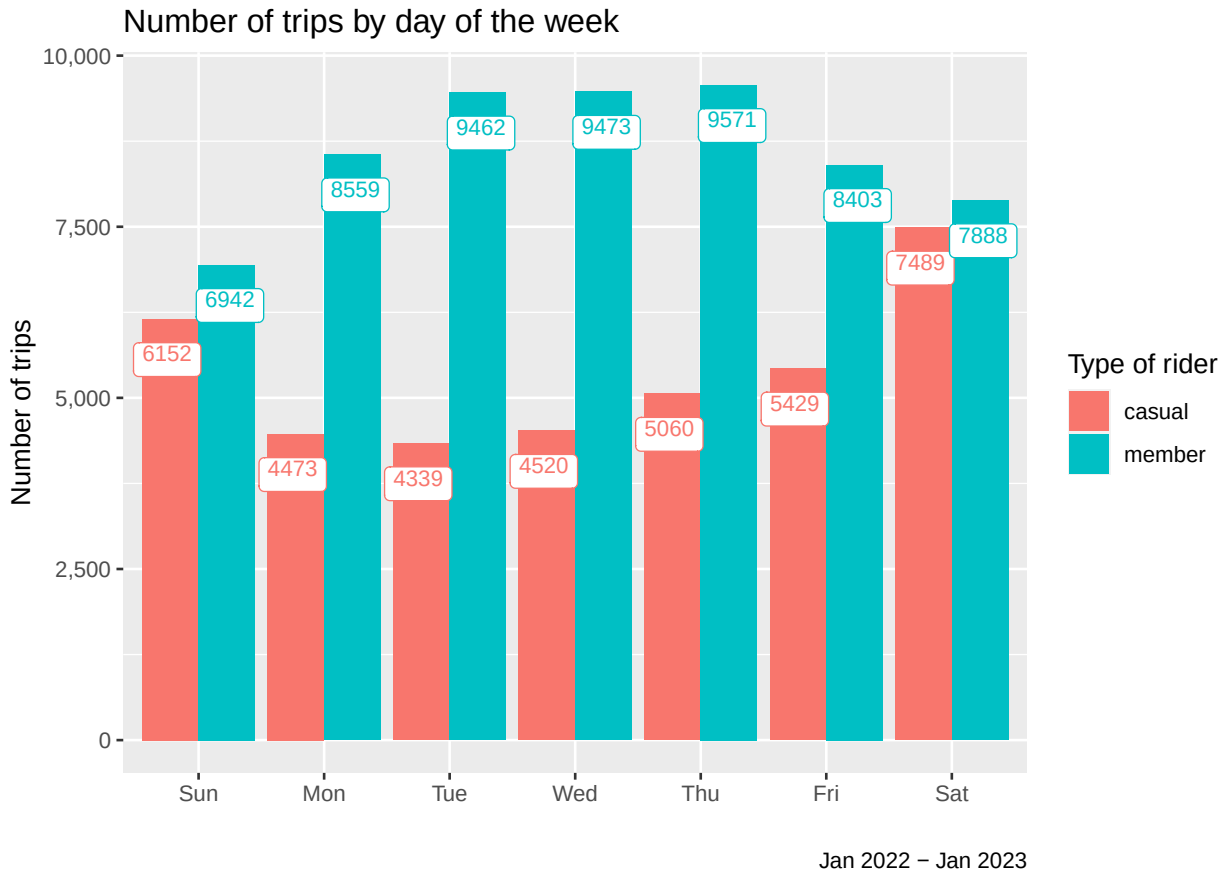
Jan

The mode of day of week throughout a year

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

num_weeks <- as.numeric(max(df$ended_at) - min(df$ended_at),"weeks")
df %>%
  group_by(member_casual, weekday) %>%
  summarise(mode = Mode(weekday),
            count = n() / num_weeks )  %>%
  # arrange(desc(count))
```

```
  ggplot() +
  geom_col(aes(y = count, x = weekday, fill = member_casual ),
           position = "dodge") + #, show.legend = FALSE, , alpha = 0.8
  labs(title = "Number of trips by day of the week",
       caption = report_caption,
       x ="", y= "Number of trips",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma()) +
  geom_label(aes(x = weekday, y = count,  label = round(count), color = member_casual),
      hjust =0.5, vjust = 1.7, show.legend = FALSE, size = 3,
      position = position_dodge(width = 1))
```




```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

num_weeks <- as.numeric(max(df$ended_at) - min(df$ended_at),"weeks")

df %>%
  mutate(Month=  month(started_at, label = TRUE)) %>%

  crosstable(c("   " =Month), by=c(" "=member_casual,"  "=weekday),  total="both", showNA="no",
        percent_digits=0, percent_pattern="{n} ({p_col})") %>%
```

```
  as_flextable(compact=TRUE, keep_id=FALSE) %>%
  set_caption(paste("Cross table. Number of trips by Month and by day of the week", report_caption))
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

variable	=Sun		=Mon		=Tue	
	=casual	=member	=casual	=member	=casual	=me
Jan	8031 (2%)	24195 (6%)	7465 (3%)	34877 (7%)	8655 (4%)	41872 (8%
Feb	3672 (1%)	11381 (3%)	4013 (2%)	18015 (4%)	2537 (1%)	15967 (3%
Mar	14234 (4%)	21562 (5%)	12497 (5%)	28844 (6%)	9163 (4%)	33757 (6%
Apr	16679 (5%)	24822 (6%)	10764 (4%)	33206 (7%)	13087 (5%)	39592 (7%
May	47572 (14%)	47682 (12%)	41294 (16%)	60772 (13%)	31241 (13%)	58353 (11
Jun	57394 (16%)	47897 (12%)	32815 (13%)	45782 (9%)	34858 (14%)	53720 (10
Jul	69053 (20%)	57212 (15%)	39100 (15%)	48714 (10%)	37405 (15%)	56153 (10
Aug	42769 (12%)	41865 (11%)	38181 (15%)	61177 (13%)	46558 (19%)	74859 (14
Sep	32293 (9%)	34801 (9%)	27904 (11%)	46291 (10%)	27151 (11%)	55757 (10
Oct	40347 (12%)	49014 (12%)	24877 (10%)	56630 (12%)	14541 (6%)	38723 (7%
Nov	11183 (3%)	20647 (5%)	9498 (4%)	31568 (7%)	14668 (6%)	45027 (8%
Dec	4821 (1%)	11681 (3%)	4630 (2%)	18329 (4%)	5589 (2%)	21515 (4%
Total	348048 (6%)	392759 (7%)	253038 (5%)	484205 (9%)	245453 (4%)	535295 (1

Number of trips by day of the week. Insights:

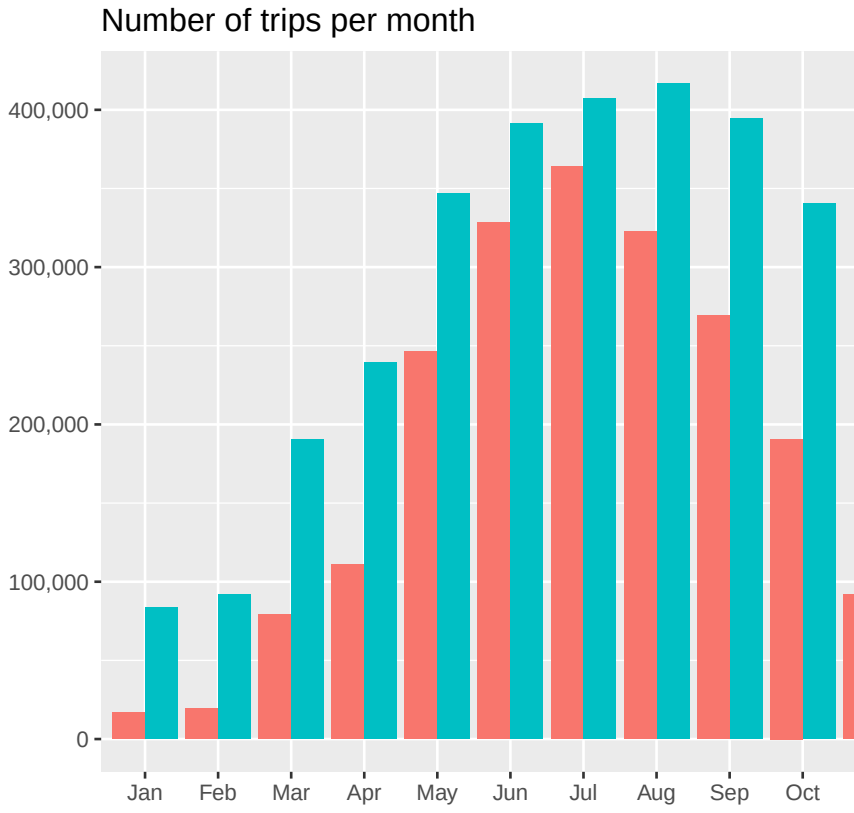
Rider	The mode	Anti-mode (the least frequent score)
Member riders	Tue, Wed, Thu - roughly equal	Sunday
Casual riders	Saturday	Monday - Tuesday
Population	**Saturday** is the most busy day at Cyclistic's mainly thanks to casual riders.	Monday is the least busy day of the week

Insights:

- **Monday** is the least busy day of the week (anti-mode).
- **Saturday** is the most busy day at Cyclistic's mainly thanks to casual riders.

```
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  group_by(member_casual, Month=  month(started_at, label = TRUE)) %>%
  summarise(count = n()) %>%

  ggplot(aes(x = Month, y= count, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title = "Number of trips per month",
       caption = "2022 calendar year data",
       x ="", y= "",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma())
```



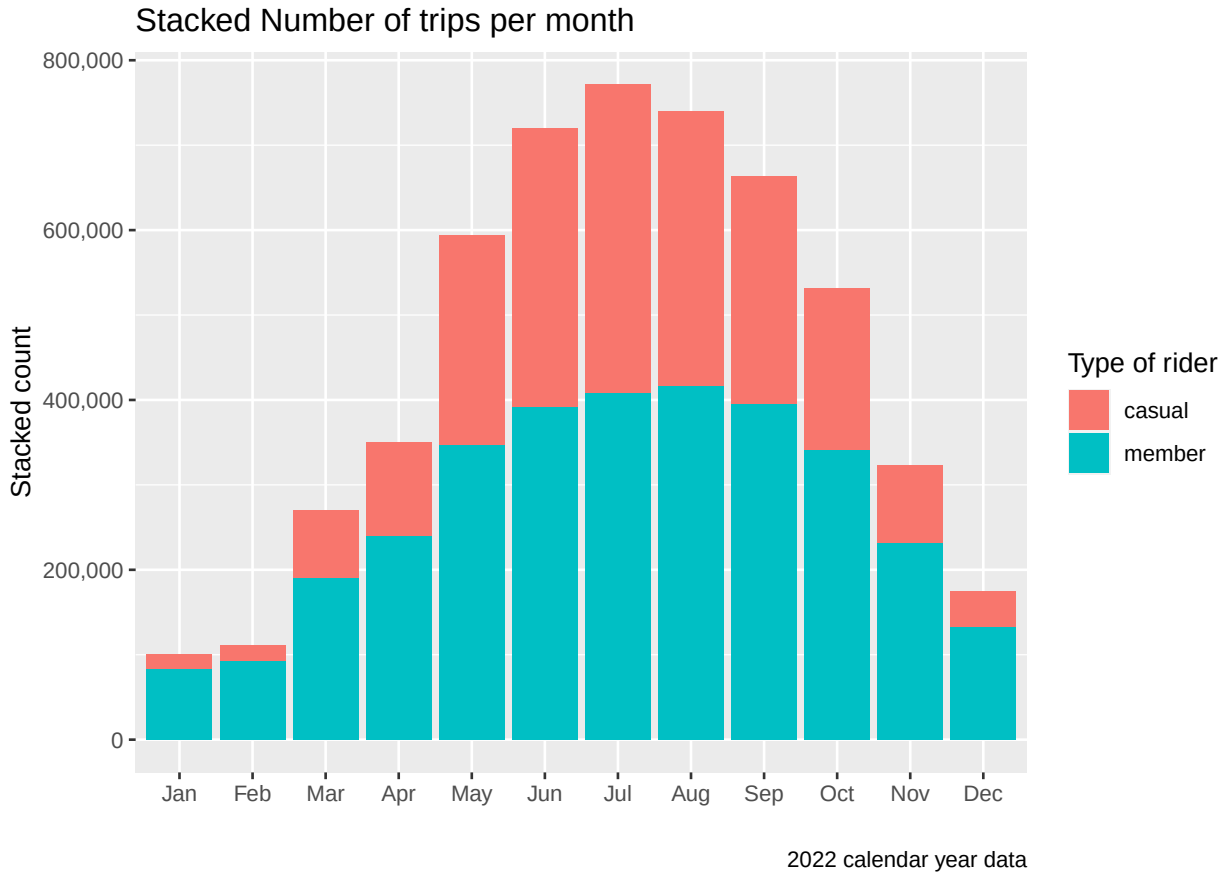
2022 calen

Rider behavior depending on the season

```
# +
#   geom_text(aes(x = Month, y= count, label = count),
#       hjust =0.5, vjust = 1.7, show.legend = FALSE, size = 2,
#       position = position_dodge(width = 1))
```

```
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  group_by(member_casual, Month=  month(started_at, label = TRUE)) %>%
  summarise(count = n()) %>%

  ggplot(aes(x = Month, y= count, fill = member_casual)) +
  geom_col(position = "stack") +
  labs(title = "Stacked Number of trips per month",
       caption = "2022 calendar year data",
       x ="", y= "Stacked count",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma())
```



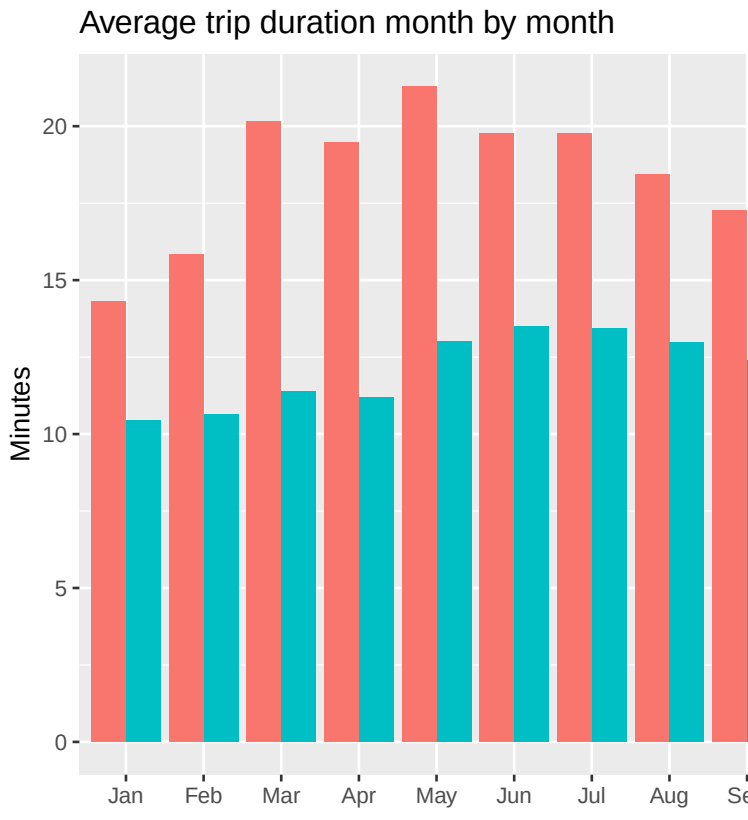

```
# +
#   geom_text(aes(x = Month, y= count, label = count),
#       hjust =0.5, vjust = 1.7, show.legend = FALSE, size = 2,
#       position = position_dodge(width = 1))
```

Insights:

- **Jan and Feb are the toughest month at Cyclistics**. The income is only ~ 16% of year peaks.
- **Members** generated ~**80% of income** during **Jan and Feb**.
- In the summer months, the proportion begins to level off.

```
# plotly interactive doesn't work with github output
# insp_plt <-
df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  group_by(member_casual, Month=  month(started_at, label = TRUE)) %>%
  summarise(aver = mean(trip_duration)) %>%

  ggplot() +
  geom_col(aes(x = Month, y=aver, fill = member_casual),
           position = "dodge", alpha = 1) +
  labs(title = "Average trip duration month by month",
       caption = report_caption,
       x ="", y= "Minutes",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma())
```



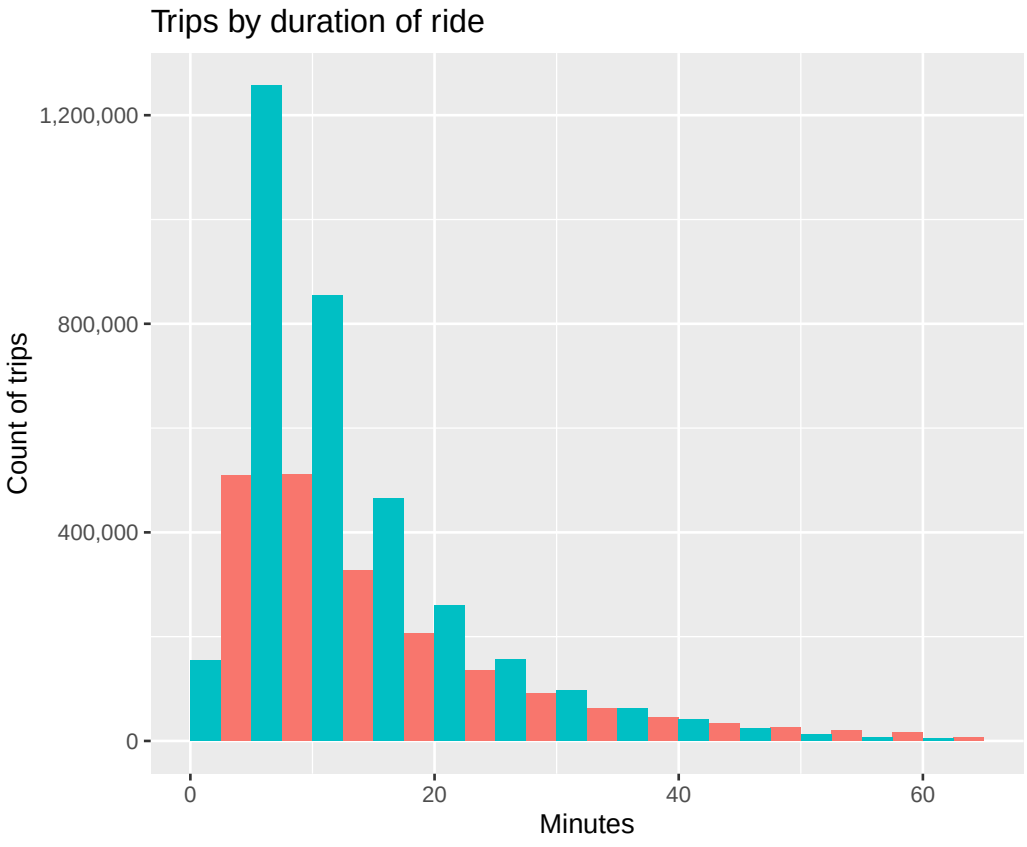
Average trip duration by day of week, month, rider

```
# ggplotly(insp_plt)
```

Insights:

The variation of trip duration among **members** is much lower. This means that they follow some **patterns** throughout the year.
However average trip duration is higher among **casual** riders. They are **not time constrained** as members.

```
df %>%
  ggplot(aes(x = trip_duration, fill = member_casual)) +
  geom_histogram(binwidth= 5,position = "dodge") + #
  labs(title = "Trips by duration of ride",
       caption = "Bin duration: 5 min",
       x ="Minutes", y= "Count of trips",
       fill='Type of rider') +
  scale_y_continuous(labels = label_comma())+
  xlim(0, 65)
```



Number of trips by duration

Insights:

- Most trips (21%) are made in the 5 to 15 minute range.
- Members ride more often, but less time.

```r
# df %>%
# summary(df$trip_duration )

df %>%
  mutate(Month=  month(started_at, label = TRUE),
         Ride_duration = round(trip_duration)) %>%

  crosstable(c(" "=Ride_duration), by=c("Rider" = member_casual ), funs = c(mean, median), total="both"
        percent_digits=0, percent_pattern="{n} ({p_col})") %>%
  as_flextable(compact=TRUE, keep_id=FALSE) %>%
  set_caption(paste("Mean and median trip durations.", report_caption))
```

```
## Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘ engine is
## used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning by using the
## ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a compatible engine
## by defining ‘latex_engine: xelatex‘ in the YAML header of the R Markdown
## document.
```

Table 23: Mean and median trip durations. Jan 2022 - Jan 2023

	Rider		Total
	casual	member	
mean	18.4	12.1	14.5
median	12.0	9.0	10.0

5. SHARE

Guiding questions:

1. Were you able to answer the question of how annual members and casual riders use Cyclistic bikes differently?
2. What story does your data tell?
3. How do your findings relate to your original question?
4. Who is your audience? What is the best way to communicate with them?
5. Can data visualization help you share your findings?
6. Is your presentation accessible to your audience?

How casual riders and annual members use Cyclistic bikes differently?

We have figured out a significant differences in behaviour among riders.

	members	casual riders
number of trips min-max per month	83482 - 416457	16997 - 363992
spread of trip duration per month	10-13 min (Dec-Jun)	12-22 min (Dec-May)
year average trip duration	12.1	18,4
mode of day of week	Tue, Wed, Thu	Sat
anti-mode of day of week	Sun	Mon-Tue
bike type preference	doesn't matter	e-bike
weather dependence	medium	high
seasonal patterns	medium	very high
Jan -Jul rides	1:5	1:21
most likely purpose of use	**Commute to work or school, run errands, performing daily duties during workdays**	**Explore Chicago, get to appointments or social engagements on weekends**

Members use Cyclistics to complete their daily activities, such as commuting to work or school, running errands.
Members ride more often but their trips are shorter. They are more time constrained.
Casual riders most likely use bike share to explore Chicago, get to appointments or social engagements, and more. And casual riders prefer e-bikes.

Why would casual riders buy Cyclistic annual memberships?

So, how can we steer casual riders to became a Cyclistics member?
We should accent our offerings on the preferences that have been figured out in the report.
Casual riders are not avid bikers. They are ready to pay more for e-bike rent.
We should outline that our proposition is more profitable than rent a e-bike from time to time.

For example, Annual weekend membership program -
any number of rides totally up to 60 min on Saturdays and 60 min on Sundays included
for 200$ a year. (from April till September - 48 weekend days means only 4$ on e-bike a day).
If we take into account full year - 200$ / 104 d means only 2$ per weekend day on e-bike.

How can Cyclistic use digital media to influence casual riders to become members?

1. Use social networks to populate bike sharing.
2. Use direct targeting on potential customers. Send direct proposals using mobile app every customers have already installed.
3. Collaboration with local parks and attractions to populate bike using. Invest into popular bike apps like Komoot, Strava to inform local customers about Cyclistics.

6. ACT

1. Our top recommendations:

    1. Work out a new e-bike membership which includes 30-60 minutes of **e-bike rides on weekends**. The price should be higher than ordinary membership, say 200 $ but
    Additional research should be conducted about specific pricing parameters.
    Let's don't forget about our current customers. New proposal shouldn't contradict their current membership (accented on classic bike usage).
    2. Customers may consider to buy personal e-bikes, but we should launch **an awareness campaign** about the benefits of e-bike sharing (charging, service, does not take up space in the apartment, not have to worry a single second about a bike being stolen, etc.)
    3. Investigate most popular casual riders routes in Chicago and inform potential customers about future locations of dock stations. Conduct a survey among potential customers where to locate stations.
    4. Invest time and efforts into additional exploration of potential customers and their needs, polishing special proposals among customers segments. Conduct a survey about potential locations of dock-stations.
    5. Explore Chicago 2022 weather (temperature, precipitation) data and how it affected riders trips.

Additional research

How does weather affect riders?

To answer this question we use weather data set. It's a first party data.

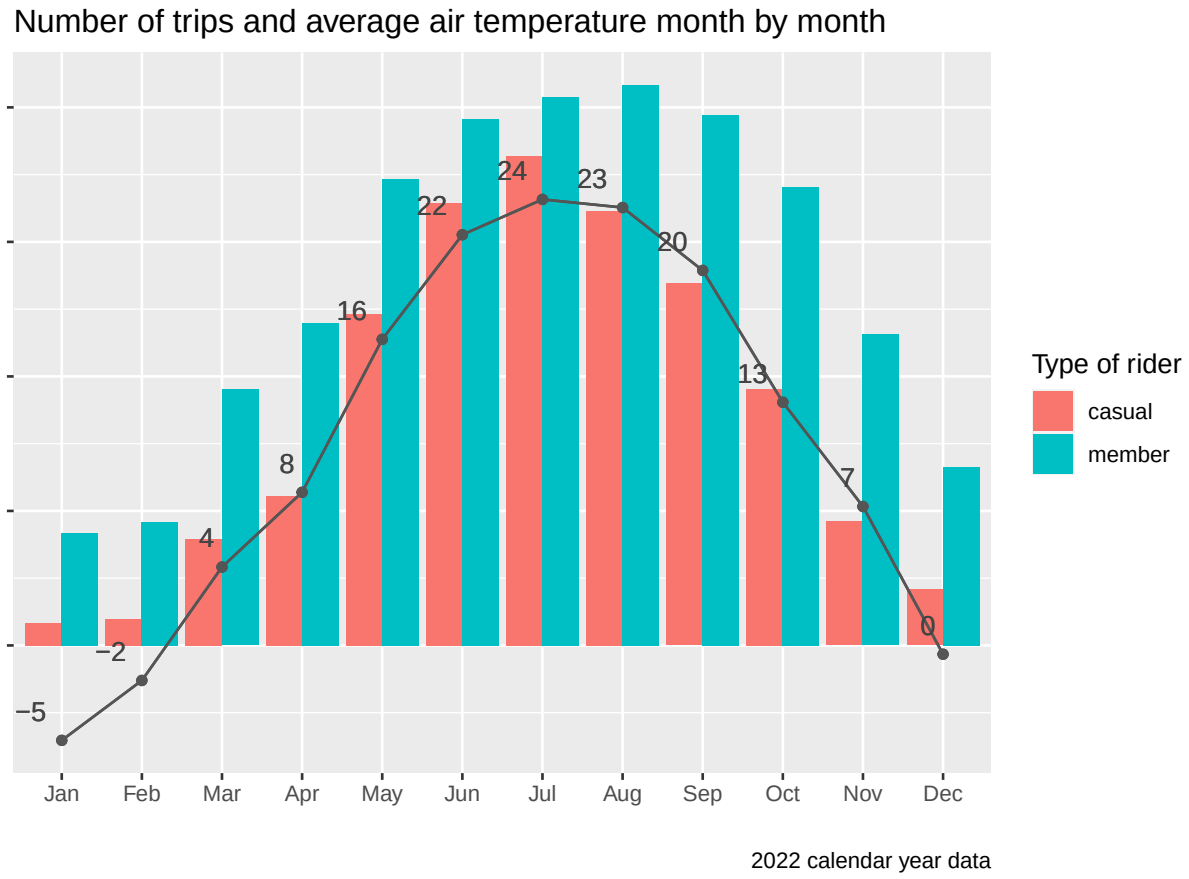
```
w_df <- read.csv("Divvy_tripdata/Beach_Weather_Stations_-_Automated_Sensors.csv")
# casting date
w_df$weather_date = mdy_hms(w_df$Measurement.Timestamp)
```

```
w_df %>%
  filter(weather_date >= "2022-01-01" & weather_date <= "2022-12-31") %>%
  select("weather_date","Air.Temperature", "Total.Rain") %>%
  # filter(hour(weather_date) == 11) %>%
  group_by( Month = month(weather_date, label = TRUE)) %>%
  summarise( aver_temp = mean(Air.Temperature),
             aver_rain = mean(Total.Rain, na.rm = TRUE)) -> w2_df

df %>%
  filter(started_at < ymd("2023-01-01")) %>% # limit to a calendar year
  group_by(member_casual, Month=  month(started_at, label = TRUE)) %>%
  summarise(ride_count = n()) %>%
  left_join(w2_df, by = c("Month"), multiple = "first") -> w2_df
```

```
w2_df %>%
  # filter(member_casual == "casual") %>%
  ggplot(label = aver_temp) +
  geom_col(aes(x = Month, y= c(ride_count), fill = member_casual),
           position = "dodge", alpha = 1) +
  labs(title = "Number of trips and average air temperature month by month",
```

```
      caption = "2022 calendar year data",
      x ="", y= "",
      fill='Type of rider') +
scale_y_continuous(labels = label_comma()) +
geom_point(aes(x= Month, y = aver_temp *1.4e4 ), color= "#555555" ) +
geom_line(aes(x= Month, y = aver_temp *1.4e4 , group= member_casual), color= "#555555" ) +
geom_text(aes(x= Month, y = aver_temp *1.4e4,label = round(aver_temp)), hjust=1.5, vjust=-1, color= "
# add temperature line
# remove y axis scale
  theme(axis.text.y=element_blank() )
```



The behavior of casual riders is more dependent on air temperature. Members are more loyal customers especially in cold months.