Case Study 1: How does a bike-share navigate speedy success?

Tien-Chi Lin 2024-12-15

STEP 0: LOAD PACKAGES FOR THE CASE STUDY

```
library(tidyverse)
```

```
## ── 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   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── 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
```

```
library(lubridate)
library(stringr)
library(ggplot2)
library(dplyr)
```

STEP 1: COLLECT DATA

Upload Divvy datasets (csv files) here

```
nov_2024 <- read.csv("../origin_csv/202411-divvy-tripdata.csv")
oct_2024 <- read.csv("../origin_csv/202410-divvy-tripdata.csv")
sep_2024 <- read.csv("../origin_csv/202409-divvy-tripdata.csv")
aug_2024 <- read.csv("../origin_csv/202408-divvy-tripdata.csv")
jul_2024 <- read.csv("../origin_csv/202407-divvy-tripdata.csv")
jun_2024 <- read.csv("../origin_csv/202406-divvy-tripdata.csv")
may_2024 <- read.csv("../origin_csv/202405-divvy-tripdata.csv")
apr_2024 <- read.csv("../origin_csv/202404-divvy-tripdata.csv")
mar_2024 <- read.csv("../origin_csv/202403-divvy-tripdata.csv")
feb_2024 <- read.csv("../origin_csv/202402-divvy-tripdata.csv")
jan_2024 <- read.csv("../origin_csv/202401-divvy-tripdata.csv")
dec_2023 <- read.csv("../origin_csv/202312-divvy-tripdata.csv")
```

STEP 2: WRANGLE DATA AND COMBINE INTO A SINGLE FILE

Inspect dataframes

```
str(nov_2024)
```

```
## 'data.frame':	335075 obs. of  13 variables:
##  $ ride_id           : chr  "578DDD7CE1771FFA" "78B141C50102ABA6" "1E794CF36394E2D7" "E5DD2CAB58D73F
98" ...
##  $ rideable_type     : chr  "classic_bike" "classic_bike" "classic_bike" "classic_bike" ...
##  $ started_at        : chr  "2024-11-07 19:21:58.206" "2024-11-22 14:49:00.431" "2024-11-08 09:24:0
0.238" "2024-11-24 17:51:14.144" ...
##  $ ended_at          : chr  "2024-11-07 19:28:57.301" "2024-11-22 14:56:15.475" "2024-11-08 09:28:3
3.480" "2024-11-24 18:05:32.574" ...
##  $ start_station_name: chr  "Walsh Park" "Walsh Park" "Walsh Park" "Clark St & Elm St" ...
##  $ start_station_id  : chr  "18067" "18067" "18067" "TA1307000039" ...
##  $ end_station_name  : chr  "Leavitt St & North Ave" "Leavitt St & Armitage Ave" "Damen Ave & Cortla
nd St" "Clark St & Drummond Pl" ...
##  $ end_station_id    : chr  "TA1308000005" "TA1309000029" "13133" "TA1307000142" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.7 -87.7 -87.7 -87.6 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
```

Stack individual monthly dataframes into one full-year dataframe

```
all_trips <- bind_rows(nov_2024, oct_2024, sep_2024, aug_2024,
                       jul_2024, jun_2024, may_2024, apr_2024,
                       mar_2024, feb_2024, jan_2024, dec_2023)
```

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

Add columns that list the date, month, weekday, and year of each ride This will help us aggregate data for further analysis

```
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)

```
all_trips$ride_length <- difftime(all_trips$ended_at,all_trips$started_at)
```

Convert the class of “ride_length” from difftime to numeric so we can run further calculations

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

Check the reliability of “ride_length”, which should be no smaller than 0

```
min(all_trips$ride_length)
```

```
## [1] -164899
```

Store the total number of records and the number of bad records

```
count_all_data <- length(all_trips$ride_length)
count_all_data
```

```
## [1] 5906269
```

```
count_bad_data <- sum(all_trips$ride_length<0)
count_bad_data
```

```
## [1] 237
```

There is the reliability issue about “ride_length”, so clean data by removing “bad” data We will create a new version of the dataframe (v2) since data is being removed

```
all_trips_v2 <- all_trips[!(all_trips$ride_length<0),]
```

Check again the modified result

```
min(all_trips_v2$ride_length)
```

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

Check the reliability of “member_casual”, which should only has value “member” and “casual” In this case, no wrong string value exists, no need for cleaning

```
count_wrong_word <- sum(!(all_trips_v2$member_casual %in% c("member", "casual")))
count_wrong_word
```

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

STEP 4: CONDUCT DESCRIPTIVE ANALYSIS

Use `summary()` to see all measured metric: mean, median, min, max, etc.

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   332.7   582.5  1039.3  1034.0 93596.0
```

Optional: Compare members and casual users

```
#aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean)
#aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median)
#aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max)
#aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min)
```

Fix the order of “day of week” from Sunday to Saturday

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

Optional: 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 = mea
n)
```

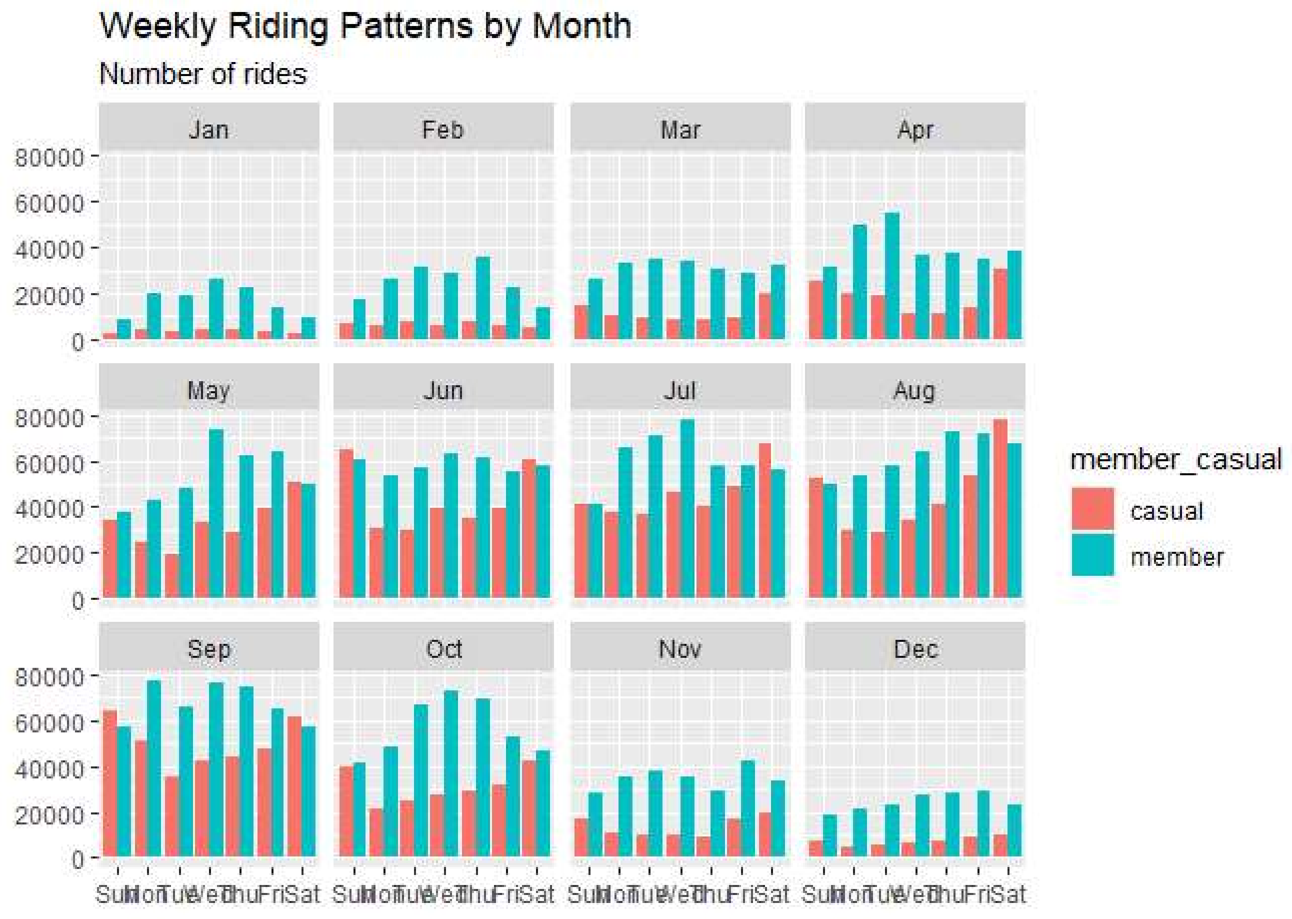
Analyze monthly ridership data by type and weekday

```
data_monthly <- all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%                          # Create weekday field
using wday()
  mutate(month = factor(month.abb[as.numeric(month)], levels = month.abb)) %>%   # Transform the numeri
c value of month into month abbreviation
  group_by(month, member_casual, weekday) %>%                                    # Group by month, type
and weekday
  summarise(number_of_rides = n()                                                               # Calcul
ate the number of rides and average duration
          ,average_duration = mean(ride_length)) %>%                               # Calculate the aver
age duration
  arrange(month, member_casual, weekday)                                                          # Sort
```

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

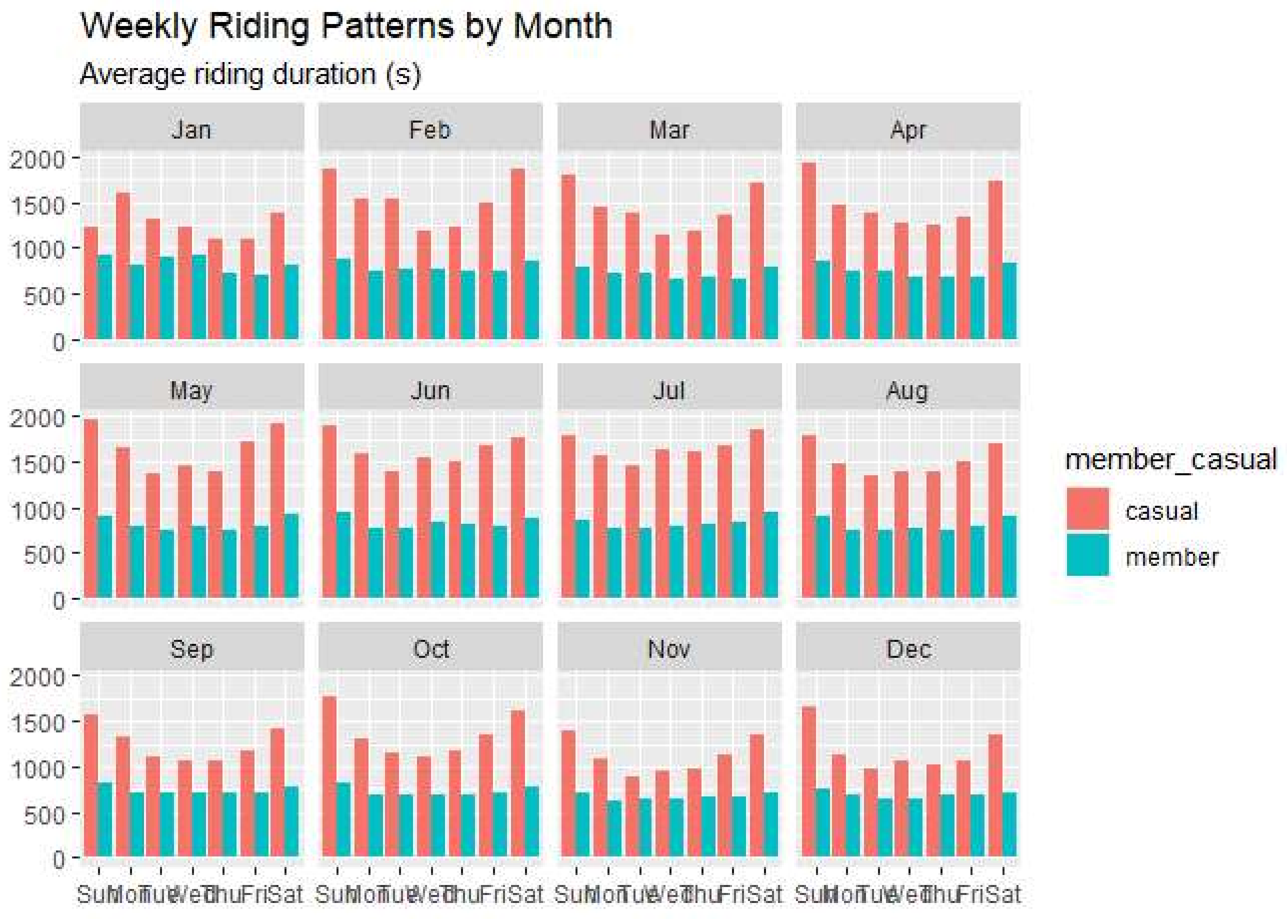
Visualize the number of rides by rider type

```
data_monthly  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  facet_wrap(~month) +
  geom_col(position = "dodge") +
  labs(title='Weekly Riding Patterns by Month', subtitle='Number of rides', x=' ', y='')
```



Visualize the average duration by rider type

```
data_monthly  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  facet_wrap(~month) +
  geom_col(position = "dodge") +
  labs(title='Weekly Riding Patterns by Month', subtitle='Average riding duration (s)', x=' ', y='')
```



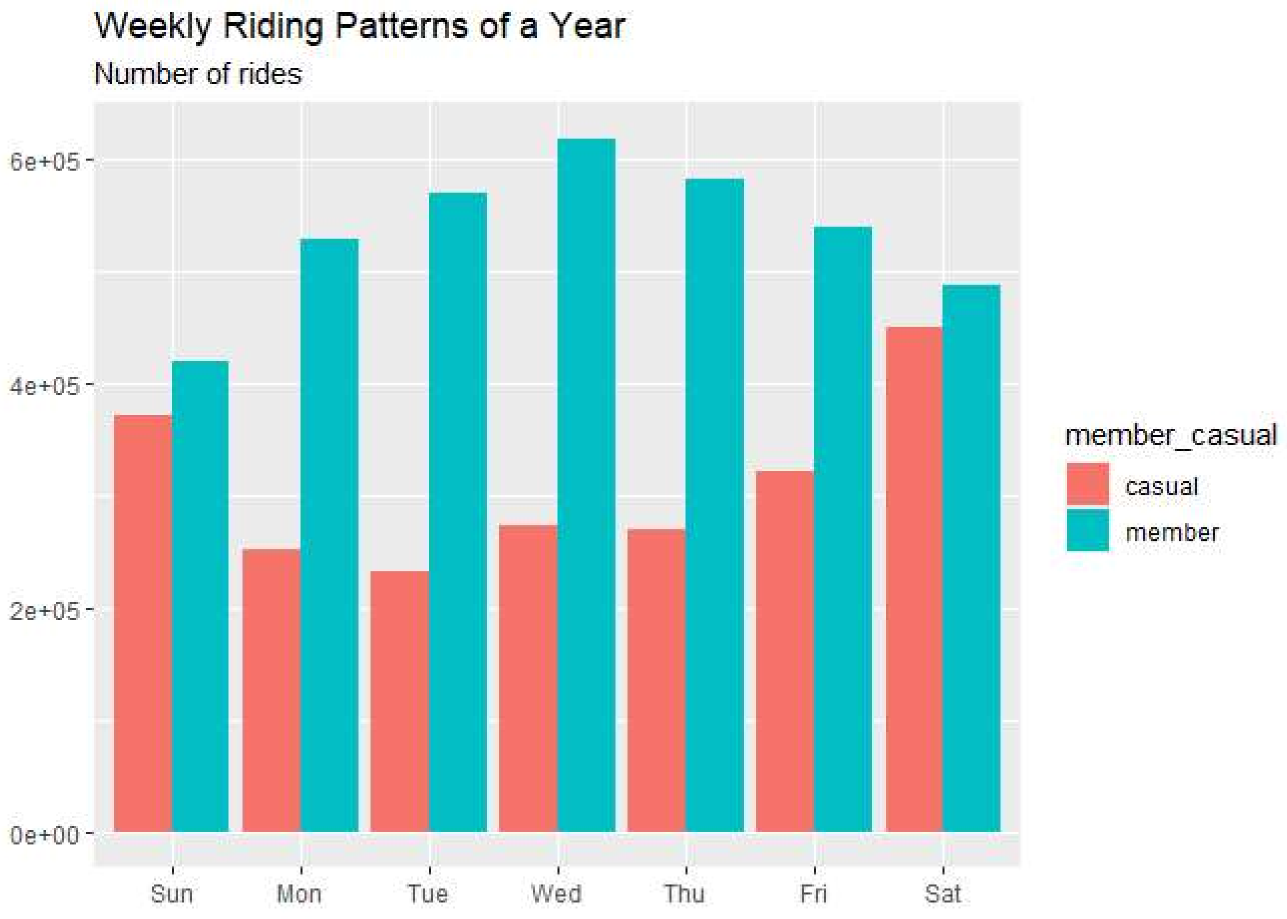
Analyze full-year ridership data by type and weekday

```
data_fullyear <- all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%    # Create weekday field
  group_by(member_casual, weekday) %>%                    # Group by type and weekday
  summarise(number_of_rides = n()                                      # Calculate the number of rides
and average duration
            ,average_duration = mean(ride_length)) %>%      # Calculate the average duration
  arrange(member_casual, weekday)                         # Sort
```

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

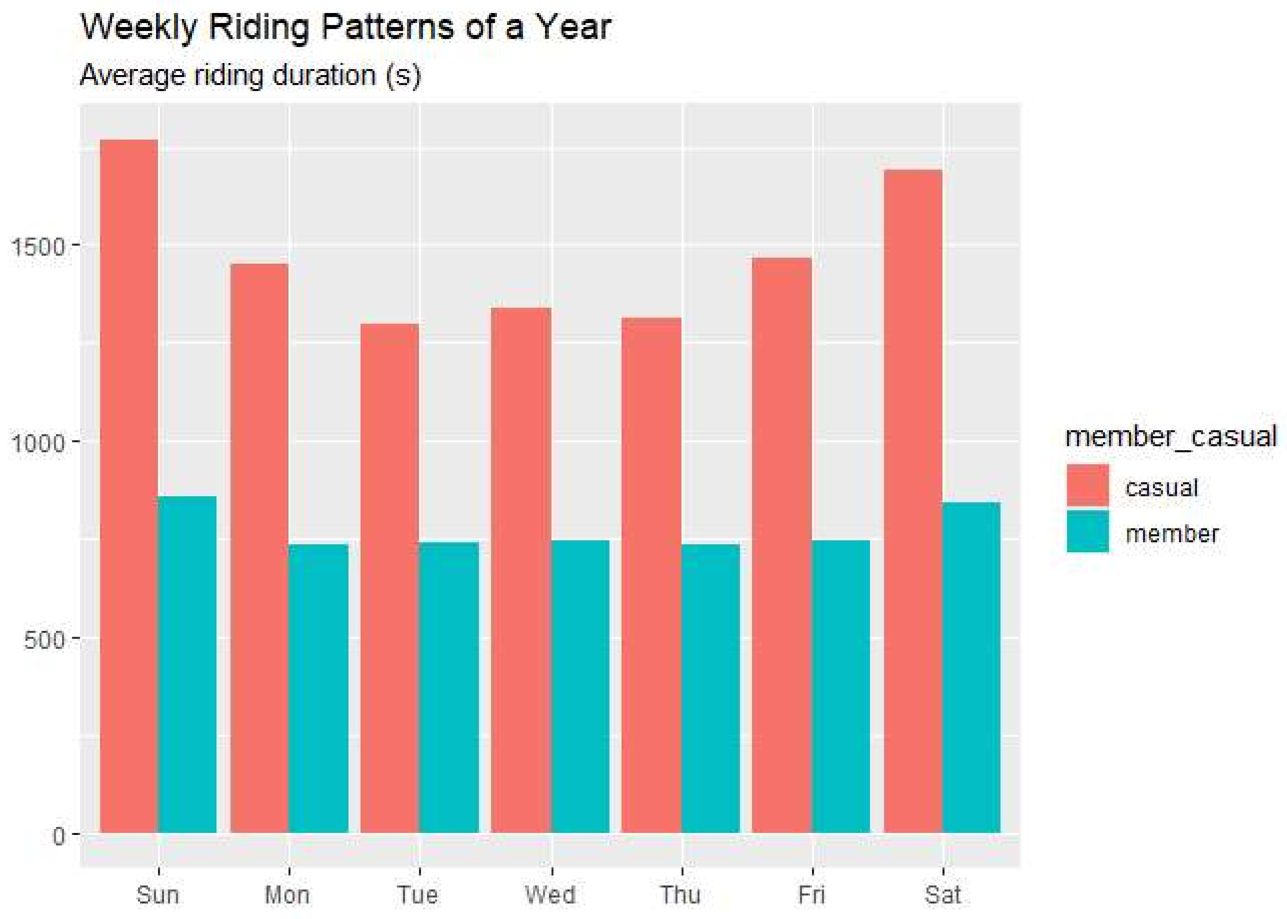
Visualize the number of rides by rider type

```
data_fullyear  %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title='Weekly Riding Patterns of a Year', subtitle='Number of rides', x=' ', y='')
```



Visualize the average duration by rider type

```
data_fullyear  %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge") +
  labs(title='Weekly Riding Patterns of a Year', subtitle='Average riding duration (s)', x=' ', y='')
```



STEP 5: EXPORT SUMMARY FILE FOR FURTHER ANALYSIS

Create csv files that we will visualize in Excel, Tableau, or presentation software

```
#write.csv(data_monthly, file = 'data_monthly.csv')
#write.csv(data_fullyear, file = 'data_fullyear.csv')
```