Case Study:How Does a Bike-Share Navigate Speedy Success?

Thomas Arias

21/3/2022

In this case study a data analysis was made for a fictional company, Cyclistic, a bike-share company in Chicago in order to answer key business questions, mainly:

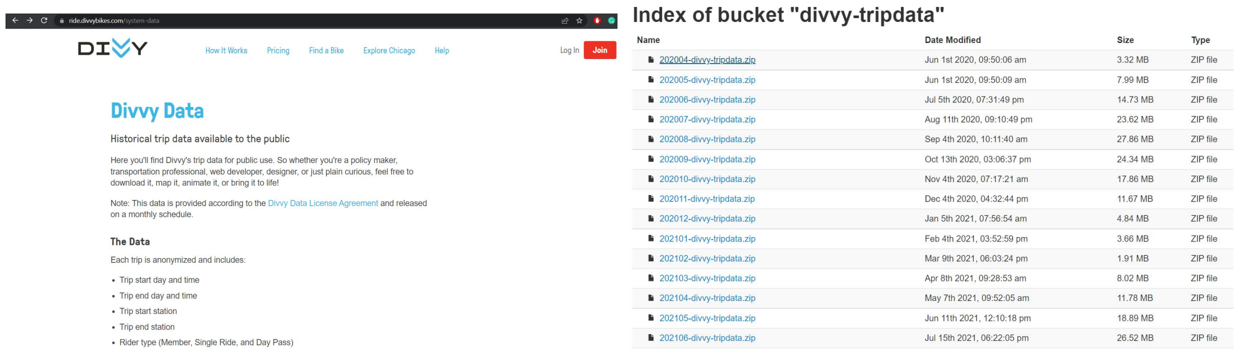
1. How casual riders and member riders use Cyclistic bikes differently.
2. Design a new marketing strategy to convert casual riders into annual members.

More info:

Cyclistic's current marketing strategy relies on building general awareness and appeal to broad consumer segments. One such approach was the flexibility of its pricing plans: single-ride passes, full-day passes and annual memberships. Customers who purchase single-ride passes or full-day passes are considered casual riders, while customers who purchase annual memberships are considered members.

Cyclistic's financial analysts believe that the largest amount of profit comes from clients who purchase annual memberships, and they want to convert more of their casual clients into annual members. To accomplish this, they need to design a new marketing strategy aimed at casual riders, but first they need to understand how casual and member riders differ. They are interested in analyzing the Cyclistic historical bike trip data to identify potential trends and answer the business questions.

The data was obtained from the Divvy Data website. For more details on the data, click here: link



Divvy Data

Historical trip data available to the public

Here you'll find Divvy's trip data for public use. So whether you're a policy maker, transportation professional, web developer, designer, or just plain curious, feel free to download it, map it, animate it, or bring it to life!

Note: This data is provided according to the Divvy Data License Agreement and released on a monthly schedule.

The Data

Each trip is anonymized and includes:

- Trip start day and time
- Trip end day and time
- Trip start station
- Trip end station
- Rider type (Member, Single Ride, and Day Pass)

Index of bucket "divvy-tripdata"

Name	Date Modified	Size	Type
202004-divvy-tripdata.zip	Jun 1st 2020, 09:50:06 am	3.32 MB	ZIP file
202005-divvy-tripdata.zip	Jun 1st 2020, 09:50:09 am	7.99 MB	ZIP file
202006-divvy-tripdata.zip	Jul 5th 2020, 07:31:49 pm	14.73 MB	ZIP file
202007-divvy-tripdata.zip	Aug 11th 2020, 09:10:49 pm	23.62 MB	ZIP file
202008-divvy-tripdata.zip	Sep 4th 2020, 10:11:40 am	27.86 MB	ZIP file
202009-divvy-tripdata.zip	Oct 13th 2020, 03:06:37 pm	24.34 MB	ZIP file
202010-divvy-tripdata.zip	Nov 4th 2020, 07:17:21 am	17.86 MB	ZIP file
202011-divvy-tripdata.zip	Dec 4th 2020, 04:32:44 pm	11.67 MB	ZIP file
202012-divvy-tripdata.zip	Jan 5th 2021, 07:56:54 am	4.84 MB	ZIP file
202101-divvy-tripdata.zip	Feb 4th 2021, 03:52:59 pm	3.66 MB	ZIP file
202102-divvy-tripdata.zip	Mar 9th 2021, 06:03:24 pm	1.91 MB	ZIP file
202103-divvy-tripdata.zip	Apr 8th 2021, 09:28:53 am	8.02 MB	ZIP file
202104-divvy-tripdata.zip	May 7th 2021, 09:52:05 am	11.78 MB	ZIP file
202105-divvy-tripdata.zip	Jun 11th 2021, 12:10:18 pm	18.89 MB	ZIP file
202106-divvy-tripdata.zip	Jul 15th 2021, 06:22:05 pm	26.52 MB	ZIP file

Figure 1: Divvy Data website

Once the data has been downloaded the next step is to clean the data.

Data cleaning process

1. I Checked that the types of variables agree with the variables themselves (numbers as numeric instead of characters,etc).

2. I deleted rows with missing information.
3. I checked that the values for latitude were positive and longitude were all negative.
4. I separated the beginning and the end of trips into *date_start*, *date_end*, *time_start* and *time_end*.
5. I changed the column name of *member* to *member_type*
6. I checked that trips started and ended on the same day.
7. I checked that *end_time* came later than *start_time*.
8. I eliminated the columns *start_station*, *end_station*, and *station_id* because they didn't provide meaningful information.
9. Instead of using the whole dataset, I took a sample from each csv file by month. To get the sample size I employed Cochran's formula:

$$n_0 = \frac{Z^2 pq}{e^2}$$

Where:

- n_0 is the sample size.
- Z^2 is the critical value of the Z score .
- p is the percentage of the population that is represented in the formula.
- q is equal to 1 - p.
- e^2 is the margin of error squared.

With a 99% confidence level I used a Z score of 2.576, and a margin of error of 5%, which left me with a sample of 662 rows for every csv file, which I then merged into one. The p value was left at 0.5 for variablity purposes.

How to read the files

The data is read and loaded into a dataframe called *divvy_trip_data*. The columns for *date_start, time_start, date_end and time_end* are formatted to hours, minutes, day, month and year to use the *Date* function later. A small preview of the data is shown.

```r
#import data from csv file
divvy_trip_data <- read_csv("data/merged_data/divvy-trip-data.csv",
                            col_types = cols(date_start = col_date(format = "%d/%m/%Y"),
                                             time_start = col_time(format = "%H:%M"),
                                             date_end = col_date(format = "%d/%m/%Y"),
                                             time_end = col_time(format = "%H:%M")))
#Small preview of the data along with types
str(divvy_trip_data)
```

```
## spec_tbl_df [7,386 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ride_id       : chr [1:7386] "0449139F2CB5BF9A" "0C4236EB38C3089B" "23D536DF2046804F" "537A084E3D5
##  $ rideable_type : chr [1:7386] "docked_bike" "docked_bike" "docked_bike" "docked_bike" ...
##  $ date_start    : Date[1:7386], format: "2020-04-01" "2020-04-01" ...
##  $ time_start    : 'hms' num [1:7386] 13:29:00 13:17:00 14:29:00 08:59:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ date_end      : Date[1:7386], format: "2020-04-01" "2020-04-01" ...
##  $ time_end      : 'hms' num [1:7386] 13:35:00 13:38:00 14:39:00 09:22:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ end_station_id: chr [1:7386] "332" "166" "182" "463" ...
##  $ start_lat     : num [1:7386] 41.9 41.9 41.9 41.9 41.9 ...
##  $ start_lng     : num [1:7386] -87.6 -87.6 -87.7 -87.6 -87.6 ...
##  $ end_lat       : num [1:7386] 41.9 41.9 41.9 42 41.9 ...
##  $ end_lng       : num [1:7386] -87.6 -87.7 -87.6 -87.7 -87.7 ...
##  $ member_type   : chr [1:7386] "member" "member" "member" "member" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ride_id = col_character(),
##   ..   rideable_type = col_character(),
##   ..   date_start = col_date(format = "%d/%m/%Y"),
##   ..   time_start = col_time(format = "%H:%M"),
##   ..   date_end = col_date(format = "%d/%m/%Y"),
##   ..   time_end = col_time(format = "%H:%M"),
##   ..   end_station_id = col_character(),
##   ..   start_lat = col_double(),
##   ..   start_lng = col_double(),
##   ..   end_lat = col_double(),
##   ..   end_lng = col_double(),
##   ..   member_type = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Calculating Months and Weeks using date_start

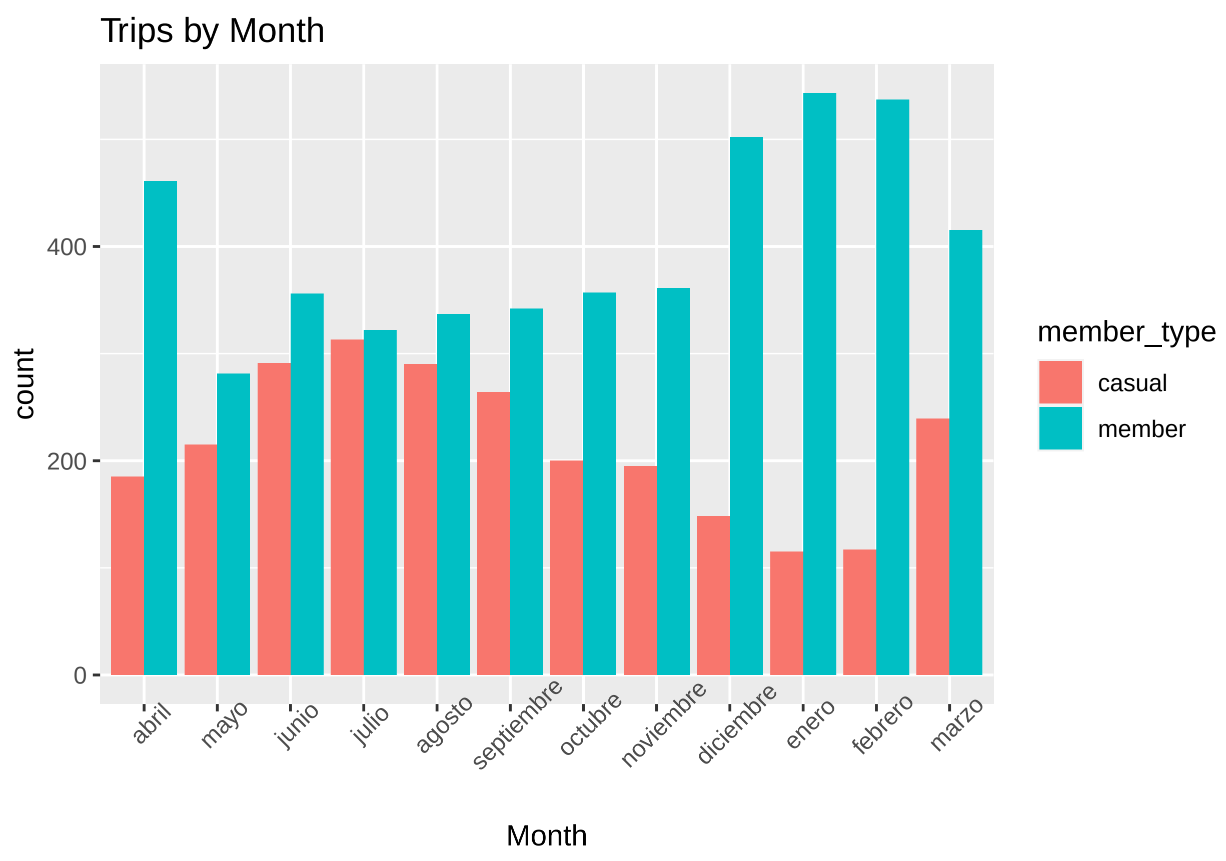
The column date_start is used to add two new columns to the dataframe : *Month* and *Weekday*.

```r
#Lets plot values based on month and day of the week
#First we create variables of the week and month of each observation:
divvy_trip_data$Month <- as.Date(cut(divvy_trip_data$date_start, breaks = "month"))
divvy_trip_data$Weekday <- as.Date(cut(divvy_trip_data$date_start,breaks = "day",start.on.monday = TRUE

divvy_trip_data <- mutate(divvy_trip_data, Month = months(Month))
divvy_trip_data <- mutate(divvy_trip_data, Weekday = wday(Weekday, label = TRUE))
```

Analysis by Month

Rides by month are plotted. We can see that member types ride more often from December to April, while casual riders ride more often from May to September, this coincides with the summer and spring months in Chicago, and might explain the raise in casual riders who come to enjoy the city as tourists.

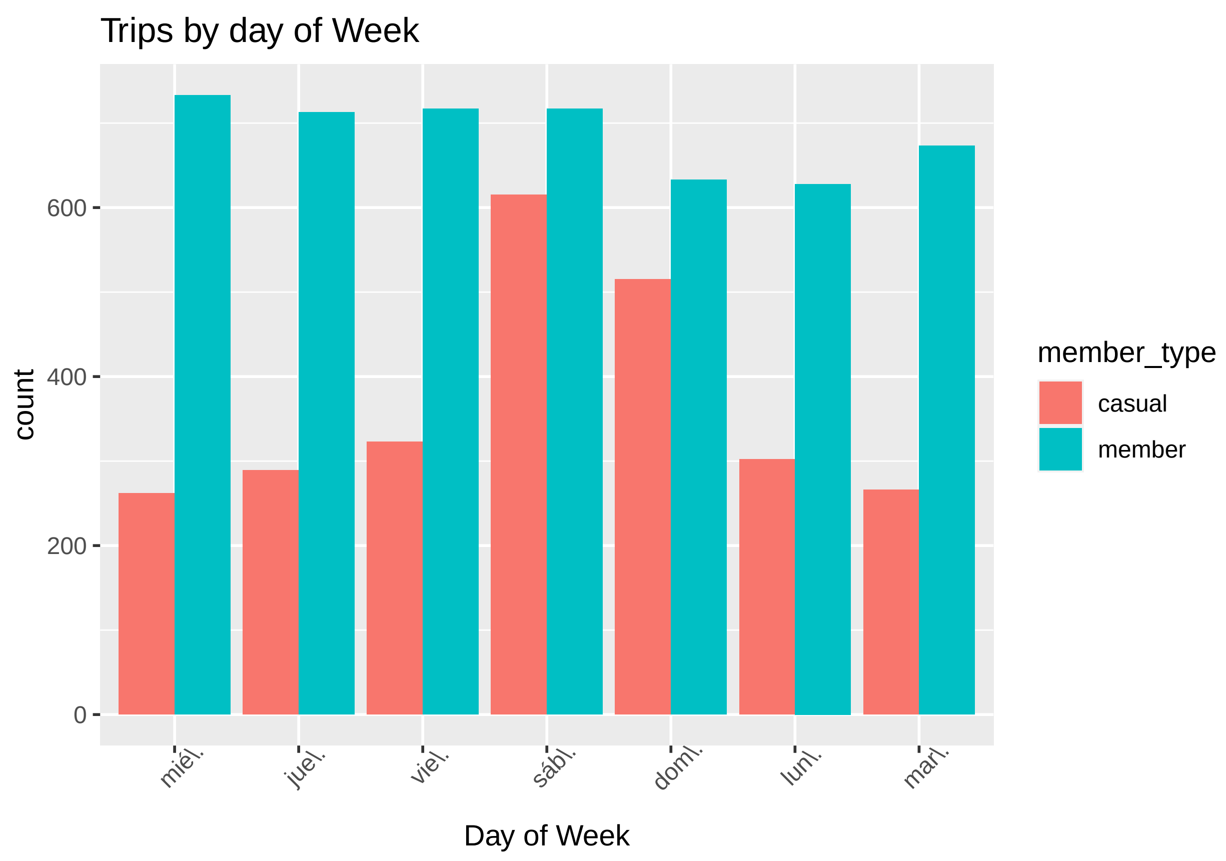
```r
#Factor is needed on Month column to order and display every label of the month
divvy_trip_data$Month <- factor(divvy_trip_data$Month, levels = unique(divvy_trip_data$Month))

#Plotting values based on month and filled by member_type
ggplot(data = divvy_trip_data) +
  geom_bar(mapping = aes(x = Month, fill = member_type), position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 45)) +
  ggtitle("Trips by Month") +
  xlab("Month")
```



Analysis by Week

Rides by day of the week are plotted. The barchart clearly shows that casual riders prefer riding on weekends, while members are constant regardless of day of the week. This might be because casual riders ride as a leisurely activity while members commute to work.

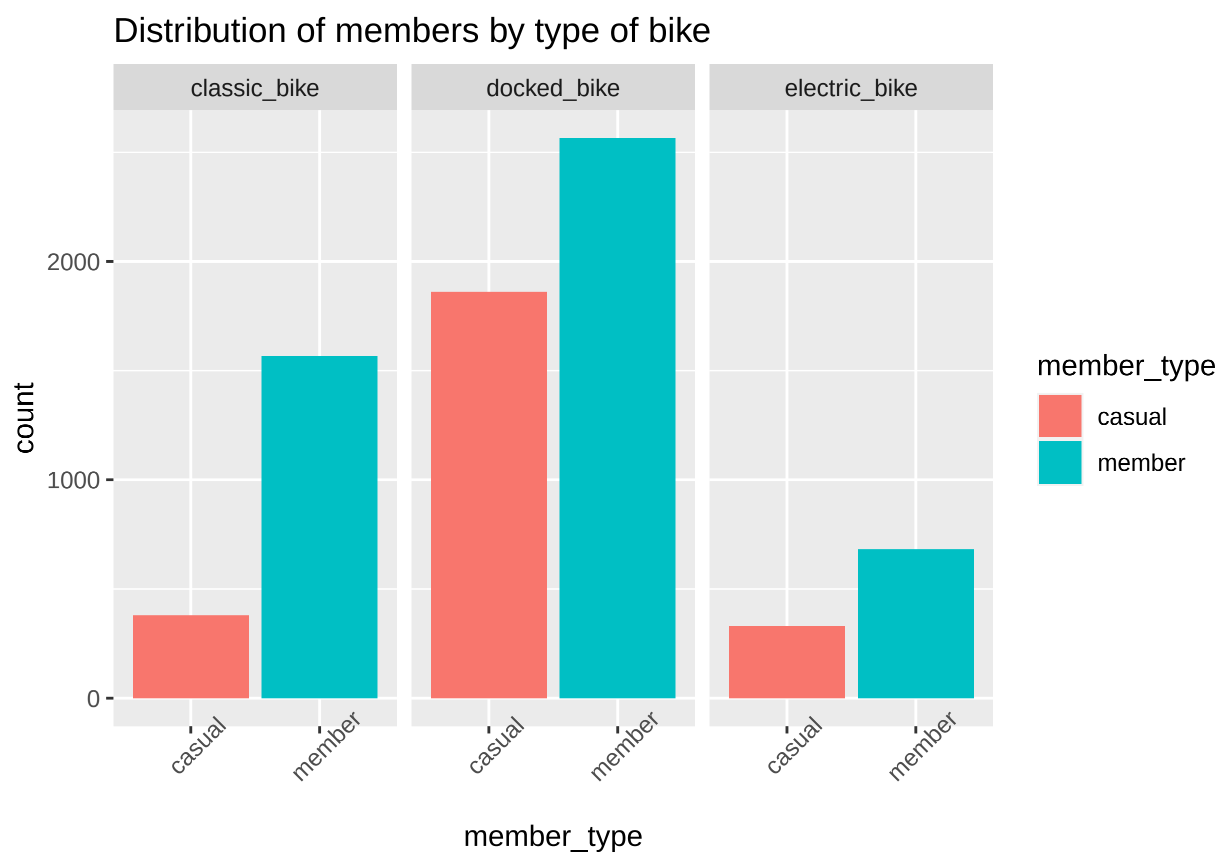
```r
#We do the same for days of the week
#Factor is needed on Month column to order and display every label of the month
divvy_trip_data$Weekday <- factor(divvy_trip_data$Weekday, levels = unique(divvy_trip_data$Weekday))


ggplot(data = divvy_trip_data) +
  geom_bar(mapping = aes(x = Weekday, fill = member_type), position=position_dodge()) +
  theme(axis.text.x = element_text(angle = 45)) +
  ggtitle("Trips by day of Week") +
  xlab("Day of Week")
```



Distribution by type of bike

There are three types of bikes: classic, docked and electric. From the plot it is observed that although each kind of bike is preferred by members, the classic bikes show a greater difference than the other two types.

```r
#Distribution of members by type of bike
ggplot(data = divvy_trip_data) +
  geom_bar(mapping = aes(x = member_type, fill = member_type)) +
  facet_wrap(~rideable_type) +
  ggtitle("Distribution of members by type of bike") +
  theme(axis.text.x = element_text(angle = 45))
```



Biketrip times

Calculating the average trip time of casual and member riders show that casual riders ride for more than double the time of member riders.

```
#calculate time difference between rides
time_diff <- with(divvy_trip_data, difftime(time_end, time_start, units = "mins") )

#add column to dataframe
divvy_trip_data<- mutate(divvy_trip_data,trip_time = time_diff)

#means
#we make a subset to calculate mean trip time by member_type

df <- subset(divvy_trip_data, select = c(trip_time, member_type))


#We calculate the mean trip time for member riders
avg_trip_time_member = df %>%
  filter(member_type == "member") %>%
  summarise(Mean = mean(trip_time))

#move element from a row into a value
mean_trip_time_member = avg_trip_time_member[1]
rm(avg_trip_time_member)
```

```r
#Do the same for casual members
avg_trip_time_casual = df %>%
  filter(member_type == "casual") %>%
  summarise(Mean = mean(trip_time))

#move element from a row into a value
mean_trip_time_casual = avg_trip_time_casual[1]
rm(avg_trip_time_casual)

print(paste("Mean time for casual riders: ", mean_trip_time_casual))
```

```
## [1] "Mean time for casual riders:  34.9681181959565"
```

```r
print(paste("Mean time for member riders: ", mean_trip_time_member))
```

```
## [1] "Mean time for member riders:  14.7833402575821"
```

Recommendations for a marketing strategy

- I would recommend getting more classic bikes as they seem to be the most attractive to member riders and might be a factor in upgrading to a membership.
- Introducing new plans for the summer months to target casual riders. This plans should be made during winter months in preparation as they have the lowest number of riders.
- Memberships rates could be lowered for the warmer months to target casual riders as these are the months they the most.

Limitations of my analysis

The data I worked with had certain limitations that should be addressed, for example, the dataset had no demographic data where I could apply an analysis to determine if member riders are mainly male or female, this would have made the analysis easier to perform and could have had an impact on the marketing strategy proposed. The data itself had no information about pay grades regarding the membership or no financial data regarding the users that hire the membership, this could have been useful to adjust the pay grade to make the membership more attractive to the specific salary range of our target audience.

References

1. Information about how to get a sample size link
2. For the use of geosphere in the code: link