Google Analytics Capstone: Case Study 1

Alecsander Guimarães

2022-04-29

Introduction

This is a case study for a fictional company, Cyclist, a bike-sharing company. The main objective is to perform many real-world tasks of a junior data analyst. The data source can be found in https://divvy-tripdata.s3.amazonaws.com/index.html.

Scenario

"You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations."

1. Ask

Business Task

The junior data analyst have to answer the following question: How do annual members and casual riders use Cyclistic bikes differently?

Objective

Design a new marketing strategy to convert casual riders into annual members.

2. Prepare

Data Source

The data used has been made available by Motivate International Inc. under this license and the can be accessed through this link.

Data Organisation

For this case study, the last twelves months of data (April 2021 - March 2022) were used. Their are stored in csv files with thirteen columns.

Credibility of the Data

The data is collected directly by the company and includes all the rides recorded. The data is also current and it is published each month.

Licensing, privacy, security, and accessibility

All the personal information was removed from the data, which is also a limitation, because it does not allow to identify recurrent users or if they are from Chicago. The data license can be accessed in https://ride.divvybikes.com/data-license-agreement.

Data information

The data contains information about the user type, initial and final station, as well as start and end time. This allows to identify the differences between the casual and the annual member.

Problems with the dataset

The data contains missing fields and some inconsistencies. These errors are mainly in the fields associated with stations and duration times, and can be solved through by data cleansing.

3. Process

The tool

R was the selected tool because it works well with a large volume of data, and contains excellent options for cleaning, processing and visualizing the data.

Data Integrity

The data was previously combined in one dataframe, with 5,723,532 rows and 13 columns. Due to hardware limitations, a sample, with 600000 rows, has been extracted and it is with this sample that the study will be carried out. As mentioned before, there are some errors in the dataset:

- Missing values in the start and end station variables
- Missing values in the end_lat and end_lng
- Negative trip times

Loading the libraries

```
library(janitor)
```

```
##
## Attaching package: ’janitor’

## The following objects are masked from ’package:stats’:
##
##     chisq.test, fisher.test
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(skimr)
library(lubridate)
```

```
##
## Attaching package: ’lubridate’

## The following objects are masked from ’package:base’:
##
##     date, intersect, setdiff, union
```

```
library(scales)
```

```
##
## Attaching package: ’scales’

## The following object is masked from ’package:purrr’:
##
##     discard

## The following object is masked from ’package:readr’:
##
##     col_factor
```

Loading the data

```
df <- read_csv('~/Cursos/Data Google/Capstone/Data/CSV/sample_dataset.csv')
```

```
## New names:
## Rows: 600000 Columns: 14
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (7): ride_id, rideable_type, start_station_name, start_station_id, end_... dbl
## (5): ...1, start_lat, start_lng, end_lat, end_lng dttm (2): started_at,
## ended_at
## 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.
## * ‘‘ -> ‘...1‘
```

```
df$...1 <- NULL
colnames(df)
```

```
##  [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"
```

Checking the data

Information about the data

```
glimpse(df)
```

```
## Rows: 600,000
## Columns: 13
## $ ride_id            <chr> "C6346A8C12AA8154", "D970AE041F6ED3E1", "863363FF77~
## $ rideable_type      <chr> "electric_bike", "electric_bike", "electric_bike", ~
## $ started_at         <dttm> 2021-08-14 15:16:52, 2021-10-02 13:33:14, 2021-11-~
## $ ended_at           <dttm> 2021-08-14 15:27:04, 2021-10-02 14:54:58, 2021-11-~
## $ start_station_name <chr> "Clinton St & Lake St", "State St & Harrison St", "~
## $ start_station_id   <chr> "13021", "SL-007", "13146", "KA1503000072", "13137"~
## $ end_station_name   <chr> "Wells St & Elm St", "Streeter Dr & Grand Ave", "St~
## $ end_station_id     <chr> "KA1504000135", "13022", "13276", "SL-008", "13323"~
## $ start_lat          <dbl> 41.88550, 41.87402, 41.91838, 41.88313, 41.93758, 4~
## $ start_lng          <dbl> -87.64183, -87.62771, -87.63630, -87.63732, -87.644~
## $ end_lat            <dbl> 41.90342, 41.89216, 41.93128, 41.87208, 41.95283, 4~
## $ end_lng            <dbl> -87.63458, -87.61188, -87.63880, -87.62954, -87.649~
## $ member_casual      <chr> "member", "casual", "member", "member", "casual", "~
```

Looking for null variables

```
colSums(is.na(df))
```

```
##            ride_id      rideable_type         started_at           ended_at 
##                  0                  0                  0                  0 
## start_station_name   start_station_id   end_station_name     end_station_id 
##              78149              78149              83777              83777 
##          start_lat          start_lng            end_lat            end_lng 
##                  0                  0                504                504 
##      member_casual 
##                  0
```

Checking duplicated data

```
sum(duplicated(df))
```

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

Data cleaning

Creating new columns and filtering

During the analysis the trip duration by hour, day and month will be evaluated. Therefore it is necessary to create new columns that provide this data. In addition, trips with negative time will be discarded. The empty station fields will be removed in a next step, because they have no influence at first.

```
df_1 <- df %>%
  mutate(
    day_week = wday(started_at, label = TRUE, abbr = FALSE),
    month = month(started_at, label = TRUE, abbr = FALSE),
    hour = as.factor(hour(started_at)),
    rideable_type = as.factor(rideable_type),
    member_casual = as.factor(member_casual),
    trip_time = difftime(ended_at, started_at, units = 'mins')
  ) %>%
  filter(trip_time > 0)
```

Removing empty stations

```
df_station_cleaned <- df_1 %>%
  drop_na(c('start_station_name', 'end_station_name'))
colSums(is.na(df_station_cleaned))
```

```
##            ride_id      rideable_type         started_at           ended_at 
##                  0                  0                  0                  0 
## start_station_name   start_station_id   end_station_name     end_station_id 
##                  0                  0                  0                  0 
##          start_lat          start_lng            end_lat            end_lng
##                 0                   0                   0                   0
##     member_casual            day_week               month                hour
##                 0                   0                   0                   0
##         trip_time
##                 0
```

As shown, the data contains null variables. I chose to divide into two dataframes, one with all the data, except the trips that are negative, and another without the null stations.

4. Analyze

Statistical evaluation

For the initial analysis, we want to know the basics about the data, for that we are going to use the function `skim_without_charts`.

```
skim_without_charts(df_1)
```

Table 1: Data summary

Name	df_1
Number of rows	599938
Number of columns	17
Column type frequency:	
character	5
difftime	1
factor	5
numeric	4
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ride_id	0	1.00	16	16	0	599938	0
start_station_name	78147	0.87	3	53	0	848	0
start_station_id	78147	0.87	3	37	0	839	0
end_station_name	83741	0.86	10	53	0	832	0
end_station_id	83741	0.86	3	37	0	824	0

Variable type: difftime

skim_variable	n_missing	complete_rate	min	max	median	n_unique
trip_time	0	1	0.02 mins	47776.7 mins	11.72 mins	11751

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
rideable_type	0	1	FALSE	3	cla: 340812, ele: 227424, doc: 31702
member_casual	0	1	FALSE	2	mem: 333128, cas: 266810
day_week	0	1	TRUE	7	sáb: 103072, dom: 91073, sex: 85902, qua: 82195
month	0	1	TRUE	12	jul: 86105, ago: 83945, set: 79267, jun: 76186
hour	0	1	FALSE	24	17: 59956, 18: 51673, 16: 49743, 15: 41898

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
start_lat	0	1	41.90	0.05	41.65	41.88	41.90	41.93	42.07
start_lng	0	1	-87.65	0.03	-87.84	-87.66	-87.64	-87.63	-87.52
end_lat	504	1	41.90	0.05	41.48	41.88	41.90	41.93	42.15
end_lng	504	1	-87.65	0.03	-87.85	-87.66	-87.64	-87.63	-87.52

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
started_at	0	1	2021-04-01 00:21:09	2022-03-31 23:55:50	2021-08-18 01:49:22	588080
ended_at	0	1	2021-04-01 00:35:30	2022-04-01 04:57:33	2021-08-18 02:43:56	588122

```
mean(df_1$trip_time)
```

```
## Time difference of 21.46205 mins
```

It shows us that:

- Busiest days are:
  - Sat: 103072
  - Sun: 91073
  - Fri: 85902
- Busiest months are:
  - Jul: 86105
  - Aug: 83945
  - Sep: 79267
- Busiest hours are:
  - 17: 59956
  - 18: 51673
  - 16: 49743
- User type:
  - Member: 333128

    - Casual: 266810

- Bike type:

    - Classic: 340812
    - Eletric: 227424
    - Docked: 31702

- The mean of trip_time is 21.4 minutes;
- The max of trip_time is 47776 minutes.

Summarizing the data

The data will be aggregated and saved to facilitate further analysis and also in the creation of graphs. They will be divided into csv files, having a summary of the data as follows:

- By user
- By hour
- By day
- By month
- By station
- By bike

All data summaries will have the total trips, the average trip time, and the sum of all trip times. The only exception is the summary data for stations, which contains only the number of trips.

By user

```
summary_users <- df_1 %>%
  group_by(member_casual) %>%
  summarise(
    total_trips_users = n(),
    mean_trip_time_users = mean(trip_time),
    sum_trip_time_users = sum(trip_time),
    .groups = 'drop'
  )
summary_users
```

```
## # A tibble: 2 x 4
##   member_casual total_trips_users mean_trip_time_users sum_trip_time_users
##   <fct>                     <int> <drtn>               <drtn>
## 1 casual                   266810 31.60857 mins        8433484 mins
## 2 member                   333128 13.33546 mins        4442415 mins
```

```
write_csv(summary_users, 'summary_users.csv')
```

By hour

```
summary_hour <- df_1 %>%
  group_by(hour, member_casual) %>%
  summarise(
    total_trips_hour = n(),
    mean_trip_time_hour = mean(trip_time),
    sum_trip_time_hour = sum(trip_time),
    .groups = 'drop'
  )
summary_hour
```

```
## # A tibble: 48 x 5
##    hour  member_casual total_trips_hour mean_trip_time_hour sum_trip_time_hour
##    <fct> <fct>                    <int> <drtn>              <drtn>
##  1 0     casual                    5690 36.60832 mins       208301.35 mins
##  2 0     member                    3548 14.02192 mins        49749.78 mins
##  3 1     casual                    4133 45.52979 mins       188174.62 mins
##  4 1     member                    2379 15.95584 mins        37958.95 mins
##  5 2     casual                    2707 34.58711 mins        93627.30 mins
##  6 2     member                    1342 12.56533 mins        16862.67 mins
##  7 3     casual                    1481 43.96340 mins        65109.80 mins
##  8 3     member                     791 15.11104 mins        11952.83 mins
##  9 4     casual                    1023 53.57489 mins        54807.12 mins
## 10 4     member                     924 13.66228 mins        12623.95 mins
## # ... with 38 more rows
```

```
write_csv(summary_hour, 'summary_day.csv')
```

By day

```
summary_day <- df_1 %>%
  group_by(day_week, member_casual) %>%
  summarise(
    total_trips_day = n(),
    mean_trip_time_day = mean(trip_time),
    sum_trip_time_day = sum(trip_time),
    .groups = 'drop'
  )
summary_day
```

```
## # A tibble: 14 x 5
##    day_week      member_casual total_trips_day mean_trip_time_~ sum_trip_time_d~
##    <ord>         <fct>                   <int> <drtn>           <drtn>
##  1 domingo       casual                  50410 37.31458 mins    1881028.1 mins
##  2 domingo       member                  40663 15.42027 mins     627034.2 mins
##  3 segunda-feira casual                  30599 32.63564 mins     998617.8 mins
##  4 segunda-feira member                  46253 13.00607 mins     601569.6 mins
##  5 terça-feira   casual                  29001 27.61646 mins     800905.0 mins
##  6 terça-feira   member                  51162 12.48130 mins     638568.2 mins
##  7 quarta-feira  casual                  30008 25.47934 mins     764583.9 mins
##  8 quarta-feira  member                  52187 12.55154 mins     655027.3 mins
##  9 quinta-feira  casual                  30845 27.66275 mins     853257.6 mins
## 10 quinta-feira  member                  49836 12.46944 mins     621426.9 mins
## 11 sexta-feira   casual                  38278 30.22392 mins    1156911.2 mins
## 12 sexta-feira   member                  47624 13.01151 mins     619660.3 mins
## 13 sábado        casual                  57669 34.30231 mins    1978180.1 mins
## 14 sábado        member                  45403 14.95779 mins     679128.6 mins
```

```
write_csv(summary_day, 'summary_week.csv')
```

By month

```
summary_month <- df_1 %>%
  group_by(month, member_casual) %>%
  summarise(
    total_trips_month = n(),
    mean_trip_time_month = mean(trip_time),
    sum_trip_time_month = sum(trip_time),
    .groups = 'drop'
  )
summary_month
```

```
## # A tibble: 24 x 5
##    month     member_casual total_trips_month mean_trip_time_mo~ sum_trip_time_m~
##    <ord>     <fct>                     <int> <drtn>             <drtn>
##  1 janeiro   casual                     1900 54.61225 mins      103763.3 mins
##  2 janeiro   member                     8976 11.94692 mins      107235.5 mins
##  3 fevereiro casual                     2250 23.42467 mins       52705.5 mins
##  4 fevereiro member                     9786 11.22260 mins      109824.3 mins
##  5 março     casual                     9551 36.00960 mins      343927.7 mins
##  6 março     member                    20486 11.78569 mins      241441.6 mins
##  7 abril     casual                    14296 38.44233 mins      549571.5 mins
##  8 abril     member                    20891 14.67041 mins      306479.4 mins
##  9 maio      casual                    26967 35.83284 mins      966304.2 mins
## 10 maio      member                    28823 14.66341 mins      422643.5 mins
## # ... with 14 more rows
```

```
write_csv(summary_month, 'summary_month.csv')
```

By station

```
summary_start_station <- df_station_cleaned %>%
  group_by(start_station_name, member_casual) %>%
  summarise(
    total_trips_start_station = n()
  ) %>%
  arrange(-total_trips_start_station)
```

```
## ‘summarise()‘ has grouped output by ’start_station_name’. You can override
## using the ‘.groups‘ argument.
```

```r
summary_start_station
```

```
## # A tibble: 1,554 x 3
## # Groups:   start_station_name [824]
##    start_station_name       member_casual total_trips_start_station
##    <chr>                    <fct>                             <int>
##  1 Streeter Dr & Grand Ave  casual                             6984
##  2 Millennium Park          casual                             3340
##  3 Michigan Ave & Oak St    casual                             2915
##  4 Kingsbury St & Kinzie St member                             2509
##  5 Wells St & Concord Ln    member                             2429
##  6 Clark St & Elm St        member                             2428
##  7 Shedd Aquarium           casual                             2208
##  8 Theater on the Lake      casual                             2137
##  9 Wells St & Elm St        member                             2129
## 10 Wells St & Concord Ln    casual                             1993
## # ... with 1,544 more rows
```

```r
write.csv(summary_start_station, 'summary_start_station.csv')
```

```r
summary_end_station <- df_station_cleaned %>%
  group_by(end_station_name, member_casual) %>%
  summarise(
    total_trips_end_station = n()
    ) %>%
  arrange(-total_trips_end_station)
```

```
## ‘summarise()‘ has grouped output by ’end_station_name’. You can override using
## the ‘.groups‘ argument.
```

```r
summary_end_station
```

```
## # A tibble: 1,540 x 3
## # Groups:   end_station_name [819]
##    end_station_name         member_casual total_trips_end_station
##    <chr>                    <fct>                           <int>
##  1 Streeter Dr & Grand Ave  casual                           7182
##  2 Millennium Park          casual                           3429
##  3 Michigan Ave & Oak St    casual                           3128
##  4 Kingsbury St & Kinzie St member                           2522
##  5 Wells St & Concord Ln    member                           2514
##  6 Clark St & Elm St        member                           2427
##  7 Theater on the Lake      casual                           2261
##  8 Shedd Aquarium           casual                           2247
##  9 Wells St & Elm St        member                           2187
## 10 Dearborn St & Erie St    member                           2008
## # ... with 1,530 more rows
```

```r
write_csv(summary_end_station, 'summary_end_station.csv')
```

By bike

```
summary_bike <- df_1 %>%
  group_by(rideable_type, member_casual, day_week) %>%
  summarise(
    total_trips_rideable_type = n(),
    mean_trip_time_rideable_type = mean(trip_time),
    sum_trip_time_rideable_type = sum(trip_time)
  )
```

```
## ‘summarise()‘ has grouped output by ’rideable_type’, ’member_casual’. You can
## override using the ‘.groups‘ argument.
```

```
summary_bike
```

```
## # A tibble: 35 x 6
## # Groups:   rideable_type, member_casual [5]
##    rideable_type member_casual day_week      total_trips_ridea~ mean_trip_time_~
##    <fct>         <fct>         <ord>                      <int> <drtn>
##  1 classic_bike  casual        domingo                    26733 32.63785 mins
##  2 classic_bike  casual        segunda-feira              14350 29.33746 mins
##  3 classic_bike  casual        terça-feira                12996 26.02313 mins
##  4 classic_bike  casual        quarta-feira               14007 25.50816 mins
##  5 classic_bike  casual        quinta-feira               14506 26.02200 mins
##  6 classic_bike  casual        sexta-feira                18400 26.96039 mins
##  7 classic_bike  casual        sábado                     31050 30.36735 mins
##  8 classic_bike  member        domingo                    26451 16.08438 mins
##  9 classic_bike  member        segunda-feira              29046 13.29500 mins
## 10 classic_bike  member        terça-feira                31602 13.03655 mins
## # ... with 25 more rows, and 1 more variable:
## #   sum_trip_time_rideable_type <drtn>
```

```
write.csv(summary_bike, 'summary_bike.csv')
```

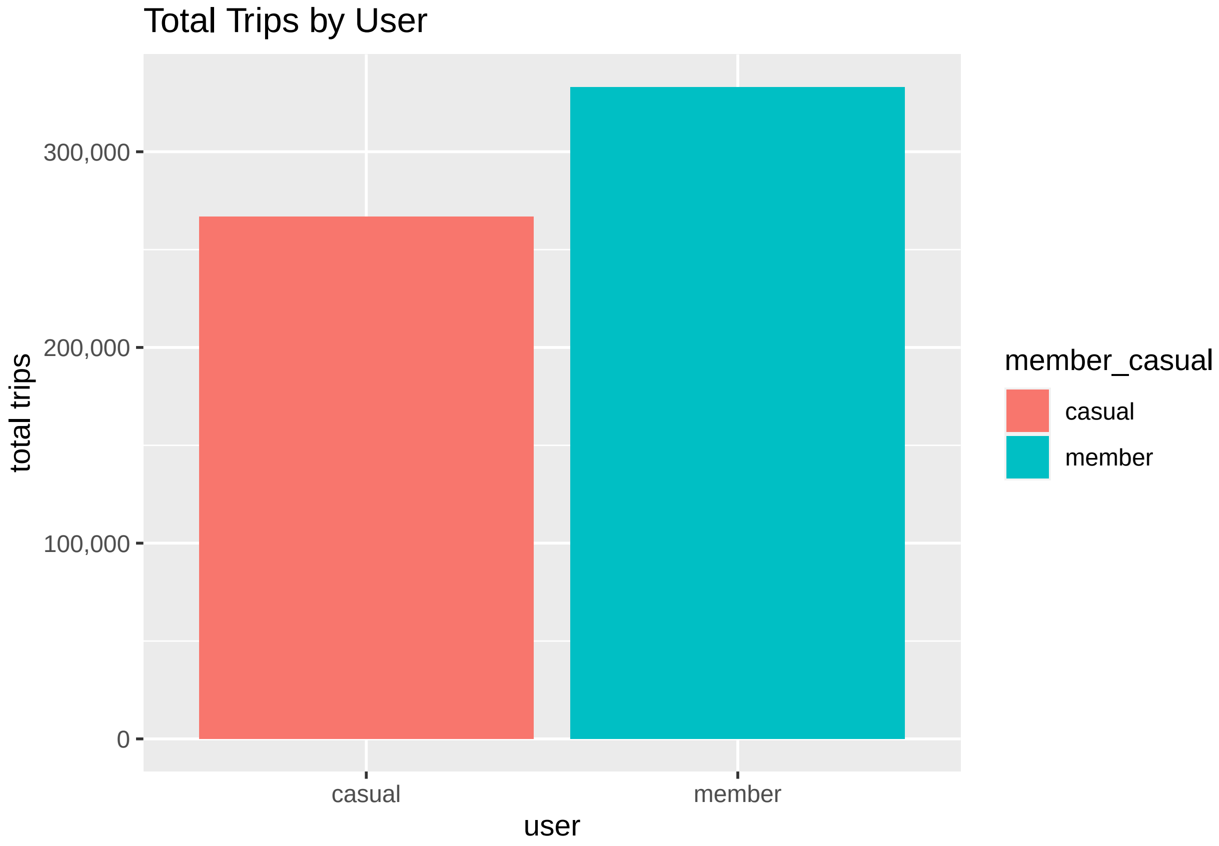
Visualizing the data

The graphs will be made from the summarized data and divided in the same way.

By user

```
summary_users %>%
  ggplot(aes(member_casual, total_trips_users, fill = member_casual))+
  geom_col()+
  labs(
    title = 'Total Trips by User',
    x = 'user',
    y = 'total trips'
  )+
  scale_y_continuous(labels = comma)
```

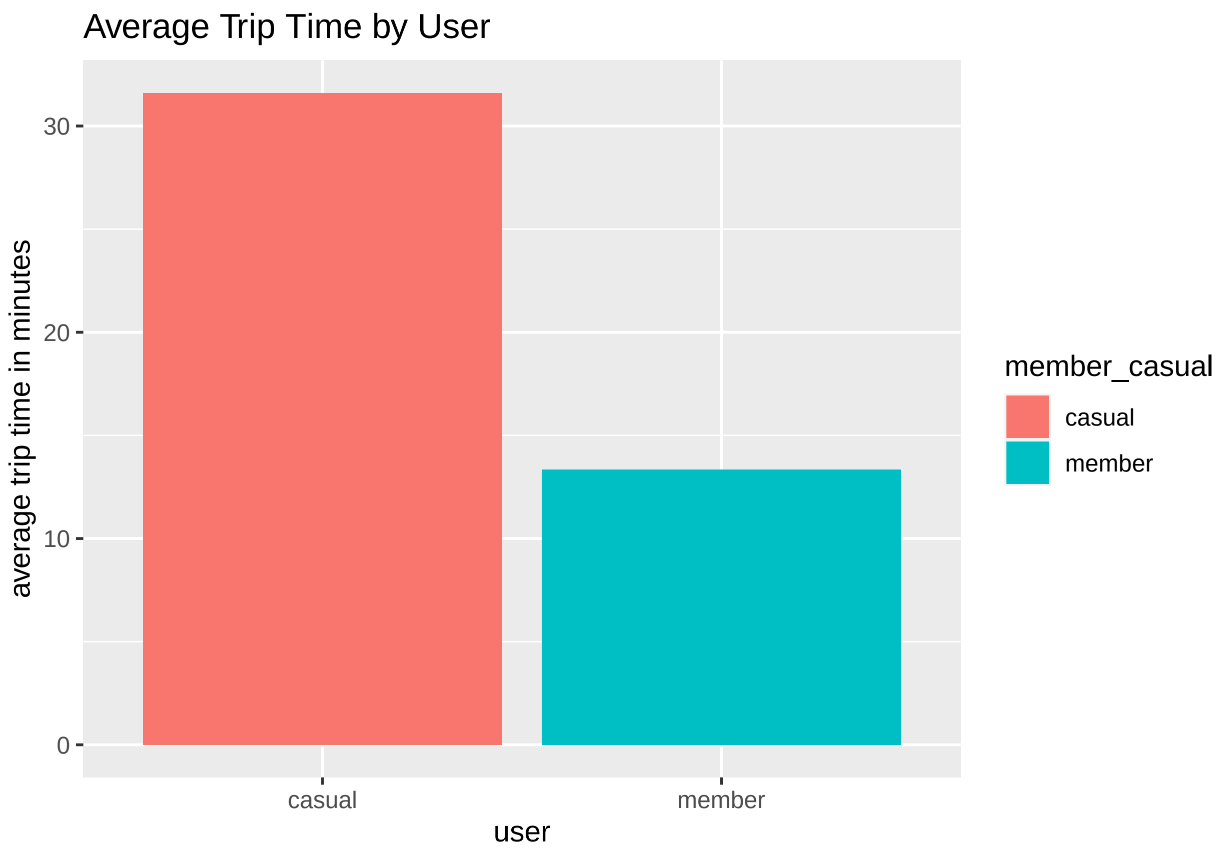
Total Trips by User



```
summary_users %>%
  ggplot(aes(member_casual, mean_trip_time_users, fill = member_casual))+
  geom_col()+
  labs(
    title = 'Average Trip Time by User',
    x = 'user',
    y = 'average trip time in minutes'
  )
```

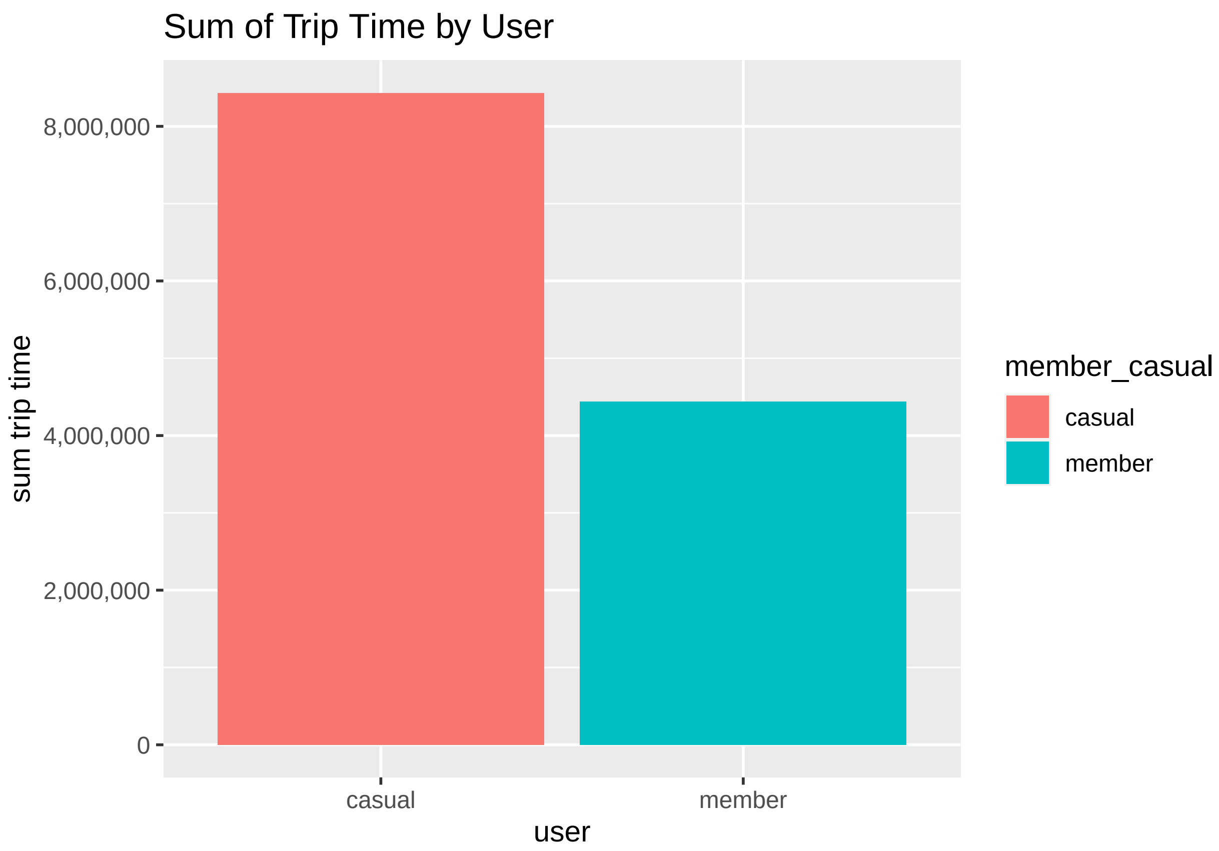
Average Time Trip by User

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



```
summary_users %>%
  ggplot(aes(member_casual, sum_trip_time_users, fill = member_casual))+
  geom_col()+
  labs(
    title = 'Sum of Trip Time by User',
    x = 'user',
    y = 'sum trip time'
  )+
  scale_y_continuous(labels = comma)
```

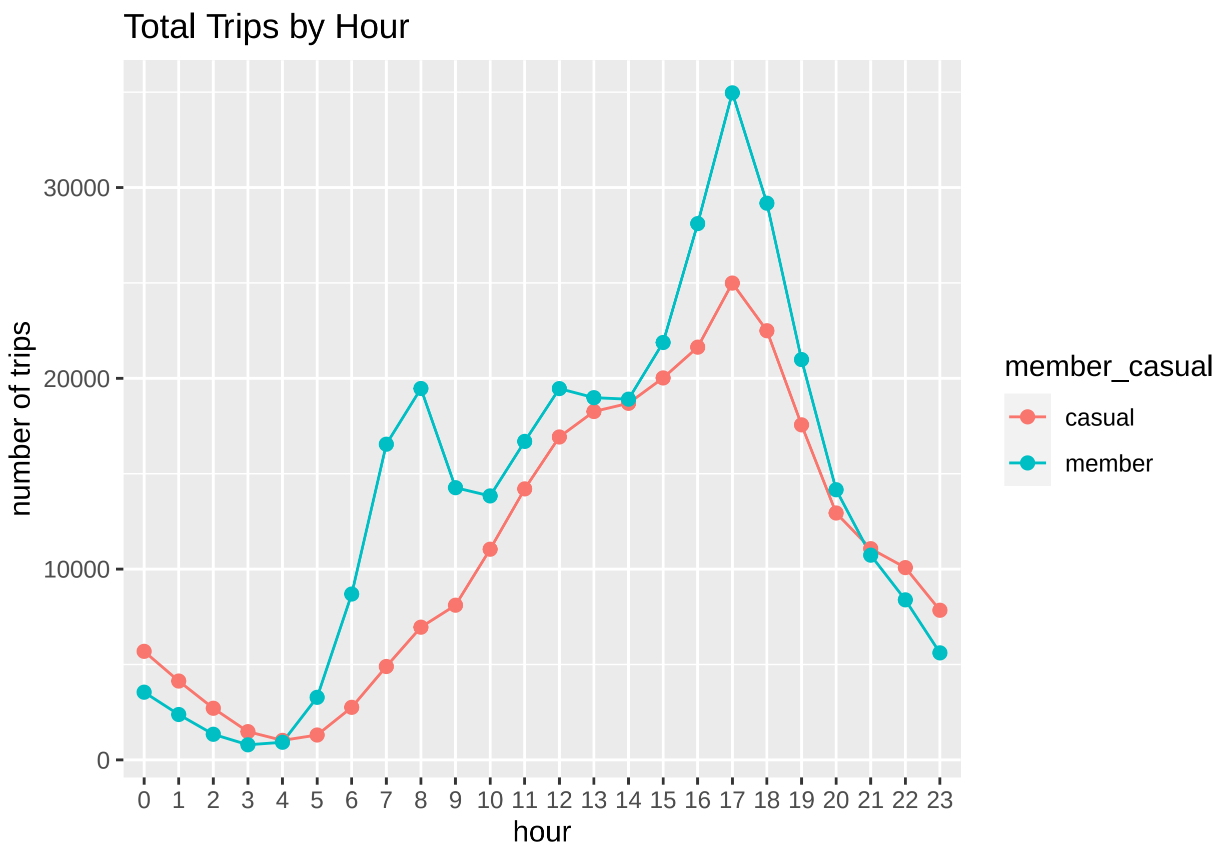
Total Trip Time by User



By hour

```
summary_hour %>%
  ggplot(aes(hour, total_trips_hour, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  labs(
    title = 'Total Trips by Hour',
    y = 'number of trips'
  )
```

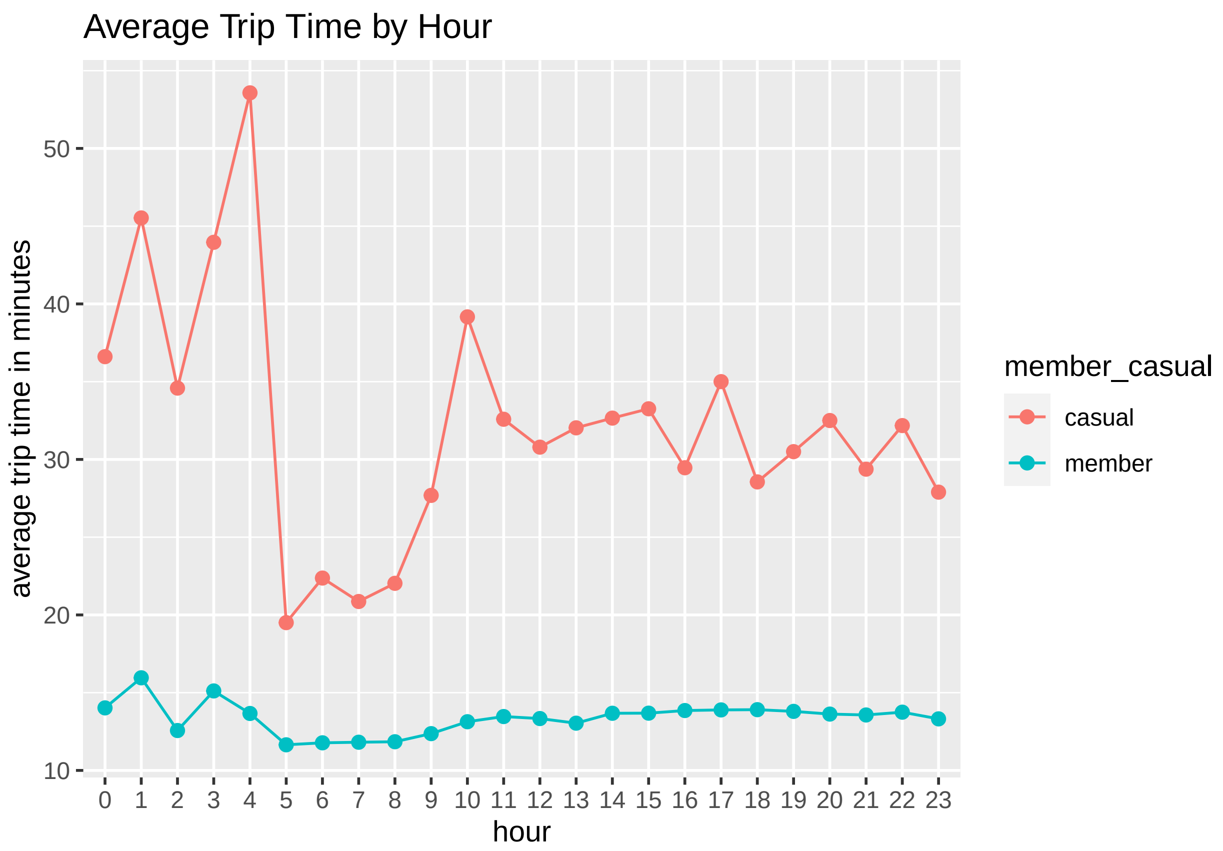
Total Trips by Hour



```
summary_hour %>%
  ggplot(aes(hour, mean_trip_time_hour, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  labs(
    title = 'Average Trip Time by Hour',
    y = 'average trip time in minutes'
  )
```

Average Trip Time by Hour

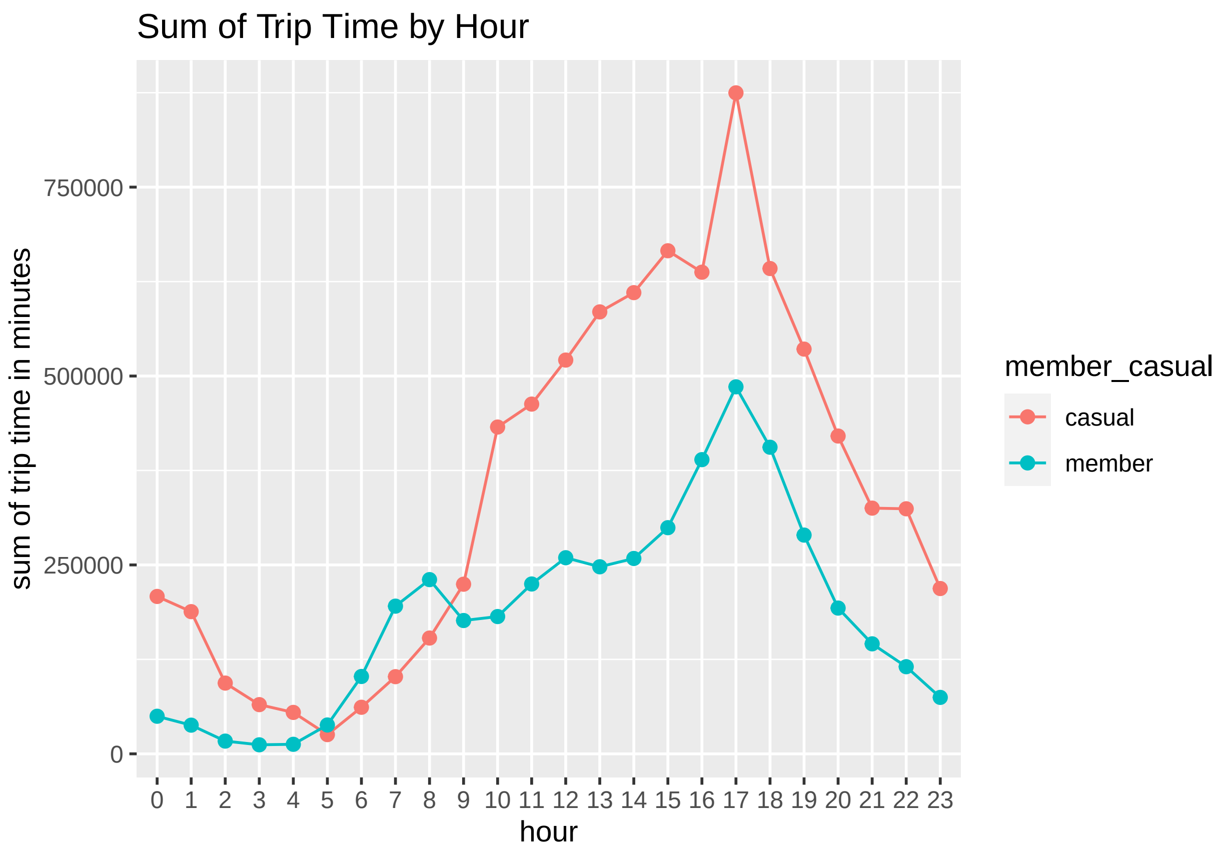
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



```
summary_hour %>%
  ggplot(aes(hour, sum_trip_time_hour, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  labs(
    title = 'Sum of Trip Time by Hour',
    y = 'sum of trip time in minutes'
  )
```

Total Trip Time by Hour

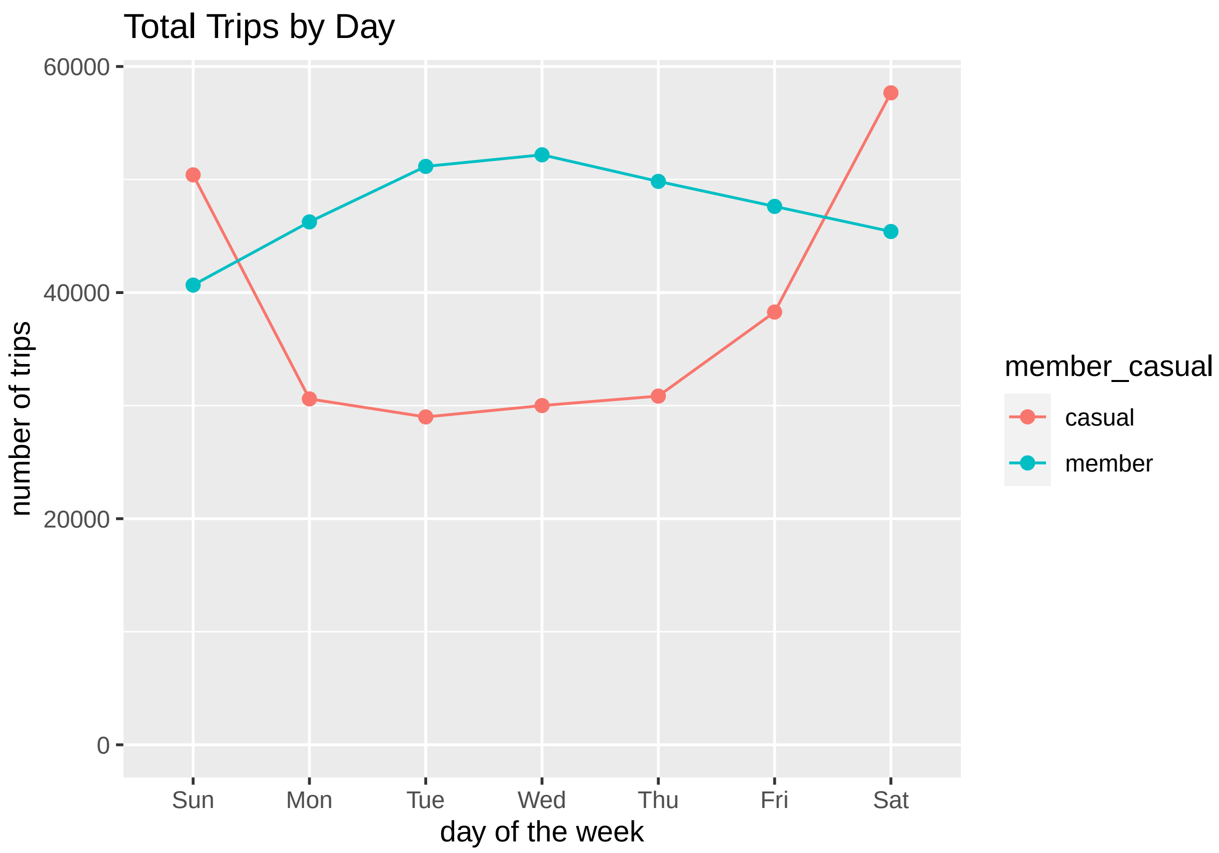
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



By day

```
summary_day %>%
  ggplot(aes(day_week, total_trips_day, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  expand_limits(y = 0)+
  scale_x_discrete(
    labels = c(
      'domingo' = 'Sun', 'segunda-feira' = 'Mon', 'terça-feira' = 'Tue',
      'quarta-feira' = 'Wed', 'quinta-feira' = 'Thu', 'sexta-feira' = 'Fri',
      'sábado' = 'Sat'
    ))+
  labs(
    title = 'Total Trips by Day',
    x = 'day of the week',
    y = 'number of trips'
  )
```

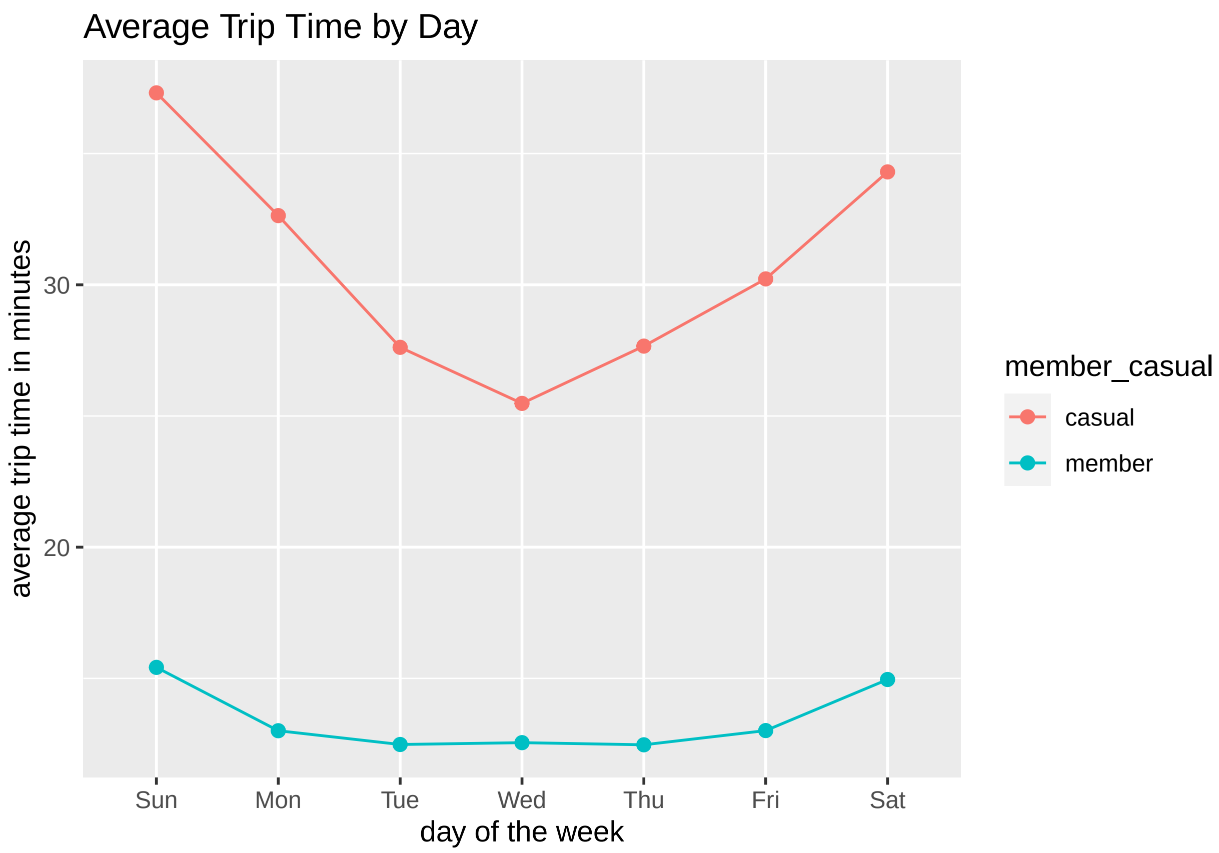
Total Trips by Day



```
summary_day %>%
  ggplot(aes(day_week, mean_trip_time_day, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  scale_x_discrete(
    labels = c(
      'domingo' = 'Sun', 'segunda-feira' = 'Mon', 'terça-feira' = 'Tue',
      'quarta-feira' = 'Wed', 'quinta-feira' = 'Thu', 'sexta-feira' = 'Fri',
      'sábado' = 'Sat'
    ))+
  labs(
    title = 'Average Trip Time by Day',
    x = 'day of the week',
    y = 'average trip time in minutes'
  )
```

Average Trip Time by Day

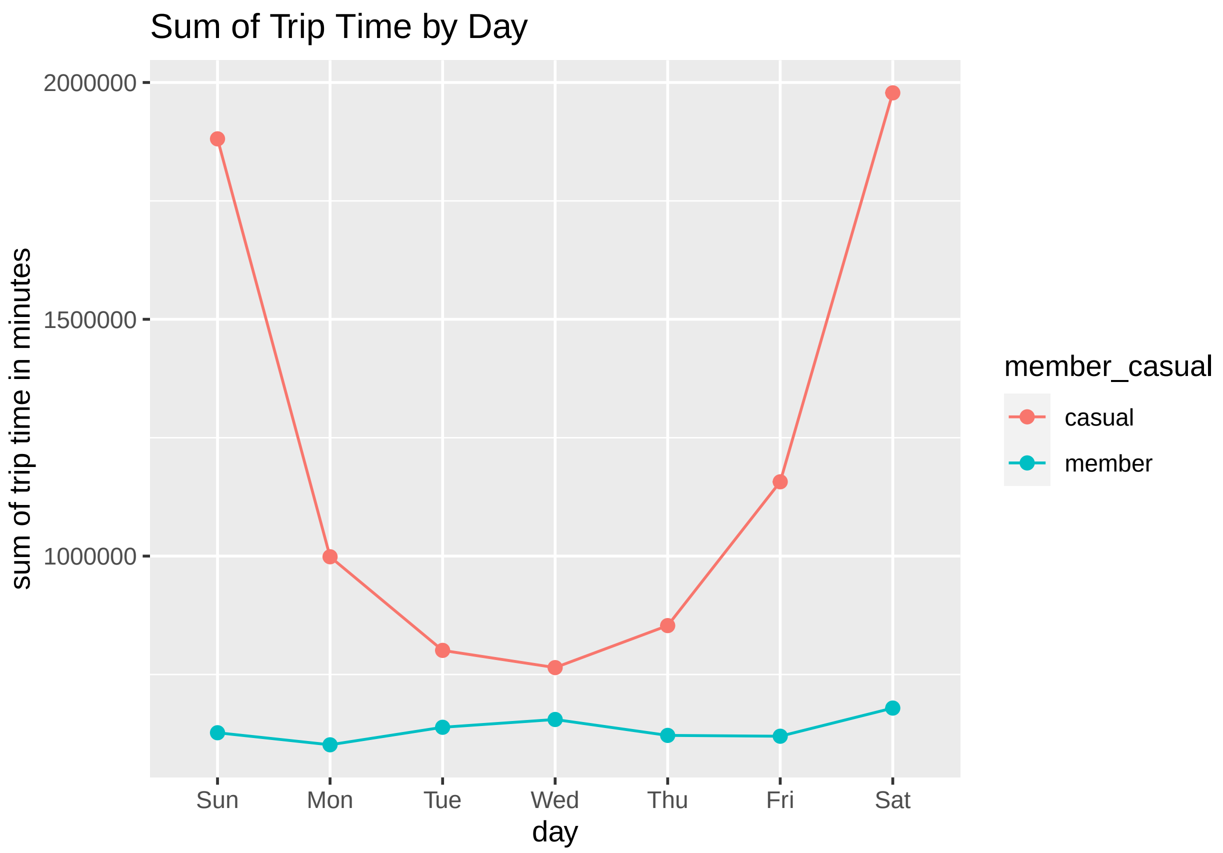
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



```
summary_day %>%
  ggplot(aes(day_week, sum_trip_time_day, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  scale_x_discrete(
    labels = c(
      'domingo' = 'Sun', 'segunda-feira' = 'Mon', 'terça-feira' = 'Tue',
      'quarta-feira' = 'Wed', 'quinta-feira' = 'Thu', 'sexta-feira' = 'Fri',
      'sábado' = 'Sat'
    ))+
  labs(
    title = 'Sum of Trip Time by Day',
    x = 'day',
    y = 'sum of trip time in minutes'
  )
```

Total Trip Time by Day

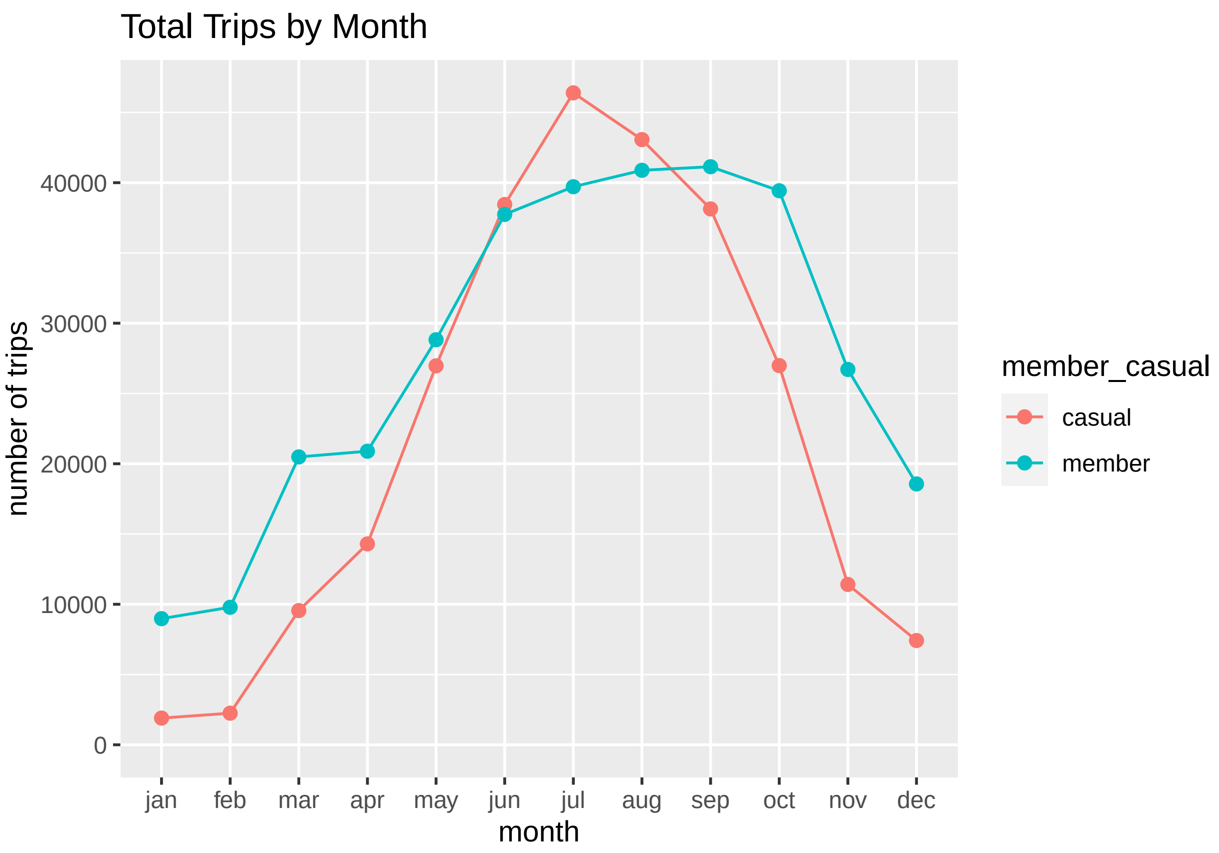
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



By month

```
summary_month %>%
  ggplot(aes(month, total_trips_month, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  expand_limits(y = 0)+
  scale_x_discrete(
    labels = c(
      'janeiro' = 'jan', 'fevereiro' = 'feb', 'março' = 'mar',
      'abril' = 'apr', 'maio' = 'may', 'junho' = 'jun',
      'julho' = 'jul', 'agosto' = 'aug', 'setembro' = 'sep',
      'outubro' = 'oct', 'novembro' = 'nov', 'dezembro' = 'dec'
    ))+
  labs(
    title = 'Total Trips by Month',
    x = 'month',
    y = 'number of trips'
  )
```

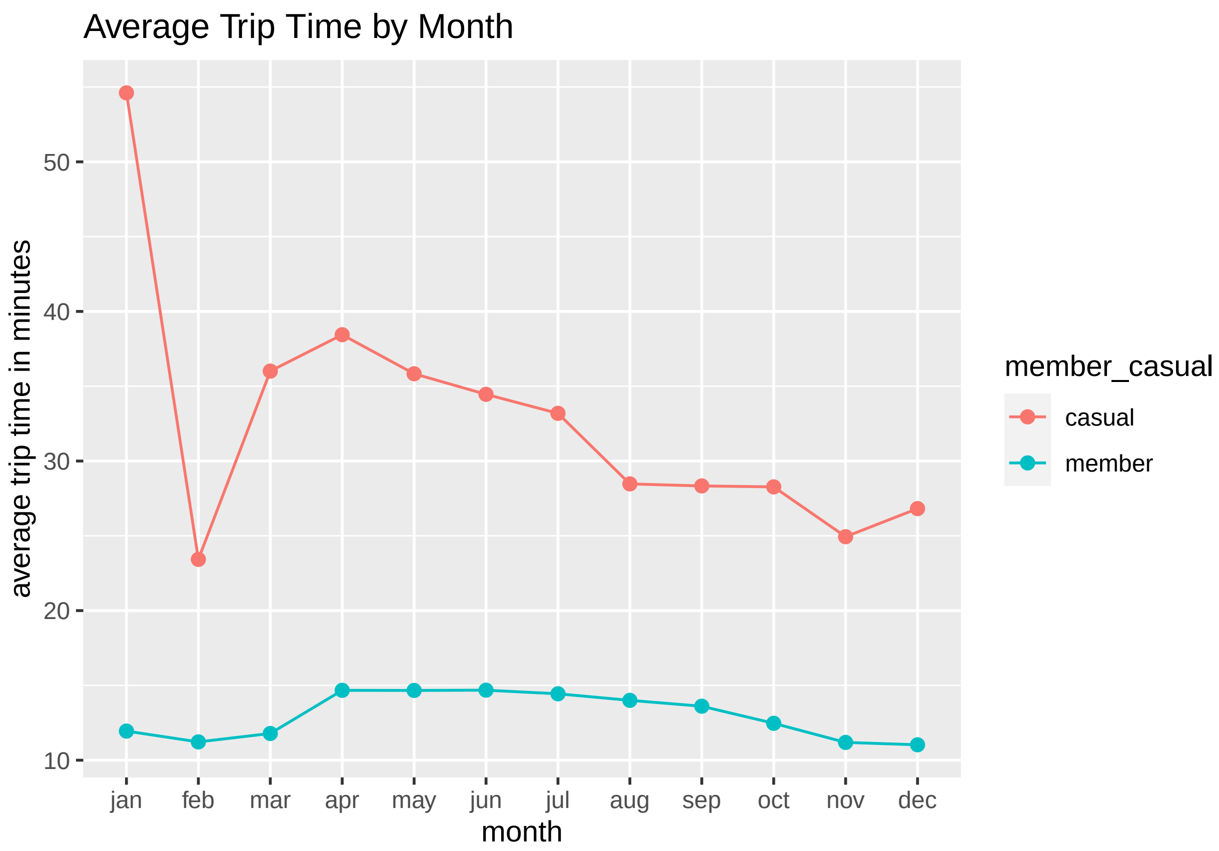
Total Trips by Month



```
summary_month %>%
  ggplot(aes(month, mean_trip_time_month, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  scale_x_discrete(
    labels = c(
      'janeiro' = 'jan', 'fevereiro' = 'feb', 'março' = 'mar',
      'abril' = 'apr', 'maio' = 'may', 'junho' = 'jun',
      'julho' = 'jul', 'agosto' = 'aug', 'setembro' = 'sep',
      'outubro' = 'oct', 'novembro' = 'nov', 'dezembro' = 'dec'
    ))+
  labs(
    title = 'Average Trip Time by Month',
    x = 'month',
    y = 'average trip time in minutes'
  )
```

Average Trip Time by Month

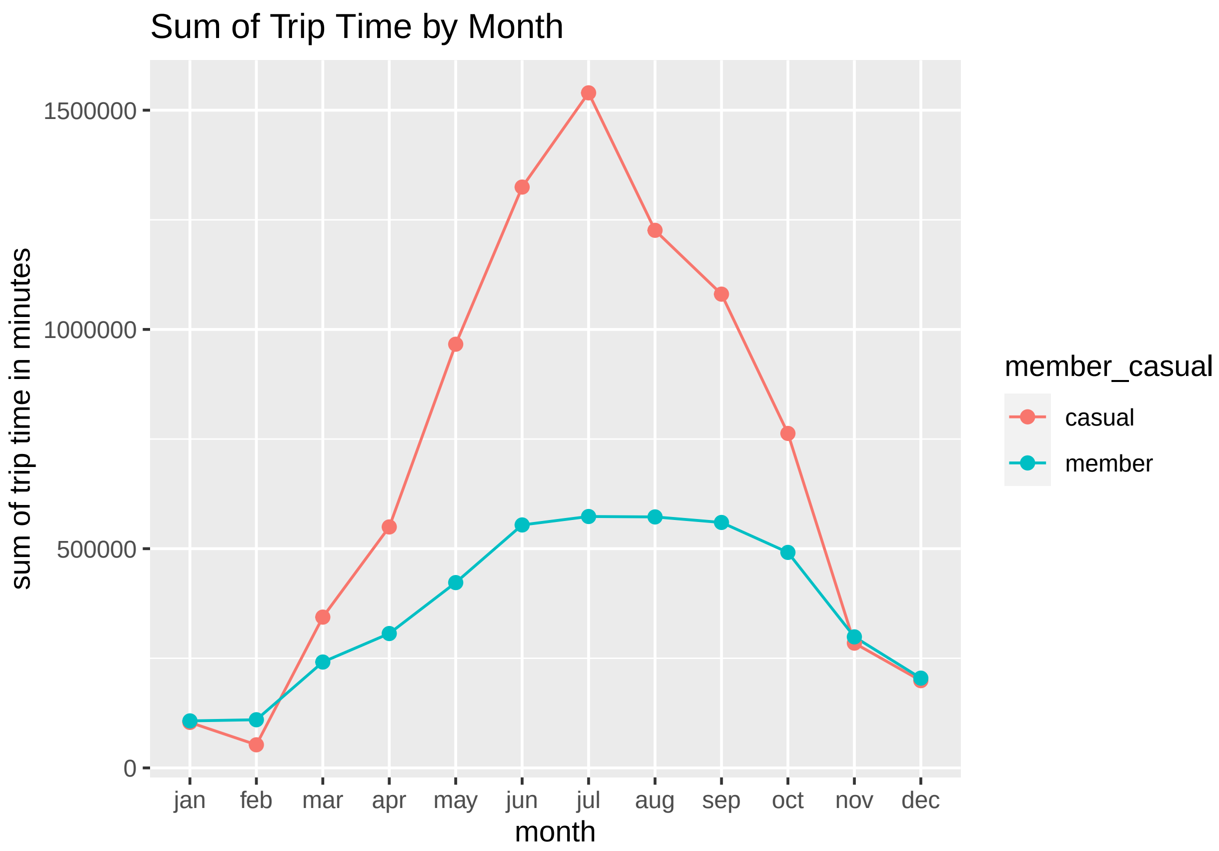
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



```
summary_month %>%
  ggplot(aes(month, sum_trip_time_month, group = member_casual))+
  geom_line(aes(color=member_casual))+
  geom_point(aes(color = member_casual), size = 2)+
  scale_x_discrete(
    labels = c(
      'janeiro' = 'jan', 'fevereiro' = 'feb', 'março' = 'mar',
      'abril' = 'apr', 'maio' = 'may', 'junho' = 'jun',
      'julho' = 'jul', 'agosto' = 'aug', 'setembro' = 'sep',
      'outubro' = 'oct', 'novembro' = 'nov', 'dezembro' = 'dec'
    ))+
  labs(
    title = 'Sum of Trip Time by Month',
    x = 'month',
    y = 'sum of trip time in minutes'
  )
```

Total Trip Time by Month

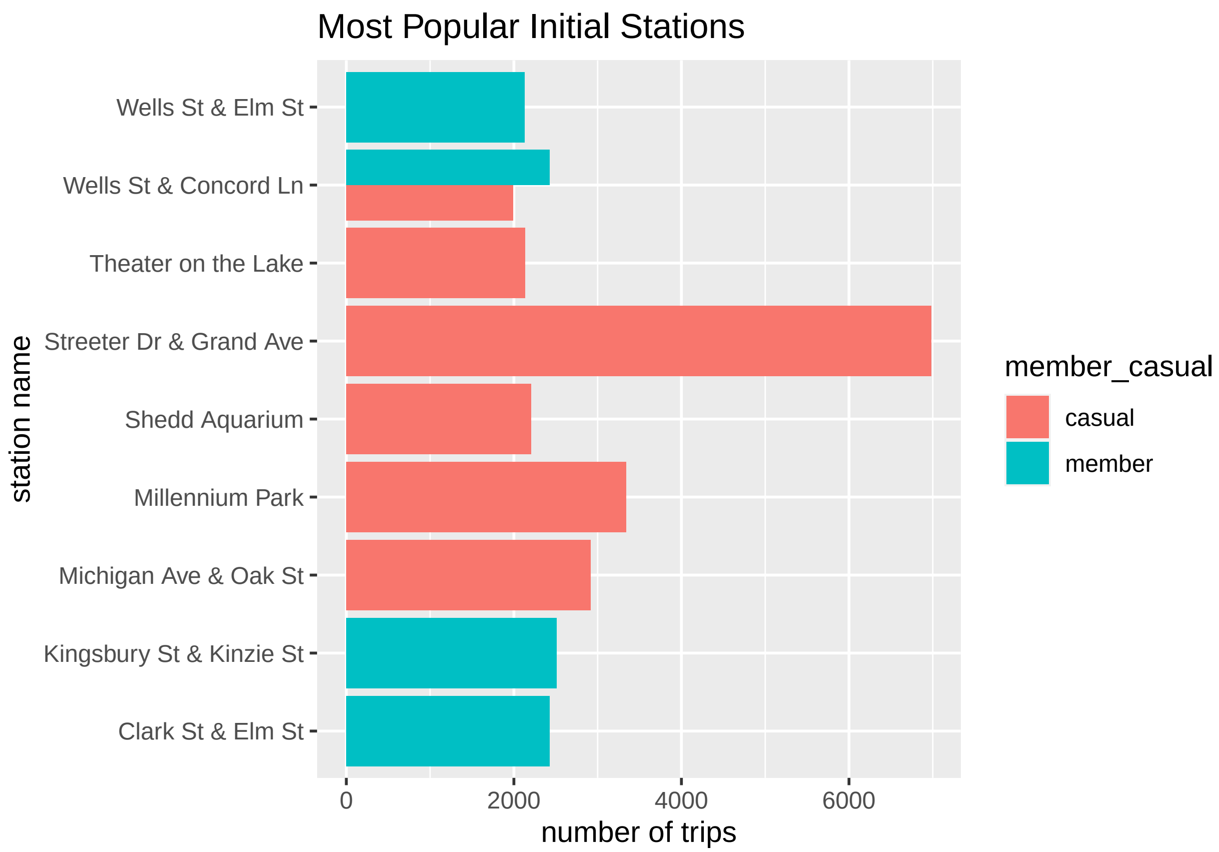
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



By station

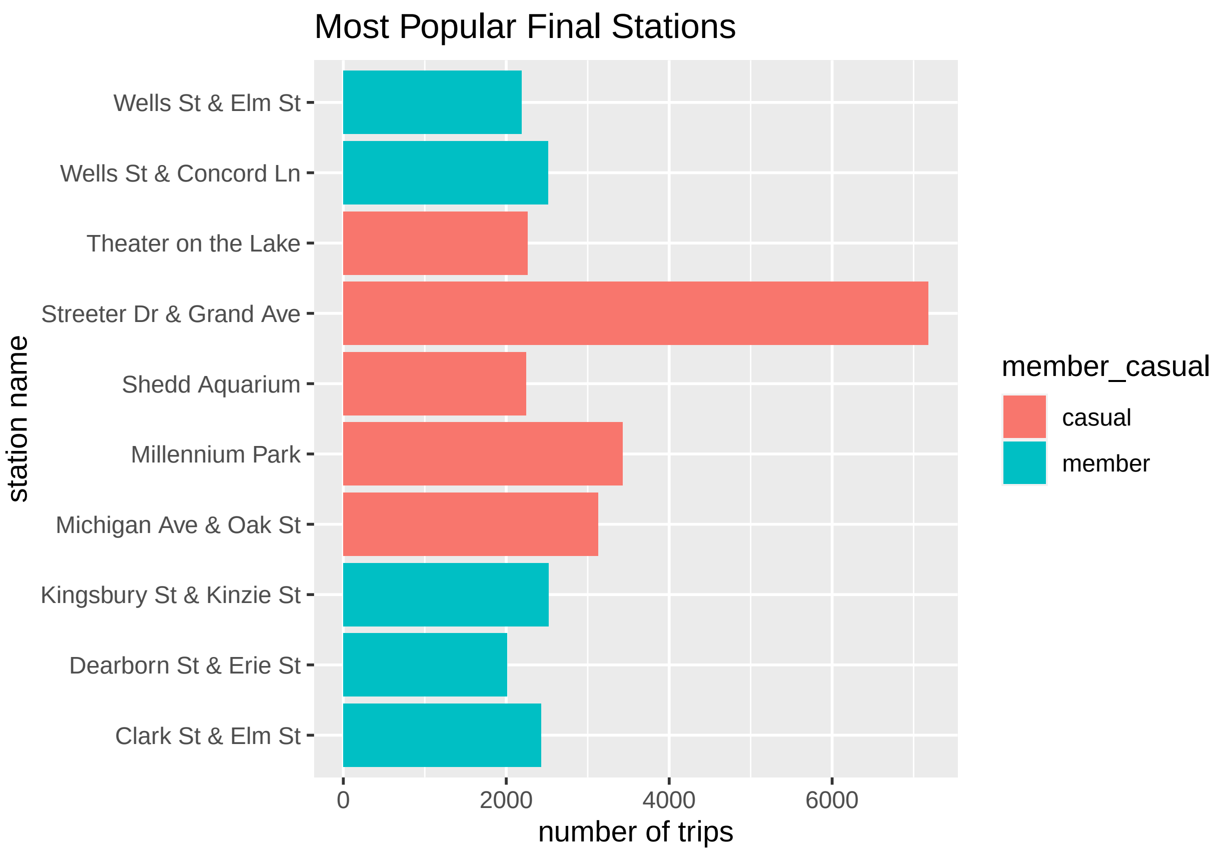
```
summary_start_station[1:10, ] %>%
  ggplot(aes(start_station_name, total_trips_start_station, fill = member_casual))+
  geom_col(position = 'dodge')+
  coord_flip()+
  labs(
    title = 'Most Popular Initial Stations',
    x = 'station name',
    y = 'number of trips'
  )
```

Most Popular Initial Stations



```
summary_end_station[1:10, ] %>%
  ggplot(aes(end_station_name, total_trips_end_station, fill = member_casual))+
  geom_col(position = 'dodge')+
  coord_flip()+
  labs(
    title = 'Most Popular Final Stations',
    x = 'station name',
    y = 'number of trips'
  )
```

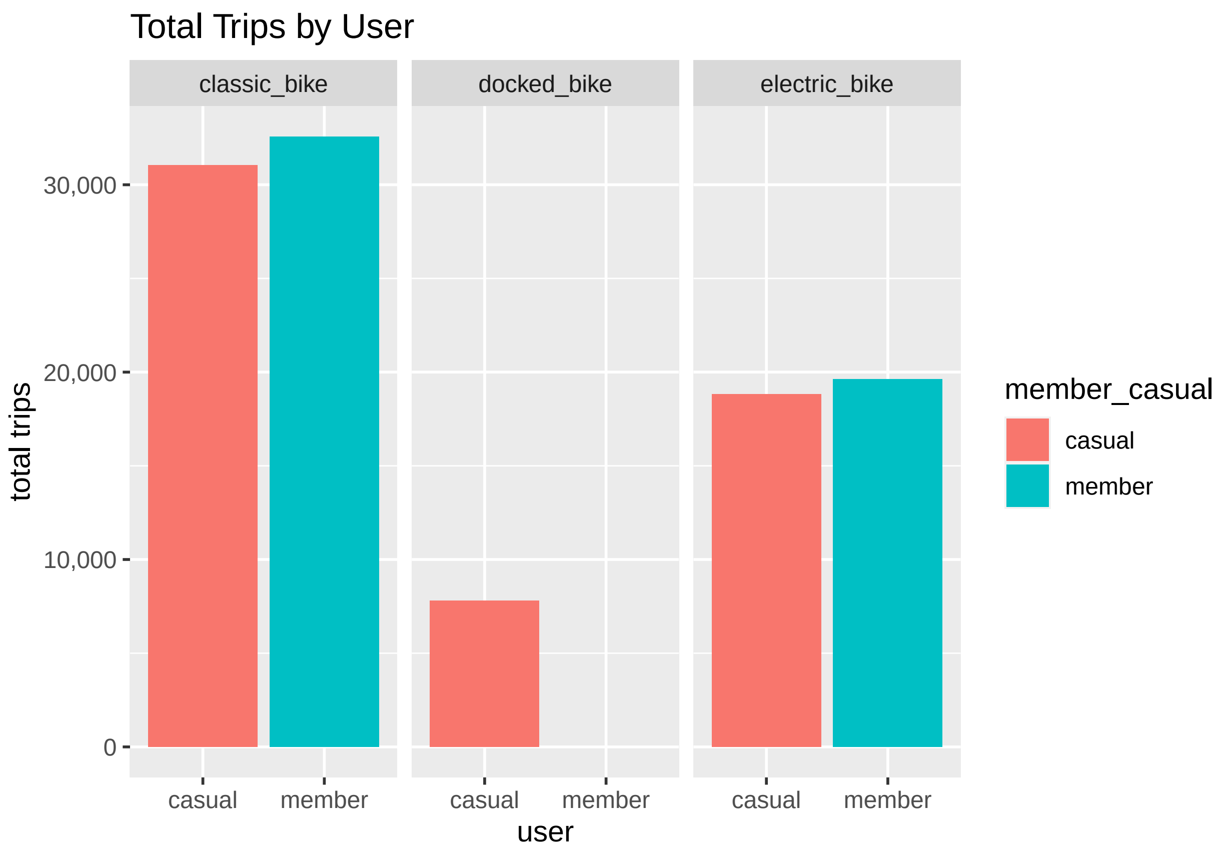
Most Popular Final Stations



By bike

```
summary_bike %>%
  ggplot(aes(member_casual, total_trips_rideable_type, fill = member_casual))+
  geom_col(position = 'dodge')+
  facet_wrap(~rideable_type)+
  labs(
    title = 'Total Trips by User',
    x = 'user',
    y = 'total trips'
  )+
  scale_y_continuous(labels = comma)
```

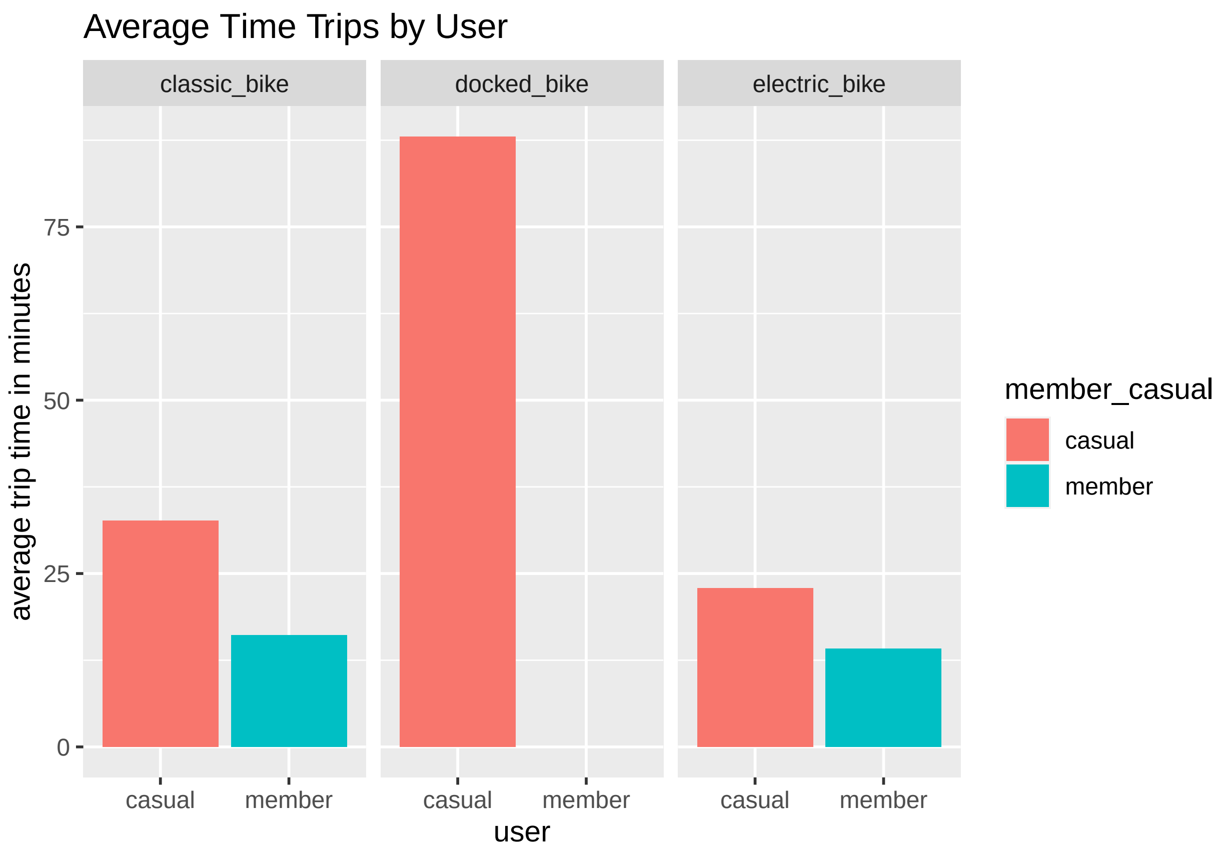
Total Trips by Bike



```
summary_bike %>%
  ggplot(aes(member_casual, mean_trip_time_rideable_type, fill = member_casual))+
  geom_col(position = 'dodge')+
  facet_wrap(~rideable_type)+
  labs(
    title = 'Average Time Trips by User',
    x = 'user',
    y = 'average trip time in minutes'
  )
```

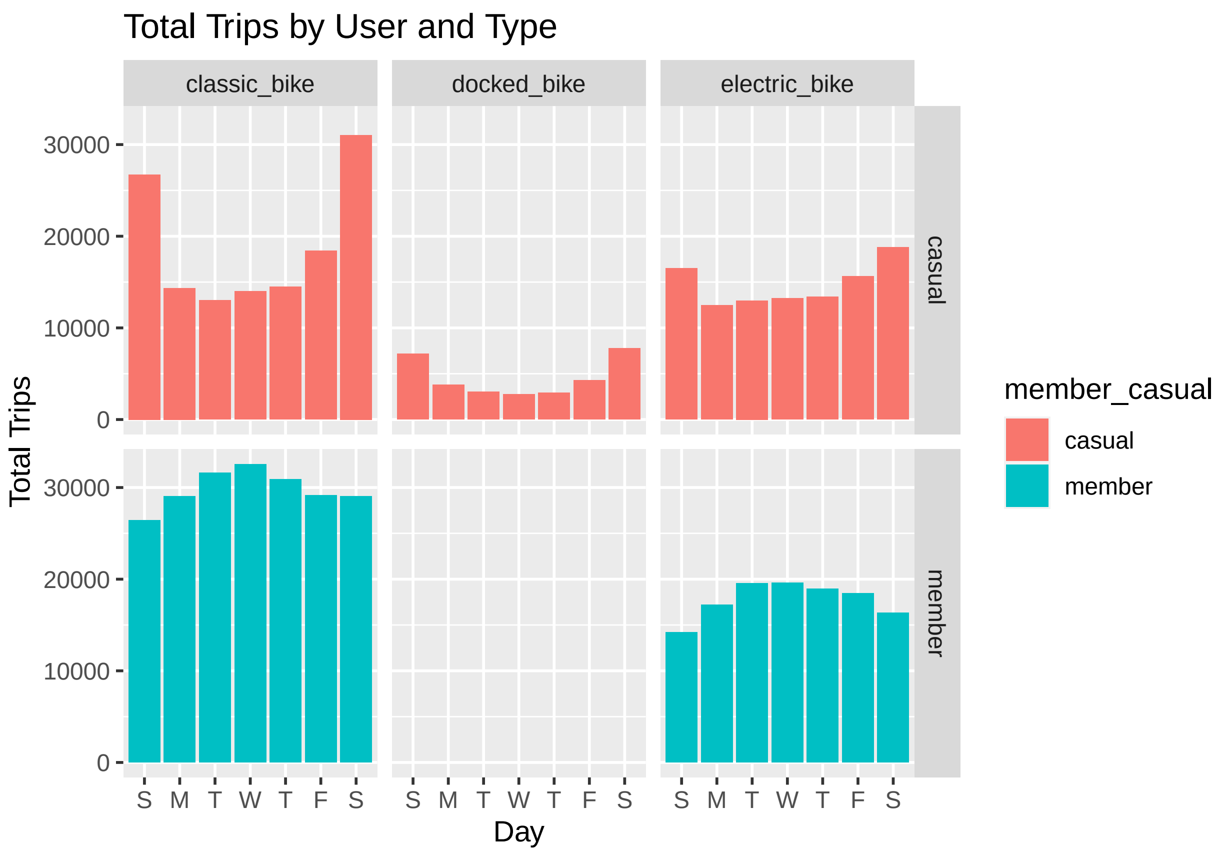
Average Time Trip by Bike

```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
```



```
summary_bike %>%
  ggplot(aes(day_week, total_trips_rideable_type, fill = member_casual))+
  geom_col(position = 'dodge')+
  facet_grid(~member_casual~rideable_type)+
  scale_x_discrete(
    labels = c(
      'domingo' = 'S', 'segunda-feira' = 'M', 'terça-feira' = 'T',
      'quarta-feira' = 'W', 'quinta-feira' = 'T', 'sexta-feira' = 'F',
      'sábado' = 'S'
    ))+
  labs(
    title = 'Total Trips by User and Type',
    x = 'Day',
    y = 'Total Trips'
    )
```

Total Trips by Bike



5. Share

The analysis can be acessed through this link

6. Act

Limitations

The main limitation of the data is the impossibility of observing individual user behavior and even knowing whether the user is from the city or not.

Recomendations

1. The marketing team should create a focused campaign for the busiest periods. This corresponds to weekends and Friday for weekdays, the months of July to September, and the late afternoon and early evening hours, from 3 pm to 6 pm
2. The most visited stations could also be the focus of a campaign
3. Another behavior that could be observed by the marketing team is the longer average travel time of casual users
4. Focusing on the type of bicycle, one opportunity is the combination of the classic type of bicycle and weekend trips that have a high volume