Cyclistic Bike-Share Case Study

Arya Tiwari

2024-03-21

Case study for data analysis step by step process

First install and load the required libraries

```r
library(tidyverse) #helps wrangle data
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
# Use the conflicted package to manage conflicts
library(conflicted)
# Set dplyr::filter and dplyr::lag as the default choices
conflict_prefer("filter", "dplyr")
```

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

```r
conflict_prefer("lag", "dplyr")
```

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

Step 1 Ask:

Company Background: Cyclistic is a bike-sharing company in Chicago, offering a fleet of bicycles for public use across the city. Cyclistic operates on a subscription-based model, with options for both casual riders and annual members. The company is committed to an inclusive and accessible mode of transportation, striving to cater to the needs of diverse user segments.

Data Context: Our data, sourced from Cyclistic's historical trip records, covers a year's worth of ride information, providing insights into the usage patterns of both casual riders and annual members. The key variables in our dataset include the start and end times of each trip, start and end station details, rideable type, and whether the rider is a casual user or an annual member.

Problem Statement: Understanding the differences in the usage patterns of Cyclistic bikes between casual riders and annual members to design a new marketing strategy aimed at converting casual riders into annual members.

Business Task: To analyze Cyclistic's historical bike trip data to understand how casual riders and annual members use Cyclistic bikes differently, and use these insights to design a new marketing strategy aimed at converting casual riders into annual members.

Key Stakeholders: Lily Moreno: The director of marketing who is responsible for developing campaigns and initiatives to promote the bike-sharing program. Cyclistic marketing analytics team: This team is responsible for collecting, analyzing, and reporting data that guides Cyclistic's marketing strategy. Cyclistic's executive team: This detail-oriented executive team will decide whether to approve the recommended marketing program.

Key Questions: The analysis will aim to answer the following key questions:

How do annual members and casual riders use Cyclistic bikes differently? What are the patterns or behaviors unique to casual riders that could potentially be addressed by a targeted marketing campaign? What are the trends over time for these two user groups? Are there seasonal patterns that could inform the timing of the marketing campaign?

Assumptions and Limitations: Assumptions: We are operating under the assumption that the provided data is accurate, up-to-date, and representative of the larger rider population. We also assume that the riders' behaviors are largely influenced by their status as casual riders or annual members, not by other unrecorded factors. Limitations: While the dataset offers rich information about ride details, it lacks direct demographic data of users such as age, gender, or socioeconomic status. Therefore, our analysis and conclusions are drawn based on the ride patterns and cannot account for the influence of these demographic factors.

Step 2 Prepare: Prepare the data by organinsing

Data Location and Organization: The data for this case study is provided by Motivate International Inc and is publicly available for download. This data represents the historical trip data of Cyclistic bikes. The data might be in a structured format like CSV or Excel, with each row representing a bike trip and columns indicating features such as trip duration, start time, end time, start station, end station, bike type, and user type (casual or member). *We will be using 2 files which contain data for 2019 and 2020

Licensing, Privacy, Security, and Accessibility: The data has been provided under a specific license, which permits its use for analysis but prohibits sharing the data as a standalone dataset. Privacy is preserved as no personally identifiable information is included in the data. Security will involve storing the downloaded data in a secure location, perhaps encrypted if needed. Accessibility refers to ensuring that the data and the subsequent analysis are available to all stakeholders involved in the project.

Key Tasks: Fixing Data Types and Removing Duplicates: Looking at the data, it seems like started_at and ended_at should be datetime objects, and start_station_id and end_station_id might be treated as categorical data (strings). Let's create a function that fixes the corresponding data types of these columns. While we're at it, let's also remove duplicate entries within the data.

Dropping Irrelevant Columns: Upon inspecting the corresponding columns I made an assumption that the every station name have a corresponding station ID aggregated for the columns, with this I decided to drop the columns with ids and keep their categorical equivalent. Given the potential inconsistencies in the start_station_id and end_station_id columns, it seems reasonable to drop these columns if your analysis is not heavily dependent on them. Since station names (start_station_name and end_station_name) are more understandable and user-friendly, we can keep these columns for our analysis.

Step 3 Process: Cleaning Up and adding additional data to prepare for analysis

Data Cleaning and Transformation: During this phase, we'll take steps to check for errors in our dataset and rectify them if necessary. This could involve dealing with missing or duplicate data, handling high cardinality variables, or even removing irrelevant data that doesn't contribute to our analysis.

Key Tasks: 1. Check the data for errors: Any errors or anomalies in the data could skew our analysis and lead to inaccurate insights. It's essential to identify and rectify these early on. 2. Transform the data: This could involve a variety of steps, such as correcting data types, dealing with missing data, or creating new features. 3. Document the cleaning process: Keeping a clear record of what steps were taken during the cleaning process can help ensure the reproducibility of our analysis and maintain the integrity of our data.

```r
# Inspect the new table that has been created
colnames(all_trips) #List of column names
```

```
## [1] "ride_id"            "started_at"         "ended_at"
## [4] "rideable_type"      "start_station_id"   "start_station_name"
## [7] "end_station_id"     "end_station_name"   "member_casual"
```

```r
nrow(all_trips) #How many rows are in data frame?
```

```
## [1] 791956
```

```r
dim(all_trips) #Dimensions of the data frame?
```

```
## [1] 791956      9
```

```r
head(all_trips) #See the first 6 rows of data frame. Also tail(all_trips)
```

```
##    ride_id          started_at            ended_at rideable_type
## 1 21742443 2019-01-01 00:04:37 2019-01-01 00:11:07          2167
## 2 21742444 2019-01-01 00:08:13 2019-01-01 00:15:34          4386
## 3 21742445 2019-01-01 00:13:23 2019-01-01 00:27:12          1524
## 4 21742446 2019-01-01 00:13:45 2019-01-01 00:43:28           252
## 5 21742447 2019-01-01 00:14:52 2019-01-01 00:20:56          1170
## 6 21742448 2019-01-01 00:15:33 2019-01-01 00:19:09          2437
##   start_station_id                  start_station_name end_station_id
## 1              199              Wabash Ave & Grand Ave             84
## 2               44              State St & Randolph St            624
## 3               15                Racine Ave & 18th St            644
## 4              123      California Ave & Milwaukee Ave            176
## 5              173 Mies van der Rohe Way & Chicago Ave             35
## 6               98           LaSalle St & Washington St             49
##               end_station_name member_casual
## 1      Milwaukee Ave & Grand Ave    Subscriber
## 2 Dearborn St & Van Buren St (*)    Subscriber
## 3  Western Ave & Fillmore St (*)    Subscriber
```

```
## 4              Clark St & Elm St    Subscriber
## 5        Streeter Dr & Grand Ave    Subscriber
## 6        Dearborn St & Monroe St    Subscriber
```

```r
str(all_trips) #See list of columns and data types (numeric, character, etc)
```

```
## 'data.frame':    791956 obs. of  9 variables:
##  $ ride_id           : chr  "21742443" "21742444" "21742445" "21742446" ...
##  $ started_at        : chr  "2019-01-01 00:04:37" "2019-01-01 00:08:13" "2019-01-01 00:13:23" "2019-0
##  $ ended_at          : chr  "2019-01-01 00:11:07" "2019-01-01 00:15:34" "2019-01-01 00:27:12" "2019-0
##  $ rideable_type     : chr  "2167" "4386" "1524" "252" ...
##  $ start_station_id  : int  199 44 15 123 173 98 98 211 150 268 ...
##  $ start_station_name: chr  "Wabash Ave & Grand Ave" "State St & Randolph St" "Racine Ave & 18th St"
##  $ end_station_id    : int  84 624 644 176 35 49 49 142 148 141 ...
##  $ end_station_name  : chr  "Milwaukee Ave & Grand Ave" "Dearborn St & Van Buren St (*)" "Western Ave
##  $ member_casual     : chr  "Subscriber" "Subscriber" "Subscriber" "Subscriber" ...
```

```r
summary(all_trips) #Statistical summary of data. Mainly for numerics
```

```
##    ride_id           started_at          ended_at          rideable_type     
##  Length:791956      Length:791956      Length:791956      Length:791956     
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                            
##                                                                            
##                                                                            
##                                                                            
##  start_station_id start_station_name end_station_id  end_station_name  
##  Min.   :  2.0    Length:791956      Min.   :  2.0   Length:791956     
##  1st Qu.: 77.0    Class :character   1st Qu.: 77.0   Class :character  
##  Median :174.0    Mode  :character   Median :174.0   Mode  :character  
##  Mean   :204.4                       Mean   :204.4                     
##  3rd Qu.:291.0                       3rd Qu.:291.0                     
##  Max.   :675.0                       Max.   :675.0                     
##                                      NA's   :1                         
##  member_casual     
##  Length:791956     
##  Class :character  
##  Mode  :character  
##                    
##                    
##                    
##                    
```

After inspection We find out there are some problems with our data:

(1) In the "member_casual" column, there are two names for members ("member" and consolidate that from four to two labels.
(2) The data can only be aggregated at the ride-level, which is too granular. We will want to add some additional columns of data – such as day, month, year – that provide additional opportunities to aggregate the data.

(3) We will want to add a calculated field for the length of the ride since the 2020Q1 data did not have the “tripduration” column. We will add “ride_length” to the entire data frame for consistency.
(4) There are some rides where trip-duration shows up as negative, including several hundred rides where Divvy took bikes out of circulation for Quality Control reasons. We will want to delete these rides.

We will solve these problems one by one:

1. In the “member_casual” column, replace “Subscriber” with “member” and “Customer” with “casual” Before 2020, Divvy used different labels for these two types of riders … we will want to make our data frame consistent with their current nomenclature.

```
# Begin by seeing how many observations fall under each usertype
table(all_trips$member_casual)
```

```
##
##     casual   Customer     member Subscriber
##      48480      23163     378407     341906
```

```
# Reassign to the desired values (we will go with the current 2020 labels)
all_trips <- all_trips %>%
  mutate(member_casual = recode(member_casual
                                ,"Subscriber" = "member"
                                ,"Customer" = "casual"))
```

2. We will add coloumns like day, month and year to our dataframe.

```
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")
```

3. We will add coloum for Ride length (in seconds) to maintain consistency

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

4. We will remove the invalid data from the dataframe, so that it does not affect our analysis.

```
is.factor(all_trips$ride_length)
```

```
## [1] FALSE
```

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

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

```
# Remove "bad" data
all_trips_v2 <- all_trips[!(all_trips$start_station_name == "HQ QR" | all_trips$ride_length<0),]
```

Now our data is clean, consistent and ready for analysis.

Step 4 Analyse: Conduct descreptive analysis of the data

Data Organization and Formatting: In this phase, it's essential to organize the data in a manner that makes it accessible and convenient for performing various analyses. This involves appropriately formatting data, ensuring consistency across all variables, and creating a unified view of the data.

Descriptive Analysis: We'll conduct descriptive analysis, which helps understand the central tendencies and distribution of the data. This provides an overview of the patterns and trends within the data and can reveal surprising insights.

Identifying Trends and Relationships: In this step, we'll identify key trends, patterns, and relationships between different variables in our data. This may involve looking at correlations between variables, analyzing patterns over time, or identifying factors that influence a particular outcome.

Key Tasks:

1. Aggregate the data: Aggregating the data in different ways can reveal new insights and make the data easier to work with.
2. Organize and format the data: Proper organization and formatting of the data is essential for effective analysis.
3. Perform calculations: Calculations can help us understand the data better and can form the basis for our insights.
4. Identify trends and relationships: Identifying key trends and relationships in the data is a critical part of the analysis process.

First we calculate the mean, median, max, and min of ride lengths.

```
# Descriptive analysis on ride_length (all figures in seconds)
mean(all_trips_v2$ride_length) #straight average (total ride length / rides)
```

```
## [1] 1189.459
```

```
median(all_trips_v2$ride_length) #midpoint number in the ascending array of ride lengths
```

```
## [1] 539
```

```
max(all_trips_v2$ride_length) #longest ride
```

```
## [1] 10632022
```

```
min(all_trips_v2$ride_length) #shortest ride
```

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

```r
# You can condense the four lines above to one line using summary() on the specific attribute
summary(all_trips_v2$ride_length)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        1      331      539     1189      912 10632022
```

Now we use aggregate function to compare between member and casual riders

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = mean)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                5372.7839
## 2                     member                 795.2523
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = median)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                     1393
## 2                     member                      508
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = max)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                 10632022
## 2                     member                  6096428
```

```r
aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual, FUN = min)
```

```
##   all_trips_v2$member_casual all_trips_v2$ride_length
## 1                     casual                        2
## 2                     member                        1
```

To compare this on weekday basis we can add the following code,

```r
all_trips_v2$day_of_week <- ordered(all_trips_v2$day_of_week, levels=c("Sunday", "Monday", "Tuesday", "W

aggregate(all_trips_v2$ride_length ~ all_trips_v2$member_casual + all_trips_v2$day_of_week,
          FUN = mean)
```

```
##    all_trips_v2$member_casual all_trips_v2$day_of_week all_trips_v2$ride_length
## 1                      casual                   Sunday                5061.3044
## 2                      member                   Sunday                 972.9383
## 3                      casual                   Monday                4752.0504
## 4                      member                   Monday                 822.3112
## 5                      casual                  Tuesday                4561.8039
## 6                      member                  Tuesday                 769.4416
## 7                      casual                Wednesday                4480.3724
## 8                      member                Wednesday                 711.9838
## 9                      casual                 Thursday                8451.6669
```

```
## 10                        member              Thursday              707.2093
## 11                        casual                Friday             6090.7373
## 12                        member                Friday              796.7338
## 13                        casual              Saturday             4950.7708
## 14                        member              Saturday              974.0730
```

```
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>% #creates weekday field using wday()

group_by(member_casual, weekday) %>% #groups by usertype and weekday
  summarise(number_of_rides = n() #calculates the number of rides and average duration
            ,average_duration = mean(ride_length)) %>% # calculates the average duration

arrange(member_casual, weekday) # sorts
```

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

```
## # A tibble: 14 x 4
## # Groups:   member_casual [2]
##    member_casual weekday number_of_rides average_duration
##    <chr>         <ord>             <int>            <dbl>
##  1 casual        Sun               18652            5061.
##  2 casual        Mon                5591            4752.
##  3 casual        Tue                7311            4562.
##  4 casual        Wed                7690            4480.
##  5 casual        Thu                7147            8452.
##  6 casual        Fri                8013            6091.
##  7 casual        Sat               13473            4951.
##  8 member        Sun               60197             973.
##  9 member        Mon              110430             822.
## 10 member        Tue              127974             769.
## 11 member        Wed              121902             712.
## 12 member        Thu              125228             707.
## 13 member        Fri              115168             797.
## 14 member        Sat               59413             974.
```

Step 5 Share: Create plots to share the insights

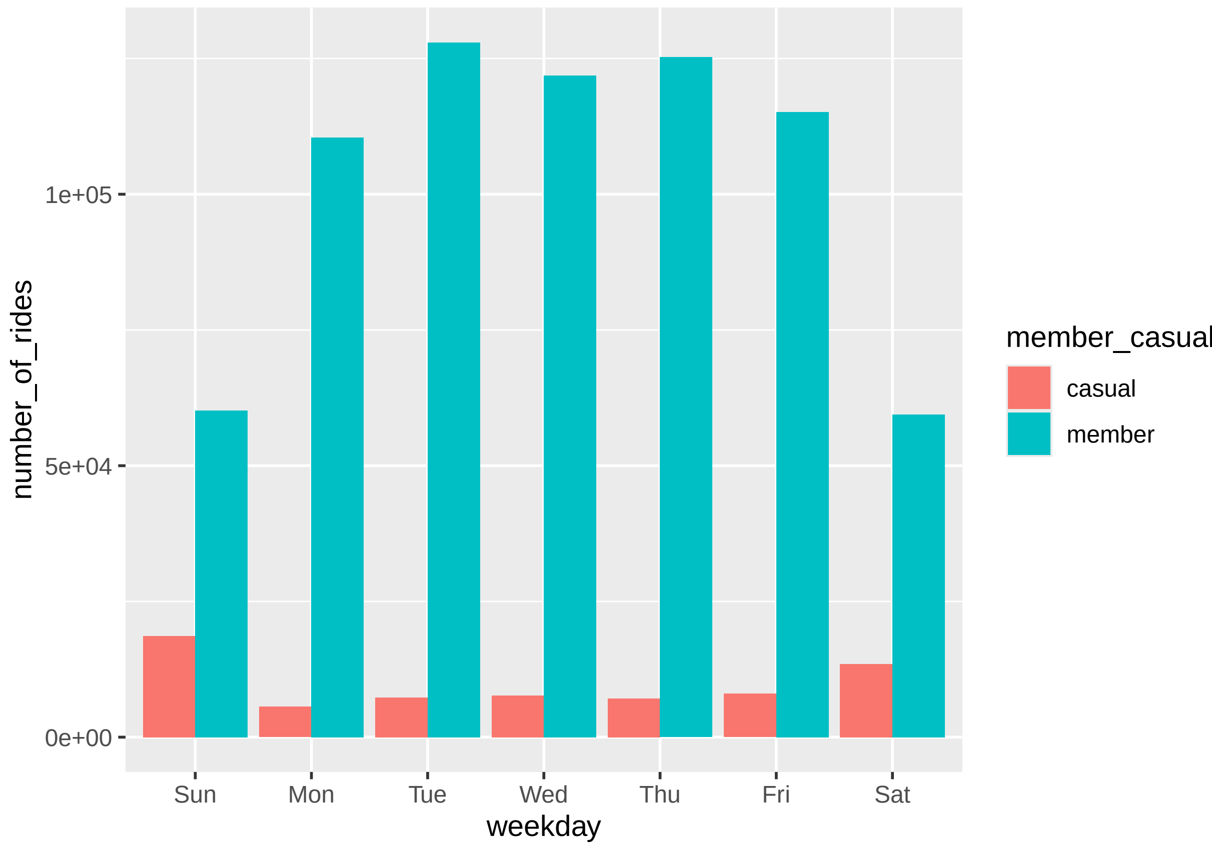
It is important to create figures to visualise, it helps us share our finding with the stakeholders and co-workers.

Below is the plot for Number of rides take by diffrent riders during the weekdays,

```
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%

  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x = weekday, y = number_of_rides, fill = member_casual)) +
  geom_col(position = "dodge")
```

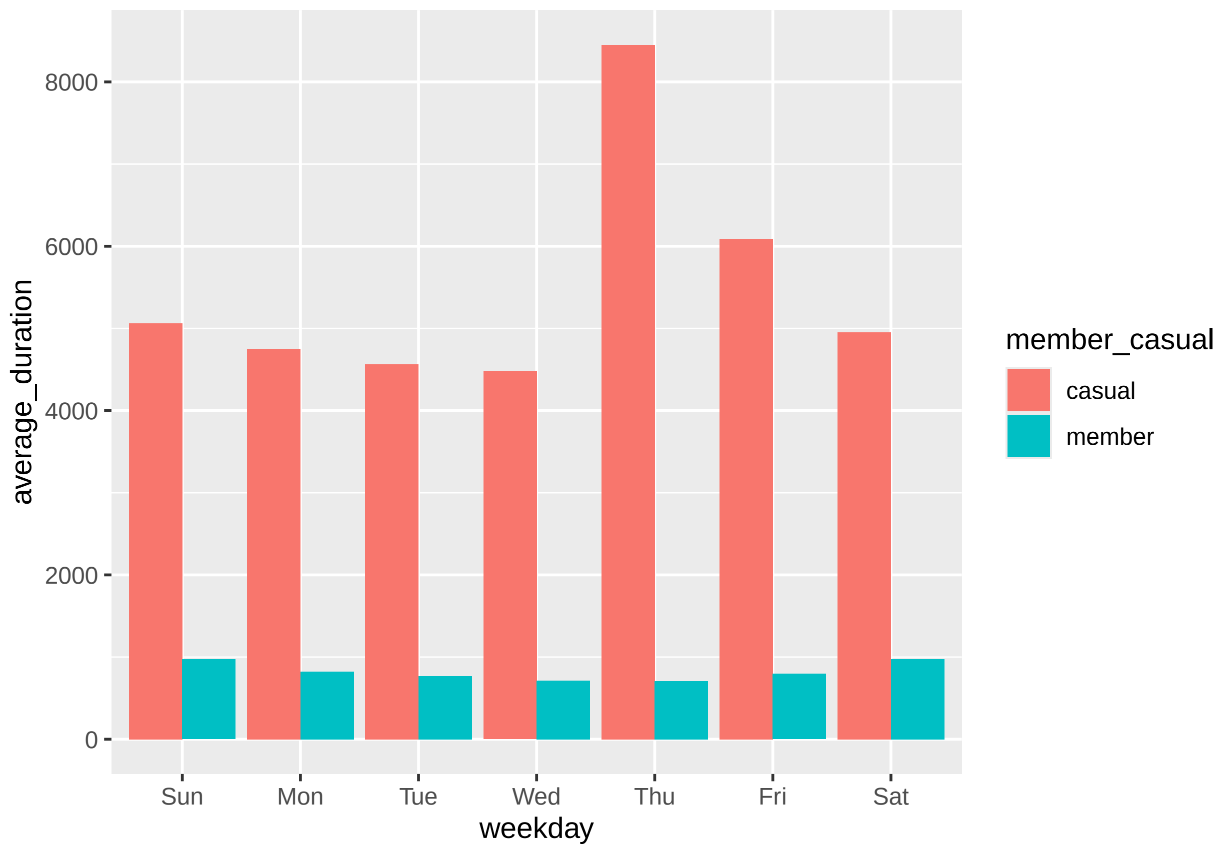
```
## 'summarise()' has grouped output by 'member_casual'. You can override using the
## '.groups' argument.
```



Also ,add a plot for average duration(in seconds) of the trip,

```r
all_trips_v2 %>%
  mutate(weekday = wday(started_at, label = TRUE)) %>%
  group_by(member_casual, weekday) %>%
  summarise(number_of_rides = n()
            ,average_duration = mean(ride_length)) %>%
  arrange(member_casual, weekday) %>%
  ggplot(aes(x = weekday, y = average_duration, fill = member_casual)) +
  geom_col(position = "dodge")
```

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


8000
6000
4000
2000
0
average_duration
Sun
Mon
Tue
Wed
Thu
Fri
Sat
weekday
member_casual
casual
member