# Exploring Toronto Bike Share Ridership

**Haiyue Yang, January 2020**

## Introduction

In recent years, Bike Sharing has become increasingly popular aroung the world. It is a both convenient and ecological way to explore the city. Thus, I was attracted be the Bike Share Ridership in Toronto and decided to explore the ridership using Python.

My exploration focused on the difference of riderships between casual users and members and the inflence of weather on the ridership.

By understanding these traits of the bikeshare data, can we know how to develop the bikeshare program in the future.

## Outline

- **Data Wrangling**:
  - About the data: The dataframe and variables used for analysis.
  - Data Cleaning and Combination:
    - Bikeshare Ridership Data:
      - Import bikeshare ridership data from files
      - Standardize ridership data in each quarter
      - Merge the data into the dataframe `df`
      - Save the data as a csv file `ridership_data.csv`
    - Station Data
      - Import station data from the website
      - Cleanup and extract useful data
      - Save the sation data as a csv file `station_data.csv`
      - Combine `station_data` with `ridership_data`
      - Cleanup the combined dataframe
      - Save the combined data as a csv file `bikeshare_data.csv`
    - Weather Data
      - Import weather data from the website according to the terms of use of the data
      - Combine `ridership_data` with the weather data
      - Cleanup the combined dataframe
      - Save the combined data as a csv file `weather_data.csv`
- **Analysis and Visualization**

- Relationship between `duration` and `distance`
  - Analysis
    - Two definitions of trip length
    - Scatter plots of `trip_duration_seconds` and `distance` of each trip
    - Scatter plots of the average `distance` grouped by `trip_duration_seconds`
  - Conclusion and consideration
- Difference in ridership between `casual users` and `members`
  - Analysis
    - `describe()` function
    - Boxplots of trip length of `casual users` and `members` in both definition
    - Scatter plots of `trip_duration_seconds` and `distance` of trips for two types of users
    - Relationship plots between date and trip length of both kinds of users under both definition
    - Scatter plots of the average `distance` grouped by `trip_duratoin_seconds` for each type of users
  - Conclusion and consideration
- Relationship between `weather` and `trip length`
  - Analysis
    - Mean temperature and trip length
      - Scatter plots with regression line of `mean_temp` and trip length under both definition
    - Total precipitation and trip length
      - `describe()` function
      - Boxplots of the trip lengths on both days with precipitation or not under both definition
      - Histogram of the distribution of distance when there is precipitation in a day or not.
  - Conclusion and consideration
- Instead of writing conclusions and considerations in general, I wrote them related to specific research questions.

# Data Wrangling

## 1. About the Data

My research is mainly based on the datasets of the bikeshare ridership in 2017 and the second half of 2016, provided on Open Data (https://open.toronto.ca/dataset/bike-share-toronto-ridership-data (https://open.toronto.ca/dataset/bike-share-toronto-ridership-data/)), combined with weather information on Historical Climate Data website (https://climate.weather.gc.ca/) and station information on https://open.toronto.ca/dataset/bike-share-toronto/ (https://open.toronto.ca/dataset/bike-share-toronto/).

- **ridership_data.csv**
  - `trip_id` : the unique identifier for each trip
  - `trip_start_time` : the time when the trip started
  - `trip_stop_time` : the time when the trip stopped
  - `trip_duration_seconds` : the duration of trip measured in seconds
  - `from_station_name` : the name of the station where the trip started
  - `to_station_name` : the name of the station where the trip ended
  - `user_type` : the user has the membership or is a casual user
  - `from_station_id` : the unique identifier of the station where the trip started
  - `to_station_id` : the unique identifier of the station where the trip ended
- **station_data.csv**
  - `station_id` : the unique identifier of the station
  - `name` : the name of the station
  - `lat` : the latitude of the station
  - `lon` : the longitude of the station
- **bikeshare_data.csv**
  - `user_type` : the user has the membership or is a casual user
  - `trip_duration_seconds` : the duration of trip measured in seconds
  - `distance` : the distance in straight line of the trip measured in meters
  - `date` : the date of the trip
- **weather_data.csv**
  - `mean_temp` : the mean temperature of the day
  - `total_precip` : the total precipitation of the day
  - `date` : the date of the day
  - `duration` : the mean duration of trips in the day measured in seconds
  - `distance` : the mean distance of trips in they day measure in meters

## 2. Data Cleaning and Combination

- **Bikeshare Ridership Data**

The bikeshare Ridership Data contains 6 dataframes separated by quarter. The data were retrived from Open Data (https://open.toronto.ca/dataset/bike-share-toronto-ridership-data (https://open.toronto.ca/dataset/bike-share-toronto-ridership-data/)).

During my exploration, I noticed that the data in each dataframe is not standarized. Therefore, I standarized the data fisrt.

**Firstly**, I dropped out rows with null values, made sure all data in `trip_start_time` and `trip_stop_time` in Q3 of 2016 are of type datetime and added a column representing the quarter of the data.

**Secondly**, I noticed that there were different formats in `trip_start_time` and `trip_stop_time` in Q4 of 2016. For example, look at the first and last 5 rows of the dataframe: the `trip_start_time` and `trip_stop_time` in the head is of type datetime, but the month and the day were placed in the wrong order; the `trip_start_time` and `trip_stop_time` in the tail is of type string and in the format '%d/%m/%Y %H:%M'.

Out[13]:

| | trip_id | trip_start_time | trip_stop_time | trip_duration_seconds | from_station_name | to_s |
|---|---|---|---|---|---|---|
| **0** | 462305 | 2016-01-10 00:00:00 | 2016-01-10 00:07:00 | 394 | Queens Quay W / Dan Leckie Way | Fo |
| **1** | 462306 | 2016-01-10 00:00:00 | 2016-01-10 00:09:00 | 533 | Sherbourne St / Wellesley St | |
| **2** | 462307 | 2016-01-10 00:00:00 | 2016-01-10 00:07:00 | 383 | Queens Quay W / Dan Leckie Way | Fo |
| **3** | 462308 | 2016-01-10 00:01:00 | 2016-01-10 00:27:00 | 1557 | Cherry St / Distillery Ln | Fo |
| **4** | 462309 | 2016-01-10 00:01:00 | 2016-01-10 00:27:00 | 1547 | Cherry St / Distillery Ln | Fo |
| **...** | ... | ... | ... | ... | ... | |
| **217564** | 712377 | 31/12/2016 23:26 | 31/12/2016 23:39 | 824 | Union Station | |
| **217565** | 712378 | 31/12/2016 23:26 | 31/12/2016 23:34 | 478 | Bay St / College St (East Side) | |
| **217566** | 712379 | 31/12/2016 23:33 | 31/12/2016 23:38 | 271 | Temperance St / Yonge St | |
| **217567** | 712380 | 31/12/2016 23:37 | 31/12/2016 23:58 | 1253 | Christie St / Benson Ave (Wychwood Barns) | We Yo |
| **217568** | 712381 | 31/12/2016 23:40 | 31/12/2016 23:48 | 478 | Ted Rogers Way / Bloor St E | S |

217569 rows × 7 columns

Thus, I standarized these two columns into datetime.datetime in the format '%Y-%m-%d %H:%M:%S', dropped out rows with null values, and added a column representing the quarter of the data.

**After that**, I combined the two dataframe in year 2016 and added a column representing the year.

Further more, I noticed that the data in year 2016 does not contain `from_station_id` and `to_statoin_id` , so I added this two columns with NaN value for use in the future.

Out[23]:

| | trip_id | trip_start_time | trip_stop_time | trip_duration_seconds | from_station_name | to_station_ |
|---|---|---|---|---|---|---|
| **0** | 53279 | 2016-07-09 01:03:00 | 2016-07-09 01:15:00 | 714 | Dundas St E / Regent Park Blvd | Danforth Ellerbe |
| **1** | 53394 | 2016-07-09 02:15:00 | 2016-07-09 02:22:00 | 417 | Riverdale Park North (Broadview Ave) | Dundas Regent Parl |
| **2** | 58314 | 2016-07-10 17:04:00 | 2016-07-10 17:36:00 | 1904 | Dundas St E / Regent Park Blvd | Queen Clos |
| **3** | 60784 | 2016-07-11 01:45:00 | 2016-07-11 01:58:00 | 784 | Union Station | Dundas Regent Parl |
| **4** | 93164 | 2016-07-18 13:35:00 | 2016-07-18 13:42:00 | 443 | Front St W / Blue Jays Way | Front St / St (Hocke of |

**Next**, I continued to clean up the data in 2017. Similar to the data in 2016, I dropped out rows containing null values, changed different formats in `trip_start_time` and `trip_stop_time` into datetime.datetime in the format '%Y-%m-%d %H:%M:%S' and added a column representing the quarter for each dataframe.

**Then**, I merged these four dataframes and added a column representing the year.

Also, I noticed that in Q3 and Q4 of year 2017, the columns of `from_station_id` and `to_statoin_id` is still missing, but I leaved the value of them NaN for future adjustment.

Out[32]:

| | trip_id | trip_start_time | trip_stop_time | trip_duration_seconds | from_station_id | from_station_ |
|---|---|---|---|---|---|---|
| **0** | 712382 | 2017-01-01 00:00:00 | 2017-01-01 00:03:00 | 223 | 7051.0 | Wellesley Yonge St Gr |
| **1** | 712383 | 2017-01-01 00:00:00 | 2017-01-01 00:05:00 | 279 | 7143.0 | Kendal Bernar |
| **2** | 712384 | 2017-01-01 00:05:00 | 2017-01-01 00:29:00 | 1394 | 7113.0 | Parliamer Aberdee |
| **3** | 712385 | 2017-01-01 00:07:00 | 2017-01-01 00:21:00 | 826 | 7077.0 | College Park |
| **4** | 712386 | 2017-01-01 00:08:00 | 2017-01-01 00:12:00 | 279 | 7079.0 | McGill St / C |

**After that**, we got the combined data from year 2017 and year 2016 (partial). We concatenated them and get a dataframe named `df` containing the bikeshare ridership data in whole.

Also, considering the convenience of future use, I again ensured that all data in `trip_start_time` and `trip_stop_time` are of type datetime and in the format '%Y-%m/%d %H:%M:%S', and all data in `from_station_id` and `to_station_id` are saved as float.

Out[35]:

| | trip_id | trip_start_time | trip_stop_time | trip_duration_seconds | from_station_name | to |
|---|---|---|---|---|---|---|
| **0** | 53279 | 2016-07-09 01:03:00 | 2016-07-09 01:15:00 | 714 | Dundas St E / Regent Park Blvd | |
| **1** | 53394 | 2016-07-09 02:15:00 | 2016-07-09 02:22:00 | 417 | Riverdale Park North (Broadview Ave) | Re |
| **2** | 58314 | 2016-07-10 17:04:00 | 2016-07-10 17:36:00 | 1904 | Dundas St E / Regent Park Blvd | |
| **3** | 60784 | 2016-07-11 01:45:00 | 2016-07-11 01:58:00 | 784 | Union Station | Re |
| **4** | 93164 | 2016-07-18 13:35:00 | 2016-07-18 13:42:00 | 443 | Front St W / Blue Jays Way | F |
| **...** | ... | ... | ... | ... | ... | ... |
| **1492363** | 2383642 | 2017-12-31 23:46:27 | 2017-12-31 23:46:53 | 26 | Bloor St / Brunswick Ave | |
| **1492364** | 2383643 | 2017-12-31 23:47:13 | 2018-01-01 00:11:40 | 1467 | Bloor St / Brunswick Ave | (C |
| **1492365** | 2383644 | 2017-12-31 23:47:40 | 2017-12-31 23:57:49 | 609 | Kendal Ave / Spadina Rd | |
| **1492366** | 2383645 | 2017-12-31 23:49:08 | 2017-12-31 23:49:34 | 26 | Phoebe St / Spadina Ave | |
| **1492367** | 2383646 | 2017-12-31 23:49:41 | 2017-12-31 23:57:41 | 480 | Phoebe St / Spadina Ave | |

2077890 rows × 11 columns

**Finally**, I save this dataframe `df` as a csv file named `ridership_data.csv` for reference.

- **station data**

**First**, I imported the station data from Open Data (https://open.toronto.ca/dataset/bike-share-toronto/) in order to find the latitude and longitude of each station.

These stations are not matched with station IDs.

```
WARN: Victoria St / Gould St (Ryerson University) station could not
be matched to an existing station
WARN: Bloor St / Brunswick Ave station could not be matched to an ex
isting station
WARN: Bay St / Bloor St W station could not be matched to an existin
g station
WARN: Bremner Blvd / Spadina Ave station could not be matched to an
existing station
WARN: Dockside Dr / Queens Quay E (Sugar Beach) station could not be
matched to an existing station
WARN: Temperance St / Yonge St station could not be matched to an ex
isting station
WARN: Ontario Place Blvd / Remembrance Dr station could not be match
ed to an existing station
WARN: Landsdowne Subway Green P station could not be matched to an e
xisting station
WARN: Bathurst St / Queens Quay W station could not be matched to an
existing station
WARN: Bloor GO / UP Station/ Rail Path station could not be matched
to an existing station
WARN: Stephenson Ave / Main St station could not be matched to an ex
isting station
WARN: Woodbine Subway Green P (Cedarvale Ave) station could not be m
atched to an existing station
WARN: Margueretta St / College St W station could not be matched to
an existing station
WARN: Base Station station could not be matched to an existing stati
on
WARN: Michael Sweet Ave / St. Patrick St station could not be matche
d to an existing station
WARN: Margueretta St / College St station could not be matched to an
existing station
WARN: Lansdowne Subway Green P station could not be matched to an ex
isting station
WARN: Roxton Rd / College St station could not be matched to an exis
ting station
WARN: Lake Shore Blvd W / Ontario Dr(Ontario Place) station could no
t be matched to an existing station
WARN: Fringe Next Stage - 7219 station could not be matched to an ex
isting station
```

**Then**, we got a dataframe for the `name`, `station_id`, `lat`, and `lon` for each station and saved it as a csv file named `station_data`.

Out[197]:

| | station_id | name | lat | lon |
|---|---|---|---|---|
| **0** | 7051.0 | Wellesley St E / Yonge St Green P | 43.665060 | -79.383570 |
| **2** | 7051.0 | Wellesley St E / Yonge St (Green P) | 43.665060 | -79.383570 |
| **3** | 7143.0 | Kendal Ave / Bernard Ave | 43.671513 | -79.408317 |
| **5** | 7113.0 | Parliament St / Aberdeen Ave | 43.665278 | -79.368333 |
| **7** | 7077.0 | College Park South | 43.659777 | -79.382767 |

**Next**, I merged the bikeshare ridership information with the station infomation and calculated the `distance` (trip distance) from the `lat` (latitude) and `lon` (longitude) of the stations.

Since the stations are all in Toronto, when calculating the distance, we ignored the angle and simplifed each degree of longitude or latitude to 111 km, and the results are in meters.

**After that**, I sorted out the `date` of each trip, and selected out the columns `user_type` , `trip_duration_seconds` , `distance` , and `date` for future use.

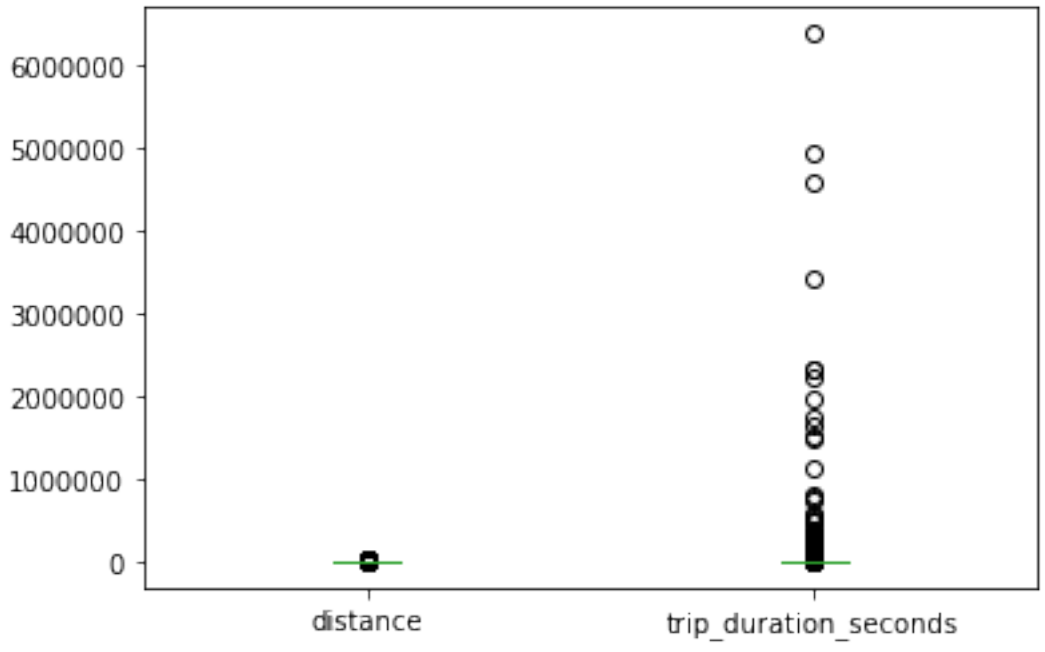
**Then**, we started to clean up the data.

I drew the boxplots for `distance` and `trip_duration_seconds` and used the `describe()` function on these two colums to see a simple profile of the data.

I discovered that there were obvious outliers in these two datasets.

Out[199]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x140f1fbb0>
```




Out[200]:

| | distance | trip_duration_seconds |
|---|---|---|
| **count** | 2.088646e+06 | 2.088646e+06 |
| **mean** | 2.058137e+03 | 9.852265e+02 |
| **std** | 1.442705e+03 | 8.276238e+03 |
| **min** | 0.000000e+00 | 1.000000e+00 |
| **25%** | 1.108542e+03 | 4.180000e+02 |
| **50%** | 1.763431e+03 | 6.550000e+02 |
| **75%** | 2.697429e+03 | 1.013000e+03 |
| **max** | 2.058279e+04 | 6.382030e+06 |

A common definition for outliers are the values 1.5*IQR larger than the 75% quantile or 1.5*IQR less than the 25% quantile.

To avoid the interference of extreme values in analysis, I removed all the outliers of `distance` and `trip_duration_seconds` under this definition.

Also, the trips less than 1 minute is considered to be invalid, so I removed the rows with `trip_duration_seconds` less than 60.

Out[204]:

| | user_type | trip_duration_seconds | distance | date |
|---|---|---|---|---|
| **311884** | Member | 60 | 304.438682 | 2017-12-04 |
| **1321925** | Casual | 60 | 545.910161 | 2017-06-24 |
| **1321926** | Casual | 60 | 545.910161 | 2017-06-24 |
| **1677842** | Member | 60 | 293.349145 | 2017-10-13 |
| **92489** | Member | 60 | 284.000422 | 2017-04-05 |
| **...** | ... | ... | ... | ... |
| **40178** | Casual | 1905 | 2917.085231 | 2017-08-10 |
| **1334895** | Member | 1905 | 2459.398093 | 2017-08-18 |
| **1232396** | Casual | 1905 | 397.869107 | 2017-09-27 |
| **1396718** | Casual | 1905 | 4330.255093 | 2017-04-10 |
| **67846** | Casual | 1905 | 3431.510926 | 2017-07-09 |

1841201 rows × 4 columns

**Finally**, I got the dataframe `data` used for my research and saved it as a csv file named `bikeshare_data.csv` for reference.

- **weather data**

**Firstly**, I scraped the wheather data from the Government of Canada Historical Climate Data website (https://climate.weather.gc.ca/climate_data/daily_data_e.html?StationID=51459&timeframe=2&StartYear=1840&EndYear=2019&Day=22&Year=2017&Month=1#).

According to the terms of use of the data:

> - We can reproduce the materials on Historical Climate Data website (https://climate.weather.gc.ca/) in whole or in part for non-commercial purposes, and in any format, without charge or further permission, provided we do the following:
>   - ensuring the accuracy of the material reproduced
>   - indicate both the complete title of the materials reproduced, as well as the author (where available)
>   - indicate that the reproduction is a copy of the a version avaible at [URL where original document is available]
> - Unless otherwise specified, we may not reproduce materials on Historical Climate Data website (https://climate.weather.gc.ca/), in whole or in part, for the purposes of commercial redistribution without prior written permission from the copyright administrator.

There are 18 dataframes from the website for different months and I had to scraped them separately. I wrote a function `month` which takes the url and the number of days in the month, and returns the dataframe for each month to scrap.

For convenience of later use, I only scraped the columns `Mean Temp Definition °C` and `Total Precip Definitinomm` for each month.

After that, I concatenated the 18 dataframes and got the dataframe `weather` .

Also, for later use, I added a column representing the `date` .

Out[62]:

| | Mean Temp Definition°C | Total Precip Definitionmm | date |
|---|---|---|---|
| **0** | 18.3 | 5.6 | 2016-07-01 |
| **1** | 19.4 | 0.0 | 2016-07-02 |
| **2** | 21.0 | 0.0 | 2016-07-03 |
| **3** | 21.8 | 0.0 | 2016-07-04 |
| **4** | 26.1 | 0.0 | 2016-07-05 |

**Secondly**, I grouped the ridership data by `date`, calculated the mean `trip_duration_seconds` and `distance`, and got a new dataframe giving the mean `trip_duration_secodns` and `distance` of each `date`.

Out[98]:

| | date | trip_duration_seconds | distance |
|---|---|---|---|
| **0** | 2016-07-01 | 755.201842 | 1813.031033 |
| **1** | 2016-07-02 | 867.896169 | 1899.371816 |
| **2** | 2016-07-03 | 862.735355 | 1879.030543 |
| **3** | 2016-07-04 | 723.247626 | 1883.836123 |
| **4** | 2016-07-05 | 714.957033 | 1905.608854 |

**Then**, I merged the new dataframe with the weather information, removed some unprecise data such as EE and MM, and got the dataframe for analysis named `weather_data`.

The `weather_data` dataframe contains the `Mean Temp Definition °C` (mean temperature measured in °C), the `Total Precip Definitionmm` (total precipitation measure in mm), `date`, the mean `trip_duration_seconds`, and the mean `distance` travelled of each day.

Out[101]:

| | Mean Temp Definition°C | Total Precip Definitionmm | date | trip_duration_seconds | distance |
|---|---|---|---|---|---|
| **0** | 18.3 | 5.6 | 2016-07-01 | 755.201842 | 1813.031033 |
| **1** | 19.4 | 0.0 | 2016-07-02 | 867.896169 | 1899.371816 |
| **2** | 21.0 | 0.0 | 2016-07-03 | 862.735355 | 1879.030543 |
| **3** | 21.8 | 0.0 | 2016-07-04 | 723.247626 | 1883.836123 |
| **4** | 26.1 | 0.0 | 2016-07-05 | 714.957033 | 1905.608854 |

**Also**, in order to avoid the influence of extreme values, I removed outliers of the `Mean Temp Definition °C` (mean temperature measured in °C), the `Total Precip Definitionmm` . The ourliers are still defined as the values 1.5*IQR larger than the 75% quantile or 1.5*IQR less than the 25% quantile.

Also, considering the convenience of use, I rename some of the columns:

- `Mean Temp Definition°C` is named `mean_temp`
- `Total Precip Definitionmm` is named `total_precip`
- `trip_duration_seconds` is named `duration`

Now, the dataframe `weather_data` is ready for analysis.

Out[104]:

| | mean_temp | total_precip | date | duration | distance |
|---|---|---|---|---|---|
| **439** | -18.6 | 0.0 | 2017-12-31 | 528.588068 | 1503.407902 |
| **380** | 15.6 | 0.0 | 2017-10-22 | 760.956976 | 2006.176339 |
| **379** | 15.4 | 0.0 | 2017-10-21 | 760.617095 | 2010.001245 |
| **212** | 6.5 | 0.0 | 2017-04-17 | 677.873105 | 1979.282584 |
| **213** | 4.7 | 0.0 | 2017-04-18 | 635.482430 | 1965.247389 |
| **...** | ... | ... | ... | ... | ... |
| **220** | 17.2 | 5.4 | 2017-04-27 | 673.977008 | 1974.296375 |
| **301** | 22.5 | 5.4 | 2017-07-27 | 718.468289 | 1988.454408 |
| **210** | 11.8 | 5.6 | 2017-04-15 | 728.440979 | 1848.022912 |
| **93** | 8.1 | 5.6 | 2016-10-21 | 598.953595 | 1829.222814 |
| **0** | 18.3 | 5.6 | 2016-07-01 | 755.201842 | 1813.031033 |

376 rows × 5 columns

**Finally**, I saved the `weather_data` dataframe as a csv file for later use.

**Now, all data are cleaned up and are ready to use.**

# Analysis and Visualization

## 1. Relationship between Duration and Distance

We have two definitions of trip length:

- duration: `trip_duration_seconds`

  ```
  The time the rider takes to ride from the starting station to the en
  ding station.
  ```

- distance: `distance`

  ```
  The distance between the starting station and the ending station.
  ```

We are interested in the relationship between these two definition.

**First**, I used the `describe()` function to see the simple profile of these two variables.

Out[67]:

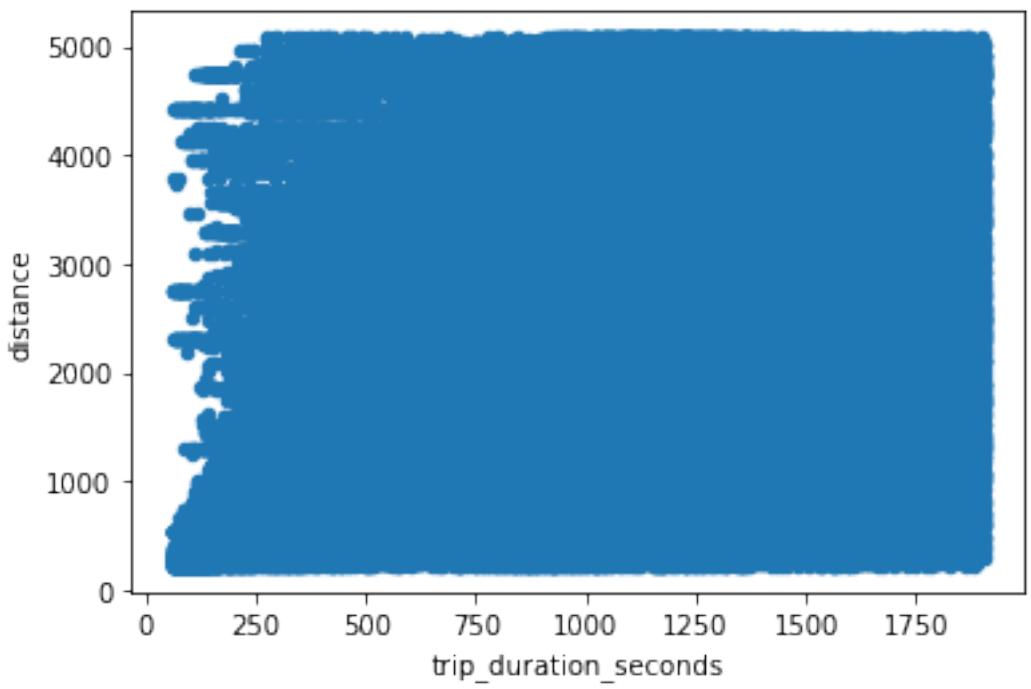
| | trip_duration_seconds | distance |
|---|---|---|
| **count** | 1.841201e+06 | 1.841201e+06 |
| **mean** | 6.900432e+02 | 1.966866e+03 |
| **std** | 3.629831e+02 | 1.039912e+03 |
| **min** | 6.000000e+01 | 2.168243e+02 |
| **25%** | 4.130000e+02 | 1.167366e+03 |
| **50%** | 6.200000e+02 | 1.766482e+03 |
| **75%** | 9.040000e+02 | 2.590728e+03 |
| **max** | 1.905000e+03 | 5.079660e+03 |

However, I was not able to identify any relationship simply from the dataframe above.

**Secondly**, I draw a scatter plot of these two variables using the builtin `plot` function of pandas.

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12461b790>
```




However, because the sample size is extremely large, numerous dots dominated the whole graph. I was still not able to discover any relationship from the graph.

**Thirdly**, I decided to group the data by one of the two variables and analyse in groups.

The variable `trip_duration_seconds` are all integers, while the variable `distance` are floats. I chose to group by data by `trip_duration_seconds` and calculated the mean of `distance` . Thus, I got a dataframe named `dis_mean` with two columns: the column `distance` represents the average distance a rider travels with the `trip_duration_seconds` .

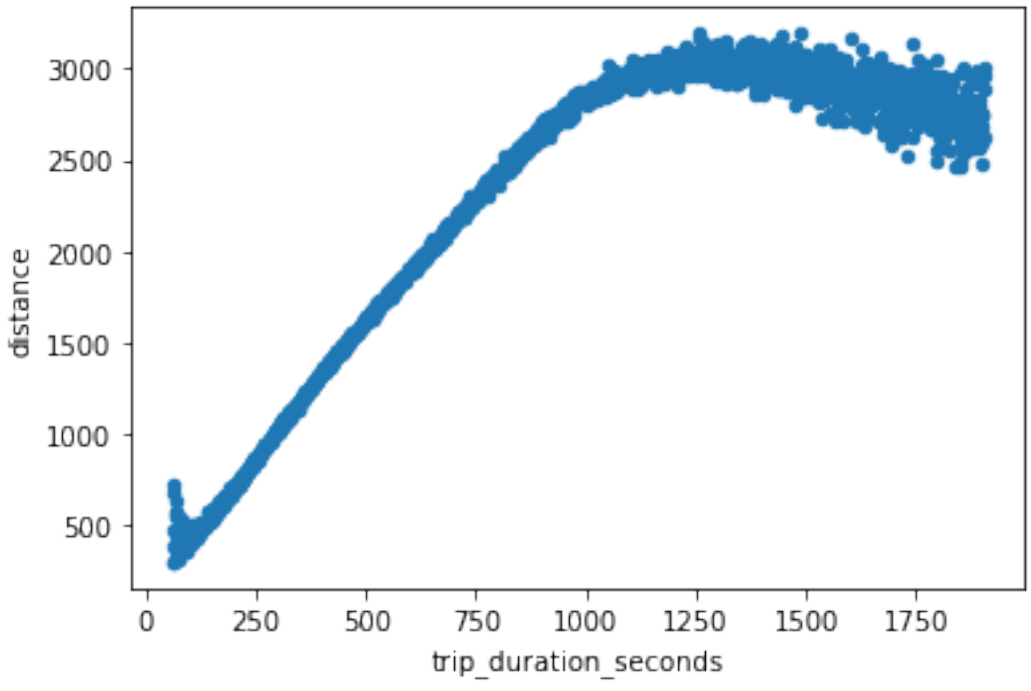
Out[72]:

| | trip_duration_seconds | distance |
|---|---|---|
| **0** | 60 | 380.998968 |
| **1** | 61 | 303.124118 |
| **2** | 62 | 684.669950 |
| **3** | 63 | 297.508107 |
| **4** | 64 | 466.710895 |

**After that**, I draw a scatter plot of the mean `distance` and `trip_duration_seconds` , with the new dataframe `dis_mean` .

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x122e4e340>
```




**Finally**, from this graph, it is obvious that the mean `distance` traveled is positively related to the `trip_duration_seconds` . The relationship is linear and extremely strong, when the `trip_duration_seconds` is between 100 to 1000 seconds. But the variance is larger when the `trip_duration_seconds` is shorter than 100 seconds or longer than 1000 seconds.

## conclusions and considerations:

- We have two definitions for trip length: `distance` (the distance of the trip) and `trip_duration_seonds` (the time duration of the trip).
- The `distance` , in average, is positively related to the `trip_duration_seconds` .
- The relationship is linear and strong, especailly with trips between 100 to 1000 seconds:
  - **explanation**: The conclusion is corresponding to our common sense that the farther the trip is, the longer time it takes.
- The correlation is weaker when the `trip_duration_seconds` is less than 100 or more than 1000 seconds:
  - **explanation**:
    - When the `trip_duration_seconds` is less than 100, probably the data is not accurate, because it is weird to travel so far within 2 minutes.
    - When the `trip_duration_secodns` is more than 1000, probably the `trip_duration_seconds` is not sololy determined by the `distance` .
      - Riders could already travel 3000 meters within 1000 seconds, and it is rare to ride farther than 3000 meters by bikeshare.
      - Users who rides more than 1000 meters might have stopped on their way and dealed with other things.
- **limitation**: We drew the conclusion only based on the mean `distance` , rather than individual trips, so it works only in aggregate and individual difference might exist.

# 2. Casual Users v.s. Members

I noticed that there are two types of users of Bikeshare: casual users and members. I would also like to investigate whether there are distinct differences in the length of trips between members vs. casual users.

**To begin with**, I used the `groupby` and `describe` function to see the simple profile of `trip_duration_seconds` and `distance` of two types of users separately.

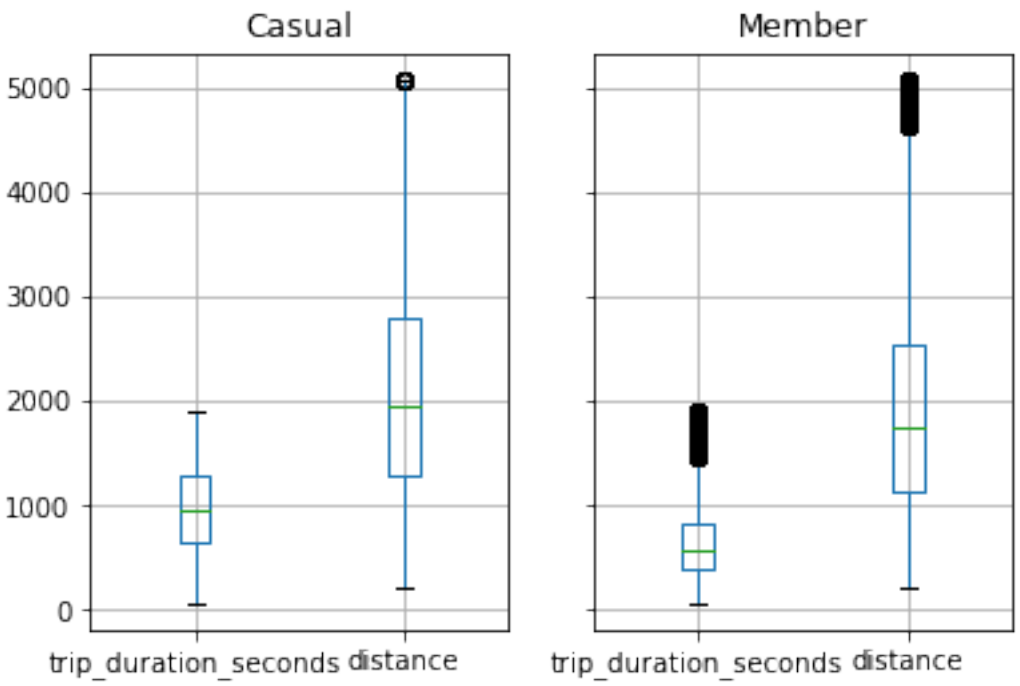
Out[74]:

| | trip_duration_seconds | | | | | | | distance | |
|---|---|---|---|---|---|---|---|---|---|
| | count | mean | std | min | 25% | 50% | 75% | max | count | n |
| user_type | | | | | | | | | | |
| **Casual** | 320483.0 | 973.762434 | 411.411103 | 60.0 | 649.0 | 945.0 | 1279.0 | 1905.0 | 320483.0 | 2 |
| **Member** | 1520718.0 | 630.250933 | 321.425057 | 60.0 | 387.0 | 572.0 | 820.0 | 1905.0 | 1520718.0 | 1 |

More adjectively, I also drew a set of boxplots for this data to visualize the difference.

Out[27]:

```
Casual          AxesSubplot(0.1,0.15;0.363636x0.75)
Member     AxesSubplot(0.536364,0.15;0.363636x0.75)
dtype: object
```

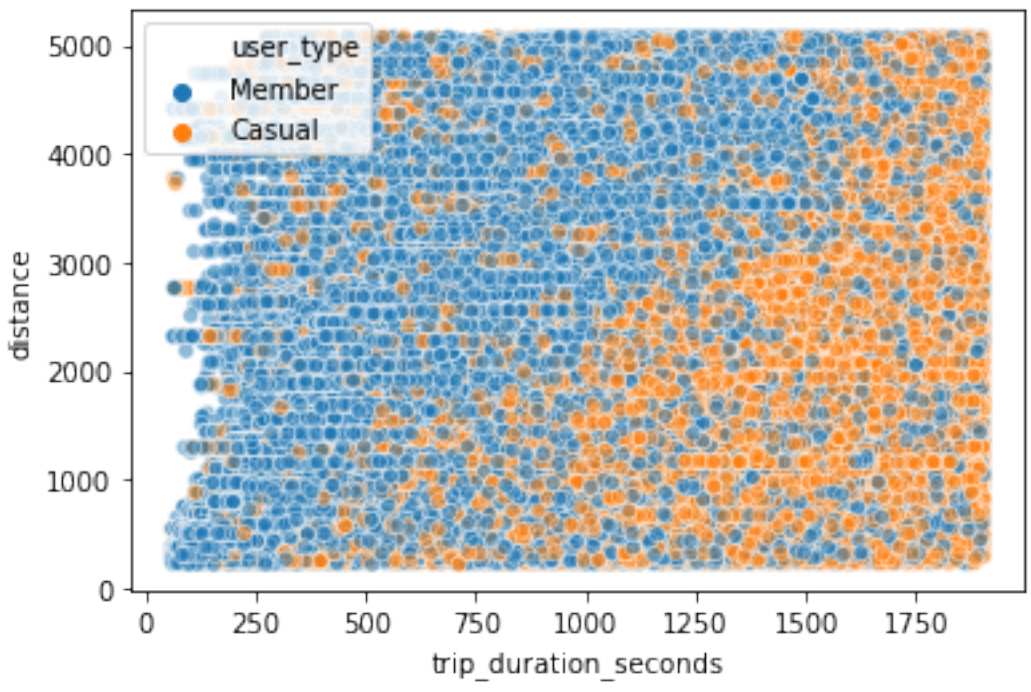


From the table and the graphs, I noticed that with the same min and max, the 25% quantile, 50% quantile, 75% quantile, and mean of both the `trip_duration_seconds` and `distance` of casual users are larger than those of members.

**Further more**, in order to find out the relationship between these two definitions of trip length of both types of users, I also drew a scatter plot to illustrate.

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a3620be0>
```

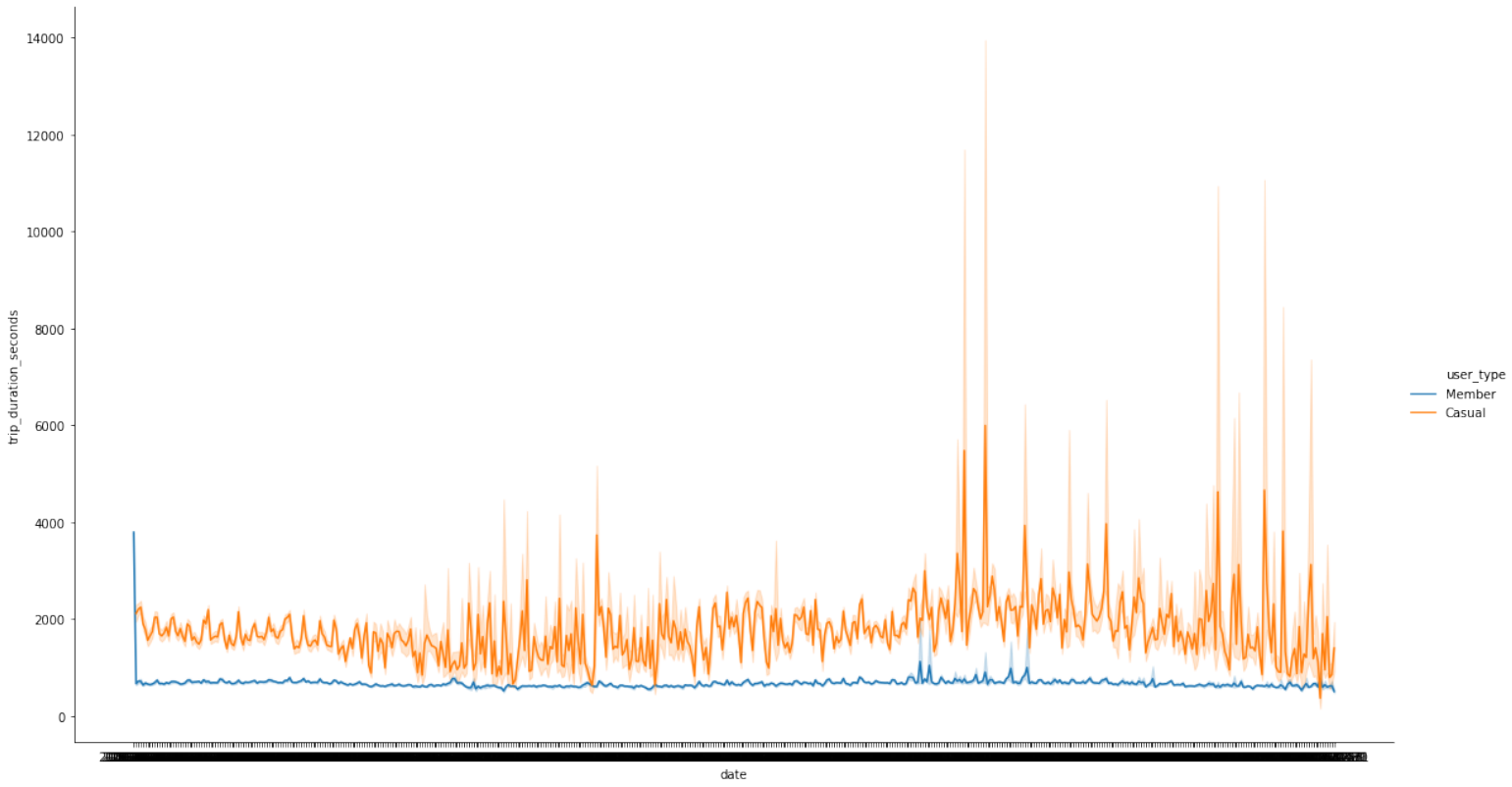



In this graph, the blue dots representing members dominates the upper-left corner, showing that members tends to take "longer" trips defined by `distance` ; the orange dots representing casual users dominates the lower-right corner, showing the casual users tends to take "longer" trips defined by `trip_duration_seconds`

However, there are too many dots on the graph, so the observation might not be accurate. Instead, I also plotted `replots` to further demonstrate the relationships.

Out[137]:

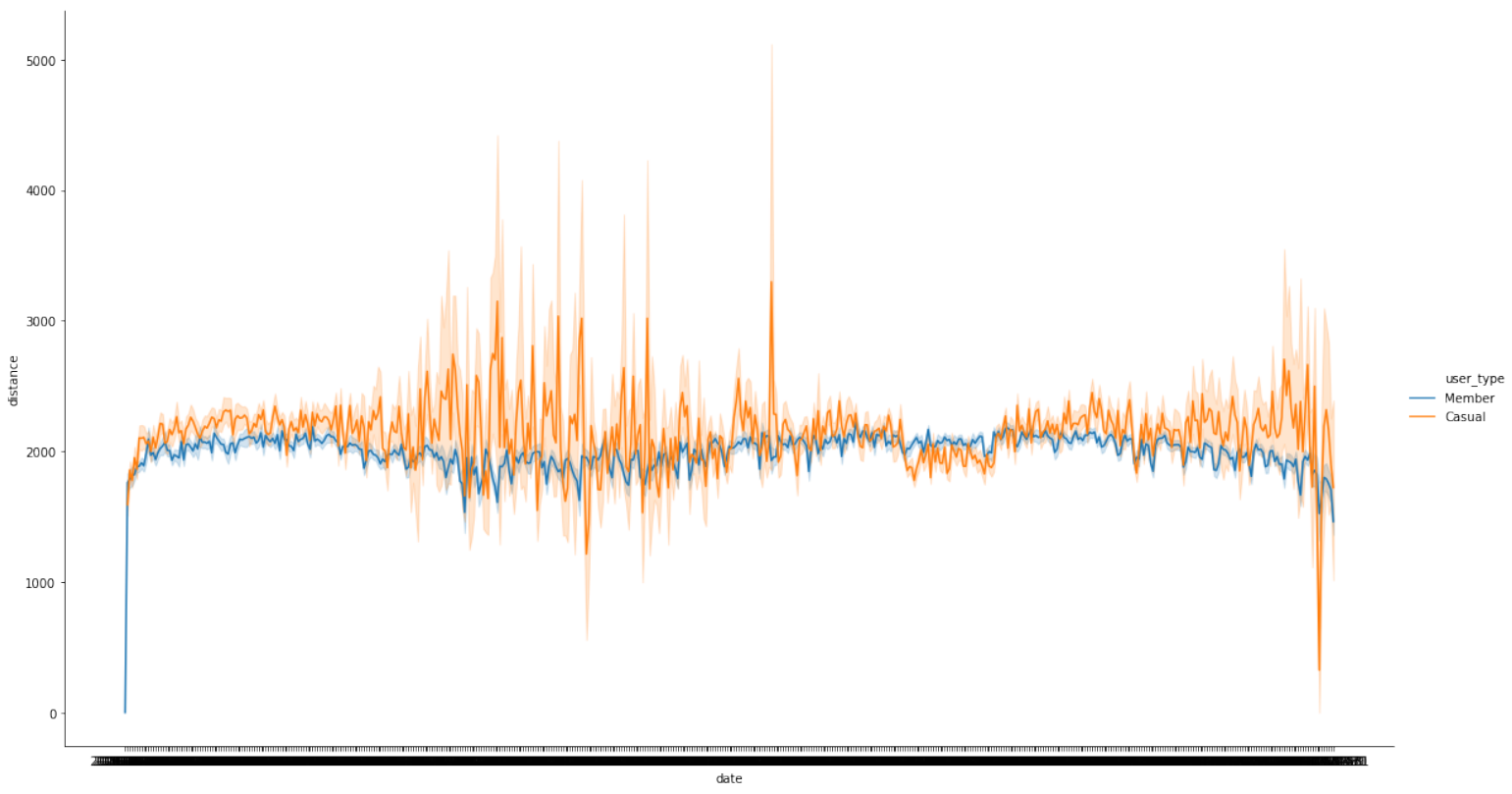
<seaborn.axisgrid.FacetGrid at 0x1a656f8e0>




From this graph, I discovered that the `trip_duration_seconds` of trips of members are obviously shorter than that of casual members. Also, the trip length defined by `trip_duration_seconds` varies more among casual users and varies less among

Out[136]:

<seaborn.axisgrid.FacetGrid at 0x1a681abb0>



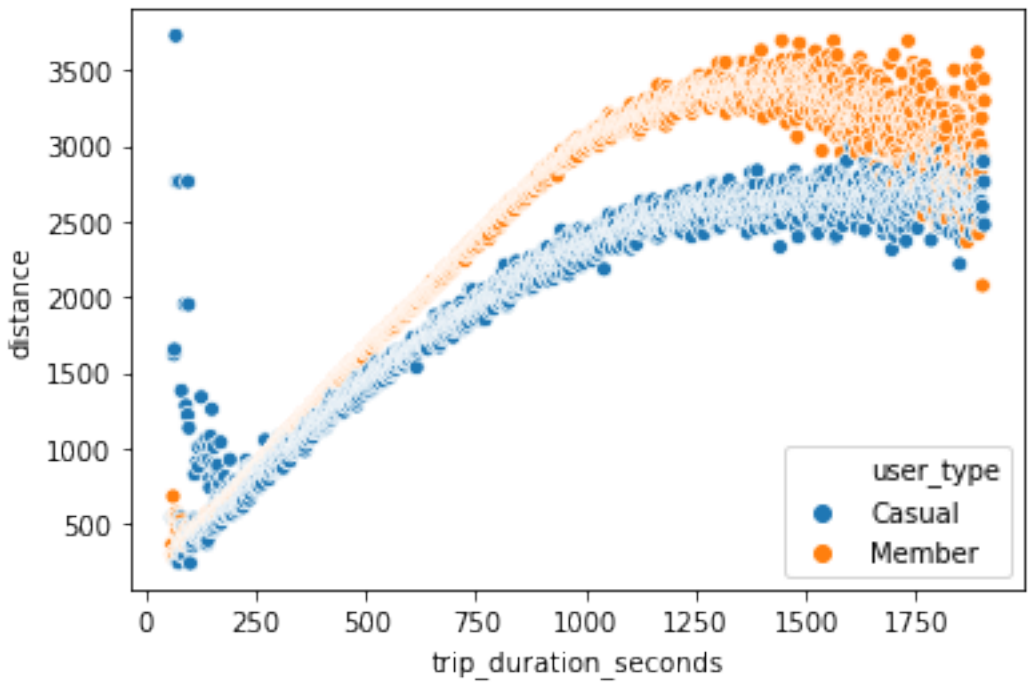
From this graph, I discovered that the trip length defined by `distance` still varies more among casual members than that among members, but the `distance` of trips of casual users is only slightly higher then that of members.

**Next**, to further investigate this relationship, I used the same strategy as investigating the relationship between `trip_duration_seconds` and `distance`.

I grouped the data by `trip_duration_seconds`, calculated the mean of `distance` in each group, and drew a new scatter plot using the grouped data.

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a22a52e0>
```



From this graph, it is obvious that with the same `trip_duration_seconds`, members tend to travel further in `distance` than casual users.

## conclusions and considerations:

- Members usually take longer trips than casual users defined by either `trip_duration_seconds` or `distance`:
  - **Limitation**: This conclusion is based on the table and boxplot, which can be pretty concise, but only gives us a overall profile.
  - **Explanation**: Considering the terms of membership, members only pay a fixed amount of fee regardless of the times they use the bikeshare system, while casual users have to buy the pass every time they use regardless of the length of their trips. Thus, members can use the bikeshare whenever they want, but it is more economical for casual members to buy the pass when they need to take longer trips.

- The trip length of casual users varies a lot. In general, the trip length of casual users is much longer than that of members when it is defined by `trip_duration_seconds`, but it is only slightly longer among casual users when it is defined by `distance`
  - This conclusion is based on the relationship plot. It is more obvious under the definition of `distance` than that of `trip_duration_seconds`, because of less varaince.
  - **A possible explanation**:
    - The `distance` that casual users take are only slightly larger than that of members, but with a large variance. This difference might not be large enough to be considered significant.
    - But the `trip_duration_seconds` that casual users take are much larger that that of members, probably because casual users, who rarely use the bikeshare, are not familiar with the routes between stations and take longer time to travel similar distance.

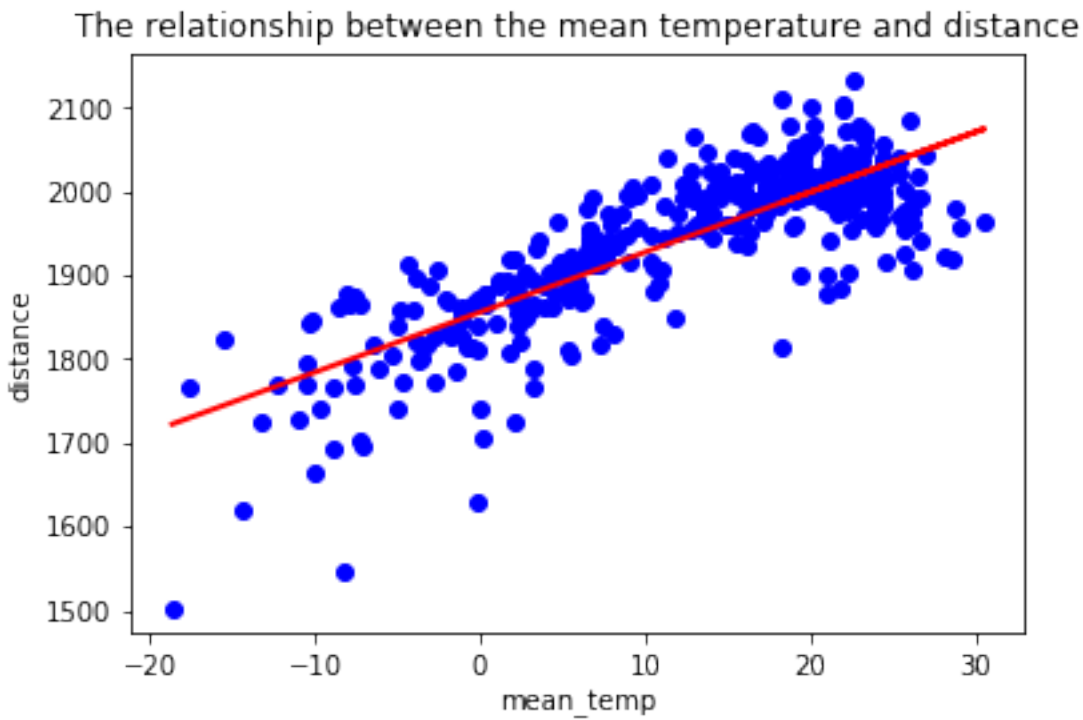
- With the same `trip_duration_seconds`, members tend to travel farther in `distance` than casual members:
  - This conclusion is based on the second scatter plot and is obvious to draw, so it can be concise.
  - **Limitation**: However, this scatter plot is based on grouped data and the mean `distance` traveled, so it only shows the aggragate data and individual difference might exist.
  - **Explanation**: Members are able to travel farther in `distance` in the same `trip_duration_seconds`, probably because members take the rides more and are more familiar with the routes.
  - This conclusion also gives an support my explanation for conclusion2: even though casual members travel only slightly longer in length defined by `distance`, they take more `trip_duration_seconds` to travel the same `distance`.

# 3. Relationship between Weather and Trip Length.

**First**, I would analyse the relationship between the ridership and the mean temperature.

- **Consider the `distance` and `mean_temp`**

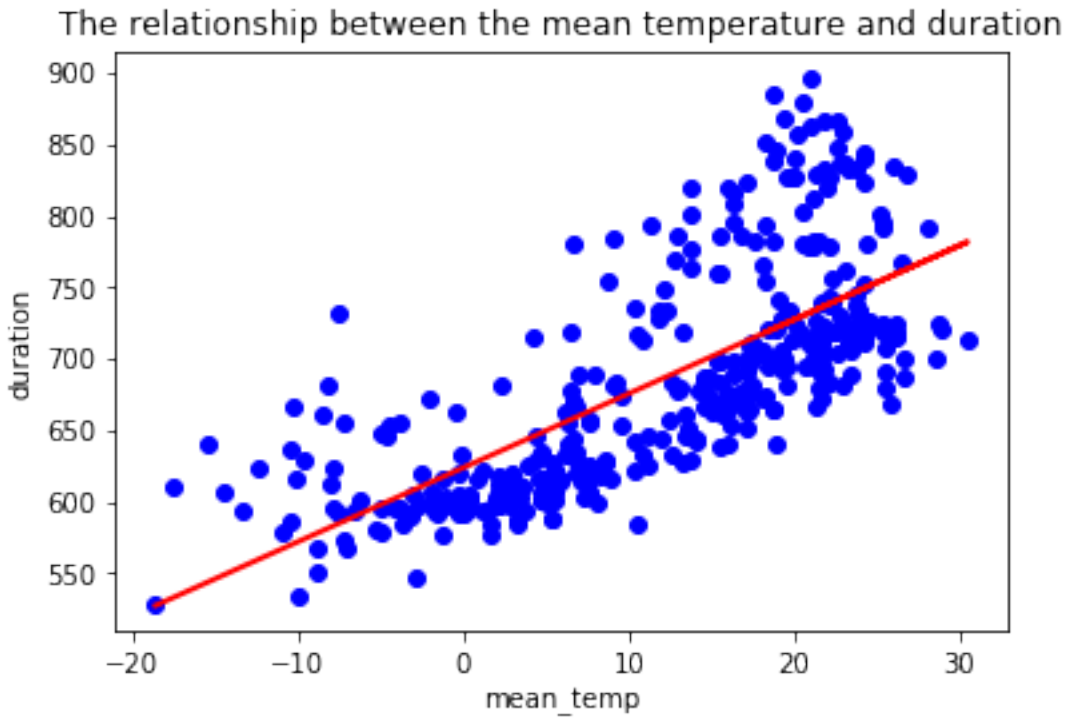
To visualize the relationship, I drew a scatter plot with regression line of the `mean_temp` and `distance` from the `weather_data` dataframe.




From this graph, we saw that the mean `distance` travelled in a day and the `mean_temp` of that day is positively related, and the relation is strong.

- **Consider the `duration` and `mean_temperature`**

Similarly, I drew a scatter plot with regression line of the `mean_temp` and `duration` from the weather_data dataframe.




From this graph, we saw that the mean `duration` of trips in a day and the `mean_temp` of that day is positively related. But the relation is weaker than that of `distance` and `mean_temp`.

**Next**, we were also interested in the relationship between the trip length and precipitation.

Instead of regarding precipitation as a numeric variable, I chose to convert it to a qualitative variable `rain` representing whether there is precipitation in a day or not.

Thus, I created a new column named `rain`. Its value is true when the `total_precip` is not zero and false when `total_precip` equals zero.

Out[191]:

| | mean_temp | total_precip | date | duration | distance | rain |
|---|---|---|---|---|---|---|
| **0** | 18.3 | 5.6 | 2016-07-01 | 755.201842 | 1813.031033 | True |
| **1** | 19.4 | 0.0 | 2016-07-02 | 867.896169 | 1899.371816 | False |
| **2** | 21.0 | 0.0 | 2016-07-03 | 862.735355 | 1879.030543 | False |
| **3** | 21.8 | 0.0 | 2016-07-04 | 723.247626 | 1883.836123 | False |
| **4** | 26.1 | 0.0 | 2016-07-05 | 714.957033 | 1905.608854 | False |

Then, I used the `describe()` function to get a simple profile of the mean `duration` and `distance` when it has precipitation or not during the day respectively.

Out[193]:

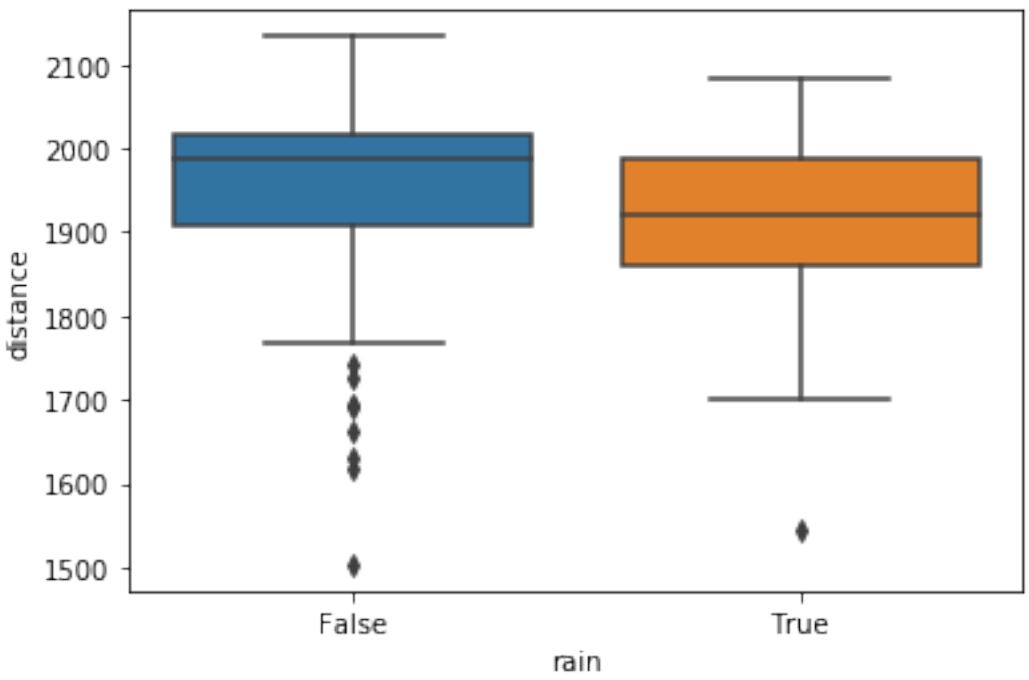
| | duration | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | count | mean | std | min | 25% | 50% | 75% | max |
| rain | | | | | | | | |
| False | 227.0 | 699.111793 | 78.344167 | 528.588068 | 637.965081 | 694.721543 | 736.164144 | 884.729 |
| True | 149.0 | 664.545126 | 70.401035 | 547.986938 | 606.116468 | 646.533937 | 707.066646 | 896.712 |

From the table, it is easy to discover that the mean, 25% quantile, 50% quantile, and 75% quantile of both the average `duration` and the `distance` of the trips in a day are larger when there is no precipitation in that day.

I also drew two sets of boxplots to visualize the above results.

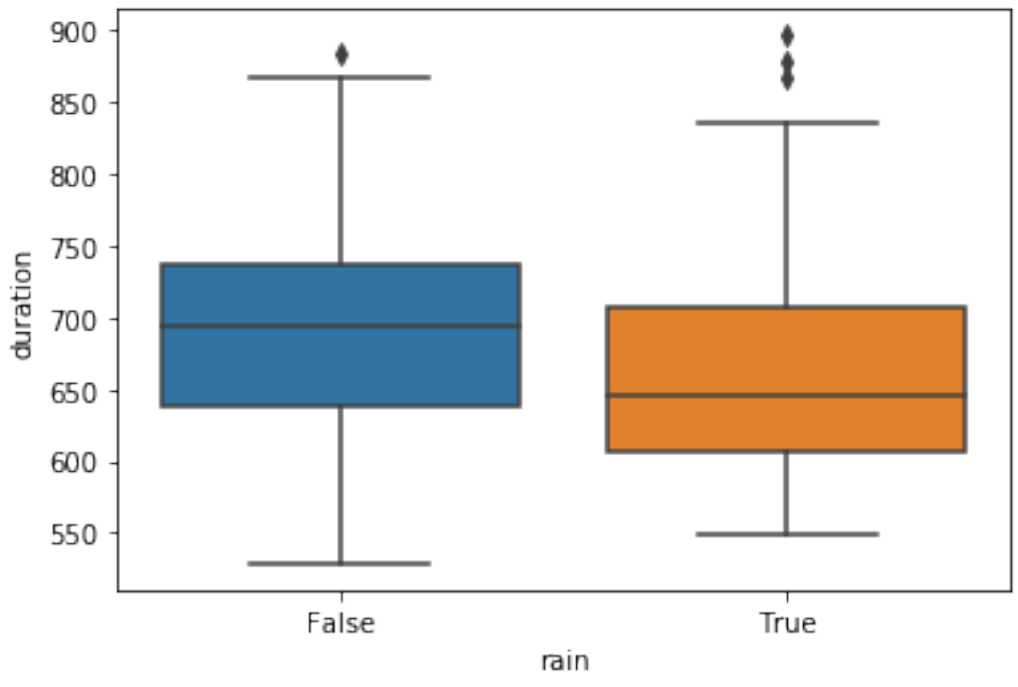
Out[186]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x125979fd0>
```



Out[187]:

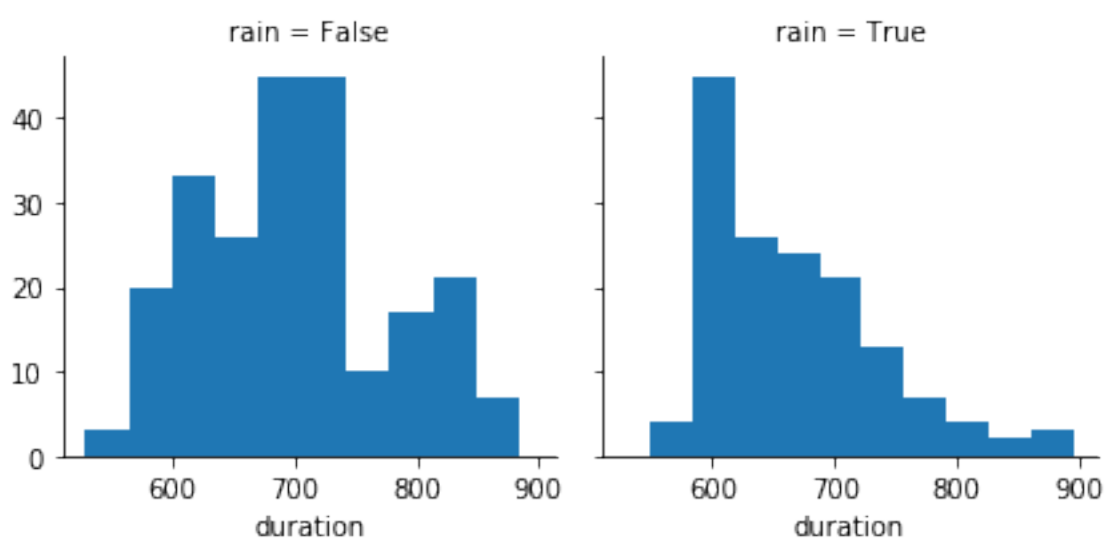
```
<matplotlib.axes._subplots.AxesSubplot at 0x125aaaeb0>
```




Also, I drew two histgrams of mean `duration` of trips in a day to further support the result.

Out[168]:

```
<seaborn.axisgrid.FacetGrid at 0x1246e1070>
```




From the histgrams, we saw that the distribution of `duration` when there is no precipitation is bimodel, with a higher peak at 700 seconds and a lower peak around 800 seconds. However, the distribution of `duration` when there is precipitation is unimodel, with the mode at 600 seconds, and the longer the duration, the smaller counts it has.

Thus, users tend to take shorter trips when there is no precipitation in that day, if the trip length is defined by `duration` ,

However, it seemed that there is not obvious patterns in the graph and the correlation between precipitaion and ridership seemed to be low.

## Conclusion and consideration.

- The trip length is positively related to the mean temperature of the day in both definition of `distance` and `duration`.
  - The higher the temperature, the longer the trips are.
    - **Explanation**: Considering the weather characteristics in Toronto, it is sometimes very cold, but seldom gets extremly hot. It is inconvenient to ride bikes when it is cold, but the higher temperature would not lead to any inconvenience.
  - The corelation between the `distance` and `mean_temp` is relatively stronger than that between `duration` and `mean_temp`
    - **Explanation**: The distance to travel depends only on the rider, but the duration of trips to travel the same distance can also be affected by the weather condition. So riders could choose to take trips shorter in distance when the temperature is lower, but it might take longer times to travel for the same distance in winter when the temperature is lower.

- The average length of trips is shorter when there is precipitation in a day, and it is longer when there is no precipitation.
  - **Explanation**: Probably because, when there is precipitaion, either rain or snow, it is not convenient to rider.
  - **Limitation**: When the data is divided into two groups: precipitation or no precipitation, the sample size of each group is not large enough, so the result might not be accurate.

Out[195]:

| rain | duration | distance |
|---|---|---|
| **False** | 227 | 227 |
| **True** | 149 | 149 |

- **Limitation**: These conclusions are only based on the mean `distance` and `duration` of all trips in each day, so it can only gives us a relationship in general. It might not be appliable to individuals as individual difference might exist.
- **Possible improvement**: The data only provides the total precipitation and mean temperature at specific weather station location. However, the temperature can vary a lot and the rainfall also might not be consistent throughout the entire day and across all Bike Share stations locations. Therefore, the conclusion based on the weather of the entire day might not be concrete and precise. Since the ridership varies depending on hour and specific stations, the weather data on an hourly level and from multiple weather stations are necessary for further investigations.

# Summary

While the Bikeshare Ridership data provided on the Open Datta is fairly simple, containing only basic information of each trips, I merged it with other public resources and finally got some sights into the characteristics of the data:

1. The duration and distance of trip are positively related. The correlation is strong for trips taking between 100 and 1000 seconds, but weaker otherwise.
    - This is corresponding to our common senses: the farther the trip, the longer time it takes.
    - Users might have stopped during their trips longer than 1000 seconds, so the duration is not sololy determined by the distance.
2. Casual users tend to take longer trips, especially longer in duration.
    - Probably because it is more economical for casual users to take longer trips.
    - Casual users might spend more time to travel the same distance considering their unfamiliarity.
3. Users tend to take longer trips, espcially in distance, when it is warmer and there is no precipitation in during the day.
    - Considering the weather characteristics of the city, Toronto, it is more convenient to ride bikes when it is warmer and with no precitipation.
    - The duration of trips to travel the same distance can also be affected by the weather condition.

There are also some limitations and possible improvements of the analysis and they are discussed in the conclusion and considerations under each specific topic.

This analysis gives us a glimpse into the bikeshare ridership in Toronto. We understood some traits of the ridership. I hope this analysis could help find out a better way providing the bikeshare service.