# Exploration_BayWheels

October 7, 2020

# 1 San Francisco Bay's bicycle sharing system data exploration

## 1.1 by (Keila González-Gómez)

### 1.1.1 Table of Contents



## Introduction

This dataset contains observations from the regional public bicycle sharing system of San Francisco Bay Area. This system is called Bay Wheels. It holds almost half million entries with variables describing the trips along the year 2017. Information covers the type of user, start and end stations, as well as the duration of the trip.

[1]:
```
# import required packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

## Preliminary Wrangling

The dataset is loaded and inspected programmatically with pandas built-in functions.

[2]:
```
# read CSV (comma-separated) file into DataFrame
trip_data = pd.read_csv('2017-fordgobike-tripdata.csv')
trip_data.info()
trip_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 519700 entries, 0 to 519699
Data columns (total 13 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             519700 non-null  int64
 1   start_time               519700 non-null  object
 2   end_time                 519700 non-null  object
 3   start_station_id         519700 non-null  int64
 4   start_station_name       519700 non-null  object
 5   start_station_latitude   519700 non-null  float64
 6   start_station_longitude  519700 non-null  float64
 7   end_station_id           519700 non-null  int64
 8   end_station_name         519700 non-null  object
 9   end_station_latitude     519700 non-null  float64
 10  end_station_longitude    519700 non-null  float64
 11  bike_id                  519700 non-null  int64
 12  user_type                519700 non-null  object
dtypes: float64(4), int64(4), object(5)
memory usage: 51.5+ MB
```

[2]:
```
   duration_sec                start_time                  end_time  \
0         80110  2017-12-31 16:57:39.6540  2018-01-01 15:12:50.2450
1         78800  2017-12-31 15:56:34.8420  2018-01-01 13:49:55.6170
2         45768  2017-12-31 22:45:48.4110  2018-01-01 11:28:36.8830
3         62172  2017-12-31 17:31:10.6360  2018-01-01 10:47:23.5310
4         43603  2017-12-31 14:23:14.0010  2018-01-01 02:29:57.5710

   start_station_id                                start_station_name  \
0                74                               Laguna St at Hayes St
1               284  Yerba Buena Center for the Arts (Howard St at …
2               245                              Downtown Berkeley BART
3                60                                8th St at Ringold St
4               239                         Bancroft Way at Telegraph Ave

   start_station_latitude  start_station_longitude  end_station_id  \
0               37.776435              -122.426244              43
1               37.784872              -122.400876              96
2               37.870348              -122.267764             245
3               37.774520              -122.409449               5
4               37.868813              -122.258764             247

                                   end_station_name  end_station_latitude  \
0  San Francisco Public Library (Grove St at Hyde…             37.778768
1                              Dolores St at 15th St             37.766210
2                             Downtown Berkeley BART             37.870348
3       Powell St BART Station (Market St at 5th St)             37.783899
```

```
4                          Fulton St at Bancroft Way             37.867789

   end_station_longitude  bike_id   user_type
0            -122.415929       96    Customer
1            -122.426614       88    Customer
2            -122.267764     1094    Customer
3            -122.408445     2831    Customer
4            -122.265896     3167  Subscriber
```

These variables could help analyzing the type of customer associated with longer or shorter trips. Do regular customers go on shorter trips? Perhaps regular commuting? Do occasional riders travel longer distances? These could be tourists that decide to get to know the city using the bike system.

[3]:
```
##initial inspection of the dataset
print(trip_data.shape)
print(trip_data.dtypes)
```

```
(519700, 13)
duration_sec                 int64
start_time                  object
end_time                    object
start_station_id             int64
start_station_name          object
start_station_latitude     float64
start_station_longitude    float64
end_station_id               int64
end_station_name            object
end_station_latitude       float64
end_station_longitude      float64
bike_id                      int64
user_type                   object
dtype: object
```

[4]:
```
# look at a sample of entries/rows and the values these could take
trip_data.sample(5)
```

[4]:
```
        duration_sec                start_time                  end_time  \
233603           212  2017-10-17 17:30:34.9940  2017-10-17 17:34:07.3700
36985            558  2017-12-14 18:31:15.4530  2017-12-14 18:40:33.7210
107763           689  2017-11-23 13:05:13.5420  2017-11-23 13:16:43.3470
182617           216  2017-10-31 19:27:40.6600  2017-10-31 19:31:17.1430
379929           251  2017-09-05 08:49:18.3800  2017-09-05 08:53:30.3470

        start_station_id        start_station_name  start_station_latitude  \
233603               201        10th St at Fallon St               37.797673
36985                 77        11th St at Natoma St               37.773507
```

```
107763                 10  Washington St at Kearny St               37.795393
182617                201        10th St at Fallon St               37.797673
379929                 76  McCoppin St at Valencia St               37.771662

        start_station_longitude  end_station_id  \
233603              -122.262997             233
36985               -122.416040              67
107763              -122.404770              19
182617              -122.262997             233
379929              -122.422423              98

                                          end_station_name  \
233603                                  12th St at 4th Ave
36985   San Francisco Caltrain Station 2  (Townsend St…
107763                                Post St at Kearny St
182617                                  12th St at 4th Ave
379929                              Valencia St at 16th St

        end_station_latitude  end_station_longitude  bike_id   user_type
233603             37.795812            -122.255555      978    Customer
36985              37.776639            -122.395526     2631  Subscriber
107763             37.788975            -122.403452     2797    Customer
182617             37.795812            -122.255555      550  Subscriber
379929             37.765052            -122.421866      248  Subscriber
```

There are some workarounds that could make this data frame deliver results in more understandable terms, for instance the duration of the trip could be expressed in hours or minutes instead of seconds and the total length of the trip could be obtained (in an Euclidean fashion) from start and end coordinates.

[5]:
```
# add column with duration of trip in minutes or hours based common values
# verification of common values
trip_data.duration_sec.value_counts()
```

[5]:
```
357      745
378      716
363      714
385      714
388      712
        …
48232      1
27762      1
21621      1
17527      1
7023       1
Name: duration_sec, Length: 13490, dtype: int64
```

[6]:
```python
# verification of the distribution of the data
trip_data.duration_sec.describe()
```

[6]:
```
count    519700.000000
mean       1099.009521
std        3444.146451
min          61.000000
25%         382.000000
50%         596.000000
75%         938.000000
max       86369.000000
Name: duration_sec, dtype: float64
```

As seen, most of the trips lasted around 20 minutes, hence the new column will store duration values in minutes. It is worth noting that some trips lasted for more than 24 hours and around. Since this is a city bike sharing system these values could come from users that forgot to return the bike or that did not find an end point to leave it, and more.

[7]:
```python
# create new column with duration in minutes
trip_data['duration_min'] = trip_data['duration_sec'] / 60
```

[8]:
```python
# inspect the dataset
trip_data.head()
trip_data['duration_min'].value_counts()
```

[8]:
```
5.950000      745
6.300000      716
6.050000      714
6.416667      714
6.466667      712
             …
57.850000       1
170.066667      1
305.300000      1
77.333333       1
648.633333      1
Name: duration_min, Length: 13490, dtype: int64
```

[9]:
```python
# verification of the distribution of the data
trip_data['duration_min'].describe()
```

[9]:
```
count    519700.000000
mean         18.316825
std          57.402441
min           1.016667
25%           6.366667
50%           9.933333
```

```
75%          15.633333
max        1439.483333
Name: duration_min, dtype: float64
```

Most trips lasted less than 15 minutes.

A new column holding the length of the trips will be created from the coordinates of start and end stations. Unfortunately, round trips whose start and end stations are the same, will not deliver valuable data.

[10]:
```
# create new column with the trip's length
# first, define a function so as to turn 'lat' and 'long' values into metric␣
↪coordinates
# obtain their eucledian distance

def haversine_np(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)

    All args must be of equal length.
    author: derricw
    """
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2

    c = 2 * np.arcsin(np.sqrt(a))
    km = 6367 * c
    return km

#Create new column
trip_data['trip_length']=␣
↪haversine_np(trip_data['start_station_longitude'],trip_data['start_station_latitude'],trip_
```

[11]:
```
# inspect the newly created distance column
trip_data.trip_length.value_counts()
```

[11]:
```
0.000000    18134
1.309522     5078
1.412660     3154
2.170170     3087
1.072082     2994
              …
0.308362        1
```

```
2.370289        1
2.817111        1
2.093119        1
4.946972        1
Name: trip_length, Length: 10845, dtype: int64
```

[12]:
```
# inspect the distance with describe()
trip_data.trip_length.describe()
```

[12]:
```
count    519700.000000
mean          1.586080
std           1.009757
min           0.000000
25%           0.899078
50%           1.399365
75%           2.071193
max          68.143976
Name: trip_length, dtype: float64
```

Interesting to see, that many trip lengths have zero values. As mentioned before, these could come from round trips but also could be from non-initialized trips or problems with the bikes GPS's system.

Another interesting aspect to analyze would be the months with more trips. This information is stored in the start_time column, but in order to extract the month, its datatype has to be changed from string to datetime.

[13]:
```
# change the datatype with to_datetime()
trip_data['start_time'] = pd.to_datetime(trip_data['start_time'])
```

[14]:
```
trip_data['end_time'] = pd.to_datetime(trip_data['end_time'])
```

[15]:
```
# extract year and month trips started
trip_data['year'] = pd.DatetimeIndex(trip_data['start_time']).year
trip_data['month'] = pd.DatetimeIndex(trip_data['start_time']).month
```

[16]:
```
# making sure the columns were created with info()
trip_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 519700 entries, 0 to 519699
Data columns (total 17 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   duration_sec           519700 non-null  int64
 1   start_time             519700 non-null  datetime64[ns]
 2   end_time               519700 non-null  datetime64[ns]
 3   start_station_id       519700 non-null  int64
```

```
 4   start_station_name       519700 non-null  object
 5   start_station_latitude   519700 non-null  float64
 6   start_station_longitude  519700 non-null  float64
 7   end_station_id           519700 non-null  int64
 8   end_station_name         519700 non-null  object
 9   end_station_latitude     519700 non-null  float64
 10  end_station_longitude    519700 non-null  float64
 11  bike_id                  519700 non-null  int64
 12  user_type                519700 non-null  object
 13  duration_min             519700 non-null  float64
 14  trip_length              519700 non-null  float64
 15  year                     519700 non-null  int64
 16  month                    519700 non-null  int64
dtypes: datetime64[ns](2), float64(6), int64(6), object(3)
memory usage: 67.4+ MB
```

[17]:
```
# inspect the datatype for user_type
type(trip_data.user_type[9])
```

[17]:
```
str
```

This datatype should be a category. It will be changed to one with two possible values.

[18]:
```
# verification of categories within user_type
trip_data.user_type.value_counts()
```

[18]:
```
Subscriber    409230
Customer      110470
Name: user_type, dtype: int64
```

[19]:
```
# change datatype to Category
user_types = ['Subscriber' , 'Customer']
categorized_var = pd.api.types.CategoricalDtype(ordered = True, categories =␣
 ↪user_types)
trip_data['user_type'] = trip_data['user_type'].astype(categorized_var)
```

[20]:
```
# test changes with info()
trip_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 519700 entries, 0 to 519699
Data columns (total 17 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   duration_sec             519700 non-null  int64
 1   start_time               519700 non-null  datetime64[ns]
 2   end_time                 519700 non-null  datetime64[ns]
 3   start_station_id         519700 non-null  int64
```

```
 4   start_station_name       519700 non-null  object
 5   start_station_latitude   519700 non-null  float64
 6   start_station_longitude  519700 non-null  float64
 7   end_station_id           519700 non-null  int64
 8   end_station_name         519700 non-null  object
 9   end_station_latitude     519700 non-null  float64
 10  end_station_longitude    519700 non-null  float64
 11  bike_id                  519700 non-null  int64
 12  user_type                519700 non-null  category
 13  duration_min             519700 non-null  float64
 14  trip_length              519700 non-null  float64
 15  year                     519700 non-null  int64
 16  month                    519700 non-null  int64
dtypes: category(1), datetime64[ns](2), float64(6), int64(6), object(2)
memory usage: 63.9+ MB
```

[21]:
```
# the dataset has no missing values
trip_data.isna().sum()
```

[21]:
```
duration_sec               0
start_time                 0
end_time                   0
start_station_id           0
start_station_name         0
start_station_latitude     0
start_station_longitude    0
end_station_id             0
end_station_name           0
end_station_latitude       0
end_station_longitude      0
bike_id                    0
user_type                  0
duration_min               0
trip_length                0
year                       0
month                      0
dtype: int64
```

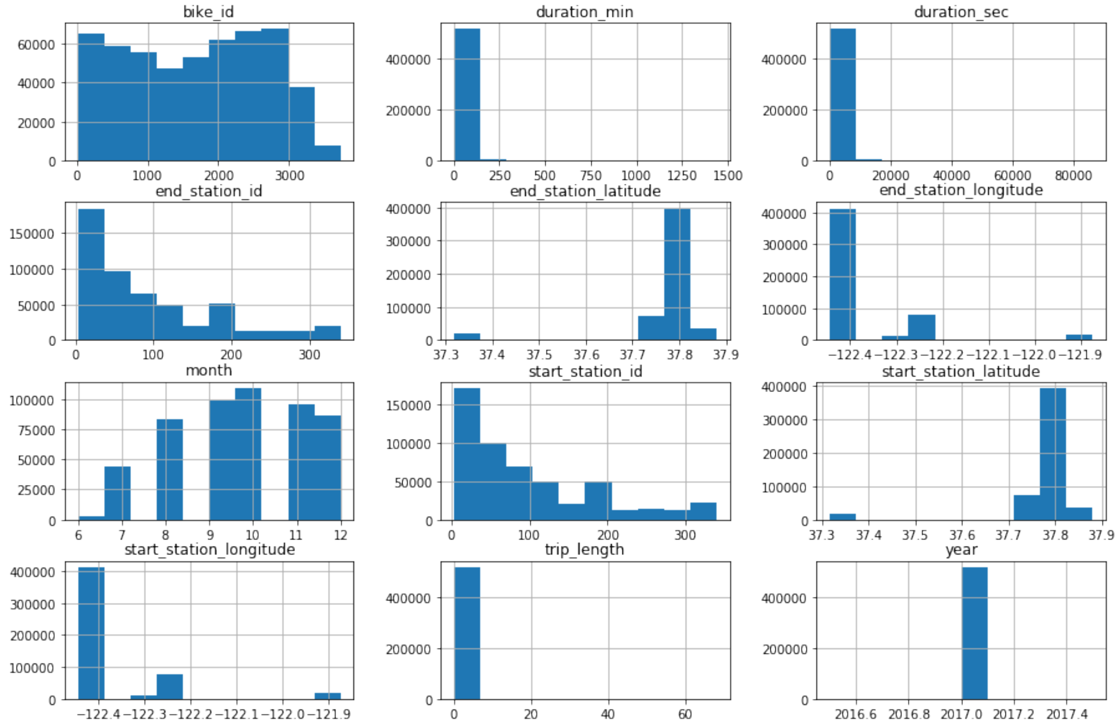
[22]:
```
# a programmatic inspection with describe()
trip_data.describe()
```

[22]:
```
        duration_sec  start_station_id  start_station_latitude  \
count  519700.000000     519700.000000           519700.000000
mean     1099.009521         95.034245               37.771653
std      3444.146451         86.083078                0.086305
min        61.000000          3.000000               37.317298
25%       382.000000         24.000000               37.773492
```

```
50%       596.000000        67.000000                37.783521
75%       938.000000       139.000000                37.795392
max     86369.000000       340.000000                37.880222

       start_station_longitude  end_station_id  end_station_latitude  \
count            519700.000000   519700.000000         519700.000000
mean               -122.363927       92.184041             37.771844
std                   0.105573       84.969491              0.086224
min                -122.444293        3.000000             37.317298
25%                -122.411726       23.000000             37.774520
50%                -122.398870       66.000000             37.783830
75%                -122.391034      134.000000             37.795392
max                -121.874119      340.000000             37.880222

       end_station_longitude        bike_id   duration_min    trip_length  \
count          519700.000000  519700.000000  519700.000000  519700.000000
mean             -122.363236    1672.533079      18.316825       1.586080
std                 0.105122     971.356959      57.402441       1.009757
min              -122.444293      10.000000       1.016667       0.000000
25%              -122.410345     787.000000       6.366667       0.899078
50%              -122.398525    1728.500000       9.933333       1.399365
75%              -122.391034    2520.000000      15.633333       2.071193
max              -121.874119    3733.000000    1439.483333      68.143976

           year          month
count  519700.0  519700.000000
mean     2017.0       9.731716
std         0.0       1.566787
min      2017.0       6.000000
25%      2017.0       8.000000
50%      2017.0      10.000000
75%      2017.0      11.000000
max      2017.0      12.000000
```

[23]:

```
# a visual inspection with hist()
trip_data.hist(figsize = (15,10));
```




Unfortunately, most of the variables within this dataset contain 'numerical' data but are not useful *quantitative* variables. For instance, the bike_id is a unique identifier and could be considered a categorical variable. The geospatial data as well is useful with spatial references but not with this kind of visuals. Still, repetitions on latitude and longitude values along end and start stations could show which stations are more popular over others to start and end trips, still, these insights could be obtained as well with station_ids.

Furthermore, many of these histograms are skewed to the right. This means that lower values of trip duration and trip's length are the majority.

### 1.1.2 What is the structure of your dataset?

This dataset holds almost half a million (519,700) logs of bike rentals. The descriptive table originally contained 13 variables with information regarding the duration of the trip (with start and end time), the user type, the start and end stations of the trip and their respective coordinates. From these 13 columns, 4 more variables were extracted in order better comprehend the relationship between variables. These were the transformation of durations from seconds to minutes, the obtention of the trip's length (making use of their coordinates) and the months when trips were made (extracted from the start time column). Most variables are numeric but do not possess numerical characteristics as they are identifiers of bikes, trips and stations.

### 1.1.3 What is/are the main feature(s) of interest in your dataset?

I would like to see how the variables are related to each other. For example, which months hold more trips, which type of customer travels farther or for longer time, are there 'favorite' stations with more users than the rest? Do longer trips start and end at the same station? and so on.

### 1.1.4 What features in the dataset do you think will help support your investigation into your feature(s) of interest?

I think the newly created variable named "months" will highlight which months are more popular among riders. The customer type and with more repetitions will be the one type that made more use of that sharing service during that year. The statistics of the station's ids will shade light on whether or not there is a favorite station. In addition, I suppose there will be a strong relationship between trips length and duration (the further the trip the longer it takes). Out of all 17 variables, 6 will be mainly inspected these are: start station, end station, user type, month, duration in minutes and trip length and month. The first four are categorical and the last two quantitative.

## Univariate Exploration

The first variable to be inspected is the trip's duration. What is the mean trip duration across this dataset? Does it follow a normal distribution?

[24]:
```
# inspection to determine suitable bin sizes
trip_data.duration_min.max()
```

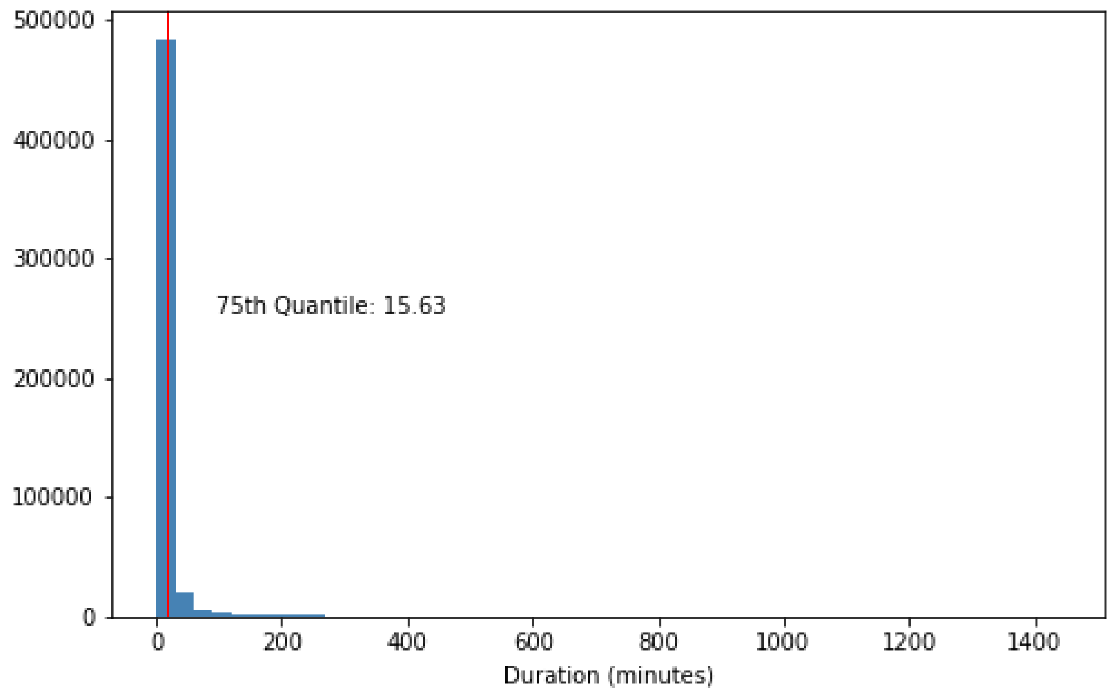
[24]: 1439.4833333333333

[25]:
```
# inspection to determine suitable bin sizes
trip_data.duration_min.min()
```

[25]: 1.0166666666666666

[26]:
```
# plot with seaborn
sb_colors = sb.color_palette()[2] #set colors
sb.countplot(data = trip_data, x = 'duration_min', color = sb_colors);
plt.axis('off');
```

[27]:
```
# from the general histogram plot the binsize changing multiple values
# a very large bin size
binsize = 30
bins = np.arange(0, trip_data['duration_min'].max()+binsize, binsize)

plt.figure(figsize=[8, 5])
plt.hist(data = trip_data, x = 'duration_min', bins = bins, color = 'steelblue')
plt.xlabel('Duration (minutes)')

# bar to show where the 75th quantile of the values is
plt.axvline(trip_data.duration_min.quantile(0.75), color='red', linewidth = 1)
min_ylim, max_ylim = plt.ylim()
plt.text(trip_data.duration_min.quantile(0.75)*6, max_ylim*0.5, '75th Quantile:␣
↪{:.2f}'.format(trip_data.duration_min.quantile(0.75)));
```

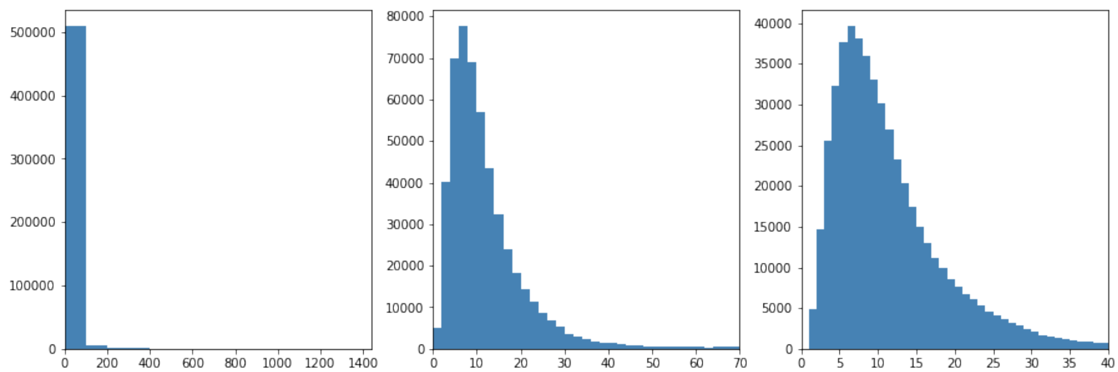



As seen in the histogram plots of the whole dataset, this variable shows a very skewed distribution due to few extremely-long-lasting trips. Most durations last less than 15 minutes (3rd Quantile). Usually these bikes are used as last and first mile options, so these values are supportive of this. Minimum values are around 1 minute.

This distribution shows a long tail, perhaps due to some outliers spotted during the programmatically inspection. These values hold durations that span several hours. In order to get better insights of the data two things will be done, first zoom into more interesting parts of the histogram and second plot the data on a log scale.

[28]:
```
# first: zoom in
plt.figure(figsize = [15, 5])

# histogram on the left: full data with large bin size
plt.subplot(1, 3, 1)
binsize = 100
bin_edges = np.arange(0, trip_data['duration_min'].max()+1, binsize)
plt.hist(data = trip_data, x = 'duration_min', bins = bin_edges, color =␣
 ↪'steelblue')
plt.xlim(0, trip_data['duration_min'].max()+1);

# center: zoom into trips lasting little more than 1 hour
plt.subplot(1, 3, 2)
binsize2 = 2
bin_edges = np.arange(0, 70+binsize2, binsize2)
```

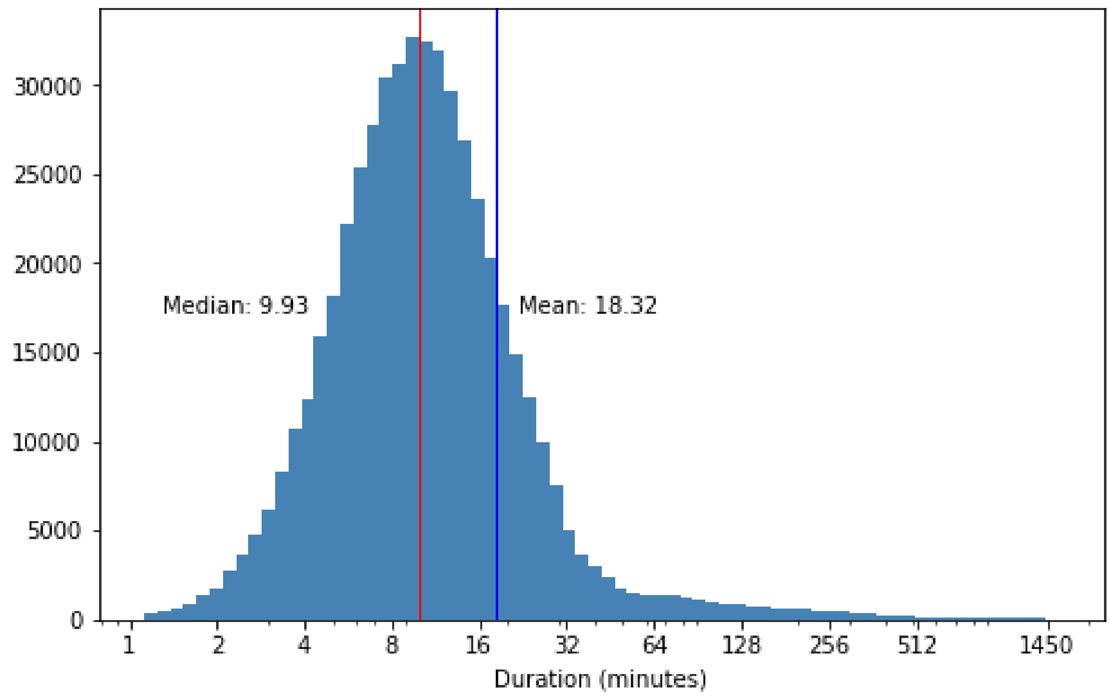
```
plt.hist(data = trip_data, x = 'duration_min', bins = bin_edges, color =␣
 ↪'steelblue')
plt.xlim(0, 70);

# histogram on right: smoother histogram with most common data
plt.subplot(1, 3, 3)
binsize3 = 1
bin_edges = np.arange(0, 40+binsize2, binsize3)
plt.hist(data = trip_data, x = 'duration_min', bins = bin_edges, color =␣
 ↪'steelblue')
plt.xlim(0, 40);
```




In general, the three graphs show a skewed distribution with mean higher than median and one mode.

[29]:
```
# scale transformation intended at a better representation of the data

log_binsize = 0.045
bins = 10 ** np.arange(0.05, np.log10(trip_data['duration_min'].
 ↪max())+log_binsize, log_binsize)

plt.figure(figsize=[8, 5])
plt.hist(data = trip_data, x = 'duration_min', bins = bins, color = 'steelblue')
plt.xscale('log')
tick_locs = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1450]
plt.xticks(tick_locs, tick_locs)
plt.xlabel('Duration (minutes)')


# bar to show where the mean and median are
plt.axvline(trip_data.duration_min.mean(), color='blue', linewidth = 1)
min_ylim, max_ylim = plt.ylim()
plt.text(trip_data.duration_min.mean()*1.2, max_ylim*0.5, 'Mean: {:.2f}'.
 ↪format(trip_data.duration_min.mean()))
```

```
plt.axvline(trip_data.duration_min.median(), color='red', linewidth = 1)
min_ylim, max_ylim = plt.ylim()
plt.text(trip_data.duration_min.median()*0.13, max_ylim*0.5, 'Median: {:.2f}'.
↪format(trip_data.duration_min.median()))
plt.show();
```




This graph looks more like a normal distribution. As seen before the median is almost 10 minutes with a mean of 18 minutes; this mean is heavily affected by the larger values. Perhaps analyzing the data by homogeneous groups would change how it looks like.

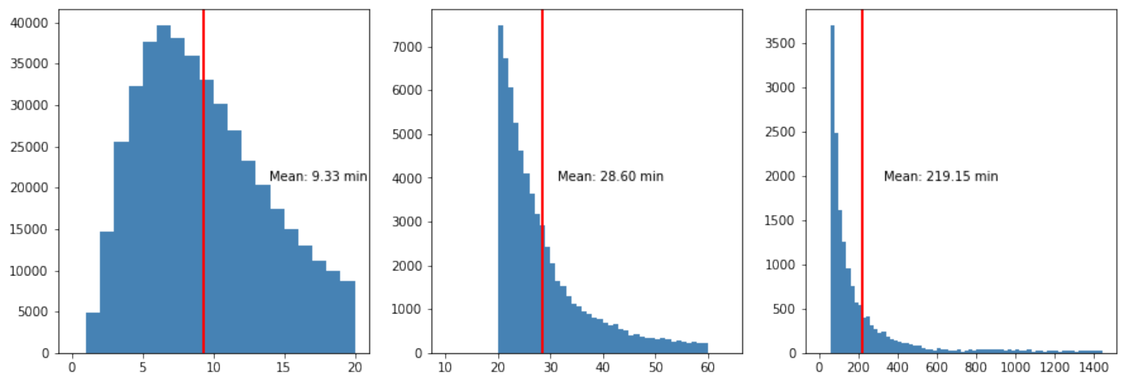
Having inspected the data visually and programmatically suggests that most trips are indeed short ones, but longer trips could be also interesting to analyze. One option could be separate the data between short, medium and long trips and explore them separately. In that regard, revised literature on the subject indicates that most sharing services have 9 minutes as the most common trip duration. Also, trips shorter than 30 min are commonly encouraged (with lower rates) by many public biking services so as to have more bikes available. The dataset will be divided in short tips, those lasting less than 20 min, medium distance trips lasting 20 minutes to 1 hour and longer trips, those lasting more than an hour.

[30]:
```
# segmentation
short_trips = trip_data.drop(trip_data.loc[trip_data['duration_min'] > 20].
↪index)
```

```
medium_trips = trip_data.loc[(trip_data['duration_min'] > 20) &␣
↪(trip_data['duration_min'] < 60)]
long_trips = trip_data.drop(trip_data.loc[trip_data['duration_min'] < 60].index)
```

See how ploting this datasets will look like:

[31]:
```
# first: zoom in
plt.figure(figsize = [15, 5])

# histogram on left: full data with large bin size
plt.subplot(1, 3, 1)
binsize = 1
bin_edges = np.arange(0, short_trips['duration_min'].max()+1, binsize)
plt.hist(data = short_trips, x = 'duration_min', bins = bin_edges, color =␣
↪'steelblue');

## line showing the mean
plt.axvline(short_trips.duration_min.mean(), color='red', linewidth = 2)
min_ylim, max_ylim = plt.ylim()
plt.text(short_trips.duration_min.mean()*1.5, max_ylim*0.5, 'Mean: {:.2f} min'.
↪format(short_trips.duration_min.mean()))

# histogram on right: zoom into trips lasting little more than 1 hour
plt.subplot(1, 3, 2)
binsize2 = 1
bin_edges = np.arange(10, medium_trips['duration_min'].max()+5, binsize2)
plt.hist(data = medium_trips, x = 'duration_min', bins = bin_edges, color =␣
↪'steelblue');

## line showing the mean
plt.axvline(medium_trips.duration_min.mean(), color='red', linewidth = 2)
min_ylim, max_ylim = plt.ylim()
plt.text(medium_trips.duration_min.mean()*1.1, max_ylim*0.5, 'Mean: {:.2f} min'.
↪format(medium_trips.duration_min.mean()))

# histogram on right: zoom into trips lasting little more than 1 hour
plt.subplot(1, 3, 3)
binsize2 = 20
bin_edges = np.arange(0, long_trips['duration_min'].max()+1, binsize2)
plt.hist(data = long_trips, x = 'duration_min', bins = bin_edges, color =␣
↪'steelblue');

## line showing the mean
plt.axvline(long_trips.duration_min.mean(), color='red', linewidth = 2)
min_ylim, max_ylim = plt.ylim()
plt.text(long_trips.duration_min.mean()*1.5, max_ylim*0.5, 'Mean: {:.2f} min'.
↪format(long_trips.duration_min.mean()));
```




These distributions are very skewed. Short tips have a mean of 9.3 minutes, an average of almost half an hour for medium trips and 3 hours for the longest trips. Longer trips could be outliers, it would be useful to inspect their length.

[32]:
```
long_trips.trip_length.describe()
```

[32]:
```
count    16186.000000
mean         1.318647
std          1.775488
min          0.000000
25%          0.000000
50%          0.972941
75%          1.962442
max         68.143976
Name: trip_length, dtype: float64
```

A 0 length could not necessarily be a miscalculation as it could describe a long round trip.

[33]:
```
long_trips.duration_min.quantile(0.95)
```

[33]:
```
839.6541666666667
```

[34]:
```
#Duration of the longest trip
trip_data.loc[(trip_data['duration_min'] == 1439.4833333333333)]
```

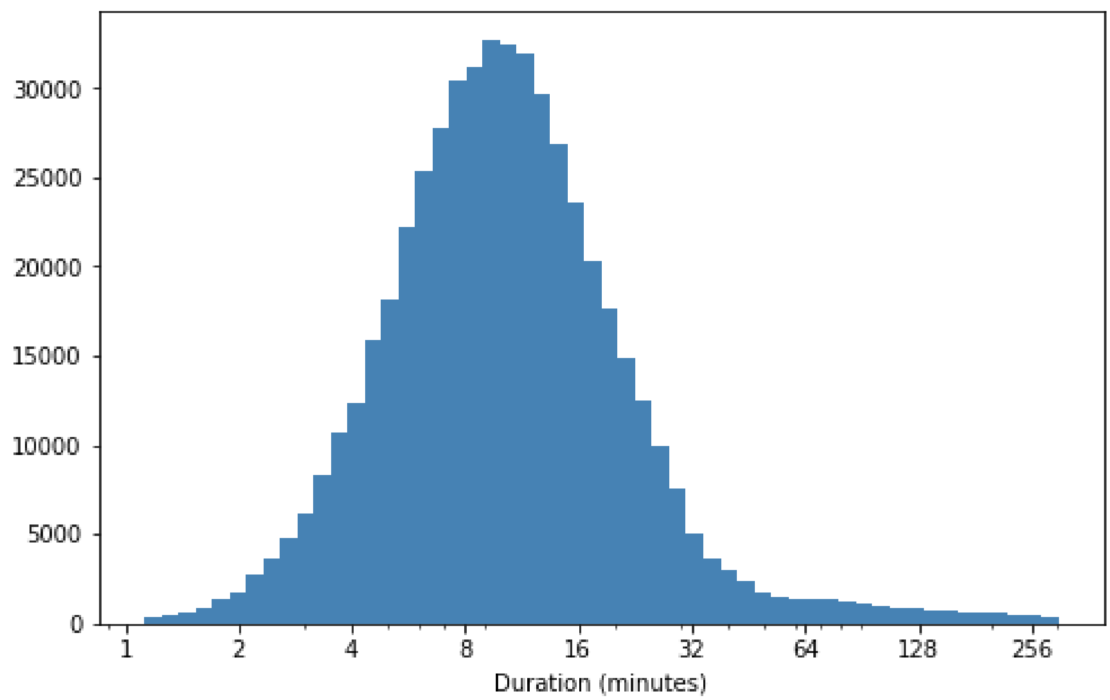
[34]:
```
        duration_sec              start_time                end_time  \
138862         86369 2017-11-12 10:44:32.043 2017-11-13 10:44:01.820

        start_station_id start_station_name  start_station_latitude  \
138862               308    San Pedro Square               37.336802

        start_station_longitude  end_station_id  end_station_name  \
138862               -121.89409             308  San Pedro Square
```

```
        end_station_latitude  end_station_longitude  bike_id user_type  \
138862             37.336802             -121.89409     2231  Customer

        duration_min  trip_length  year  month
138862   1439.483333          0.0  2017     11
```

Inspecting the values of the longest trips of the dataset various things could be inferred. First, that the longest trip could be a misplacement of the bike, or it could have been a very long road trip (since its length is zero) but as sharing systems are intended to serve shorter trips this one will be considered as an invalid observation. Same for trips that last longer than 5 hours as they are not part of the interested population because they do not serve the main purpose of the service.

[35]:
```
# trips lasting more than 5 hours will be dropped
trip_data.drop(trip_data.loc[trip_data['duration_min'] > 300].index, inplace =␣
↪True)
```

[36]:
```
# Re plot the data with the log transformation
log_binsize = 0.045
bins = 10 ** np.arange(0.05, np.log10(trip_data['duration_min'].
↪max())+log_binsize, log_binsize)

plt.figure(figsize=[8, 5])
plt.hist(data = trip_data, x = 'duration_min', bins = bins, color = 'steelblue')
plt.xscale('log')
tick_locs = [1, 2, 4, 8, 16, 32, 64, 128, 256]
plt.xticks(tick_locs, tick_locs)
plt.xlabel('Duration (minutes)');
```




Now the durations display the shape of a normal distribution.

The second variable to be inspected is the trip's length. What is the mean trip length? Is the mean trip length equal to the median and mode?

[37]:
```
# inspection to determine suitable bin sizes
trip_data.trip_length.max()
```

[37]:
```
67.8856258864343
```

[38]:
```
# inspection of the longest trip
trip_data.loc[(trip_data['trip_length'] == 67.8856258864343)]
```

[38]:
```
        duration_sec              start_time                end_time  \
146554          9580 2017-11-10 14:02:14.719 2017-11-10 16:41:55.054

        start_station_id              start_station_name  \
146554               314  Santa Clara St at Almaden Blvd

        start_station_latitude  start_station_longitude  end_station_id  \
146554               37.333988              -121.894902              74

             end_station_name  end_station_latitude  end_station_longitude  \
146554  Laguna St at Hayes St             37.776435            -122.426244
```

```
        bike_id user_type  duration_min  trip_length  year  month
146554     2118  Customer    159.666667    67.885626  2017     11
```

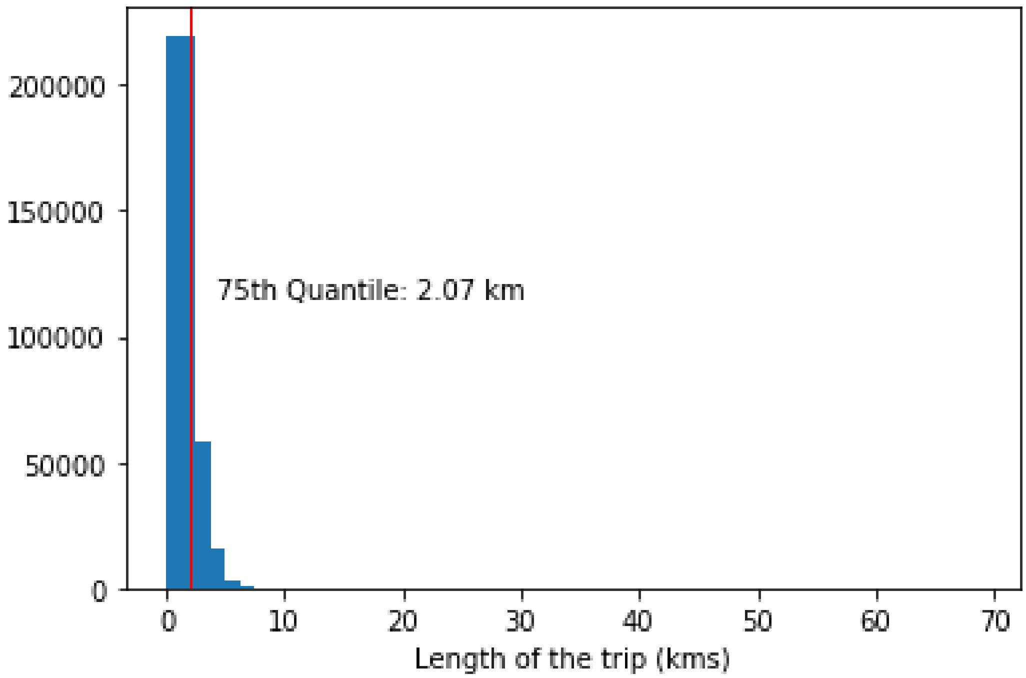
[39]:
```
# inspection to determine suitable bin sizes
trip_data.trip_length.min()
```

[39]: 0.0

[40]:
```
# plot with seaborn
sb_colors = sb.color_palette()[3] #set colors
sb.countplot(data = trip_data, x = 'trip_length', color = sb_colors);
plt.axis('off');
```

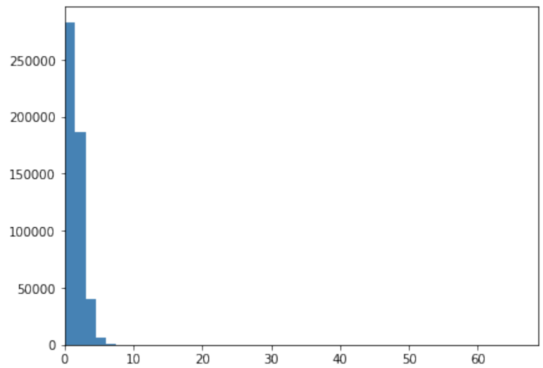
This visualization is barely visible

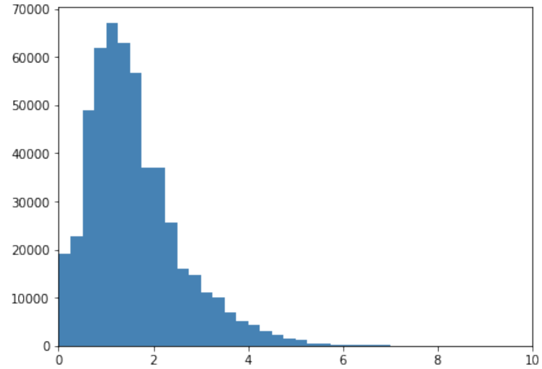
[41]:
```
# plot histogram
binsize = 1.25
bins = np.arange(0, trip_data['trip_length'].max()+binsize, binsize)
plt.hist(data = trip_data, x = 'trip_length', bins = bins)
plt.xlabel('Length of the trip (kms)')

# bar to show where the 75th quantile of the values is
plt.axvline(trip_data.trip_length.quantile(0.75), color='red', linewidth = 1)
min_ylim, max_ylim = plt.ylim()
plt.text(trip_data.trip_length.quantile(0.75)*2, max_ylim*0.5, '75th Quantile:␣
 ↪{:.2f} km'.format(trip_data.trip_length.quantile(0.75)));
```




The graph shows that most trips have very short lengths and very few trips span up to 60 kilometers and more. It could be useful to zoom where the bulk of data is:

[42]:
```
# first: zoom in
plt.figure(figsize = [15, 5])

# histogram on left: full data with large bin size
plt.subplot(1, 2, 1)
binsize = 1.5
bin_edges = np.arange(0, trip_data['trip_length'].max()+1, binsize)
plt.hist(data = trip_data, x = 'trip_length', bins = bin_edges, color =␣
 ↪'steelblue')
plt.xlim(0, trip_data['trip_length'].max()+1);

# histogram on right: zoom into trips lasting little more than 1 hour
plt.subplot(1, 2, 2)
binsize2 = 0.25
bin_edges = np.arange(0, 70+binsize2, binsize2)
plt.hist(data = trip_data, x = 'trip_length', bins = bin_edges, color =␣
 ↪'steelblue')
plt.xlim(0, 10);
```



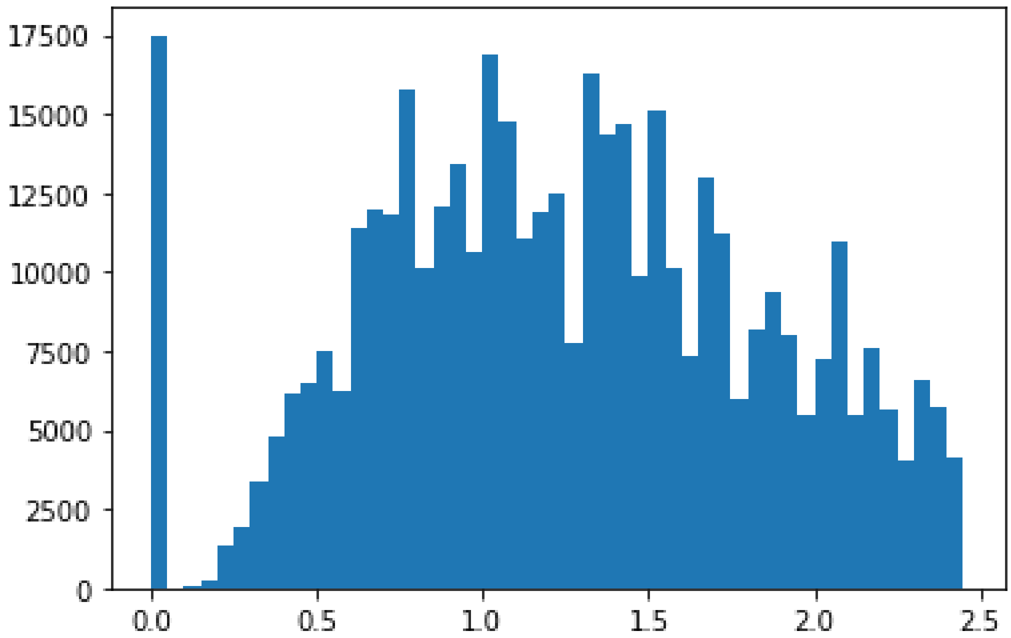




[43]:
```
# inspect quantiles
trip_data['trip_length'].describe()
```

[43]:
```
count    516758.000000
mean          1.587049
std           1.002314
min           0.000000
25%           0.903003
50%           1.400935
75%           2.071193
max          67.885626
Name: trip_length, dtype: float64
```

There's a very long tail of trip's lengths. Having zoomed in on trips lasting less than 2 km, so that a smaller bin size could be used, gave a more detailed look at the main data distribution.

[44]:
```
# plot histogram of the 75th percentile

bin_edges = np.arange(0, 2.5, 0.05); #Set wide bin edges
plt.hist(data = trip_data, x = 'trip_length', bins = bin_edges);
```




[45]:
```
# inspect mode
trip_data['trip_length'].mode()
```

[45]:
```
0    0.0
dtype: float64
```

As explained before, trips with length 0 could be errors or round trips, these values constitute the mode. Unfortunately, they add noise to the observations.

[46]:
```
trip_data.loc[(trip_data['trip_length'] == 0)].describe()
```

[46]:
```
       duration_sec  start_station_id  start_station_latitude  \
count  17452.000000      17452.000000            17452.000000
mean    2751.670238        121.627607               37.755065
std     3236.200930         99.735672                0.133730
min       61.000000          3.000000               37.317298
25%      649.000000         24.000000               37.773311
50%     1664.000000         97.000000               37.789677
75%     3435.250000        197.000000               37.804770
max    17997.000000        338.000000               37.880222

       start_station_longitude  end_station_id  end_station_latitude  \
count             17452.000000    17452.000000          17452.000000
mean               -122.318574      121.627607             37.755065
std                   0.149014       99.735672              0.133730
min                -122.444293        3.000000             37.317298
```

```
25%                 -122.408445       24.000000            37.773311
50%                 -122.394203       97.000000            37.789677
75%                 -122.265192      197.000000            37.804770
max                 -121.874119      338.000000            37.880222

       end_station_longitude       bike_id  duration_min  trip_length  \
count           17452.000000  17452.000000  17452.000000      17452.0
mean             -122.318574   1562.789251     45.861171          0.0
std                 0.149014    938.869629     53.936682          0.0
min              -122.444293     10.000000      1.016667          0.0
25%              -122.408445    732.000000     10.816667          0.0
50%              -122.394203   1514.000000     27.733333          0.0
75%              -122.265192   2358.000000     57.254167          0.0
max              -121.874119   3730.000000    299.950000          0.0

          year         month
count  17452.0  17452.000000
mean    2017.0      9.408320
std        0.0      1.617981
min     2017.0      6.000000
25%     2017.0      8.000000
50%     2017.0      9.000000
75%     2017.0     11.000000
max     2017.0     12.000000
```

Trips with length zero are 17 thousand and have a mean duration of 45 minutes which is reasonable. On the other hand, trips with length 0 which last less than 3 minutes could be errors or non-initialized trips, because this duration does seem to justify a round trip.

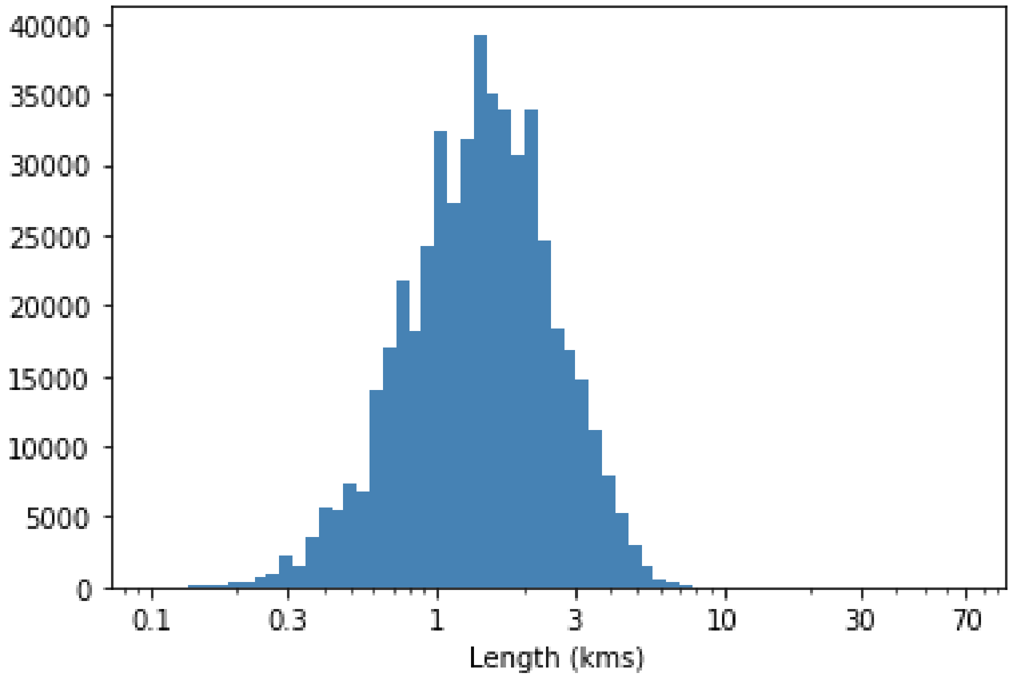
[47]:
```python
# identification of trips with zero length and zero duration
outliers = trip_data.loc[(trip_data['duration_min'] < 3) &␣
 ↪(trip_data['trip_length'] == 0)]
len(outliers)
```

[47]: 2056

[48]:
```python
# trips lasting less than 3 minutes with length 0
trip_data.drop(trip_data.loc[(trip_data['duration_min'] < 3) &␣
 ↪(trip_data['trip_length'] == 0)].index, inplace = True)
```

[49]:
```python
# re plot the data with log transformation
log_binsize = 0.045
bins = 10 ** np.arange(-1, np.log10(trip_data['trip_length'].
 ↪max())+log_binsize, log_binsize)
plt.hist(data = trip_data, x = 'trip_length', bins = bins, color = 'steelblue')
plt.xscale('log')
```

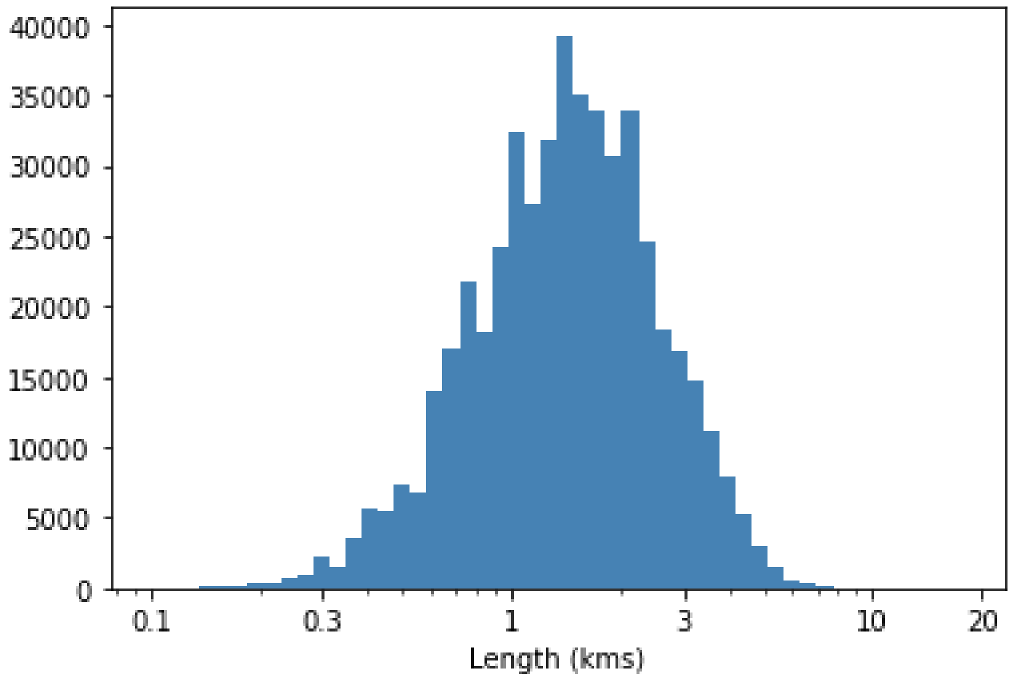
```
tick_locs = [0.1, 0.3, 1, 3, 10, 30, 70]
plt.xticks(tick_locs, tick_locs)
plt.xlabel('Length (kms)');
```




The distribution of the length values seems uniform with two tails. First values closer to zero and those longer than 30 kilometers. These two tales are a bit problematic. These could be valid observations from real trips but on the one hand, trips with value zero as seen do not have the right distances as they could be round trips or non-initiated trips (either way, something is wrong). On the other hand, trips lasting longer than 1 hour are not the main interest of the analysis.

[50]:
```
# new dataset without long distance trips
trip_data2 = trip_data.drop(trip_data.loc[(trip_data['trip_length'] > 20)].
 ↪index)
```

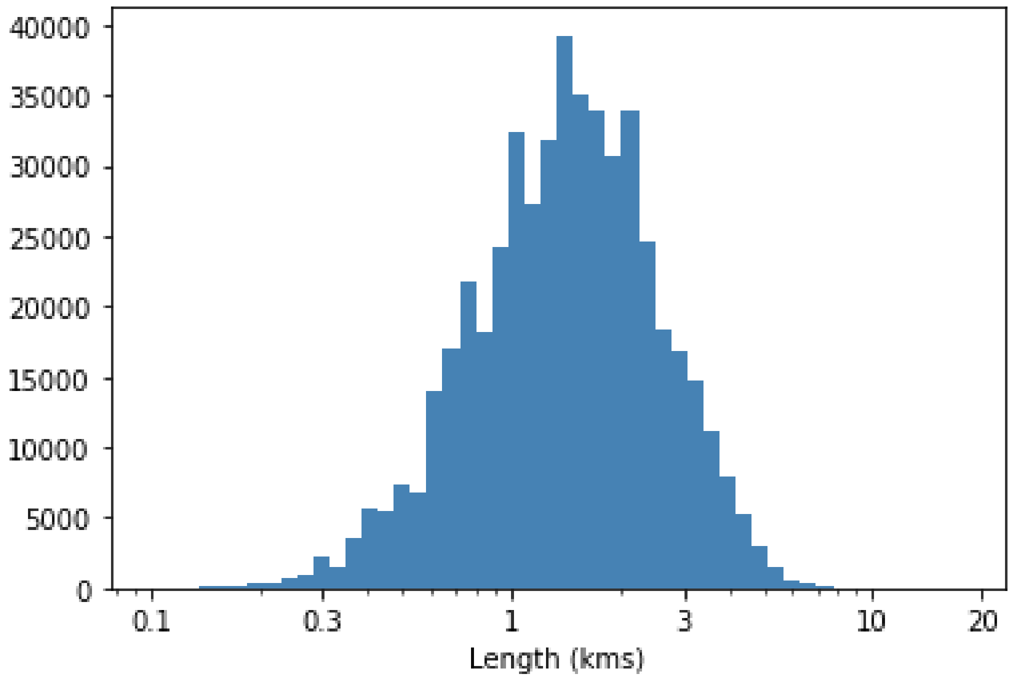
[51]:
```
# Re plot the data with the log transformation
log_binsize = 0.045
bins = 10 ** np.arange(-1, np.log10(trip_data2['trip_length'].
 ↪max())+log_binsize, log_binsize)
plt.hist(data = trip_data2, x = 'trip_length', bins = bins, color = 'steelblue')
plt.xscale('log')
tick_locs = [0.1, 0.3, 1, 3, 10, 20]
plt.xticks(tick_locs, tick_locs)
plt.xlabel('Length (kms)');
```




This plot describes a normal distribution.

[52]:
```
# new dataset without confusing zero length trips
trip_data3 = trip_data2.drop(trip_data.loc[(trip_data['trip_length'] == 0)].
 ↪index)
```

[53]:
```
# Re plot the data with the log transformation
log_binsize = 0.045
bins = 10 ** np.arange(-1, np.log10(trip_data3['trip_length'].
 ↪max())+log_binsize, log_binsize)
plt.hist(data = trip_data3, x = 'trip_length', bins = bins, color = 'steelblue')
plt.xscale('log')
tick_locs = [0.1, 0.3, 1, 3, 10, 20]
plt.xticks(tick_locs, tick_locs)
plt.xlabel('Length (kms)');
```




This also depicts a normal distribution.

Next inspection will be carried out along categorical variables. Are there common stations among trips?

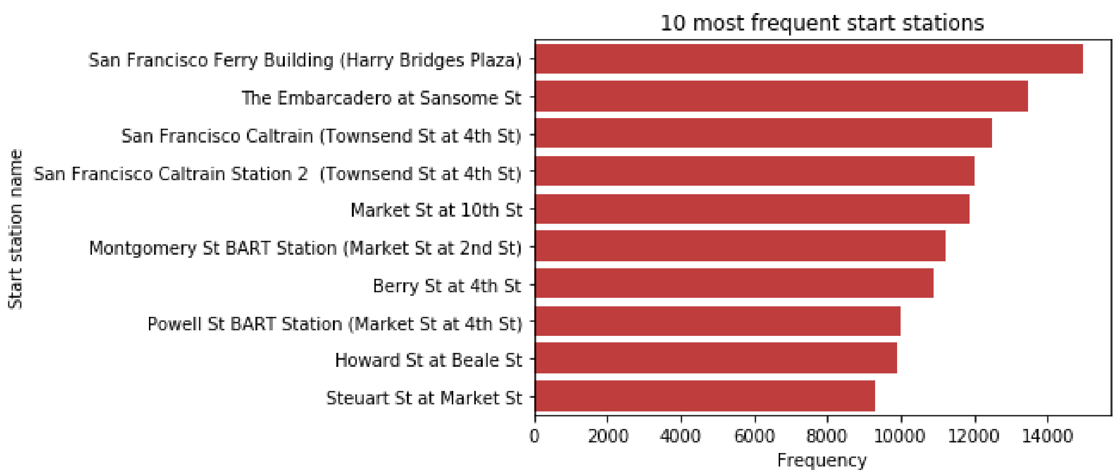
First stations Id's and names and their frequency:

[54]:
```
# inspection of categorical variables
trip_data.start_station_id.value_counts()
trip_data.end_station_id.value_counts()
```

[54]:
```
30     17324
15     16909
6      16279
67     13622
21     13378
       …
294        7
340        4
339        3
292        2
268        2
Name: end_station_id, Length: 272, dtype: int64
```

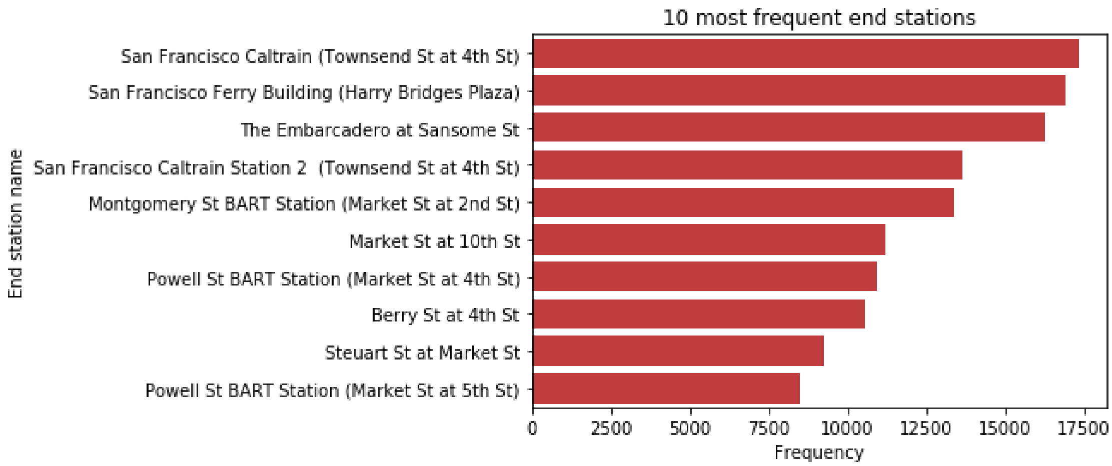
[55]:
```
# first the 10 most frequent stations
# n is the number of stations to retrieve
```

```
n = 10

# Create New variable for common stations

freq_St = trip_data.start_station_name.value_counts()[:n].index.tolist()
Top_start_stations = trip_data[trip_data['start_station_name'].isin(freq_St)]

default_color = sb.color_palette()[3];
order1 = Top_start_stations.start_station_name.value_counts().index;
sb.countplot(data = Top_start_stations, y = 'start_station_name', color =␣
↪default_color, order = order1);
plt.ylabel('Start station name');
plt.xlabel('Frequency');
plt.title('10 most frequent start stations');
```




[56]:
```
# first the 10 most frequent stations
# n is the number of stations to retrieve
n = 10

# Create New variable for common stations

freq_St = trip_data.end_station_name.value_counts()[:n].index.tolist()
Top_end_stations = trip_data[trip_data['end_station_name'].isin(freq_St)]

default_color = sb.color_palette()[3];
order1 = Top_end_stations.end_station_name.value_counts().index;
sb.countplot(data = Top_end_stations, y = 'end_station_name', color =␣
↪default_color, order = order1);
plt.ylabel('End station name');
plt.xlabel('Frequency');
```

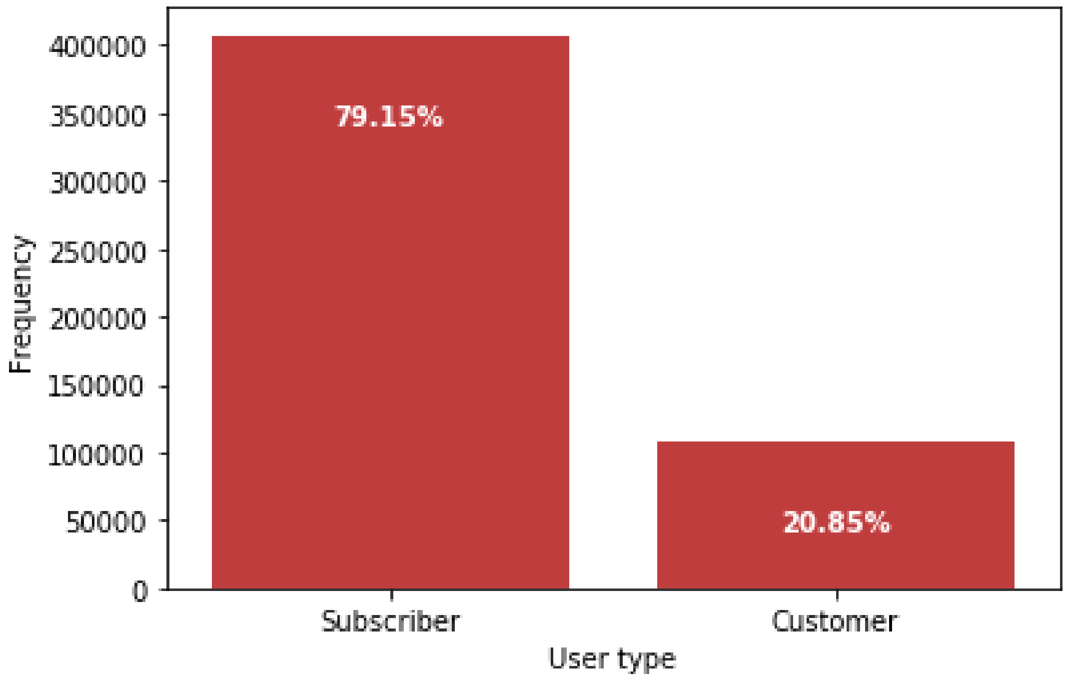
```
plt.title('10 most frequent end stations');
```




The top three stations are the most famous for ending and starting a trip.

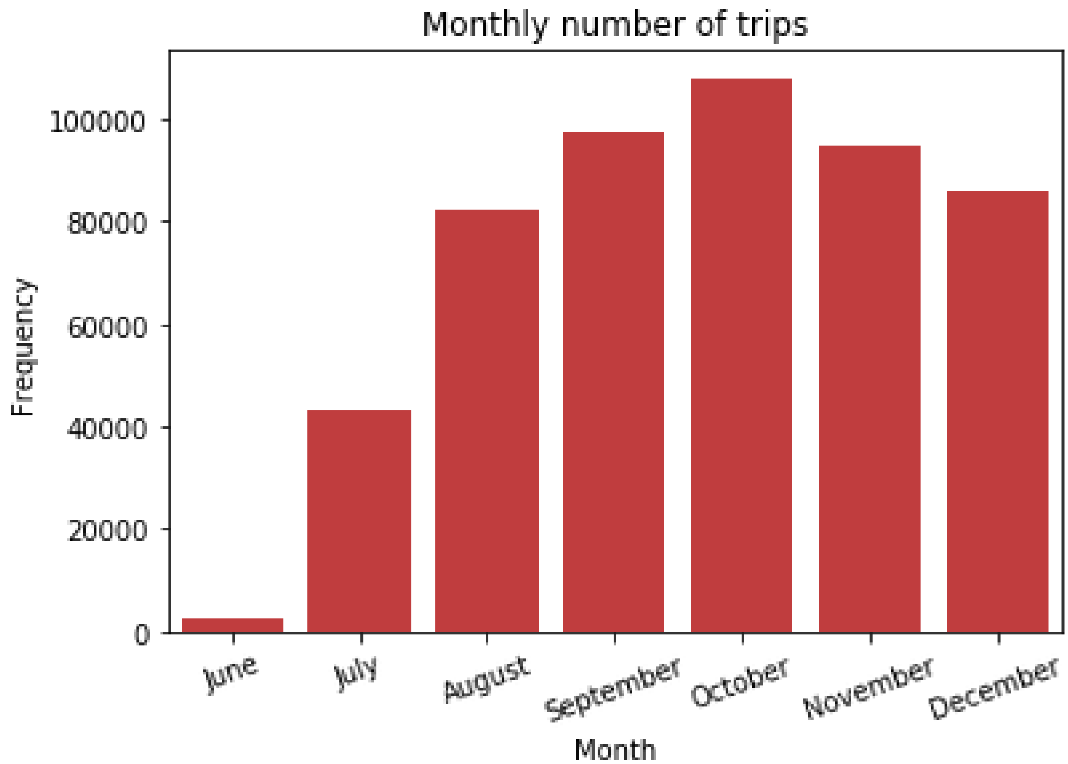
Type of user. Users are either Subscribers (members of the service) or Customer (casual users). Which user goes on more trips?

[57]:
```
# plot for the Type of user
default_color = sb.color_palette()[3];
order1 = trip_data.user_type.value_counts().index;
sb.countplot(data = trip_data, x = 'user_type', color = default_color);
plt.xlabel('User type');
plt.ylabel('Frequency');


# add text of relative value
total = trip_data.shape[0]
type_counts = trip_data['user_type'].value_counts()
locs, labels = plt.xticks()


for loc, label in zip(locs, labels):
    count = type_counts[label.get_text()]
    perc_string = '{:0.2f}%'.format(100*count/total)

    plt.text(loc, count-50000, perc_string, va= 'top', ha = 'center', color =␣
 ↪'white', weight = 'bold')
```




The number of subscriber users is almost 4 times the number of customers of the bike sharing service.

[58]:
```
# plot for the month of the trip (absolute frequency)
sb.countplot(data = trip_data, x = 'month', color = default_color);
plt.title('Monthly number of trips');
plt.xlabel('Month');
plt.ylabel('Frequency');
plt.xticks(np.arange(7), ['June', 'July', 'August', 'September', 'October',␣
 ↪'November', 'December'], rotation=20);
```




October is the month with more trips. It is strange that only these months are present and not the whole year.

#### 1.1.5 Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

The quantitative variables, trip duration and length had very long right tails indicating that short and closer trips were the most frequent by far. Some trips lasted even over 20 hours which is not an expected observation for this kind of service, not even trips longer than 5 hours. These observations made the population seem as heavy-tailed distribution, both for duration and length. For durations (in minutes), first I zoomed in the bulk of the data; these zooming areas described right skewed distributions as well. After dropping the values with durations longer than what these trips could last and plot the variable in a log scale, the histogram described a normal distribution with mean higher than the median and the mode. For the distances (in kilometers) the same process was carried out, as this variable also took on a large range of values, zooming and plotting in a log scale.

### 1.1.6 Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

As mentioned in the previous section, plotting the quantitative variables with linear scales showed highly skewed distributions. Some abnormal points made the data seem to have unusual distributions. For the duration variable, as short, medium and longer trips were interesting in their own way, because they describe different purposes of the service, three data frames holding these differences were created. This changed the form of the data as three datasets were created from the original one. This was done in order to better analyze the distribution of each without their influences on each other.

## Bivariate Exploration

This examination sought to find relationships between the most relevant variables in the dataset by means of different kinds of plots.

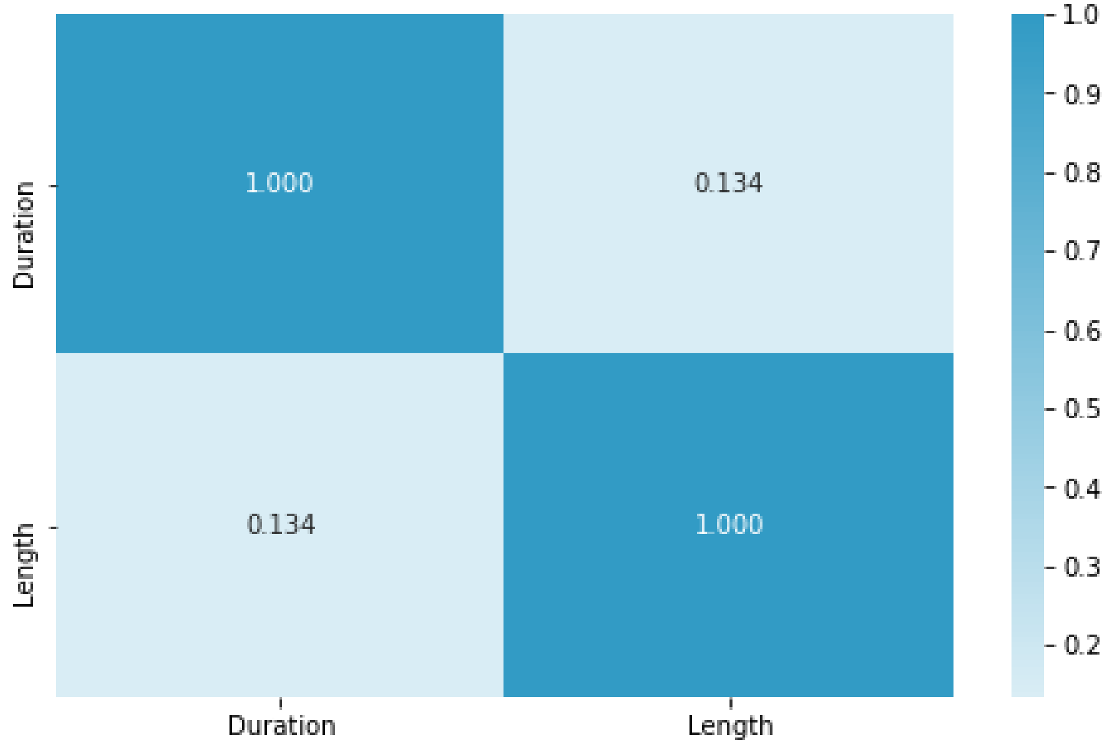
First, inspect the relationship between quantitative values, duration and length. First by means of their correlation:

[59]:
```
# verify existing correlation
quant_vars = ['trip_length', 'duration_min']
trip_data[quant_vars].corr()
```

[59]:

| | trip_length | duration_min |
|---|---|---|
| trip_length | 1.000000 | 0.133952 |
| duration_min | 0.133952 | 1.000000 |

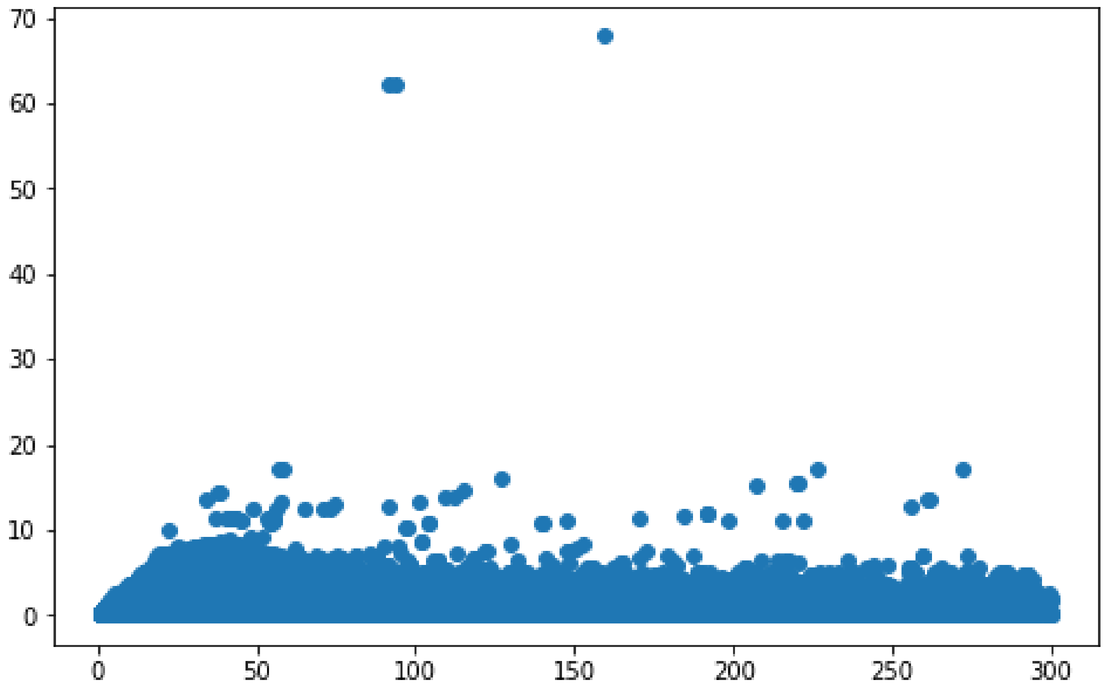
[60]:
```
# plot of the previous findings
figsize=(15, 10)
quant_vars = ['trip_length', 'duration_min']

# correlation Matrix
plt.figure(figsize = [8, 5])
cmap = sb.diverging_palette(0, 230, 90, 60, as_cmap = True) #colors
sb.heatmap(trip_data[quant_vars].corr(), annot = True, fmt = '.3f',cmap = cmap,␣
 ↪center = 0)

#labels
xticks_labels = ['Duration', 'Length']
plt.xticks(np.arange(2) + .5, labels=xticks_labels)
plt.yticks(np.arange(2) + .5, labels=xticks_labels)
plt.show()
```




The correlation coefficient between duration and length is lower than expected. This is surprising as the duration of the trip depends on its length. These values could have been affected by the flaws that we have identified earlier in the data.

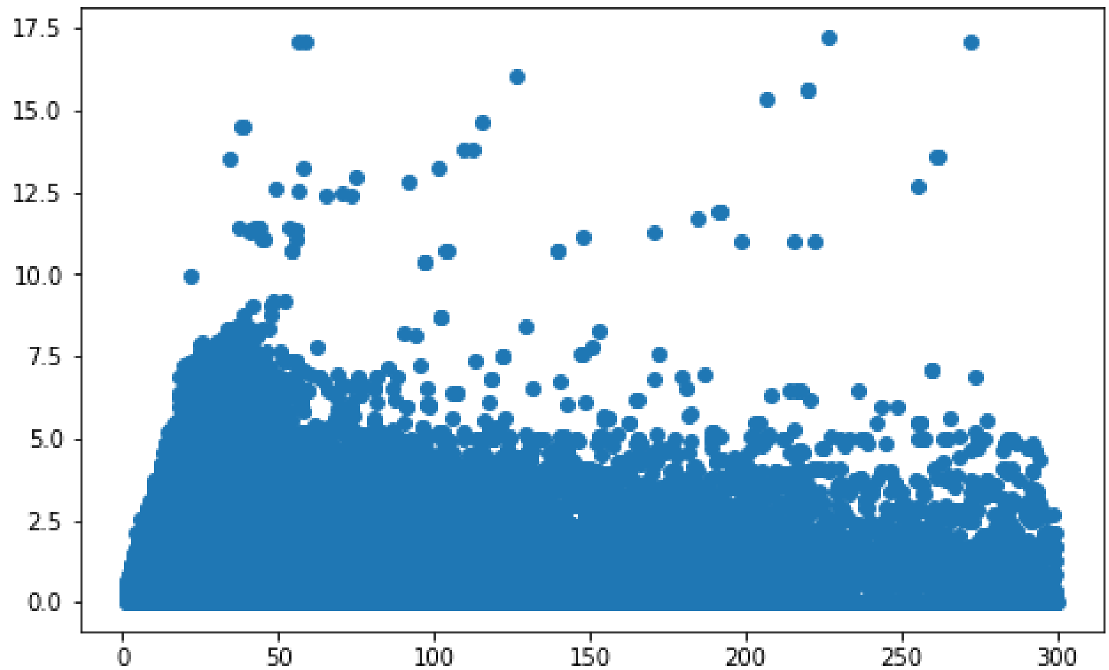
This correlation is inspected further with a scatter plot

[61]:
```
# scatter plot
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data, x = 'duration_min', y = 'trip_length');
```



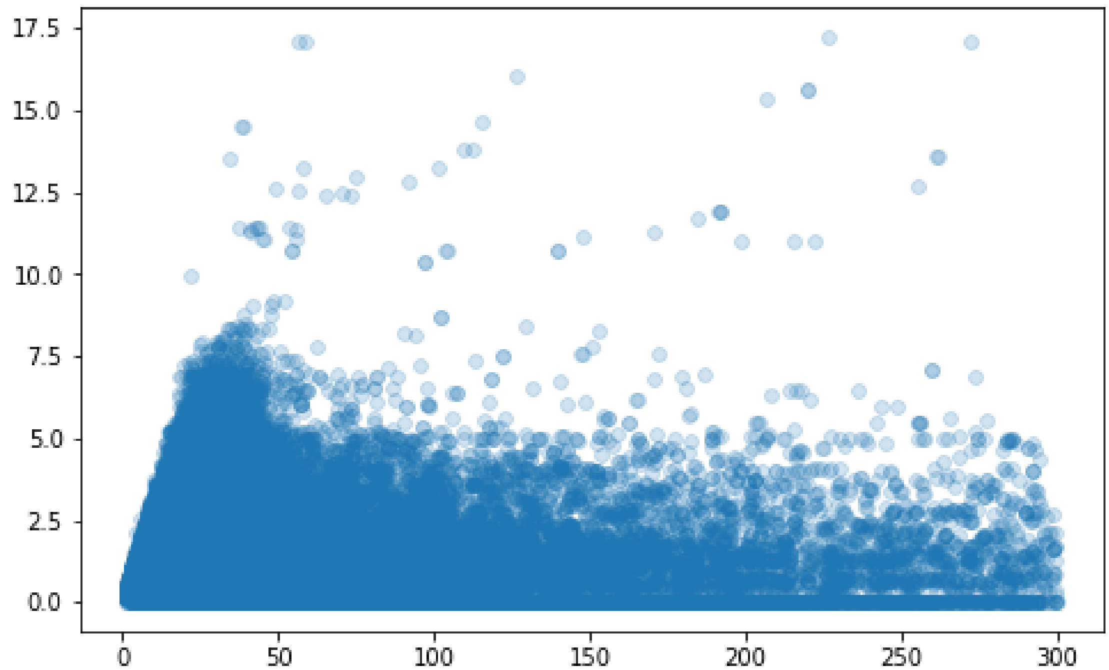

It is very difficult to infer anything from this. First, I will try plotting the dataset without particularly long trips:

[62]:
```
# scatter plot of sample data
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data2, x = 'duration_min', y = 'trip_length');
```

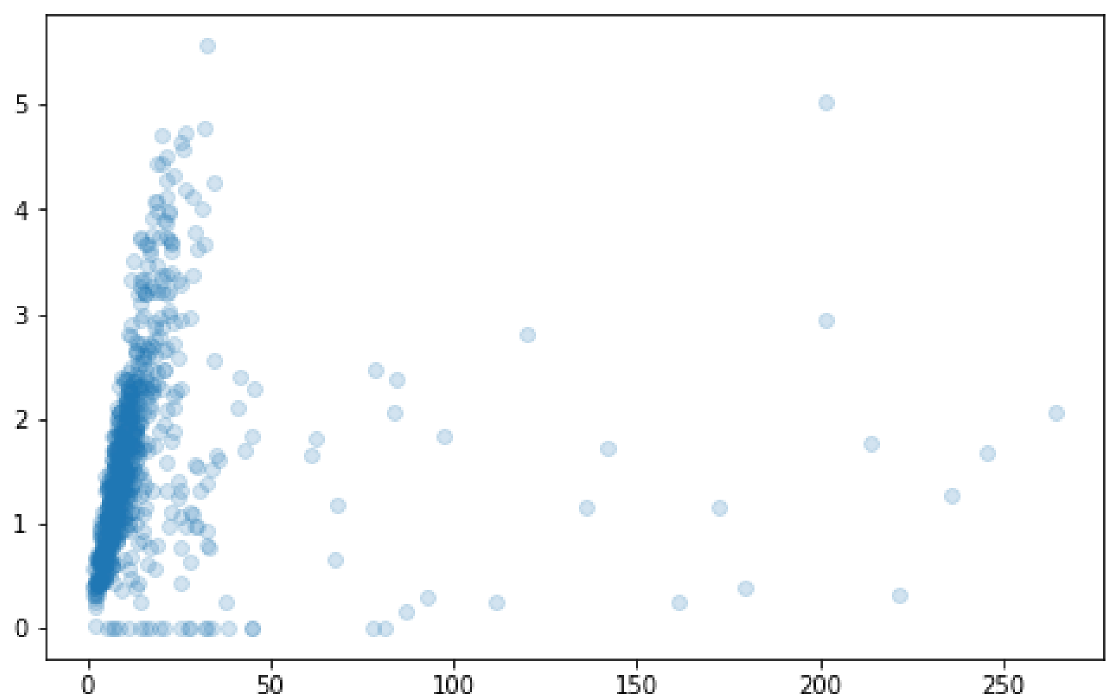



Better looking but this dataset holds too many values. before trying with a sample, the transparency of the set will be adjusted:

[63]:
```
# adding transparency:
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data2, x = 'duration_min', y = 'trip_length', alpha = 1/
 ↪5);
```

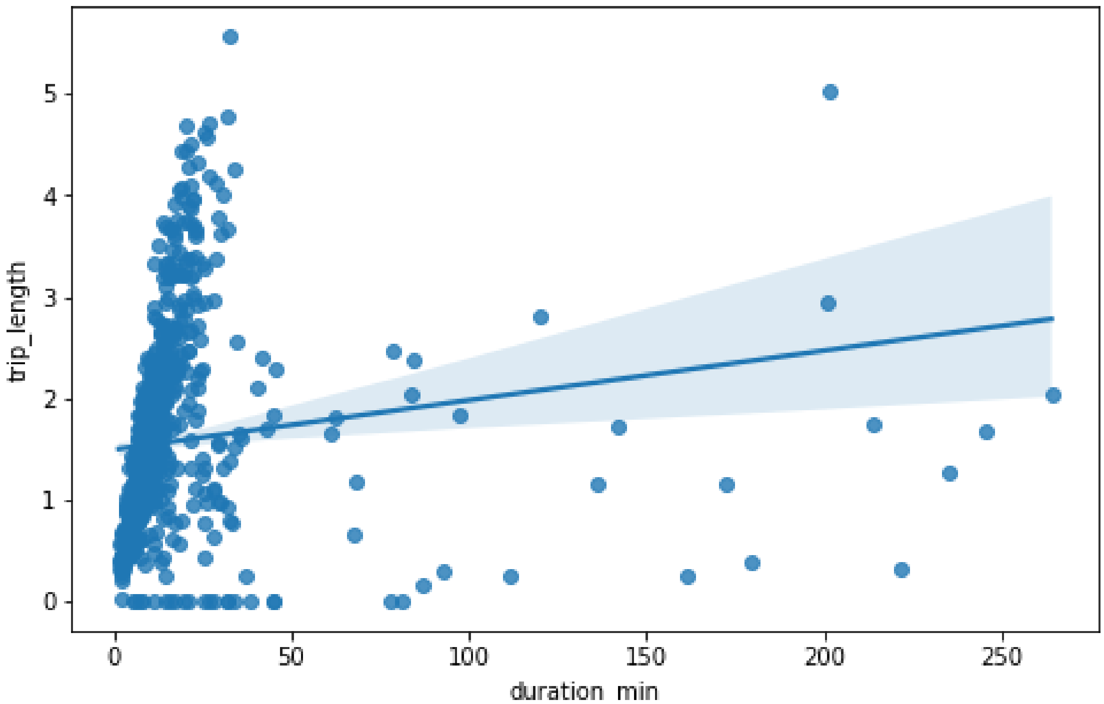



[64]:
```
# plotting a sample:
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data2.sample(800, random_state = 3), x =␣
 ↪'duration_min', y = 'trip_length', alpha = 1/5);
```



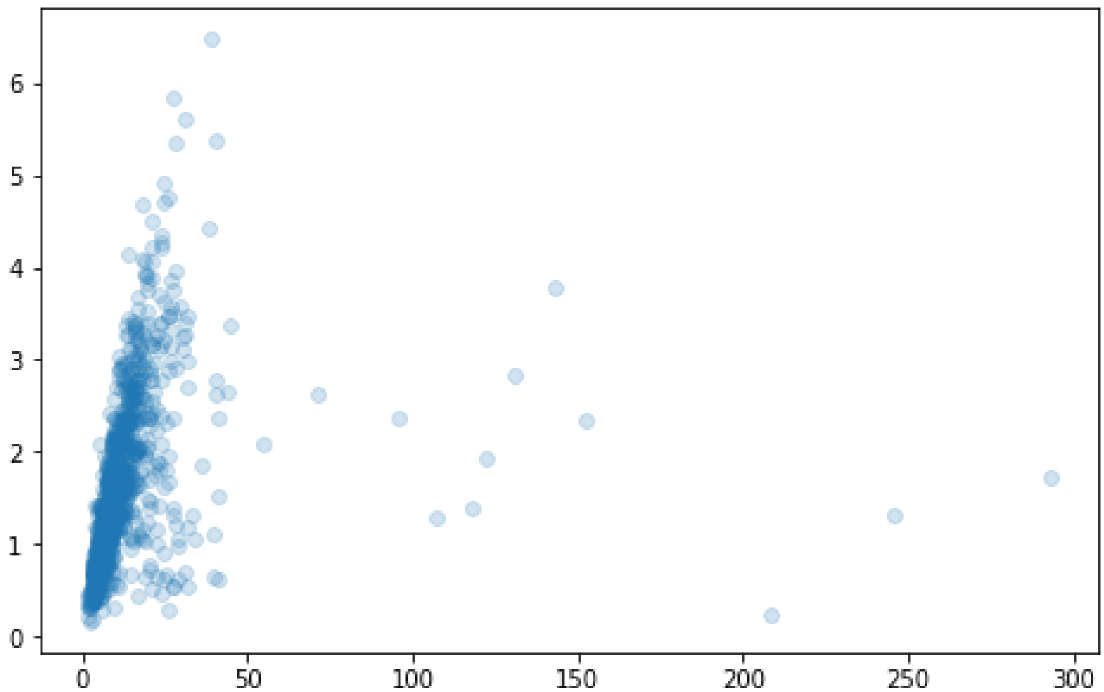
Now there seems to be strong positive relationship between length and duration as higher values of the x axis are associated with increasing values of the y axis.

[65]:
```
# plot with seaborn
plt.figure(figsize = [8, 5])
sb.regplot(data = trip_data2.sample(800, random_state = 3), x = 'duration_min',␣
↪y = 'trip_length');
```



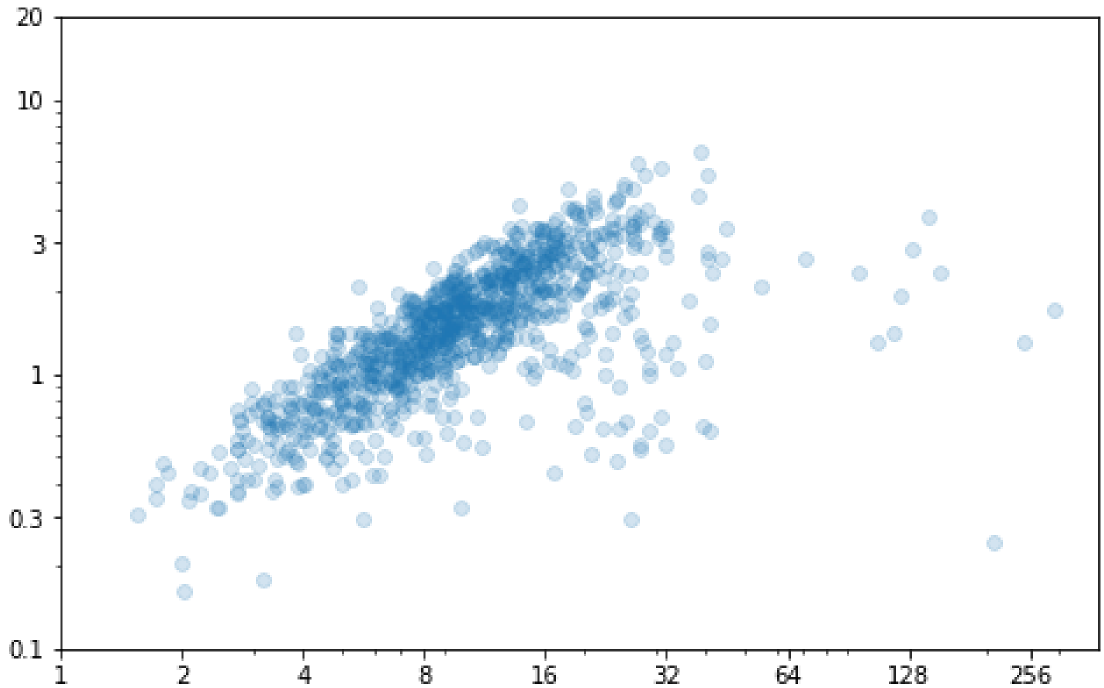

Plotting the trimmed dataset:

[66]:
```
# plotting a sample of the trimmed data
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data3.sample(1000, random_state = 4), x =␣
↪'duration_min', y = 'trip_length', alpha = 1/5);
```




This plot shows the relationship between the duration and length. These are all highly positively correlated, as expected. The longer the trip it is expected to take longer, still, variations in the shape and performance of cyclists could alter this dependency.

[67]:
```
# log-log plot
# one of them correlates with price.
plt.figure(figsize = [8, 5])
plt.scatter(data = trip_data3.sample(1000, random_state = 4), x =␣
↪'duration_min', y = 'trip_length', alpha = 1/5);
plt.xscale('log')
tick_locsx = [1, 2, 4, 8, 16, 32, 64, 128, 256]
plt.xticks(tick_locsx, tick_locsx)
plt.yscale('log')
tick_locs = [0.1, 0.3, 1, 3, 10, 20]
plt.yticks(tick_locs, tick_locs);
```



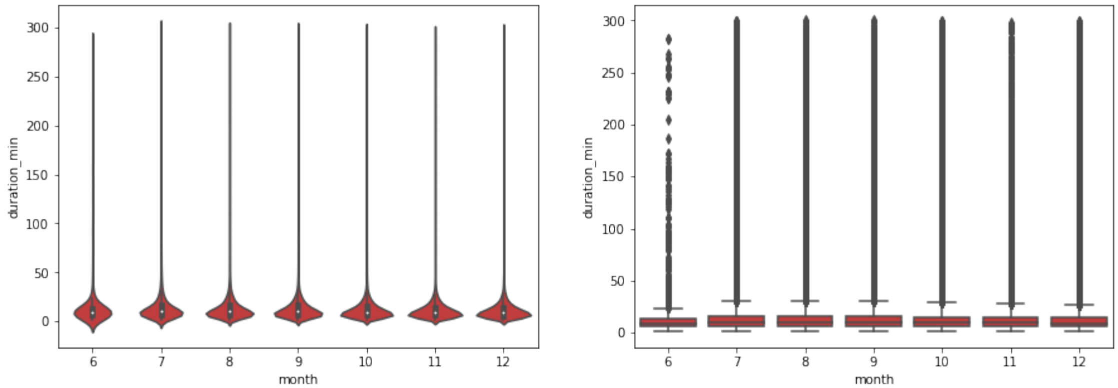

> This plot describes a strong positive correlation. As both variables take on large ranges of values, longer durations are associated with increasing values of length. Usually time is always the independent variable, but in this case, I think it could also be a dependent one.

The next plot will inspect the way months relate to duration and lengths. Various plots will depict the relationship in order to see which one provides more insights:

> First how the months of the year relate to the duration of the trips. First using normal duration values, then plots with log values and their impact on both short and medium trips.

[68]:
```
# normal plot with all values included
# figure size
fig, ax = plt.subplots(figsize=(15,5))


# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data, x = 'month', y = 'duration_min', color =␣
↪sb_colors);


# box plot on the right
```

```
plt.subplot(1, 2, 2)
sb.boxplot(data = trip_data, x = 'month', y = 'duration_min', color = sb_colors);
```

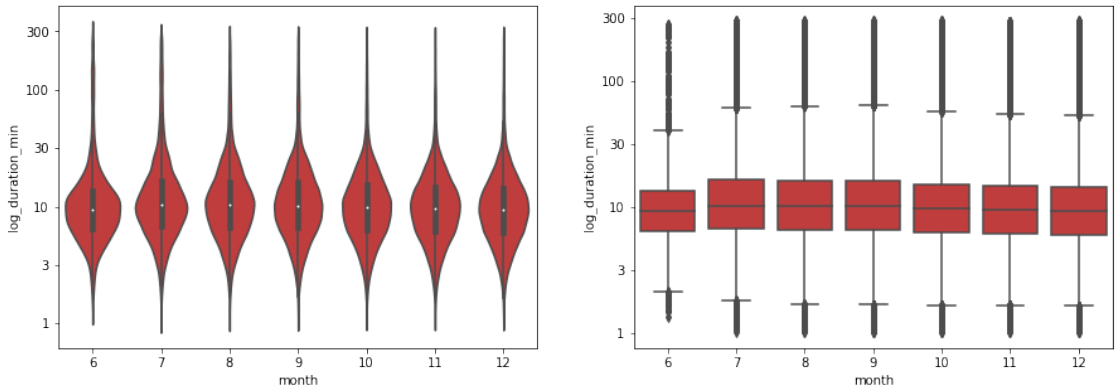



The plot seems to have too long lines/whiskers, one walkaround would be plotting the log transformed values:

[69]:
```
# function to get variables of plot and ticks
def log_trans(x, inverse = False):
    """ quick function for computing log and power operations """
    if not inverse:
        return np.log10(x)
    else:
        return np.power(10, x)

trip_data['log_duration_min'] = trip_data['duration_min'].apply(log_trans)
```

[70]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))



# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data, x = 'month', y = 'log_duration_min', color = sb_colors);
tick_locsx = [0, 0.5, 1, 1.5, 2, 2.5]
tick_labels = [1, 3, 10, 30, 100, 300]
plt.yticks(tick_locsx, tick_labels)

# box plot on the right
plt.subplot(1, 2, 2)
```

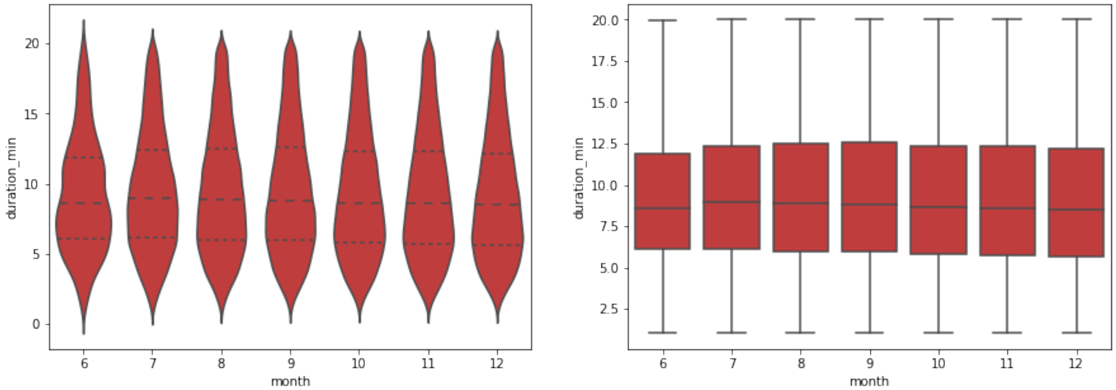
```
sb.boxplot(data = trip_data, x = 'month', y = 'log_duration_min', color =
 ↪sb_colors);
tick_locsx = [0, 0.5, 1, 1.5, 2, 2.5]
tick_labels = [1, 3, 10, 30, 100, 300]
plt.yticks(tick_locsx, tick_labels);
```




These plots are a better representation of the relationship between durations and months. They show little variation between medians for all months with many values outside the interquartile range. As previously mentioned most trips lasted around 10 minutes and that is consistent pretty much for all months. Violin plots are very comparable with values around the median and quantiles and long tails in both sides. They show that for all months trips lasting from 9 to 11 minutes are the most common ones, unlike trips that last less than 1 minute or more than 30 which are the least probable ones.

Next, plots of shorter, medium and longer trips separated. Perhaps these could provide better insights on their distribution along these months:

[71]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))



# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = short_trips, x = 'month', y = 'duration_min', color =
 ↪sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = short_trips, x = 'month', y = 'duration_min', color =
 ↪sb_colors);
```



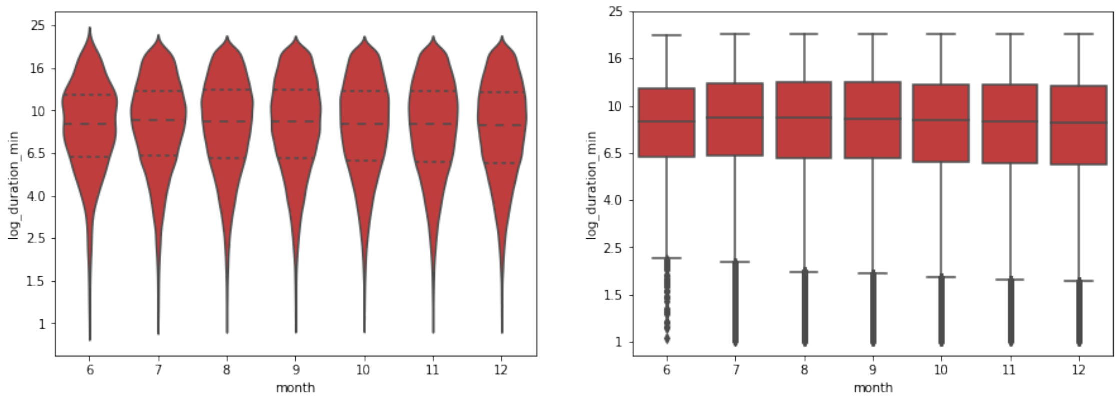

These values represent the distribution of shorter trips from June to December. The relationship between variables seem consistent as the shape of the plots does not vary a lot from month to month (except for June but not so much). It seems as if July holds the higher median (but they are all very similar). In the same way the violin plot displays analogous shapes. June and December show the smallest medians for all months (by little). I was expecting more variations due to the changes in weather between these months specially between August and December.

This plot will be remade using log values of the distances, to verify if they provide any interesting insights:

[72]:
```
# function to get variables of plot and ticks
def log_trans(x, inverse = False):
    """ quick function for computing log and power operations """
    if not inverse:
        return np.log10(x)
    else:
        return np.power(10, x)

short_trips['log_duration_min'] = short_trips['duration_min'].apply(log_trans)
```

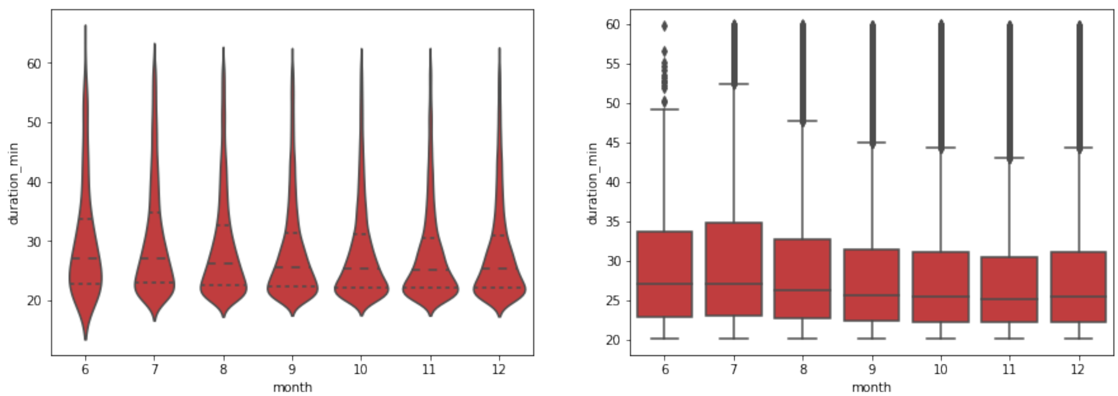
[73]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))



# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = short_trips, x = 'month', y = 'log_duration_min', color =␣
↪sb_colors, inner = 'quartile');
tick_locsx = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4]
tick_labels = [1, 1.5, 2.5, 4.0, 6.5, 10, 16, 25]
plt.yticks(tick_locsx, tick_labels)
```

```
# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = short_trips, x = 'month', y = 'log_duration_min', color =␣
↪sb_colors);
tick_locsx = [0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4]
tick_labels = [1, 1.5, 2.5, 4.0, 6.5, 10, 16, 25]
plt.yticks(tick_locsx, tick_labels);
```




Like the plots made with normal values- as oposed to log ones- this one shows similar shapes for the durations. As before medians and quantiles are very similar in value. The violin plot shows very simmilar distributions and few outliers for August, November and december. September and October show very regular distributions with most values closer to the median and tails expanding both sides. As this did not provide specially useful insights the rest of vairables will be ploted with the normal values.

[74]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))


# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = medium_trips, x = 'month', y = 'duration_min', color =␣
↪sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = medium_trips, x = 'month', y = 'duration_min', color =␣
↪sb_colors);
```

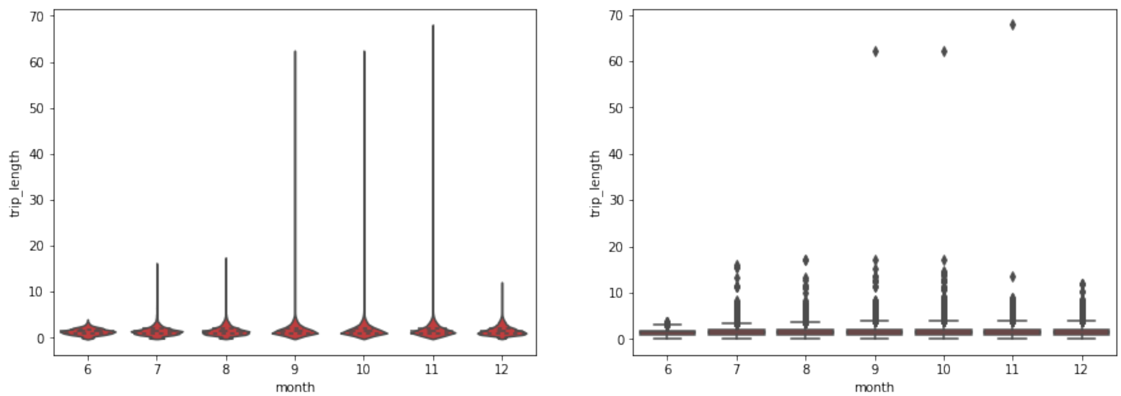



Medium duration trips show more variations between months than short ones. Still, these variations are related to the third quartile (Q3) because their medians are also very similar, with trips lasting between 25 and 30 minutes, the most common ones. As normal for a dataset obtained with a minimum value of 20 minutes, tails are longer towards longer rides. In general, it seems as if longer trips were held in July, but the longest one took place in June.

Longer trips are not to be inspected as they are not of special interest for the analysis.

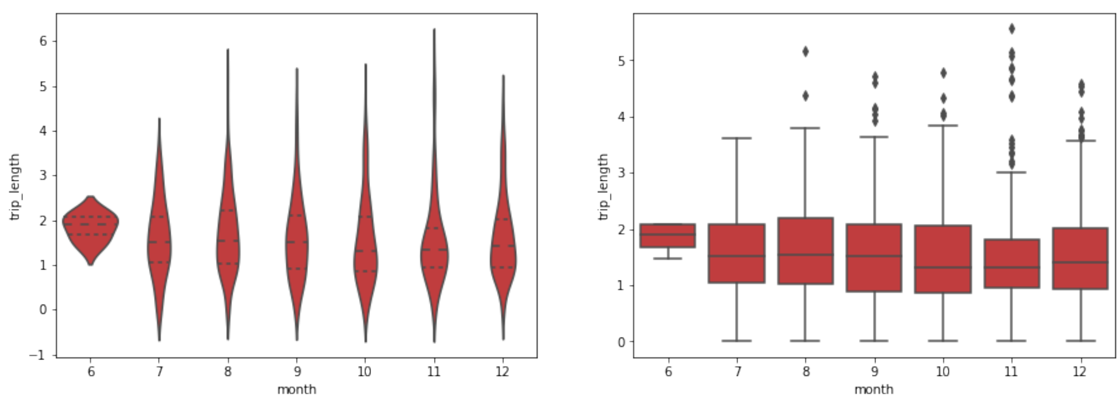
It is time to inspect the relationship between the months of the year and the length of the trips. > The first plot with show all values:

[75]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))

# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data, x = 'month', y = 'trip_length', color =␣
↪sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = trip_data, x = 'month', y = 'trip_length', color = sb_colors);
```




Sample of the values:

[76]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))

# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data.sample(1000, random_state = 3), x = 'month', y =␣
↪'trip_length', color = sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = trip_data.sample(1000, random_state = 3), x = 'month', y =␣
↪'trip_length', color = sb_colors);
```

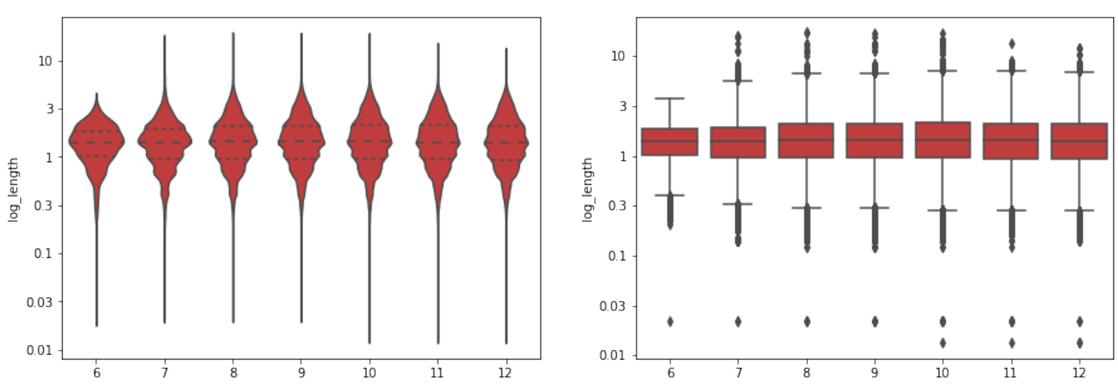



This plot shows that, in average, farther trips took place in June. Months from July to December hold values that are very similar. October and November have almost equal medians which are trips lasting almost 1.5 km. The rest of the months also show slight differences among their medians. The violin plots show how concentrated values are for June and more distributed and analogous for the rest of the months. This means that many trips had wide variety of durations from July to December.

Following the plot with log-scales:

[77]:
```
# function to get variables of plot and ticks
def log_trans(x, inverse = False):
    """ quick function for computing log and power operations """
    if not inverse:
        return np.log10(x)
    else:
        return np.power(10, x)

# The dataset with trimmed lengths will be used
trip_data3['log_length'] = trip_data3['trip_length'].apply(log_trans)
```

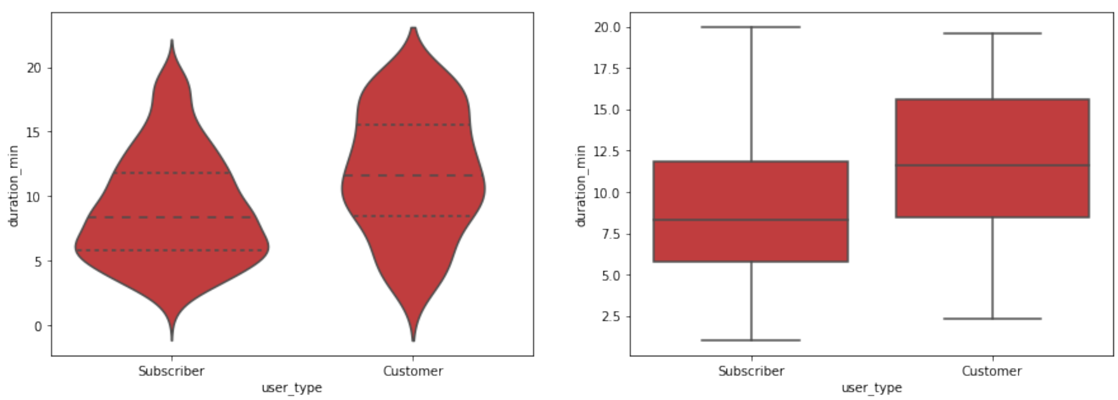
[78]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))


# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data3, x = 'month', y = 'log_length', color =␣
 ↪sb_colors, inner = 'quartile');
tick_locsx = [-2, -1.5, -1, -0.5, 0, 0.5, 1]
tick_labels = [0.01, 0.03, 0.1, 0.3, 1, 3, 10]
plt.yticks(tick_locsx, tick_labels);

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = trip_data3, x = 'month', y = 'log_length', color = sb_colors);
tick_locsx = [-2, -1.5, -1, -0.5, 0, 0.5, 1]
tick_labels = [0.01, 0.03, 0.1, 0.3, 1, 3, 10]
plt.yticks(tick_locsx, tick_labels);
```



These plots show, as before, that the length of the trips did not vary much between the revised months. This representation shows more variations than the normal plot, but still the differences between months are minimal.

Now let's explore the relationship between user types and their trip duration and length: > First duration (short trips):

[79]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))



# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = short_trips.sample(1000, random_state = 3), x =␣
↪'user_type', y = 'duration_min', color = sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = short_trips.sample(1000, random_state = 3), x = 'user_type',␣
↪y = 'duration_min', color = sb_colors);
```

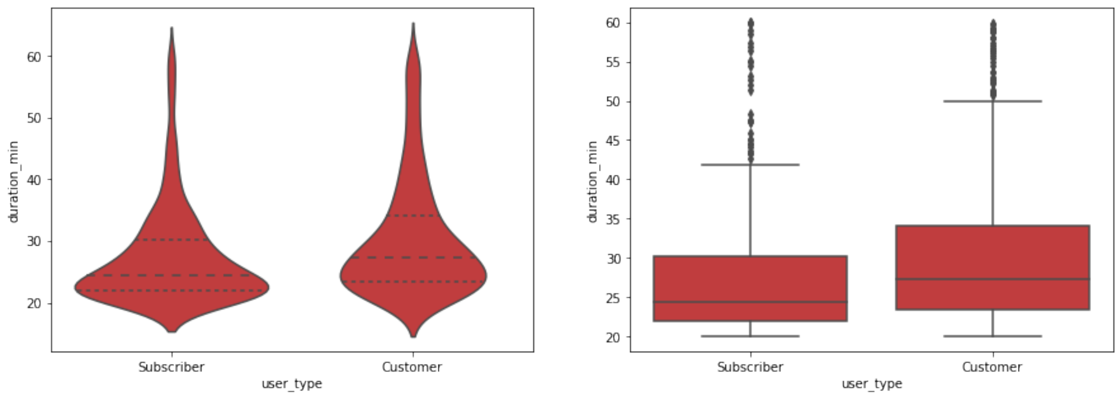



This plot shows how the type of customer relates to the duration of shorter trips. Subscribers go on shorter trips but also their data shows more variations. Customers show a smaller range of values and a more constant distribution of different durations.

Let´s inspect medium duration trips:

[80]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))
```

```
# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = medium_trips.sample(1000, random_state = 3), x =␣
↪'user_type', y = 'duration_min', color = sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
sb.boxplot(data = medium_trips.sample(1000, random_state = 3), x = 'user_type',␣
↪y = 'duration_min', color = sb_colors);
```

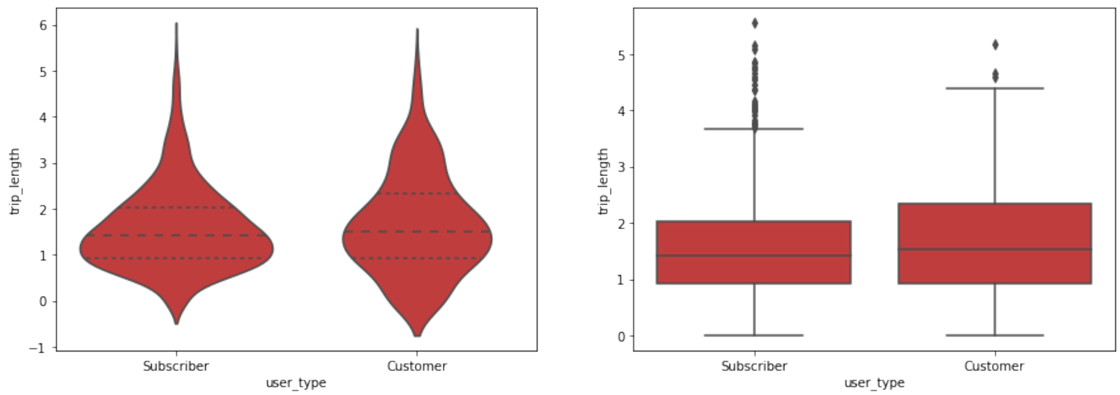



These plots show describe different relationships when compared to shorter trips. First their lower tail is short, as these values were trimmed. As before, customer users seem to go on longer trips than subscribers. Violin plots suggest that it is more likely that subscribers will go on trips lasting 22 or 23 minutes and customers would take one or two more minutes than that. There are lower chances that both of them would take trips lasting more than 50 minutes, but this probability is higher for customers.

Next their relationship with the variable length.

[81]:
```
# figure size
fig, ax = plt.subplots(figsize=(15,5))


# violin plot on the left
plt.subplot(1, 2, 1)
sb_colors = sb.color_palette()[3] #set colors
sb.violinplot(data = trip_data.sample(1000, random_state = 3), x = 'user_type',␣
↪y = 'trip_length', color = sb_colors, inner = 'quartile');

# box plot on the right
plt.subplot(1, 2, 2)
```

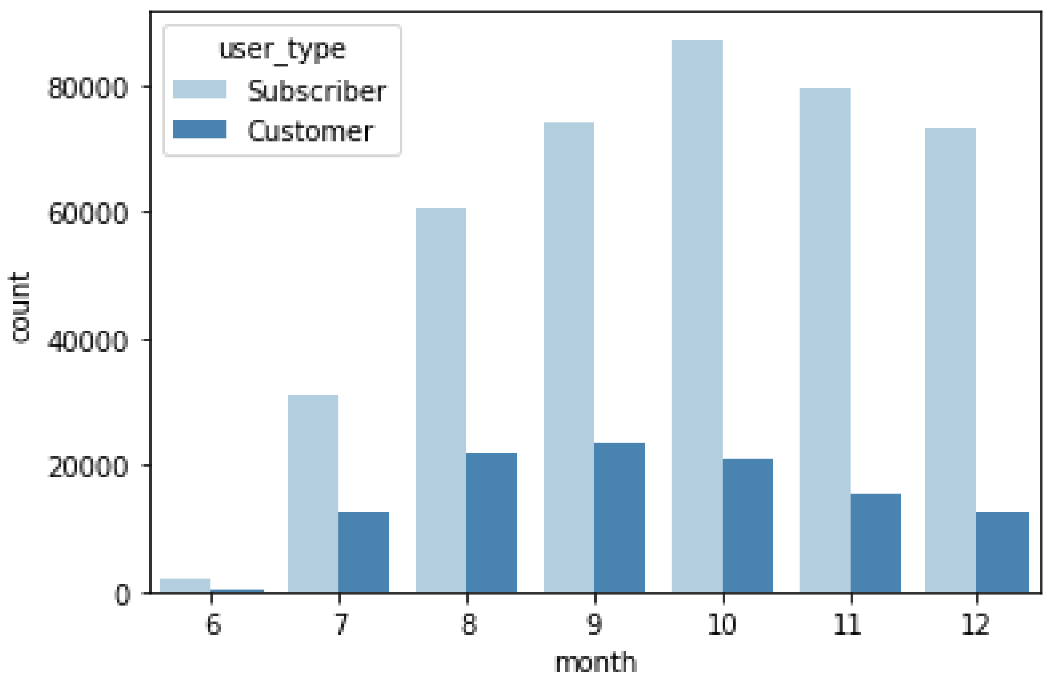
```
sb.boxplot(data = trip_data.sample(1000, random_state = 3), x = 'user_type', y␣
↪= 'trip_length', color = sb_colors);
```




Plots show that the medians of both users are almost equal, same for the first quantile. The violin plot highlights more differences between the variables where customers seem to have a more proportionate distribution while subscribers' trips are mostly around the kilometer with long tails towards longer distances.

Next the exploration of the user type along the provided months:

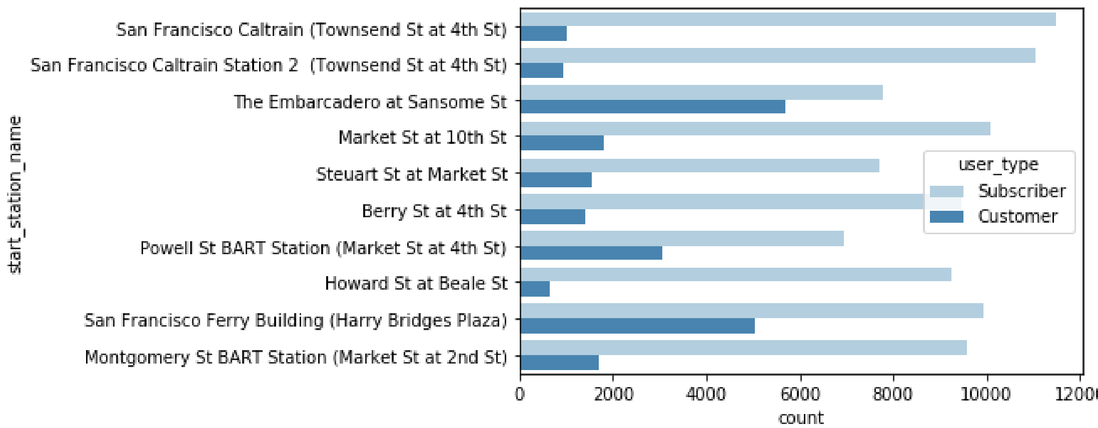
[82]:
```
# clustered bar chart for the categorical vars
sb.countplot(data = trip_data, x ='month', hue = 'user_type', palette =␣
↪'Blues');
```



Subscriber users held longer trips for all months with much difference. It is not surprising as these users constitute almost an 80% of the total (see previous section). Still for both users the data seems to display trends that increase from June to October to then decrease till December.

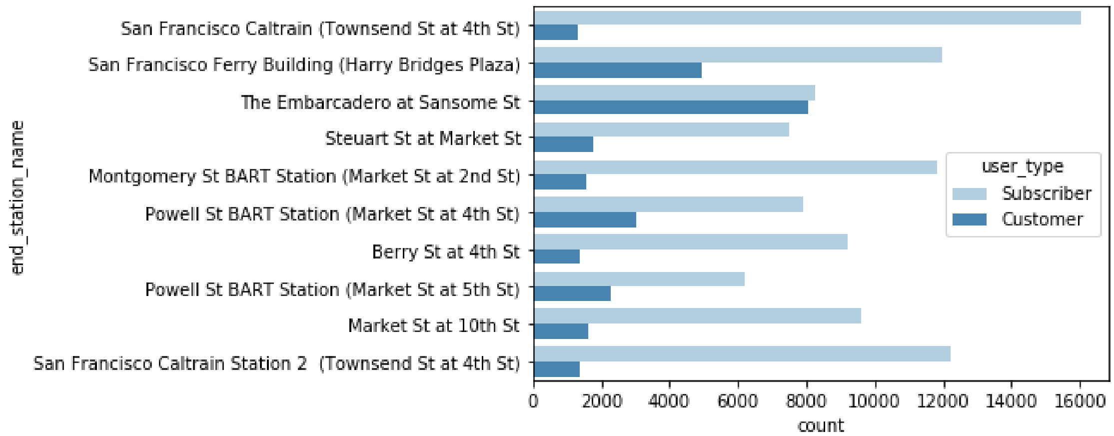
Next an inspection on the most used stations and what kind of users rent bikes there:

[83]:
```
# clustered bar chart for the categorical vars
sb.countplot(data = Top_start_stations, y ='start_station_name', hue =␣
↪'user_type', palette = 'Blues');
```




This graph shows some valuable insights. The station Ferry Building and Embarcadero are the ones more frequent among customers. On the other hand, subscribers prefer Caltrain stations. These users have a differentiated preference when it comes to start stations.

[84]:
```
# clustered bar chart for the categorical vars
sb.countplot(data = Top_end_stations, y ='end_station_name', hue = 'user_type',␣
↪palette = 'Blues');
```




As the top stations of both end and start are different it is difficult to compare them. But they do share some similarities. In this plot one station, The Embarcadero one, has the same values for customers and subscribers, this did not happen in the start stations. And as before Caltrain station is the most used by subscribers and Embarcadero the most common one for customers.

### 1.1.7 Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

The variables of interest (duration and length) showed at first a weak coefficient of correlation between them. Further plots of the data, by means of samples, transformations and segmented parts of the dataset, highlighted their strong relationship. The transformed plot suggested a strong log-log relationship.

When it comes to the relationship observed between length and duration and the categorical features, these highlighted interesting facts (mainly due to a lack of relationship as I was expecting). First, length and duration did not have special relationships with the months. For most months these values did not change drastically, but they did show differentiated shapes displayed by the violin plots. The type of user did have an impact in the duration of the trip, and less in its length. This could be based on the purpose of the trip, which impacts the time that takes to cover certain distances.

### 1.1.8 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

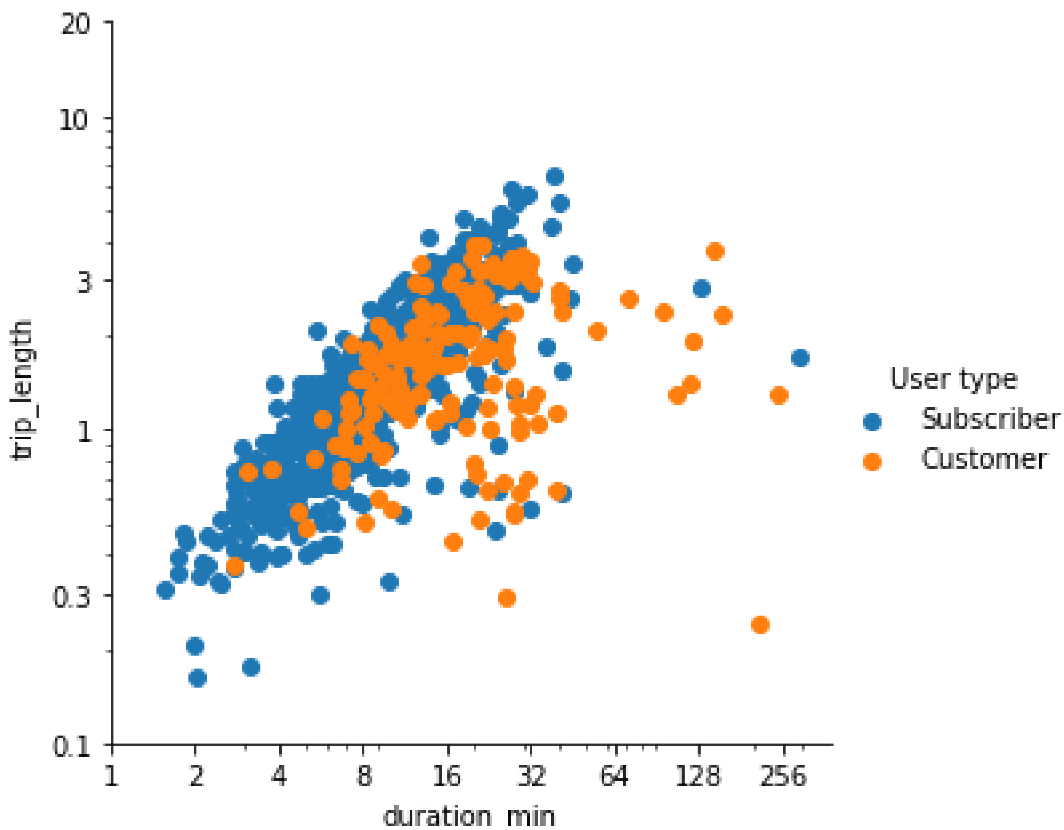
It was interesting to find out that both suscribers and customers shared the same use trend between months. The only other plot that did not contain the variables of interest was the one linking type of customers to favorite stations and it did highlited diferenciated preferences between customers and subscrbiers for both start and end stations.

## Multivariate Exploration

This section follows findings shown in previous sections. The way duration and length relate to each other and other categorical variables will be the main focus of the analysis.

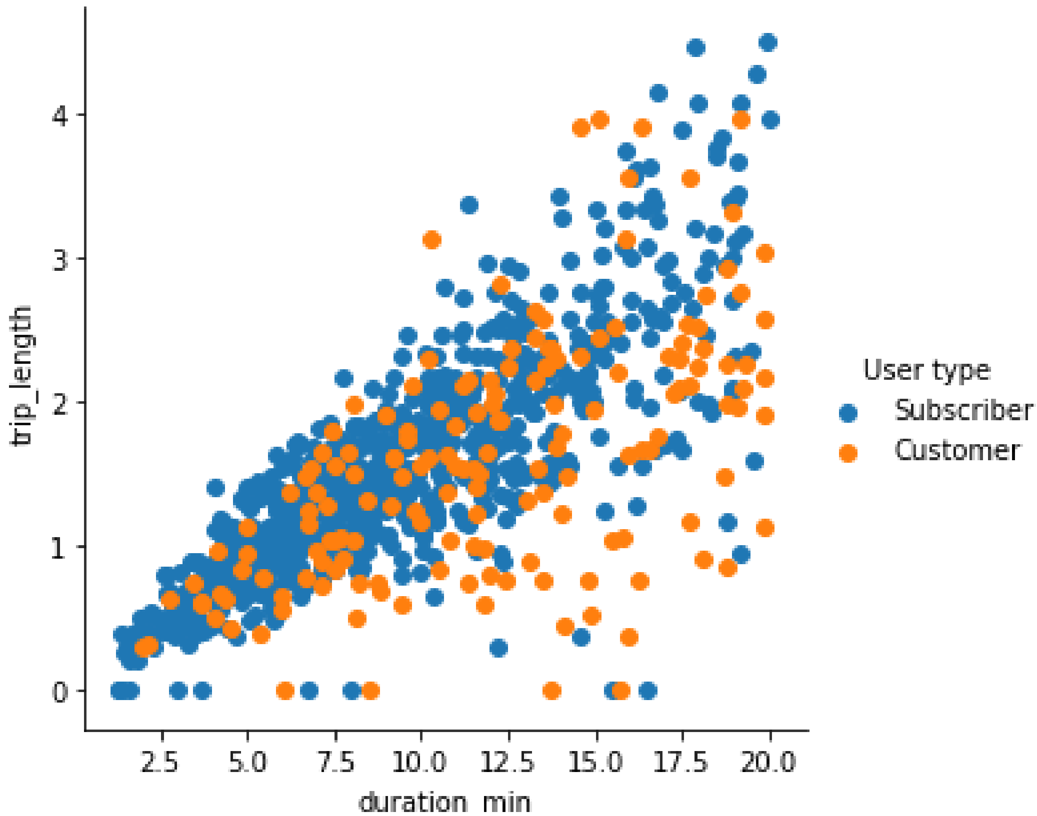
First, scatter plots of duration vs length will have third variables added:

[85]:
```
# plot two numerical plus one categorical
g = sb.FacetGrid(data = trip_data3.sample(1000, random_state = 4), hue =␣
↪'user_type', height= 4.5)
g.map(plt.scatter, 'duration_min', 'trip_length')
g.add_legend(title = 'User type');
plt.xscale('log')
tick_locsx = [1, 2, 4, 8, 16, 32, 64, 128, 256]
plt.xticks(tick_locsx, tick_locsx)
plt.yscale('log')
tick_locs = [0.1, 0.3, 1, 3, 10, 20]
plt.yticks(tick_locs, tick_locs);
```



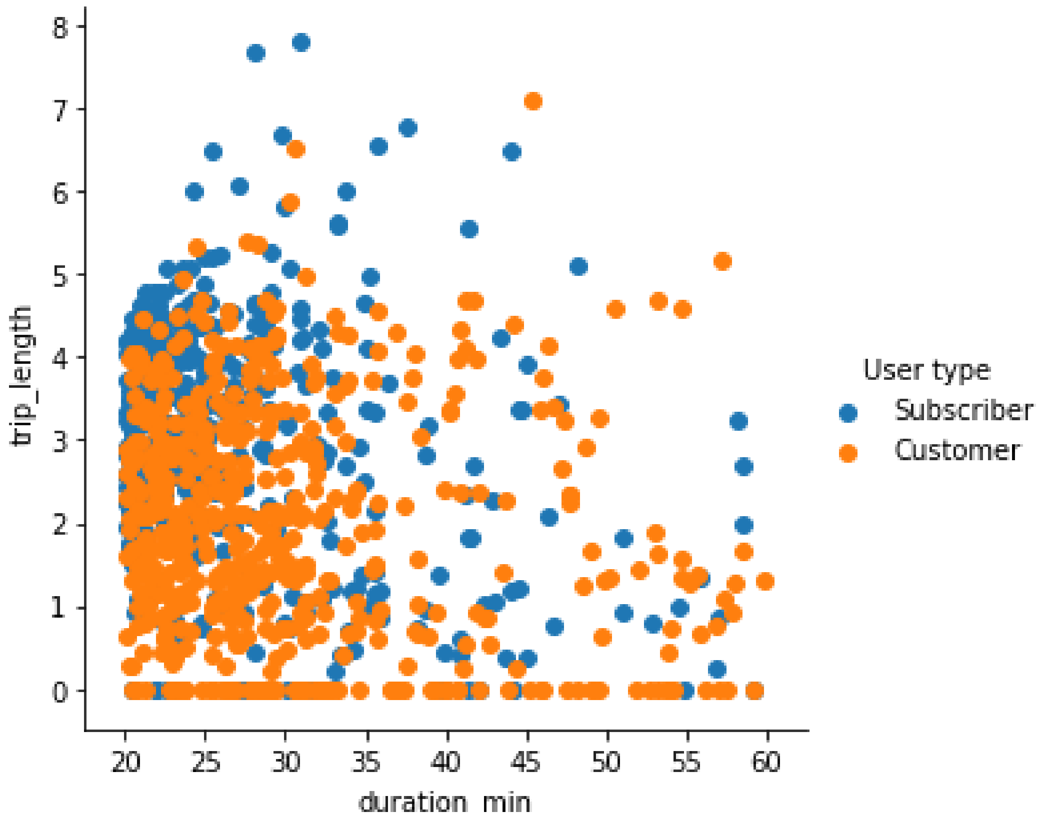

This plot depicts the log-log relationship between duration and length, by user type. The distribution of subscribers and customers will be inspected more in detail with shorter and longer trips:

[86]:
```
# plot two numerical plus one categorical
g = sb.FacetGrid(data = short_trips.sample(1000, random_state = 4), hue =␣
↪'user_type', height= 4.5)
g.map(plt.scatter, 'duration_min', 'trip_length')
g.add_legend(title = 'User type');
```



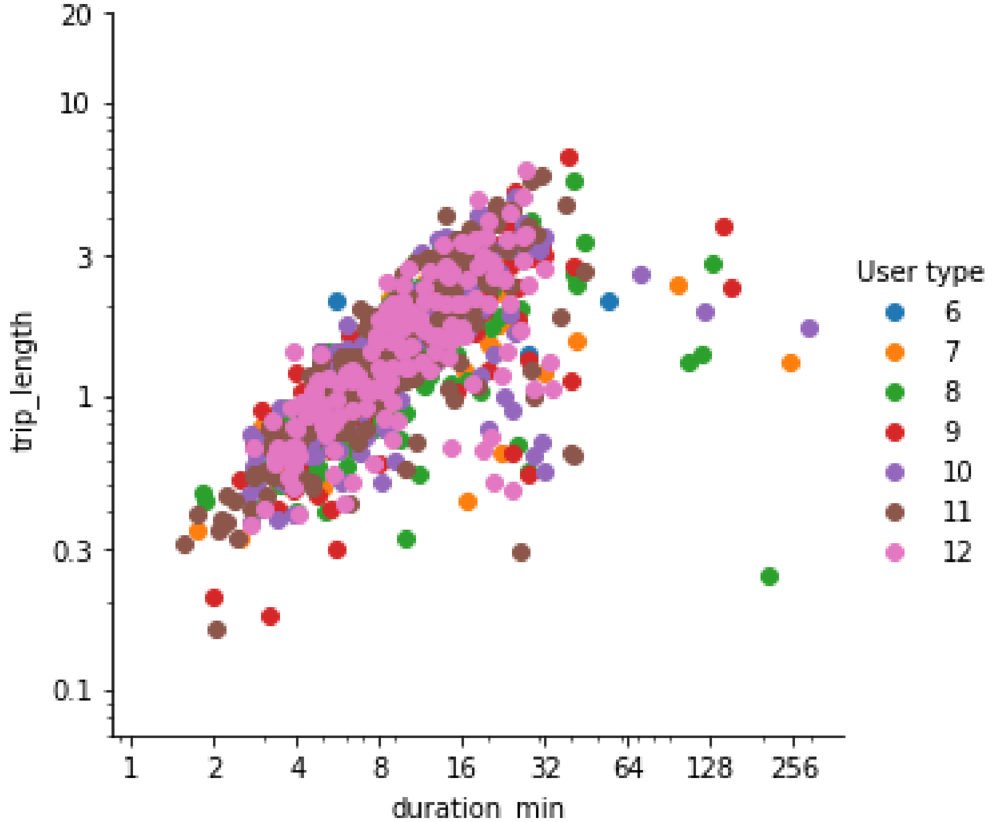

Customers seem to be more scattered among different combinations of length-duration. It suggests that it would be interesing to analyze their data separately, so as to understand each user operative characteristics better.

[87]:
```
# plot two numerical plus one categorical
g = sb.FacetGrid(data = medium_trips.sample(1000, random_state = 4), hue =␣
↪'user_type', height= 4.5)
g.map(plt.scatter, 'duration_min', 'trip_length')
g.add_legend(title = 'User type');
```




Customers are more present in trips with zero length for all durations, whether these are round trips or outliers.

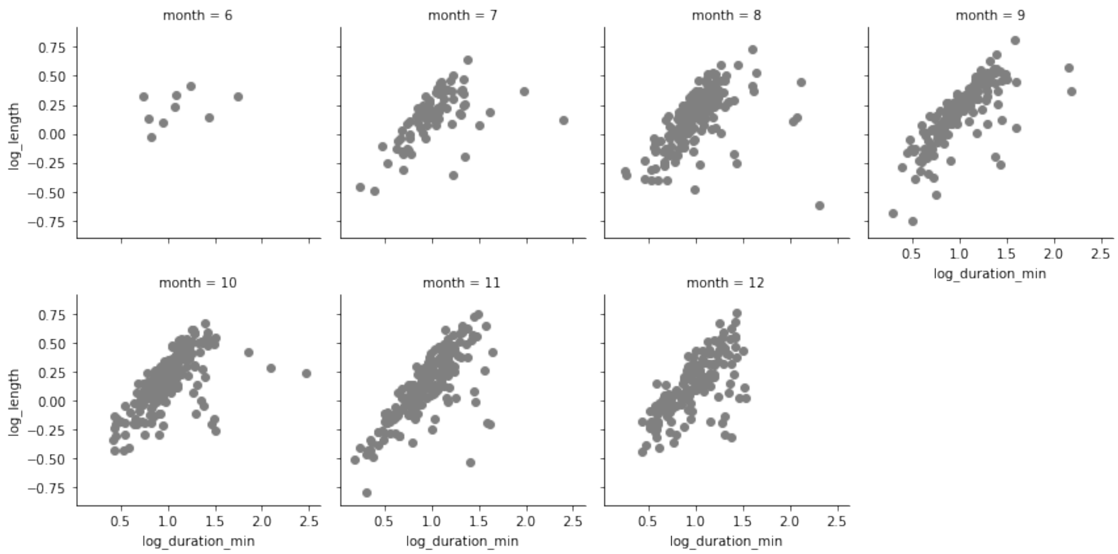
[88]:
```
# plot relationship
g = sb.FacetGrid(data = trip_data3.sample(1000, random_state = 4), hue =␣
↪'month', height= 4.5)
g.map(plt.scatter, 'duration_min', 'trip_length')
g.add_legend(title = 'User type');
plt.xscale('log')
tick_locsx = [1, 2, 4, 8, 16, 32, 64, 128, 256]
plt.xticks(tick_locsx, tick_locsx)
plt.yscale('log')
tick_locs = [0.1, 0.3, 1, 3, 10, 20]
plt.yticks(tick_locs, tick_locs);
```




With so many values this is not the best way to inspect this relationship.

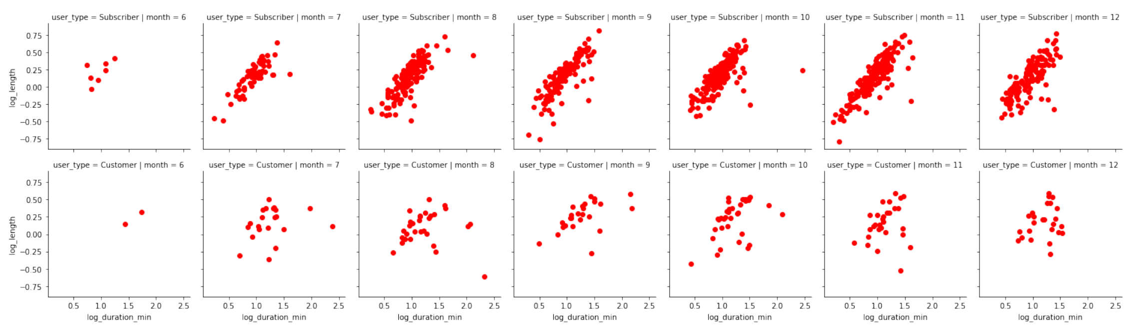
[89]:
```
# The dataset with trimmed data and log-log relationship
trip_data3['log_duration_min'] = trip_data3['duration_min'].apply(log_trans)
```

[90]:
```
# create faceted heat maps on levels of the cut variable
g = sb.FacetGrid(data = trip_data3.sample(1000, random_state = 4), col =␣
↪'month', col_wrap = 4, height = 3,)
g.map(plt.scatter, 'log_duration_min', 'log_length', color = 'grey');
```




All months seem to follow the same trend. September seems to have the les consistent data with more scattered observations.

[91]:
```
# create faceted heat maps on levels of the cut variable
g = sb.FacetGrid(data = trip_data3.sample(1000, random_state = 4), col = 'month', row = 'user_type', height = 3,)
g.map(plt.scatter, 'log_duration_min', 'log_length', color = 'red');
```




As seen before, subscribers have more consistent data, whereas customers seem to be all over.

### 1.1.9 Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

This part of the investigation highlighted the distinctive performances that customers and subscribed users had.

### 1.1.10 Were there any interesting or surprising interactions between features?

Medium duration trips seem to be more scattered when plotted along with performances (time-distance pair) and I would have guessed that longer trips were mostly done by proficient users whose speed was as well going to be higher compared to the distances they covered. Moreover, I was expecting to see some insights provided by the months, but these seem to be very similar.

### References


1. https://stackoverflow.com/questions/46908388/find-euclidean-distance-from-a-point-to-rows-in-pandas-dataframe?rq=1
2. https://stackoverflow.com/questions/29545704/fast-haversine-approximation-python-pandas
3. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html
4. https://medium.com/data-tale/how-far-do-people-travel-in-bike-sharing-systems-faf0295bc75a
5. Wu, Y. H., Kang, L., Hsu, Y. T., & Wang, P. C. (2019). Exploring trip characteristics of bike-sharing system uses: Effects of land-use patterns and pricing scheme change. International journal of transportation science and technology, 8(3), 318-331.


[ ]:

```

```