Bike_Share_Analysis Project

February 1, 2018

1 Project 2 : 2016 US Bike Share Activity Analysis

1.1 Introduction

Over the past decade, bicycle-sharing systems have been growing in number and popularity in cities across the world. Bicycle-sharing systems allow users to rent bicycles for short trips, typically 30 minutes or less.

In this project, an exploratory analysis is performed on the data which is provided by Motivate, a bike-share system provider for many major cities in the United States. The comparison made on the system usage between three large cities: New York City, Chicago, and Washington, DC. The data is examined if there are any differences within each system for those users that are registered, regular users and those users that are short-term, casual users.

Posing Questions

Question 1: Which city has the highest number of trips? Which city has the highest proportion of trips made by subscribers? Which city has the highest proportion of trips made by short-term customers?

Question 2: What is the average trip length for each city? What proportion of rides made in each city are longer than 30 minutes?

Question 3: Which type of user takes longer rides on average: Subscribers or Customers?

Question 4. What are the peak and slack hours of a day and days of a week?

Data Collection and Wrangling This project, focus on the record of individual trips taken in 2016 from City Bike ,Divvy and Capital Bikeshare in the New York City, Chicago, and Washington, DC respectively . Each of these cities has a page where the trip data could freely downloaded.: - New York City (Citi Bike): Link - Chicago (Divvy): Link - Washington, DC (Capital Bikeshare): Link

Each city has a different way of delivering its data. Chicago updates with new data twice a year, Washington DC is quarterly, and New York City is monthly. Some data wrangling of inconsistencies in timestamp format within each city has already been performed and in addition, a random 2% sample of the original data is taken to make the exploration more manageable.

```
In [8]: ## import all necessary packages and functions.
        import pandas as pd
        import csv
        from datetime import datetime
        from pprint import pprint
```

```
In [131]: data_files = ['./data/NYC-CitiBike-2016.csv',
                        './data/Chicago-Divvy-2016.csv',
                        './data/Washington-CapitalBikeshare-2016.csv']
          for data_file in data_files:
              df=pd.read_csv(data_file)
              df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 276798 entries, 0 to 276797
Data columns (total 15 columns):
tripduration               276798 non-null int64
starttime                  276798 non-null object
stoptime                   276798 non-null object
start station id           276798 non-null int64
start station name         276798 non-null object
start station latitude     276798 non-null float64
start station longitude    276798 non-null float64
end station id             276798 non-null int64
end station name           276798 non-null object
end station latitude       276798 non-null float64
end station longitude      276798 non-null float64
bikeid                     276798 non-null int64
usertype                   276081 non-null object
birth year                 245137 non-null float64
gender                     276798 non-null int64
dtypes: float64(5), int64(5), object(5)
memory usage: 31.7+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72131 entries, 0 to 72130
Data columns (total 12 columns):
trip_id              72131 non-null int64
starttime            72131 non-null object
stoptime             72131 non-null object
bikeid               72131 non-null int64
tripduration         72131 non-null int64
from_station_id      72131 non-null int64
from_station_name    72131 non-null object
to_station_id        72131 non-null int64
to_station_name      72131 non-null object
usertype             72131 non-null object
gender               54977 non-null object
birthyear            54986 non-null float64
dtypes: float64(1), int64(5), object(6)
memory usage: 6.6+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66326 entries, 0 to 66325
Data columns (total 9 columns):
Duration (ms)           66326 non-null int64
```

```
Start date              66326 non-null object
End date                66326 non-null object
Start station number    66326 non-null int64
Start station           66326 non-null object
End station number      66326 non-null int64
End station             66326 non-null object
Bike number             66326 non-null object
Member Type             66326 non-null object
dtypes: int64(3), object(6)
memory usage: 4.6+ MB
```

```
In [9]: def print_first_point(filename):
            """
            This function prints and returns the first data point (second row) from
            a csv file that includes a header row.
            """
            # print city name for reference
            city = filename.split('-')[0].split('/')[-1]
            print('\nCity: {}'.format(city))
            with open(filename, 'r') as f_in:
                ##  csv library to set up a DictReader object.
                dictreader = csv.DictReader(f_in)

                # DictReader object to read the first trip from the data file and store it in a
                first_trip =next (dictreader)
                pprint (first_trip)
            # output city name and first trip for later testing
            return (city, first_trip)

        # list of files for each city
        data_files = ['./data/NYC-CitiBike-2016.csv',
                      './data/Chicago-Divvy-2016.csv',
                      './data/Washington-CapitalBikeshare-2016.csv',]

        # print the first trip from each file, store in dictionary
        example_trips = {}
        for data_file in data_files:

            city, first_trip = print_first_point(data_file)
            example_trips[city]=first_trip
```

```
City: NYC
OrderedDict([('tripduration', '839'),
             ('starttime', '1/1/2016 00:09:55'),
             ('stoptime', '1/1/2016 00:23:54'),
             ('start station id', '532'),
             ('start station name', 'S 5 Pl & S 4 St'),
             ('start station latitude', '40.710451'),
             ('start station longitude', '-73.960876'),
             ('end station id', '401'),
             ('end station name', 'Allen St & Rivington St'),
             ('end station latitude', '40.72019576'),
             ('end station longitude', '-73.98997825'),
             ('bikeid', '17109'),
             ('usertype', 'Customer'),
             ('birth year', ''),
             ('gender', '0')])

City: Chicago
OrderedDict([('trip_id', '9080545'),
             ('starttime', '3/31/2016 23:30'),
             ('stoptime', '3/31/2016 23:46'),
             ('bikeid', '2295'),
             ('tripduration', '926'),
             ('from_station_id', '156'),
             ('from_station_name', 'Clark St & Wellington Ave'),
             ('to_station_id', '166'),
             ('to_station_name', 'Ashland Ave & Wrightwood Ave'),
             ('usertype', 'Subscriber'),
             ('gender', 'Male'),
             ('birthyear', '1990')])

City: Washington
OrderedDict([('Duration (ms)', '427387'),
             ('Start date', '3/31/2016 22:57'),
             ('End date', '3/31/2016 23:04'),
             ('Start station number', '31602'),
             ('Start station', 'Park Rd & Holmead Pl NW'),
             ('End station number', '31207'),
             ('End station', 'Georgia Ave and Fairmont St NW'),
             ('Bike number', 'W20842'),
             ('Member Type', 'Registered')])
```

Condensing the Trip Data

Its observable from the above printout that each city provides different information. Even where the information is the same, the column names and formats are sometimes different.

Duration: This has been given in seconds in the (New York, Chicago) or milliseconds (Washington). **Month**, **Hour**, **Day of Week**: Ridership volume is likely to change based on the season, time of day, and whether it is a weekday or weekend. **User Type**: Washington divides its users into two types: 'Registered' for users with annual, monthly, and other longer-term subscriptions, and 'Casual', for users with 24-hour, 3-day, and other short-term passes. The New York and Chicago data uses 'Subscriber' and 'Customer' for these groups, respectively.

```
In [11]: def duration_in_mins(datum, city):

             """
             Takes as input a dictionary containing info about a single trip (datum) and
             its origin city (city) and returns the trip duration in units of minutes.

             """
             if city != 'Washington':
                 duration = (int (datum['tripduration'])/60)
             else:
                 duration = (int(datum['Duration (ms)'])/60000)

             return duration


In [12]: def time_of_trip(datum, city):
             """
             Takes as input a dictionary containing info about a single trip (datum) and
             its origin city (city) and returns the month, hour, and day of the week in
             which the trip was made.
             """
             if city=='NYC':
                 date1 = datetime.strptime(datum['starttime'], "%m/%d/%Y %H:%M:%S")
                 month=date1.month
                 hour=date1.hour
                 day_of_week=date1.strftime("%A")
             elif city=='Chicago':
                 date2= datetime.strptime(datum['starttime'], "%m/%d/%Y %H:%M")
                 month=date2.month
                 hour=date2.hour
                 day_of_week=date2.strftime("%A")
             else:
                 date3= datetime.strptime(datum['Start date'], "%m/%d/%Y %H:%M")
                 month=date3.month
                 hour=date3.hour
                 day_of_week=date3.strftime("%A")
             return (month, hour, day_of_week)

In [13]: def type_of_user(datum, city):
             """
             Takes as input a dictionary containing info about a single trip (datum) and
             its origin city (city) and returns the type of system user that made the
             trip.
```

```
        """
        if city!='Washington':
            user_type=datum['usertype']
        else:
            user_type=datum['Member Type']
            if user_type=='Registered':
                user_type='Subscriber'
            else:
                user_type='Customer'
        return user_type
```

Condense the data

```
In [15]: def condense_data(in_file, out_file, city):
             """
             This function takes full data from the specified input file
             and writes the condensed data to a specified output file. The city
             argument determines how the input file will be parsed.

             """
             with open(out_file, 'w') as f_out, open(in_file, 'r') as f_in:
                 out_colnames = ['duration', 'month', 'hour', 'day_of_week', 'user_type']
                 trip_writer = csv.DictWriter(f_out, fieldnames = out_colnames)
                 trip_writer.writeheader()
                 ## set up csv DictReader object ##
                 trip_reader = csv.DictReader(f_in)
                 # collect data from and process each row
                 for row in trip_reader:
                     sample_data = {}
                     sample_data['duration'] = str(duration_in_mins(row,city))
                     sample_data['month'] = str(time_of_trip(row,city)[0])
                     sample_data['hour'] = str(time_of_trip(row,city)[1])
                     sample_data['day_of_week'] = str(time_of_trip(row,city)[2])
                     sample_data['user_type'] = str(type_of_user(row,city))
                     trip_writer.writerow(sample_data)
```

```
In [17]: # checking the function to print the first trip data of the 3 data sets .
         city_info = {'Washington': {'in_file': './data/Washington-CapitalBikeshare-2016.csv',
                                     'out_file': './data/Washington-2016-Summary.csv'},
                      'Chicago': {'in_file': './data/Chicago-Divvy-2016.csv',
                                  'out_file': './data/Chicago-2016-Summary.csv'},
                      'NYC': {'in_file': './data/NYC-CitiBike-2016.csv',
                              'out_file': './data/NYC-2016-Summary.csv'}}

         for city, filenames in city_info.items():
             condense_data(filenames['in_file'], filenames['out_file'], city)
             print_first_point(filenames['out_file'])
```

```
City: Washington
```

```
OrderedDict([('duration', '7.123116666666666'),
             ('month', '3'),
             ('hour', '22'),
             ('day_of_week', 'Thursday'),
             ('user_type', 'Subscriber')])

City: Chicago
OrderedDict([('duration', '15.433333333333334'),
             ('month', '3'),
             ('hour', '23'),
             ('day_of_week', 'Thursday'),
             ('user_type', 'Subscriber')])

City: NYC
OrderedDict([('duration', '13.983333333333333'),
             ('month', '1'),
             ('hour', '0'),
             ('day_of_week', 'Friday'),
             ('user_type', 'Customer')])
```

1.2 Exploratory Data Analysis

```
In [20]: def number_of_trips(filename, city):
             """
             Reads in a file with trip data and reports the number of
             trips made by subscribers, customers, and the total trip .
             """
             with open(filename, 'r') as f_in:
                 # set up csv reader object
                 reader = csv.DictReader(f_in)

                 # initialize count variables
                 n_subscribers = 0
                 n_customers = 0
                 n_trip = 0
                 # tally up ride types
                 for row in reader:
                     n_trip += 1
                     user = type_of_user(row,city)
                     if user == 'Subscriber':
                         n_subscribers += 1
                     else:
                         n_customers += 1
                 # compute total number of rides
                 n_total = n_subscribers + n_customers

                 # return tallies as a tuple
```

```
            return(n_subscribers, n_customers, n_total)
```

```
In [9]: data_files = ['./data/NYC-CitiBike-2016.csv',
                      './data/Chicago-Divvy-2016.csv',
                      './data/Washington-CapitalBikeshare-2016.csv']
        for data_file in data_files:
            city = data_file.split('-')[0].split('/')[-1]
            n_subscribers, n_customers, n_total  = number_of_trips(data_file, city)
            print(city,number_of_trips(data_file, city))
        for data_file in data_files:
            city = data_file.split('-')[0].split('/')[-1]
            n_total=number_of_trips(data_file, city)[2]
            trip_list={}
            trip_list[city]=n_total
            #a= max (trip_list['NYC'],trip_list['Chicago'], trip_list['Washington'])
            print(trip_list)
```

```
NYC (245896, 30902, 276798)
Chicago (54982, 17149, 72131)
Washington (51753, 14573, 66326)
{'NYC': 276798}
{'Chicago': 72131}
{'Washington': 66326}
```

```
In [16]: def trip_count_usertype(filename):
             """
             Creates a dictreader object ,sort users , calculates
             the count and percentage for each type of user.
             """
             # print city name
             city = filename.split('-')[0].split('/')[-1]
             print('\nCity: {}'.format(city))

             with open(filename, 'r') as f_in:


                 dictreader = csv.DictReader(f_in)
                 duration_list_subscriber=[]
                 duration_list_customer=[]
                 for row  in dictreader:

                     duration =float( row['duration'])
                     user_type=row['user_type']
                     if user_type=='Subscriber':
                         duration_list_subscriber.append (duration)
                     else:
                         duration_list_customer.append (duration)
```

```
                    count_c =len( duration_list_customer)
                    count_s = len( duration_list_subscriber)
                    count_total= count_c + count_s
                print ('Subscriber count :',count_s)
                print ('Subscriber percentage  :',"{:.1%}".format(count_s/count_total))
                print ('Customer count :',count_c)
                print ( 'Customer Percentage :',"{:.1%}".format(count_c/count_total) )
                ## Create a list using the DictReader object. ##
        trip_count_usertype('./data/NYC-2016-Summary.csv')
        trip_count_usertype('./data/Washington-2016-Summary.csv')
        trip_count_usertype('./data/Chicago-2016-Summary.csv')
```

```
City: NYC
Subscriber count : 245896
Subscriber percentage  : 88.8%
Customer count : 30902
Customer Percentage : 11.2%

City: Washington
Subscriber count : 51753
Subscriber percentage  : 78.0%
Customer count : 14573
Customer Percentage : 22.0%

City: Chicago
Subscriber count : 54982
Subscriber percentage  : 76.2%
Customer count : 17149
Customer Percentage : 23.8%
```

1.3 Question1

NYC has the highest number of trips(276798 Trips).

NYC has the highest proportion of trips made by subscribers(88.8%).

Chicago has the highest proportion of trips made by customers(23.8%).

```
In [36]: def import_durations(filename):
             """
             Creates a dictreader object from a csv file and then creates a list of durations.
             """
             # print city name
             city = filename.split('-')[0].split('/')[-1]
             print('\nCity: {}'.format(city))

             with open(filename, 'r') as f_in:
                ## csv library set up a DictReader object. ##
                dictreader = csv.DictReader(f_in)
                duration_list=[]
                for row  in dictreader:

                    duration =float( row['duration'])
                    #duration=float(duration_list[row])
                    duration_list.append (duration)
            return (duration_list)
```

```
In [37]: def average_trip(filename):

            '''calculate the average trip length using the list'''
            duration_user_list = import_durations(filename)
            average_trip =round(( sum(duration_user_list)/len(duration_user_list)),2)

            # Count the number of trips longer than 30 mins
            shorter_trips = 0
            longer_trips = 0
            for i in range(len(duration_user_list)):

                if duration_user_list[i] > 30:
                    longer_trips +=1
                else:
                    shorter_trips +=1

            print('Average trip duration : ' + str(average_trip)+' mins')
            print('Trips longer than 30 mins : ' + str(longer_trips))
            print('Trips equal or less than 30 mins : ' + str(shorter_trips))
            print('Total trips : ' + str(len(duration_user_list)))
            print ('Percentage of trips longer than 30 mins : '+ "{:.1%}".format((longer_trips)

        data_files = ['./data/Washington-2016-Summary.csv',
                      './data/Chicago-2016-Summary.csv',
                      './data/NYC-2016-Summary.csv']
        for data_file in data_files:
            print (average_trip(data_file))
```

```
City: Washington
Average trip duration : 18.93 mins
Trips longer than 30 mins : 7189
Trips equal or less than 30 mins : 59137
Total trips : 66326
Percentage of trips longer than 30 mins : 10.8%
None
```

```
City: Chicago
Average trip duration : 16.56 mins
Trips longer than 30 mins : 6010
Trips equal or less than 30 mins : 66121
Total trips : 72131
Percentage of trips longer than 30 mins : 8.3%
None

City: NYC
Average trip duration : 15.81 mins
Trips longer than 30 mins : 20213
Trips equal or less than 30 mins : 256585
Total trips : 276798
Percentage of trips longer than 30 mins : 7.3%
None
```

1.4 Question 2

The average ride duration for Customers in NYC is 32.79 mins , in Washigton 41.68 mins and in Chicago its 30.98 mins .

```
In [38]: def trip_duration_stat(filename):
             """
             Creates a dictreader object ,catagorize the user types into 2 lists and calculates
             the total count and average duration for user types
             """
             # print city name
             city = filename.split('-')[0].split('/')[-1]
             print('\nCity: {}'.format(city))

             with open(filename, 'r') as f_in:

                 dictreader = csv.DictReader(f_in)
                 duration_list_subscriber=[]
                 duration_list_customer=[]
                 for row  in dictreader:

                     duration =float( row['duration'])
                     user_type=row['user_type']
                     if user_type=='Subscriber':
                         duration_list_subscriber.append (duration)
                     else:
                         duration_list_customer.append (duration)
                     #print  ("{}:{}".format(user_type,duration))
                     count_c =len( duration_list_customer)
                     count_s = len( duration_list_subscriber)
                     sum_c= sum (duration_list_customer)
```

```
                    sum_s= sum(duration_list_subscriber)

                print ('Subscriber_count :',count_s)
                print ('Customer_count :',count_c)
                print ('Subscriber_average_duration :',(sum_s/count_s))
                print ( 'Customer_average_duration :',(sum_c/count_c) )

        trip_duration_stat('./data/NYC-2016-Summary.csv')
        trip_duration_stat('./data/Washington-2016-Summary.csv')
        trip_duration_stat('./data/Chicago-2016-Summary.csv')
```

```
City: NYC
Subscriber_count : 245896
Customer_count : 30902
Subscriber_average_duration : 13.680790523907177
Customer_average_duration : 32.77595139473187

City: Washington
Subscriber_count : 51753
Customer_count : 14573
Subscriber_average_duration : 12.528120499294745
Customer_average_duration : 41.67803139252976

City: Chicago
Subscriber_count : 54982
Customer_count : 17149
Subscriber_average_duration : 12.067201690250076
Customer_average_duration : 30.979781133982506
```

1.5 Question 3

In all the cities the data have been provided , the Customers take longer rides on average .

```
In [39]: def weekly_trip_stat(filename):
             """
             Creates a dictreader object ,catagorize trips into lists of days(mon-sun) and calcu
             the total count and average duration for each day.
             """
             # print city name
             city = filename.split('-')[0].split('/')[-1]
             print('\nCity: {}'.format(city))

             with open(filename, 'r') as f_in:

                 dictreader = csv.DictReader(f_in)
```

```
Monday_trip_list=[]
Tuesday_trip_list=[]
Wednesday_trip_list=[]
Thursday_trip_list=[]
Friday_trip_list=[]
Saturday_trip_list=[]
Sunday_trip_list=[]

for row  in dictreader:

    day_of_week = row['day_of_week']
    #user_type=row['user_type']
    if day_of_week=='Monday':
        Monday_trip_list.append (day_of_week)
    elif day_of_week=='Tuesday':
        Tuesday_trip_list.append (day_of_week)
    elif day_of_week=='Wednesday':
        Wednesday_trip_list.append (day_of_week)
    elif day_of_week=='Thursday':
        Thursday_trip_list.append (day_of_week)
    elif day_of_week=='Friday':
        Friday_trip_list.append (day_of_week)
    elif day_of_week=='Saturday':
        Saturday_trip_list.append (day_of_week)
    else:
        Sunday_trip_list.append (day_of_week)

    count_mon =len( Monday_trip_list)
    count_tue =len( Tuesday_trip_list)
    count_wen =len( Wednesday_trip_list)
    count_thu =len( Thursday_trip_list)
    count_fri =len( Friday_trip_list)
    count_sat =len( Saturday_trip_list)
    count_sun =len( Sunday_trip_list)
    count_week= sum((count_mon,count_tue,count_wen,count_thu,count_fri,count_sa
    count_weekdays=sum ((count_mon,count_tue,count_wen,count_thu,count_fri))
    count_weekends= sum ((count_sat,count_sun))
    count_all_trips=sum((count_weekdays,count_weekends))

print   (('WEEKDAYS_COUNT :',count_weekdays),
         ('Weekly_Portion:',"{:.1%}".format(count_weekdays/count_all_trips)))
print   (('Monday_c :',count_mon),
         ('Weekly_Portion:',"{:.1%}".format(count_mon/count_all_trips)))
print   (('Tuesday_c :',count_tue),
         ('Weekly_Portion:',"{:.1%}".format(count_tue/count_all_trips)))
print   (('Wedensday_c :',count_wen),
         ('Weekly_Portion:',"{:.1%}".format(count_wen/count_all_trips)))
print   (('Thursday_c:',count_thu),
                    ('Weekly_Portion:',"{:.1%}".format(count_thu/count_all_trips)))
            print   (('Friday_c :',count_fri),
                    ('Weekly_Portion:',"{:.1%}".format(count_fri/count_all_trips)))

            print   (('WEEKENDS_COUNT :',count_weekends),
                    ('Weekly_Portion:',"{:.1%}".format(count_weekends/count_all_trips)))
            print   (('Saturday_c :',count_sat),
                    ('Weekly_Portion:',"{:.1%}".format(count_sat/count_all_trips)))
            print   (('Sunday_c  :',count_sun),
                    ('Weekly_Portion:',"{:.1%}".format(count_sun/count_all_trips)))
    weekly_trip_stat('./data/NYC-2016-Summary.csv')
    weekly_trip_stat('./data/Washington-2016-Summary.csv')
    weekly_trip_stat('./data/Chicago-2016-Summary.csv')
```

```
City: NYC
('WEEKDAYS_COUNT :', 212093) ('Weekly_Portion:', '76.6%')
('Monday_c :', 39340) ('Weekly_Portion:', '14.2%')
('Tuesday_c :', 42405) ('Weekly_Portion:', '15.3%')
('Wedensday_c :', 44629) ('Weekly_Portion:', '16.1%')
('Thursday_c:', 44330) ('Weekly_Portion:', '16.0%')
('Friday_c :', 41389) ('Weekly_Portion:', '15.0%')
('WEEKENDS_COUNT :', 64705) ('Weekly_Portion:', '23.4%')
('Saturday_c :', 33353) ('Weekly_Portion:', '12.0%')
('Sunday_c  :', 31352) ('Weekly_Portion:', '11.3%')

City: Washington
('WEEKDAYS_COUNT :', 49199) ('Weekly_Portion:', '74.2%')
('Monday_c :', 9394) ('Weekly_Portion:', '14.2%')
('Tuesday_c :', 9748) ('Weekly_Portion:', '14.7%')
('Wedensday_c :', 10103) ('Weekly_Portion:', '15.2%')
('Thursday_c:', 9984) ('Weekly_Portion:', '15.1%')
('Friday_c :', 9970) ('Weekly_Portion:', '15.0%')
('WEEKENDS_COUNT :', 17127) ('Weekly_Portion:', '25.8%')
('Saturday_c :', 8900) ('Weekly_Portion:', '13.4%')
('Sunday_c  :', 8227) ('Weekly_Portion:', '12.4%')

City: Chicago
('WEEKDAYS_COUNT :', 52550) ('Weekly_Portion:', '72.9%')
('Monday_c :', 11286) ('Weekly_Portion:', '15.6%')
('Tuesday_c :', 10911) ('Weekly_Portion:', '15.1%')
('Wedensday_c :', 9604) ('Weekly_Portion:', '13.3%')
('Thursday_c:', 10008) ('Weekly_Portion:', '13.9%')
('Friday_c :', 10741) ('Weekly_Portion:', '14.9%')
('WEEKENDS_COUNT :', 19581) ('Weekly_Portion:', '27.1%')
('Saturday_c :', 9927) ('Weekly_Portion:', '13.8%')
('Sunday_c  :', 9654) ('Weekly_Portion:', '13.4%')
```

1.6 Question 4

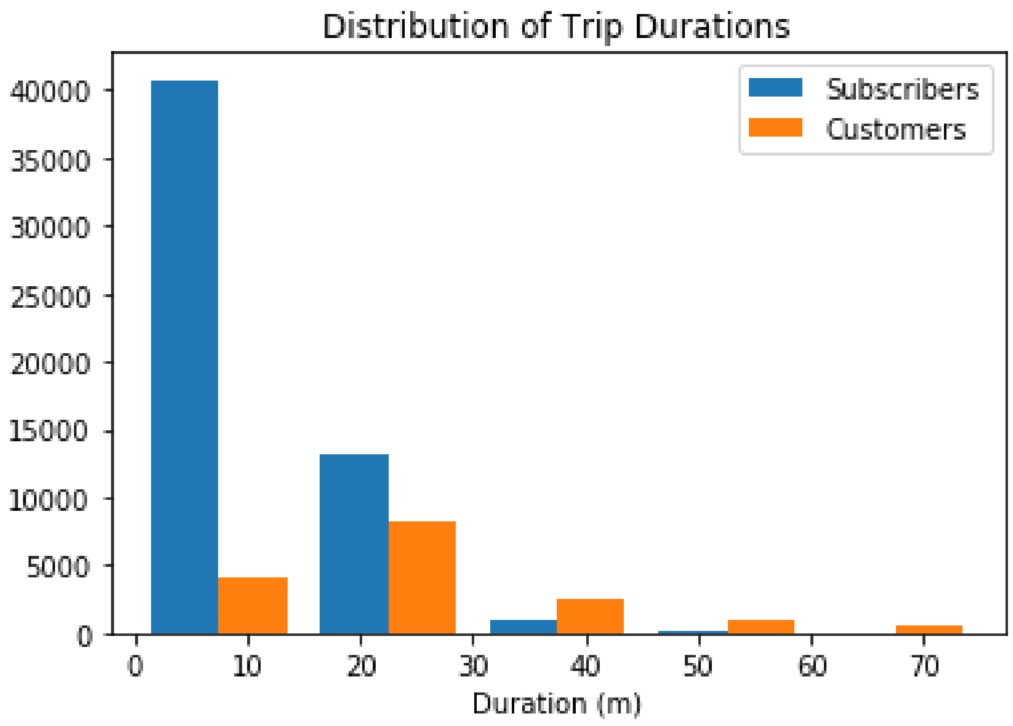
** NYC:** The number of trips is generally higher on **Wednesdays (16.12%)** and the lowest number of trips are observed in **Sundays(11.32%)**.

** WASHINGTON DC :** The number of trips is generally higher on **Wednesdays (15.22%)** and the lowest number of trips are observed in **Sundays (12.4%)**.

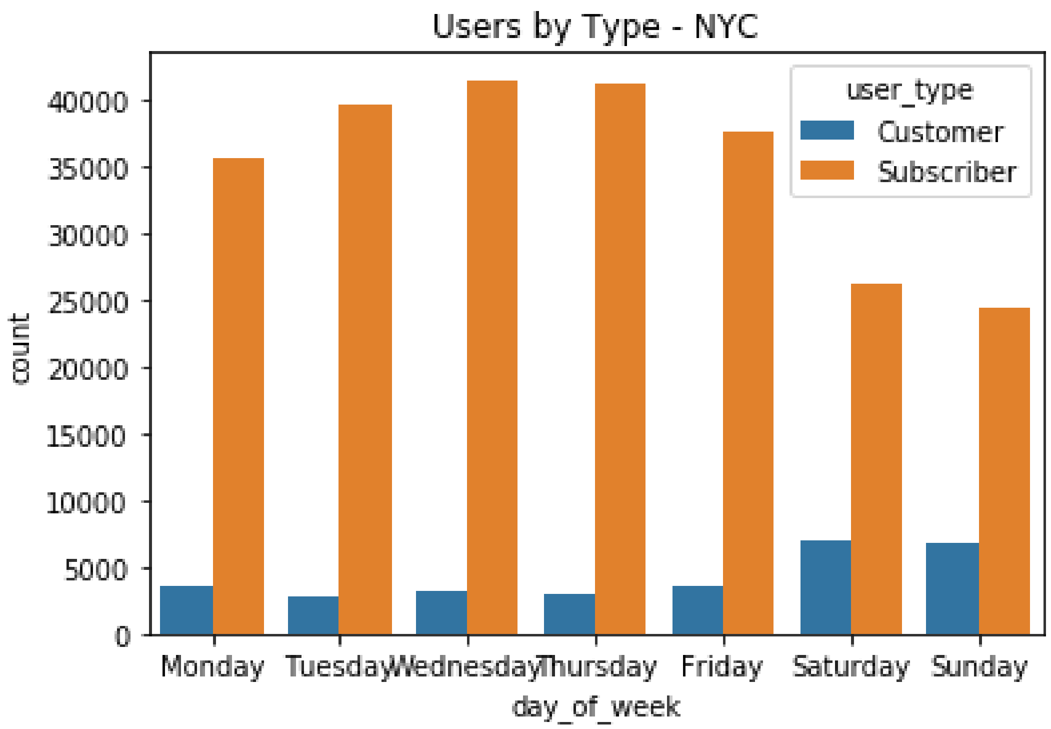
** CHICAGO :** The number of trips is generally higher on **Mondays (15.6%)** and the lowest number of trips are observed in ** Wednesdays (13.3%)**.

2 Visualizations

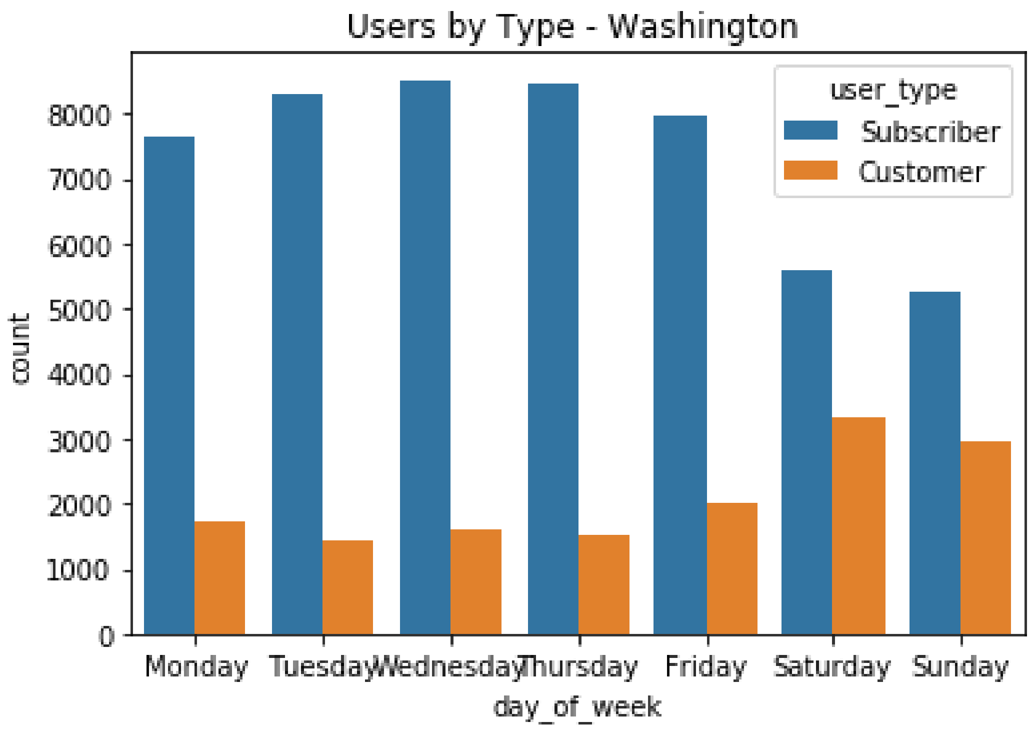
```
In [40]: import matplotlib.pyplot as plt

         %matplotlib inline
         with open(filename, 'r') as f_in:
             dictreader = csv.DictReader(f_in)

             list_of_duration_cus = []
             list_of_duration_subs = []
             for row in dictreader:
                 duration = row['duration']
                 if row['user_type']=='Subscriber':
                     list_of_duration_subs.append(float(duration))
                 else:
                     list_of_duration_cus.append(float(duration))

             plt.hist(x = (list_of_duration_subs, list_of_duration_cus), bins = 5, range = (0, 7
             plt.title('Distribution of Trip Durations')
             plt.xlabel('Duration (m)')

             plt.legend(('Subscribers', 'Customers'), loc='best')
             plt.show()
```



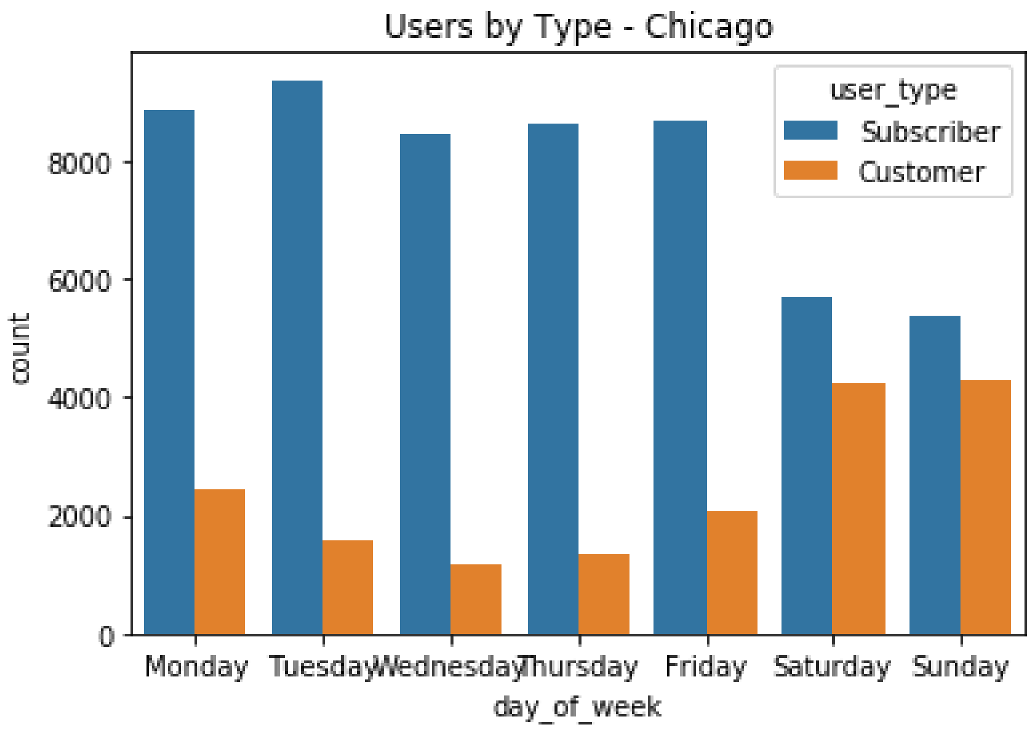
```
In [44]: import seaborn as sns
         data=pd.read_csv('./data/NYC-2016-Summary.csv')
         sns.countplot(x='day_of_week',data=data,
                      order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sund
                     hue='user_type' )
         plt.title ('Users by Type - NYC');
```



```
In [45]: data=pd.read_csv('./data/Washington-2016-Summary.csv')

         sns.countplot(x='day_of_week',data=data,
                      order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sund
                     hue='user_type' )
         plt.title ('Users by Type - Washington ');
```



```
In [48]: data=pd.read_csv('./data/Chicago-2016-Summary.csv')
         sns.countplot(x='day_of_week',data=data,
                      order=['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sund
                     hue='user_type' )
         plt.title ('Users by Type - Chicago');
```



```
In [203]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Bike_Share_Analysis.ipynb'])

Out[203]: 0
```