citiBikeExploratoryAnalysis

July 30, 2015

We undergo an initial exploratory analysis of the CitiBike NYC bike rental program. The goal here is to get an intial inspection of the data, look for interesting trends that may also require further study and analysis. Ideas for further analysis are also presented, to be implemented at a later date.

-Richard Petti

First we generate a file list and load the files. We store the contents into a list of dataframes, each element in the list corresponding to data from a different month (file)

In [1]:
```
import glob
import pandas

filelist = glob.glob("C:\Users\Rich\Documents\data\*.csv")

dfList = []
for item in filelist:
    result = pandas.read_csv(item)
    dfList.append(result)
```

Now make a list of the available months of data based on the existing files, to be used in a later plot

In [2]:
```
import datetime
import os

monthlist = []
for item in filelist:
    path, file = os.path.split(item)
    try:
        monthlist.append(datetime.datetime(int(file[0:4]), int(file[5:7]),1))
    except ValueError:
        monthlist.append(datetime.datetime(int(file[0:4]), int(file[4:6]),1))
```

Now lets just view the format of the data and see what is available

In [3]:
```
dfList[0].head()
```

Out[3]:

	tripduration	starttime	stoptime	start station id	start station name	start station latitude	start station longitude
0	634	2013-07-01 00:00:00	2013-07-01 00:10:34	164	E 47 St & 2 Ave	40.753231	-73.970325
1	1547	2013-07-01 00:00:02	2013-07-01 00:25:49	388	W 26 St & 10 Ave	40.749718	-74.002950
2	178	2013-07-01 00:01:04	2013-07-01 00:04:02	293			
3	1580	2013-07-01 00:01:06	2013-07-01 00:27:26	531			
4	757	2013-07-01 00:01:10	2013-07-01 00:13:47	382			

```
2    Lafayette St & E 8 St               40.730287               -73.990765
3   Forsyth St & Broome St               40.718939               -73.992663
4  University Pl & E 14 St               40.734927               -73.992005

   end station id         end station name  end station latitude  \
0             504          1 Ave & E 15 St             40.732219
1             459         W 20 St & 11 Ave             40.746745
2             237          E 11 St & 2 Ave             40.730473
3             499       Broadway & W 60 St             40.769155
4             410  Suffolk St & Stanton St             40.720664

   end station longitude  bikeid    usertype birth year  gender
0             -73.981656   16950    Customer         \N       0
1             -74.007756   19816    Customer         \N       0
2             -73.986724   14548  Subscriber       1980       2
3             -73.981918   16063    Customer         \N       0
4             -73.985180   19213  Subscriber       1986       1
```

I suspect that the missing age/gender information is assocatied with being a customer, rather than a subscriber. Verify this here. Just check on first data file. This could be a useful distinction.

In [4]:
```
foundone = 0

for index, row in dfList[0].iterrows():
    if row['gender']=='0' and row['usertype']=='Subscriber':
        print 'found an exception to my assumption, found a Subscriber with gender 0'
        foundone += 1
    if row['usertype']=='Customer' and (row['gender']=='1' or row['gender']=='2'):
        print 'found an exception to my assumption, found a Customer with a defined gender'
        foundone += 1
if foundone == 0:
    print "no exceptions found, missing info corresponds to being a Customer rather than a Subs
```

```
no exceptions found, missing info corresponds to being a Customer rather than a Subscriber
```

Ok great, we confirmed that assumption.

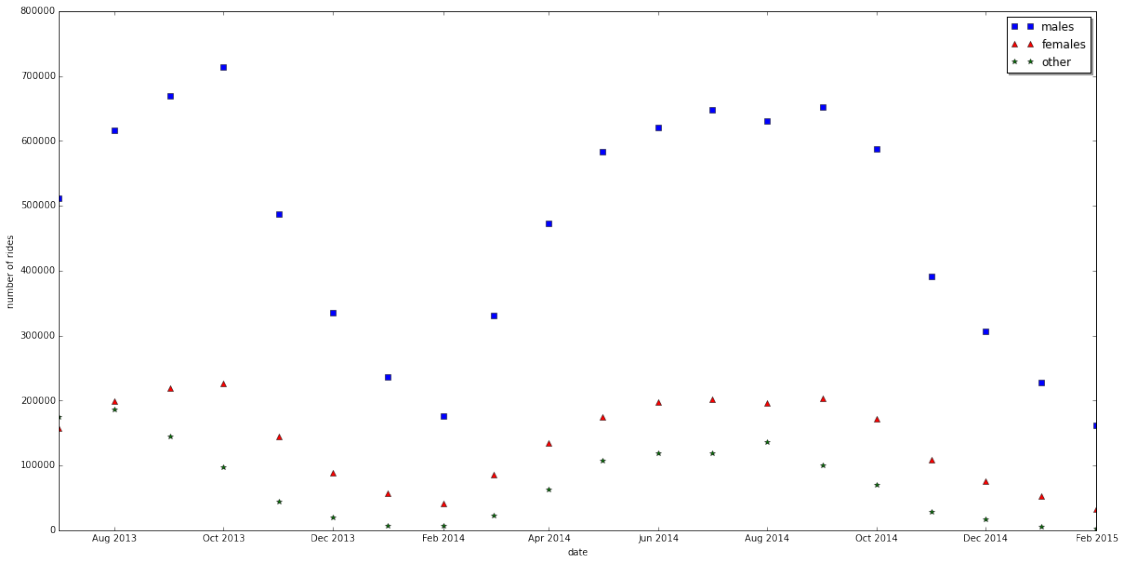
Let us get an idea of the demographic who uses the bike rentals. Let us plot the montly use as a function of time for men (1) and women (2) separately (along with the undefined Customers).

In [3]:
```
rideCounts_male = []
rideCounts_female = []
rideCounts_other = []

for item in dfList:
    rideCounts_male.append( item[item['gender']==1]['tripduration'].count())
    rideCounts_female.append( item[item['gender']==2]['tripduration'].count())
    rideCounts_other.append( item[item['gender']==0]['tripduration'].count())
```

In [6]:
```
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(20,10))
ax.plot(monthlist, rideCounts_male, 'bs', label='males')
ax.plot(monthlist, rideCounts_female, 'r^', label='females')
ax.plot(monthlist, rideCounts_other, 'g*', label='other')
```

```
    legend = ax.legend(loc='upper right', shadow=True)
    plt.ylabel('number of rides')
    plt.xlabel('date')
```

```
Out[6]: <matplotlib.text.Text at 0x1e89908>
```

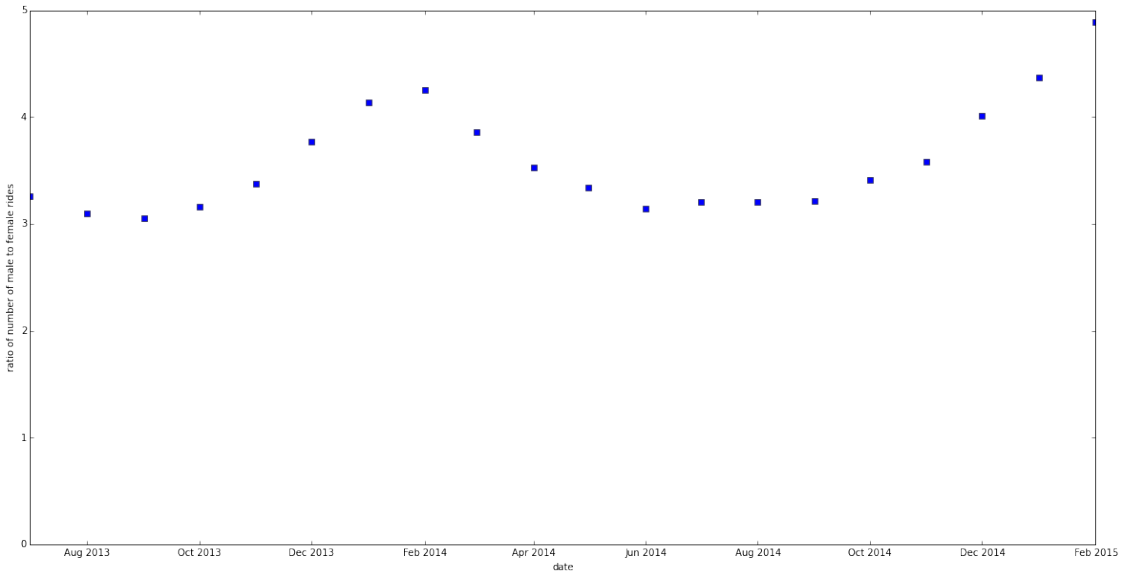


A few general observations about the above plot... 1) generally more than three as many male users as female 2) nearly all users are subscribers (gender = other category) 3) clearly observe the pattern of use due to weather (riding peaks in fall, lowest in winter)

It would be useful to come up with a stradegy to take out the weather dependence. Maybe look at the ratio of male to female riders since the usage presumably scales equally with weather conditions.

```
In [11]: import matplotlib.pyplot as plt
         %matplotlib inline
         rideCounts_ratio = []
         for i in range(0, len(rideCounts_male)):
             rideCounts_ratio.append(float(rideCounts_male[i])/float(rideCounts_female[i]))

         fig, ax = plt.subplots(figsize=(20,10))
         ax.plot(monthlist, rideCounts_ratio, 'bs')
         ax.set_ylim([0,5])
         legend = ax.legend(loc='upper right', shadow=True)
         plt.ylabel('ratio of number of male to female rides')
         plt.xlabel('date')
```

```
Out[11]: <matplotlib.text.Text at 0xc6030f28>
```



From the above plot, it looks like the effect of weather is still not taken out completely in the ratio. Additionallym there does not seem to be a strong indication of an increase or decrease in use over time, though the time span of the data is rather small.

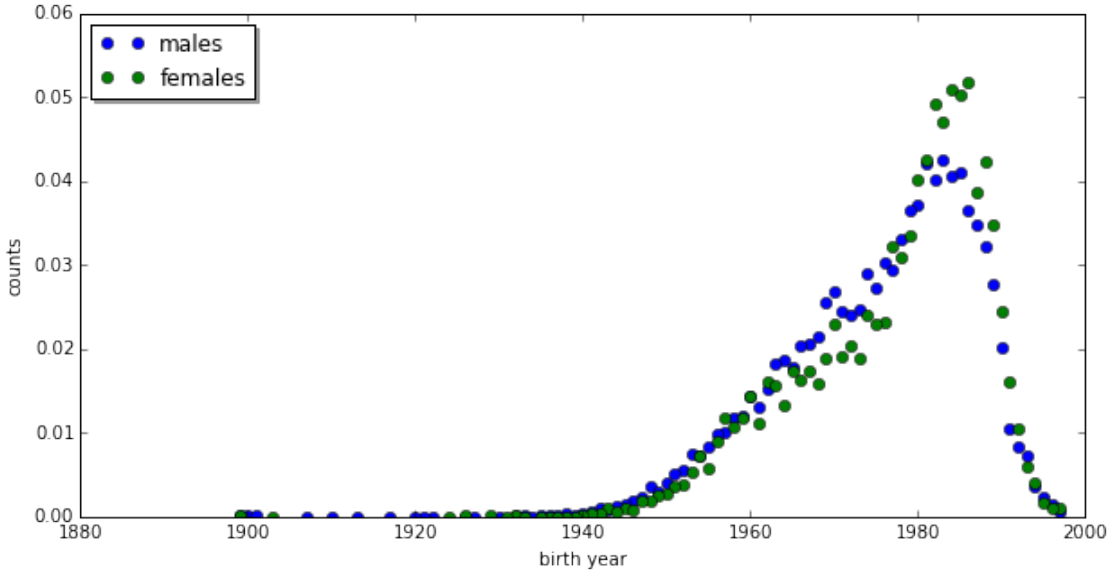
Lets look at the kind of age groups that use the service. Ideally the calculation would be done in parallel for each file (from my current analysis experience at BNL, I would use condor, but I would also be willing to learn mapreduce). But without those resources at home, we will just carry out this analysis for one specific month of data (September 2013).

```
In [1]: import glob
        import pandas

        filelist = glob.glob("C:\Users\Rich\Documents\data\*.csv")

        result = pandas.read_csv(filelist[2],na_values='\\N')
```

First break the data down into genders. I want to get an idea of the age distribution and how that may differ depending on gender.

```
In [50]: import matplotlib.pyplot as plt
         %matplotlib inline

         df_male = result[result.gender==1]
         df_female = result[result.gender==2]
         df_other = result[result.gender==0]

In [60]: df_male_by = df_male['birth year'].value_counts()
         df_female_by = df_female['birth year'].value_counts()
         df_other_by = df_other['birth year'].value_counts()
         #print df_male_by.index.values

         df_male_by = df_male_by/sum(df_male_by)
         df_female_by = df_female_by/sum(df_female_by)
         #print df_male_by.values
```

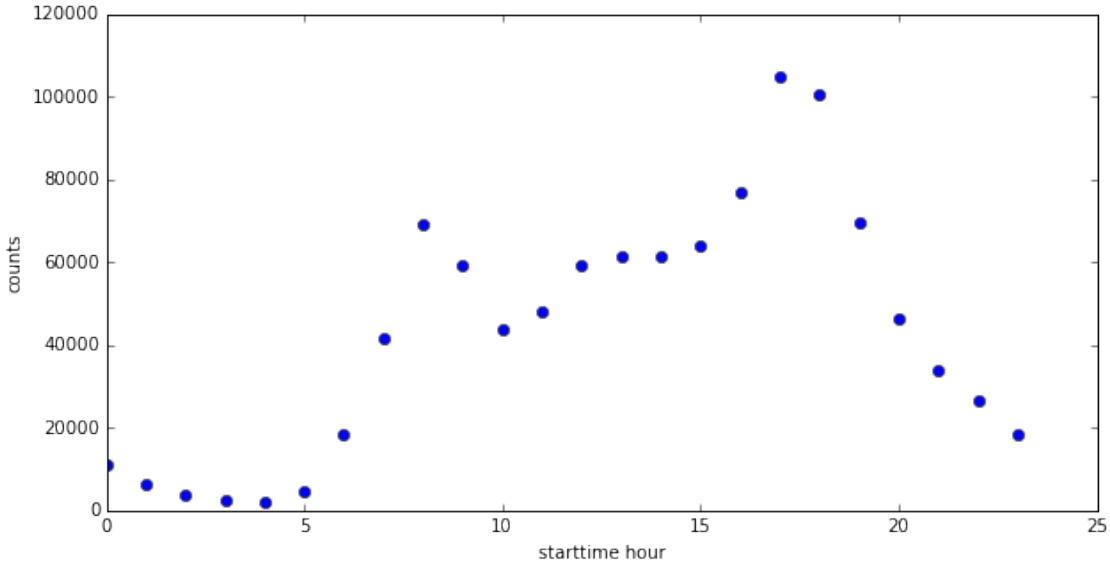
```
In [64]: fig, ax = plt.subplots(figsize=(10,5))
         ax.plot(df_male_by.index.values,df_male_by.values, 'o',label='males')
         ax.plot(df_female_by.index.values,df_female_by.values, 'o',label='females')
         legend = ax.legend(loc='upper left', shadow=True)
         plt.ylabel('counts')
         plt.xlabel('birth year')

Out[64]: <matplotlib.text.Text at 0x7d8a8550>
```



Ok. We normalize the two distributions here, keeping in mind that we already know there are more male than female rides. It appears that the two populations have different age distributions. Rides taken by females tend to be slightly younger than male riders. Both appear to be non-gaussian in shape, and by eye look as if each can be decomposed into two gaussians. It would be interesting to investigate this further and see if there are two major age groups that ride.

Next we investigate the distribution of the start time of the ride, broken down by hour.

```
In [105]: from datetime import datetime

          result_time = result
          result_time['starttime'] = pandas.to_datetime(result_time['starttime'], format='%Y-%m-%d %H:%M
          result_time['starttime'] = result_time['starttime'].apply(lambda x: x.hour)

          df_times = result_time['starttime'].value_counts()
          #print df_times

In [98]: fig, ax = plt.subplots(figsize=(10,5))
         ax.plot(df_times.index.values,df_times.values, 'o')
         legend = ax.legend(loc='upper left', shadow=True)
         plt.ylabel('counts')
         plt.xlabel('starttime hour')

Out[98]: <matplotlib.text.Text at 0x38c109e8>
```



It looks as though there are two major spikes in ride start time, one around 8 or 9am and the other around 6pm. This most likely corresponds to the start and end of the normal workday, indicating that many people use the bike system to get back and forth from work. Though this cannot account for all use, since the 6pm peak is higher than the 8pm peak. It might be interesting to look at this more diffentially, say the distribution on weekends as opposed to weekends. Maybe even break it down by season.

Now let us look at some differences between round trip and single trip rides. Round trip is defined as a ride that has the same end station id as start station id.

```
In [106]: df_roundtrip = result[result['start station id']==result['end station id']]
          df_singletrip = result[result['start station id']!=result['end station id']]

In [129]: print df_roundtrip.size
          print df_singletrip.size

          print float(df_roundtrip.size)/(float(df_singletrip.size)+float(df_roundtrip.size))
```

```
398400
15116985
0.0256777385801
```

As can be seen above, there are many more single trip rides than round trip. This is not very surprising. About 2.5% of the total rides are round trip rides.

```
In [131]: print df_roundtrip.tripduration.mean(), ' ', df_roundtrip.tripduration.std()
          print df_singletrip.tripduration.mean(), ' ', df_singletrip.tripduration.std()
```

```
1383.39506777   4288.69651843
892.413274869   1582.48063247
```

In the above output, we see that round trip rides are longer on average than single trip rides, thoughthe huge standard deviation is suspicious. We should look at the distributions.

Finally, it would be interesting to get an idea of where people are headed. Below we plot the direction vector of travel from start station to end station calculated by the lattitude and longitude coordinates of the start and end station.

In [2]:
```
import numpy as np

#print result

#soa = np.array(result[['start station latitude','start station longitude','end station latitud
soa = np.array(result[['start station latitude','start station longitude','end station latitude
X,Y,U,V = zip(*soa)

print max(X)
print min(X)
print max(Y)
print min(Y)

print soa
```

```
40.770513
40.680342423
-73.9500479759
-74.01713445
[[ 40.73532427 -73.99800419  40.71542197 -74.01121978]
 [ 40.7218158  -73.99720307  40.73704984 -73.99009296]
 [ 40.76340613 -73.97722479  40.739323   -74.008119  ]
 ...,
 [ 40.73028666 -73.9907647   40.74206539 -74.00443172]
 [ 40.76030096 -73.99884222  40.75320159 -73.9779874 ]
 [ 40.70834698 -74.01713445  40.70834698 -74.01713445]]
```

In [3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
#X = 0
#Y = 0
#U = 5
#V = 10
plt.figure(figsize=(20,10))
ax = plt.gca()
ax.quiver(X,Y,U,V,angles='xy',scale_units='xy',scale=1)
ax.set_xlim([min([min(U),min(X)]),max([max(X),max(U)])])
ax.set_ylim([min([min(V),min(Y)]),max([max(V),max(Y)])])
plt.draw()
plt.show()
```