Data Trend 0-5

April 17, 2017

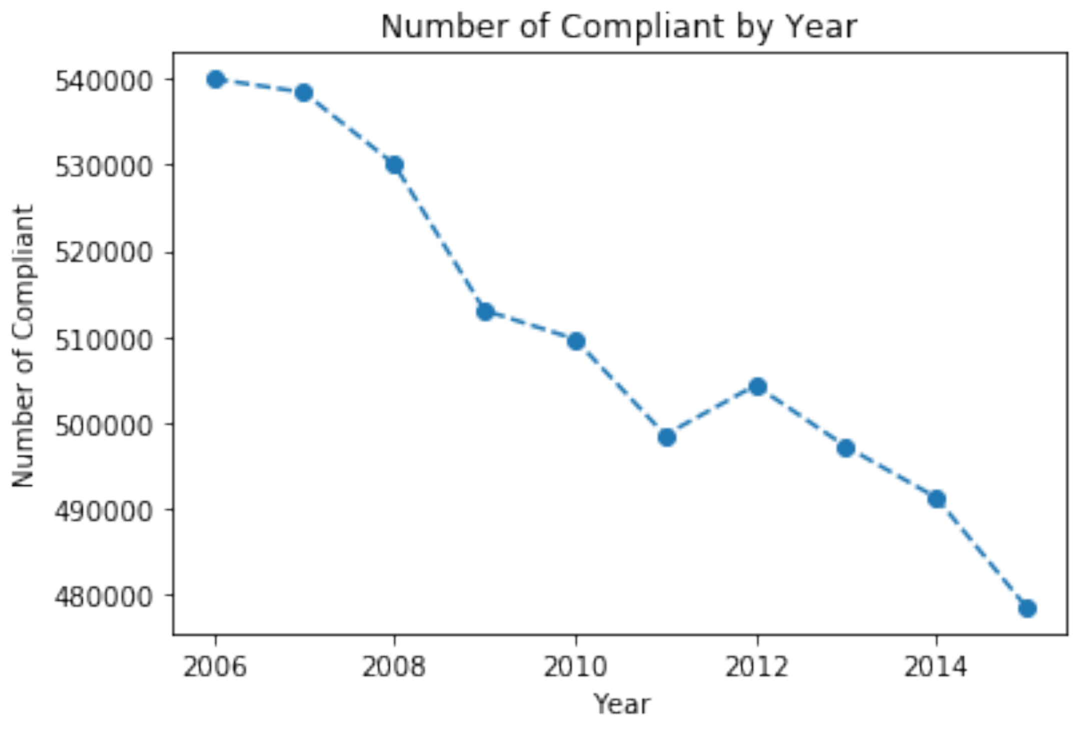
0.1 Some Trends

0.1.1 1. Number of Crimes by Year

Code: ***col5_report_year.py*** This code returns number of crimes by year.

```
In [1]: import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np

        %matplotlib inline

In [2]: df = pd.read_table('col5_report_year.out',header=-1)
        df = df.sort([0], ascending=True)
        plt.plot(list(df[0]),list(df[1]),linestyle='--', marker='o')
        plt.title("Number of Compliant by Year")
        plt.xlabel("Year")
        plt.ylabel("Number of Compliant")
        plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:2: FutureWarning: sort
  from ipykernel import kernelapp as app
```



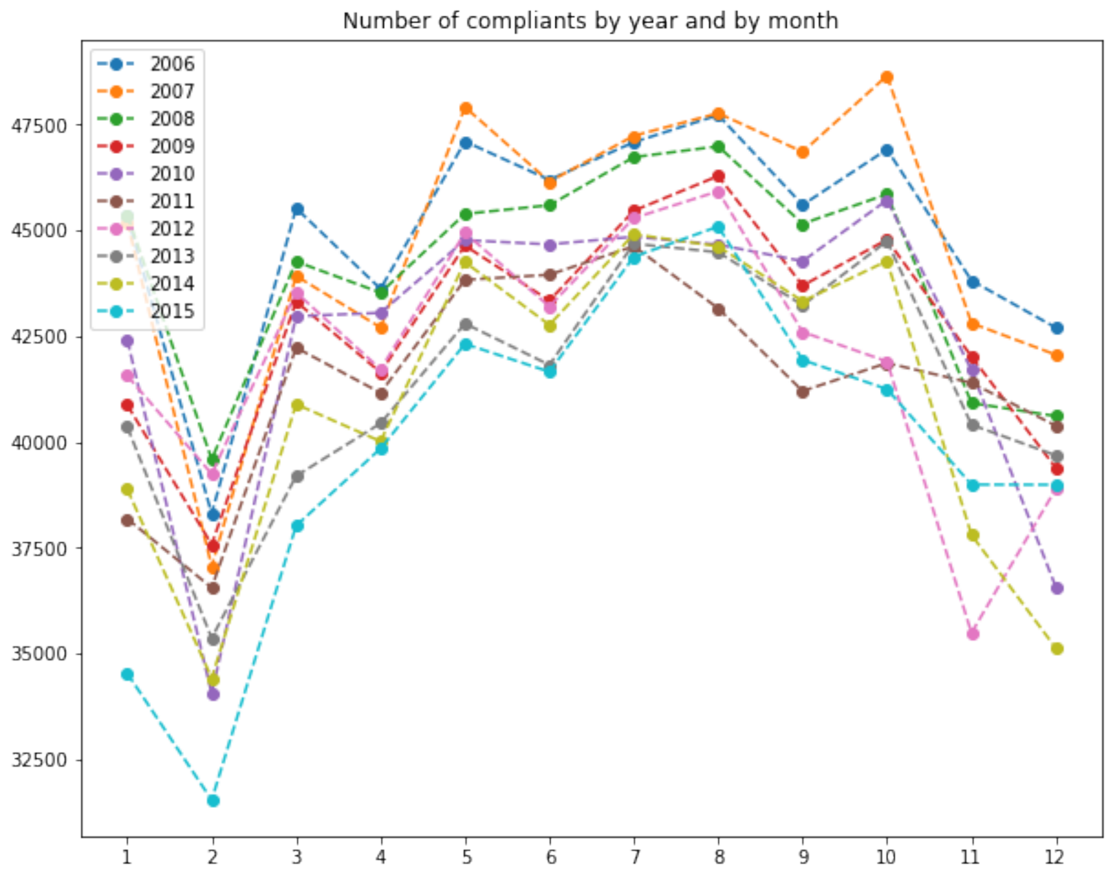
Trend: Overall, the crime decreased year over year. Our goal is to identify when, where, and why the crime decreased.

0.1.2 2. Number of crimes by year and by month

Code: ***col5_report_year_month.py*** This code returns number of crimes by each year and each month.

```
In [3]: fig = plt.figure(figsize=(10, 8))
        df = pd.read_table('col5_report_year_month.out',header=-1)
        df = df.sort([0,1], ascending=True)
        for i in df[0].unique():
            plt.plot(list(df[df[0]==i][1]),list(df[df[0]==i][2]),linestyle='--', marker='o',labe
        plt.xticks(list(range(1,13)))
        plt.legend(loc="upper left")
        plt.title("Number of compliants by year and by month")
        plt.show()
```

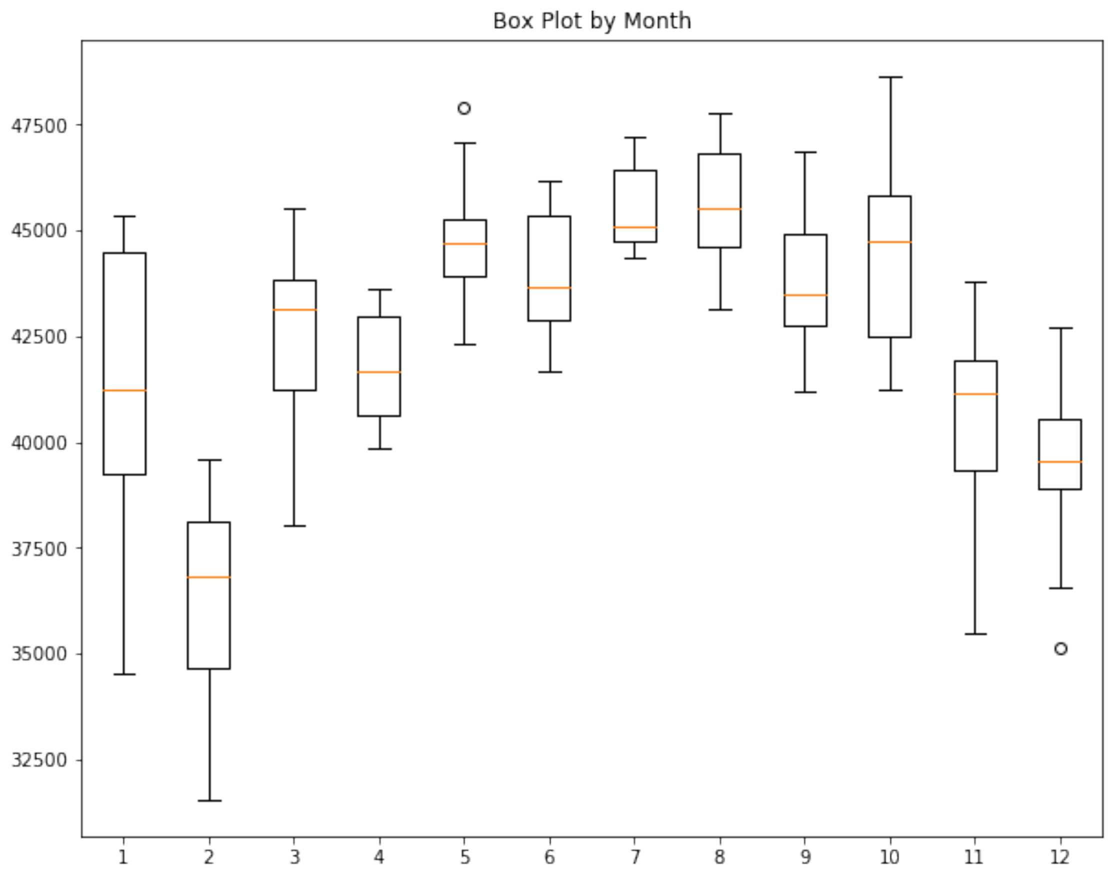
```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```



It seems that Jan and Feb in 2015 has some abnormal number of crimes compared to the other month.

I also plot a boxplot and clearly Feburay has the lowest number of crimes on average (partialy is becasue of a shorter month). It seems there is an outliar in May and December.

```
In [4]: fig = plt.figure(figsize=(10, 8))
        month = list()
        for i in (list(range(1,13))):
            month.append(df[df[1]==i][2])

        plt.boxplot(month)
        plt.title("Box Plot by Month")
        plt.show()
```

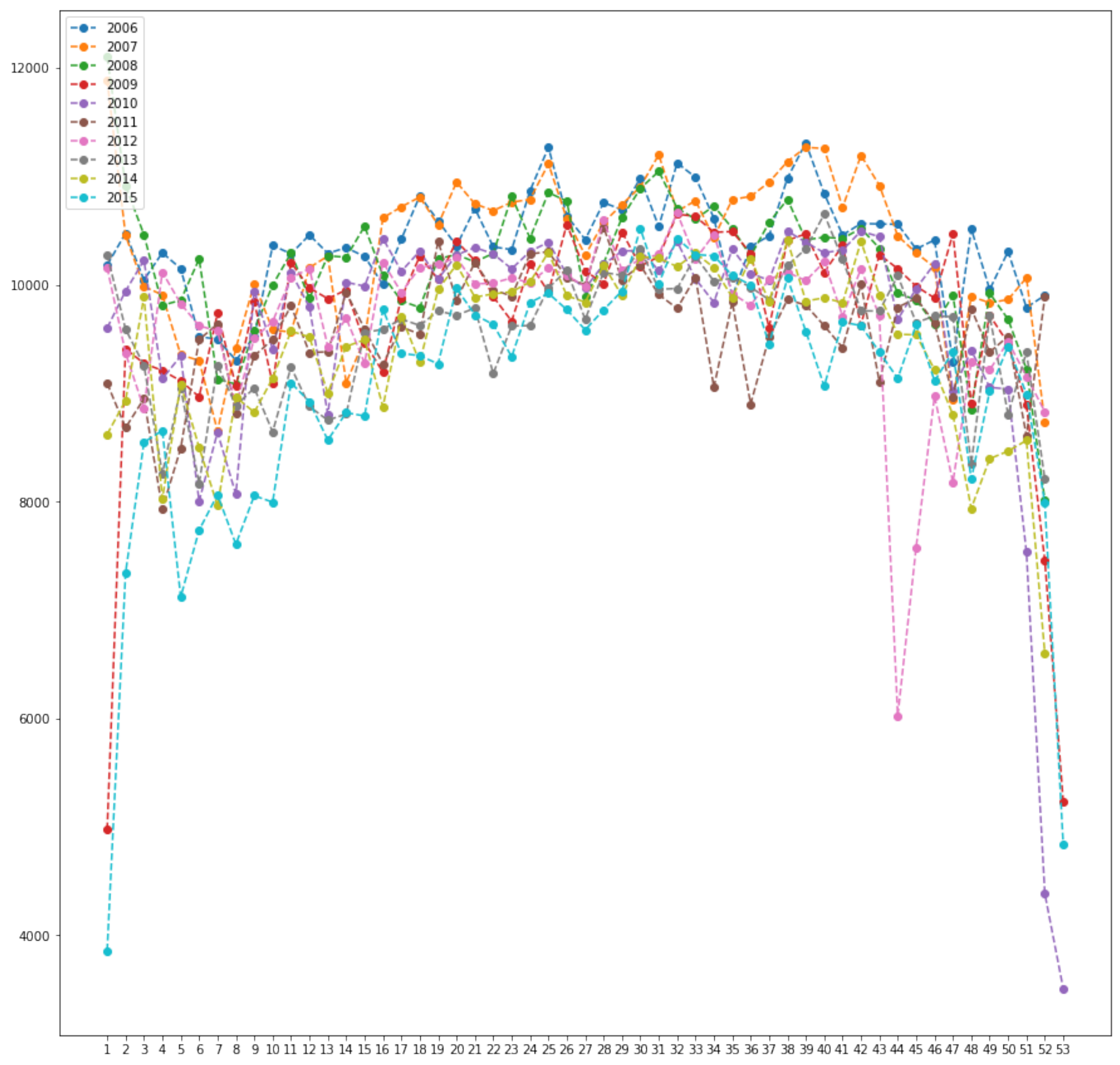


0.1.3 3. Number of crimes by year and by week

Code: ***col5_report_year_week.py*** This code provides number of crimes by year and by week.

```
In [5]: fig = plt.figure(figsize=(15, 15))
        df = pd.read_table('col5_report_year_week.out',header=-1)
        df = df.sort([0,1], ascending=True)
        for i in df[0].unique():
            plt.plot(list(df[df[0]==i][1]),list(df[df[0]==i][2]),linestyle='--', marker='o',labe
        plt.legend(loc="upper left")
        plt.xticks(list(df[1].unique()))
        plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```



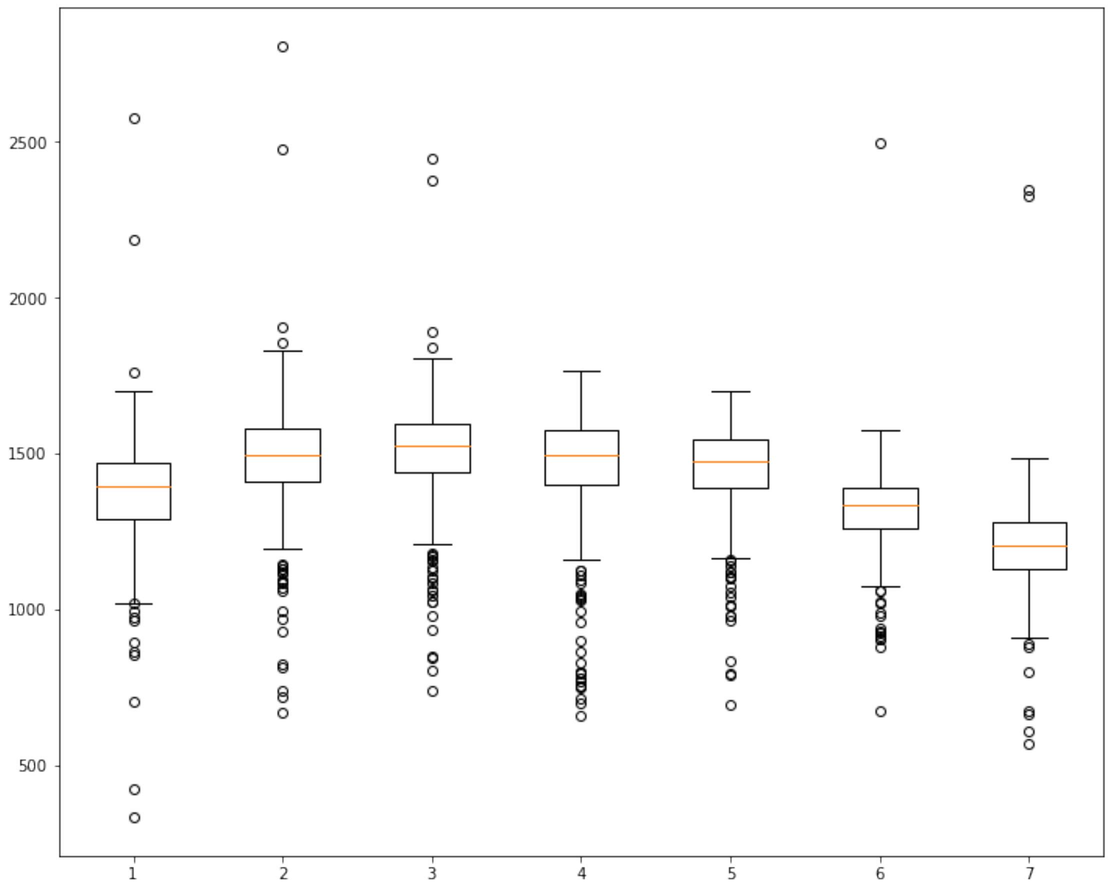
After plotting by week, it is clear that the week 44 in 2012 has an abnormal trend. It is probably because of hurricane sandy happened during that period.

0.1.4 4. Number of crimes by day of week

Code: ***col5_report_year_week_day.py*** This code provides number of crimes by year and weeknum in each day of the week.

```
In [6]: fig = plt.figure(figsize=(12, 10))
        df = pd.read_table('col5_report_year_week_day.out',header=-1)
        df = df.sort([0,1], ascending=True)
        dow = list()
        for i in (list(range(1,8))):
            dow.append(df[df[2]==i][3])
        plt.boxplot(dow)
        plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```



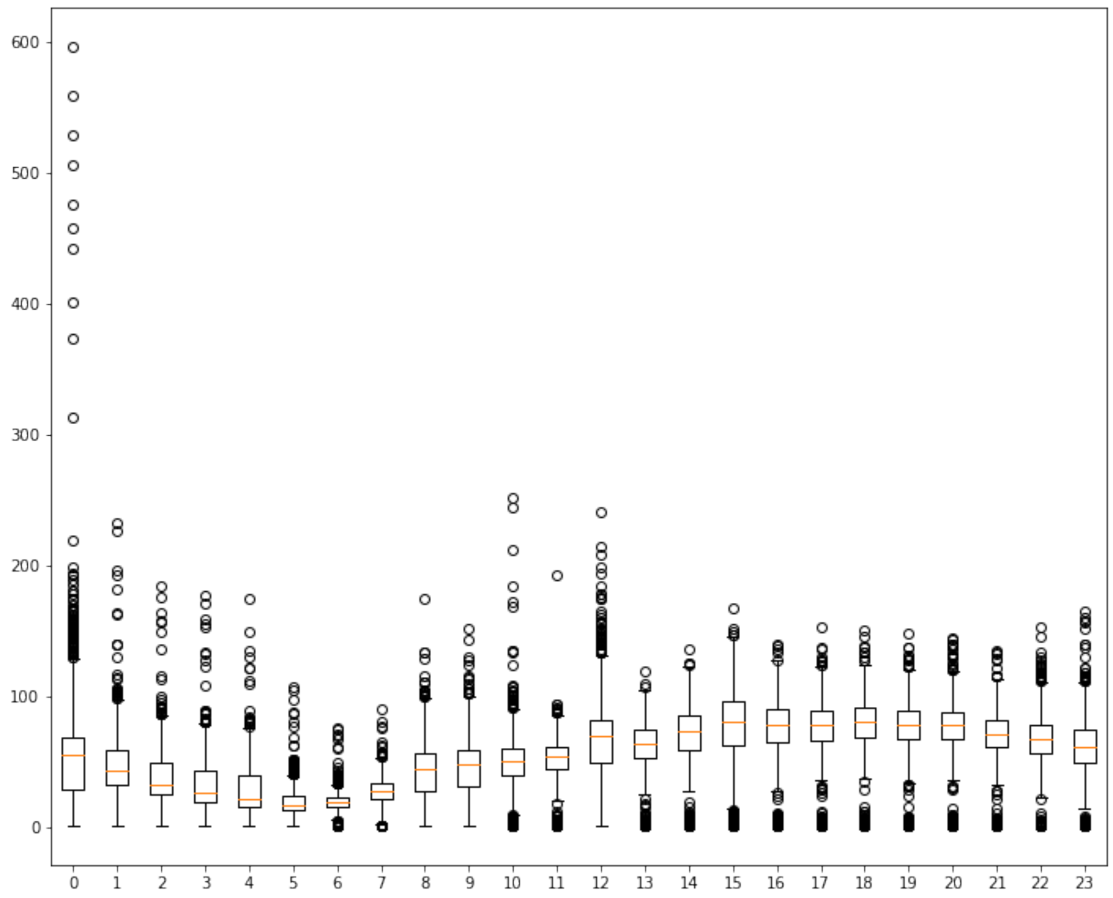
By looking at the boxplot, Tue-Fri is the peak and weekend has less compliants. Also, weekdays have more outliars.

0.1.5 5. Number of compliants by hour

Code: ***col1_2_hour.py***. This code is to check number of crimes by hour.

```
In [7]: fig = plt.figure(figsize=(12, 10))
        df = pd.read_table('col1_2_hour.out',header=-1)
        df = df.sort([0,1,2,3], ascending=True)
        dow = list()
        for i in (list(range(0,24))):
            dow.append(df[df[3]==i][4])
        plt.boxplot(dow)
        plt.xticks(list(range(1,25)),list(range(0,24)))
        plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```



Number of crimes is gradually increasing after 7 am and then start decreasing after midnight. However, there are some very high abnormal outliars happened between 12 - 1 am.

```
In [8]: dfzero = df[df[3]==0]
        dfoutliar = dfzero[dfzero[4]>300]
        dfoutliar
```

```
Out[8]:          0   1  2  3    4
        17430  2006  52  7  0  506
        17454  2007   1  1  0  596
        26214  2008   1  2  0  529
        34926  2009   1  4  0  373
        52374  2010  53  5  0  475
        61134  2011  52  6  0  558
        69894  2012  52  7  0  442
        69942  2013   1  2  0  401
```

```
78702  2014   1  3  0  458
87390  2015   1  4  0  313
```

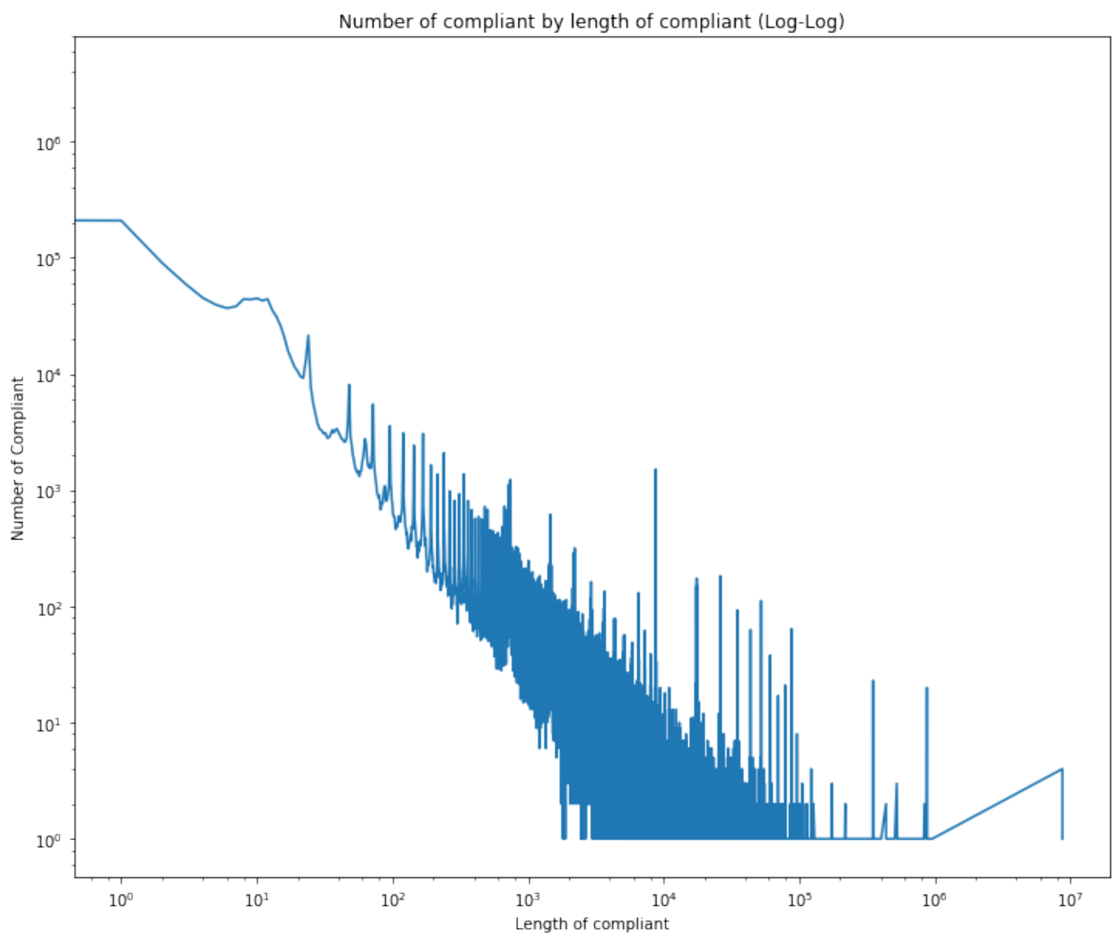
The outliars are usually between 12 and 1 am on the new year. It makes sense since the NYE celebrations.

0.1.6 6. Number of crimes by length of compliant

Code: ***length_of_compliant.py***. The code provides number of crimes by the length of complaint. When the crime has no to_date and to_time, the length will be counted as **0**.

```
In [9]: fig = plt.figure(figsize=(12, 10))
        df = pd.read_table('col1_2_3_4_length_of_compliant.out',header=-1)
        df = df.sort([0], ascending=True)

        plt.plot(list(df[0]),list(df[1]),linestyle='-')
        plt.title("Number of compliant by length of compliant (Log-Log)")
        plt.xlabel("Length of compliant")
        plt.ylabel("Number of Compliant")
        plt.xscale('log')
        plt.yscale('log')
        plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```



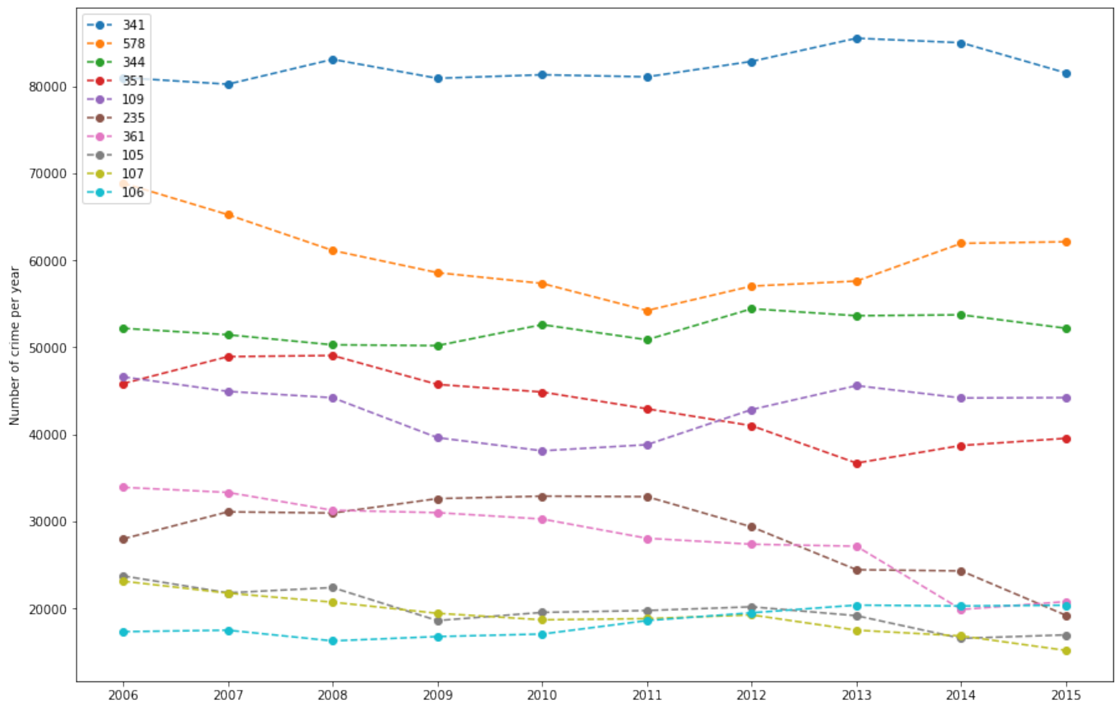
Overall,number of crimes decays over the length of the compliant after transforming to log-log.

0.1.7 7. Number of compliants by year and by crime type

Code: ***col5_6_report_kycd.py***. The code provides number of crimes by year and by KY_CD, which stands for the crime type.

```
In [10]: fig = plt.figure(figsize=(15, 10))
         df = pd.read_table('col5_6_report_year_kycd.out',header=-1)
         df.columns = ["year","kycd","count"]
         #find the top 10 KY_CD
         top_kycd = df.groupby(["kycd"], as_index=False).sum().sort(["count"], ascending=False)[
         df = df.sort(["year"],ascending=True)
         for i in list(top_kycd):
             plt.plot(list(df[df["kycd"]==i]["year"]),list(df[df["kycd"]==i]["count"]),linestyle
         plt.legend(loc="upper left")
         plt.xticks(list(df["year"].unique()))
         plt.ylabel("Number of crime per year")
         plt.show()
```

```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:5: FutureWarning: sort
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:6: FutureWarning: sort
```



From the list, the crime 341 is the most popluar and did not decrease over the year. It seems that crime 235 and 361 may contribute the year over year decrease as the trend continued going down after 2012.

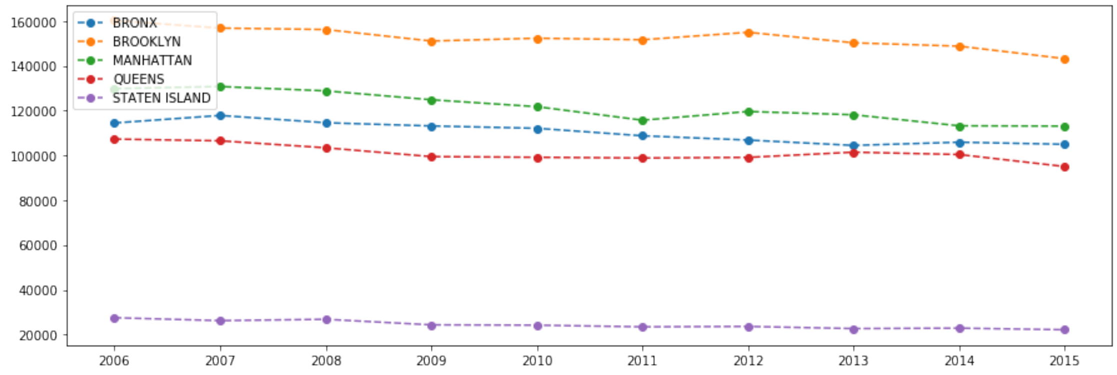
0.1.8 8. Number of compliants by year and by borough

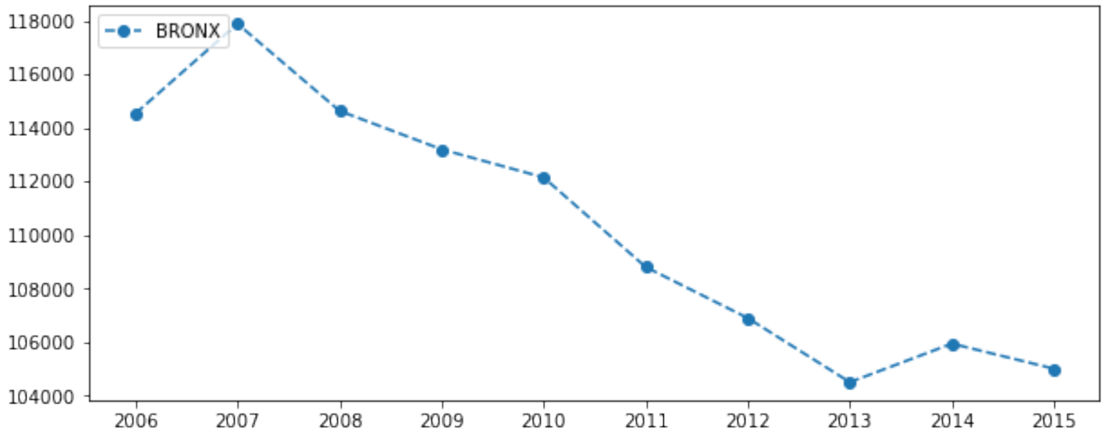
Code: ***col5_13_report_year_borough.py***. The code provides number of crime by number of compliants, year, and borough.

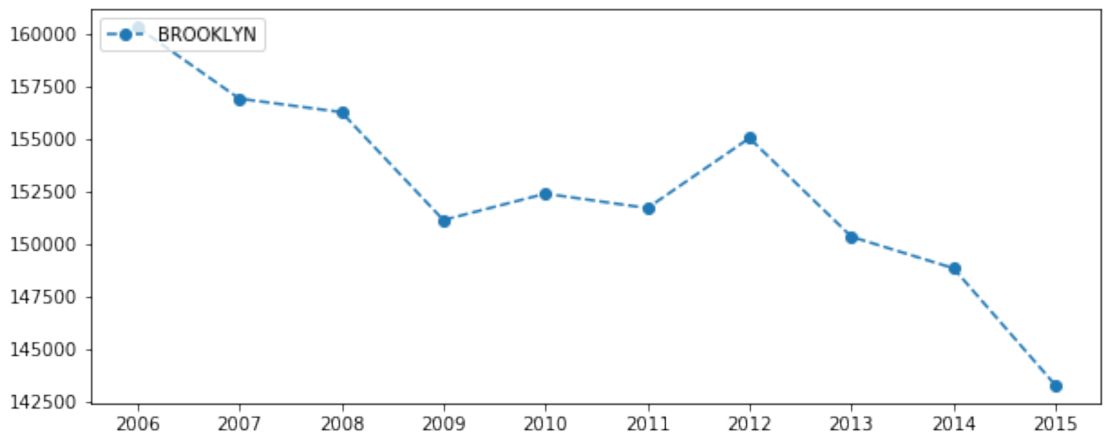
```
In [11]: fig = plt.figure(figsize=(15,5))
         df = pd.read_table('col5_13_report_year_borough.out',header=-1)
         df = df.sort([0,1], ascending=True)
         for i in df[1].unique():
             plt.plot(list(df[df[1]==i][0]),list(df[df[1]==i][2]),linestyle='--', marker='o',lab
         plt.legend(loc="upper left")
         plt.xticks(list(df[0].unique()))
         plt.show()



         for i in df[1].unique():
             fig = plt.figure(figsize=(10,4))
```

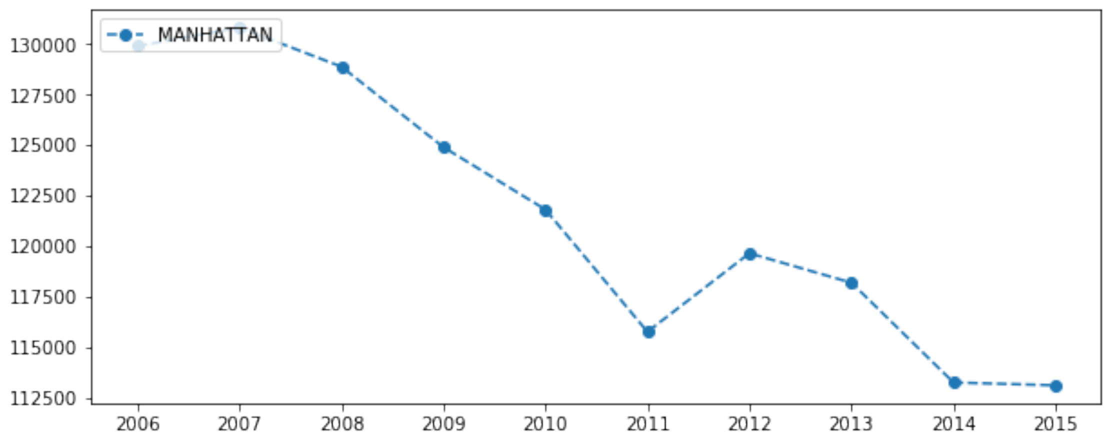
```
        plt.plot(list(df[df[1]==i][0]),list(df[df[1]==i][2]),linestyle='--', marker='o',lab
        plt.legend(loc="upper left")
        plt.xticks(list(df[0].unique()))
        plt.show()
```

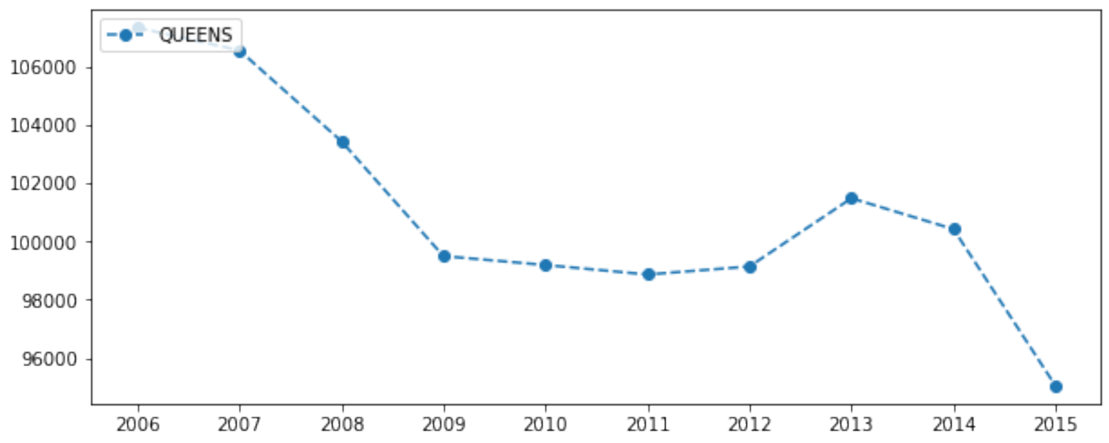
```
/Users/sunevan/anaconda/lib/python3.6/site-packages/ipykernel/__main__.py:3: FutureWarning: sort
  app.launch_new_instance()
```

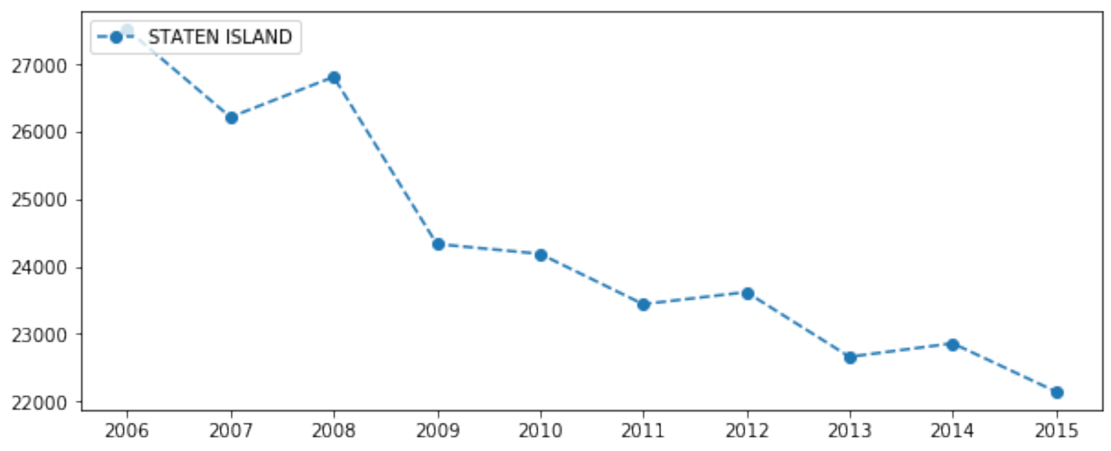





BROOKLYN
160000
157500
155000
152500
150000
147500
145000
142500
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015


MANHATTAN
130000
127500
125000
122500
120000
117500
115000
112500
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015


QUEENS
106000
104000
102000
100000
98000
96000
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015



Looking at the stacked the line chart, the decreasing trend is not very clear. After plotting the trend borough by borough, it is more clear that broklyn contributes the most to the decrease, followed by manhattan.

Data Trend 6-18

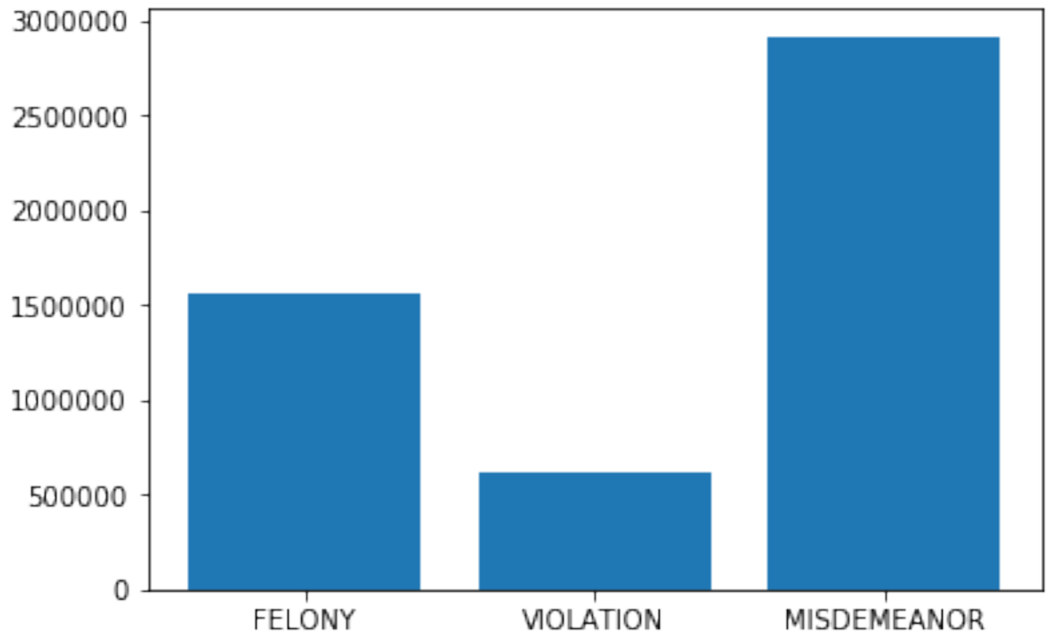
April 17, 2017

0.0.1 Column 11 - Offense Level

```
In [2]: data = pd.read_table('column11_data_quality.out', header = -1)
```

```
In [3]: stats = {}
        for level in ['FELONY','MISDEMEANOR','VIOLATION']:
            stats[level] = len(data[data[0]==level])
```

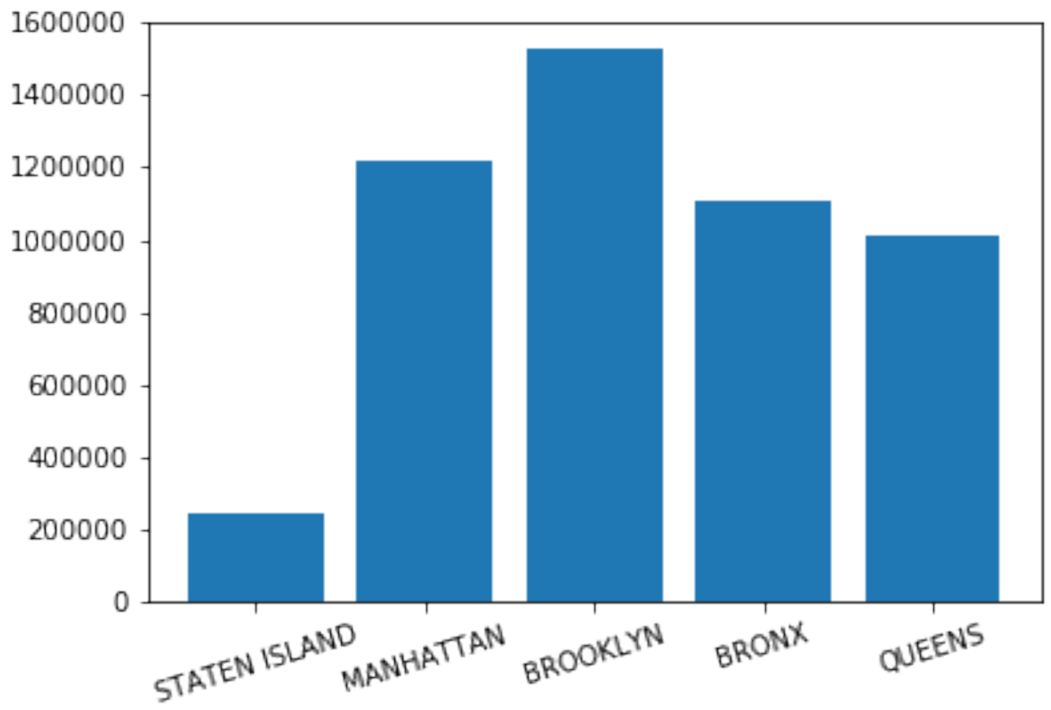
```
In [4]: plt.bar(range(len(stats)),stats.values(),align='center')
        plt.xticks(range(len(stats)), stats.keys())

        plt.show()
```



0.0.2 Column 13 - Borough Names

```
In [5]: data = pd.read_table('column13_data_quality.out', header = -1)
In [6]: stats = {}
        for boro in ['BRONX','BROOKLYN','MANHATTAN','QUEENS','STATEN ISLAND']:
            stats[boro] = len(data[data[0]==boro])
```

```
In [7]: plt.bar(range(len(stats)),stats.values(),align='center')
        plt.xticks(range(len(stats)), stats.keys(), rotation=17)

        plt.show()
```

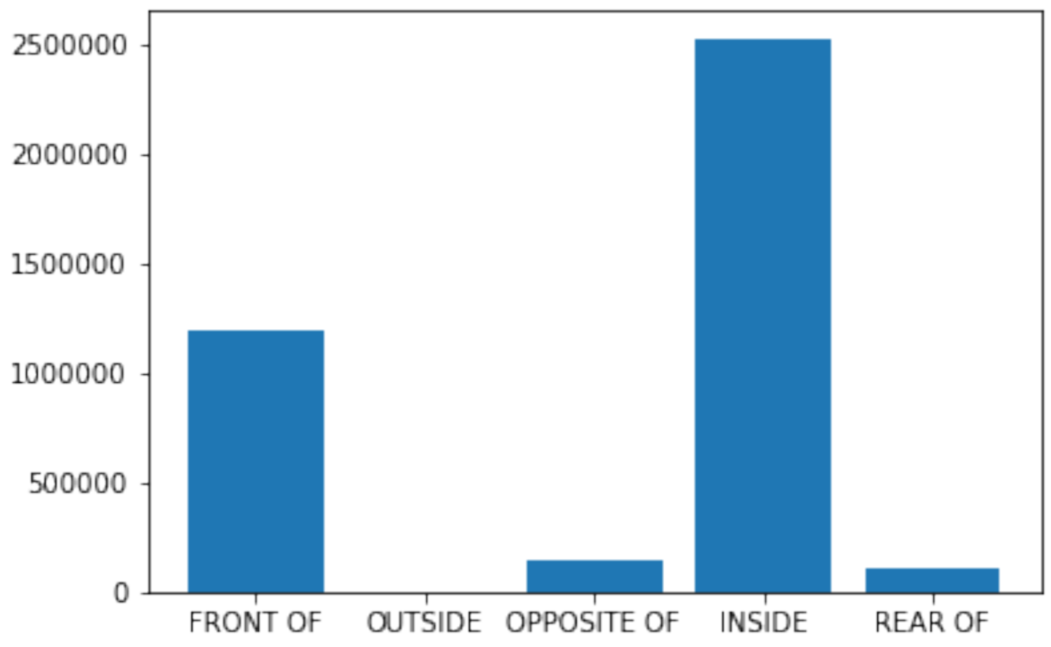


0.0.3 Column 15 - Occurrence Location Description

```
In [8]: data = pd.read_table('column15_data_quality.out', header = -1)
```

```
In [9]: stats = {}
        for loc in ['INSIDE', 'OUTSIDE', 'OPPOSITE OF', 'FRONT OF', 'REAR OF']:
            stats[loc] = len(data[data[0]==loc])
```

```
In [10]: plt.bar(range(len(stats)),stats.values(),align='center')
         plt.xticks(range(len(stats)), stats.keys())

         plt.show()
```


2500000
2000000
1500000
1000000
500000
0
FRONT OF
OUTSIDE
OPPOSITE OF
INSIDE
REAR OF

Heatmap of Crimes

April 17, 2017

0.1 Heatmap of Crimes

Now, we can plot a heatmap of crimes through history. From the heatmap we can find out whether there is any crime in an invalid location, for example, on the river.

0.1.1 Map of NYC

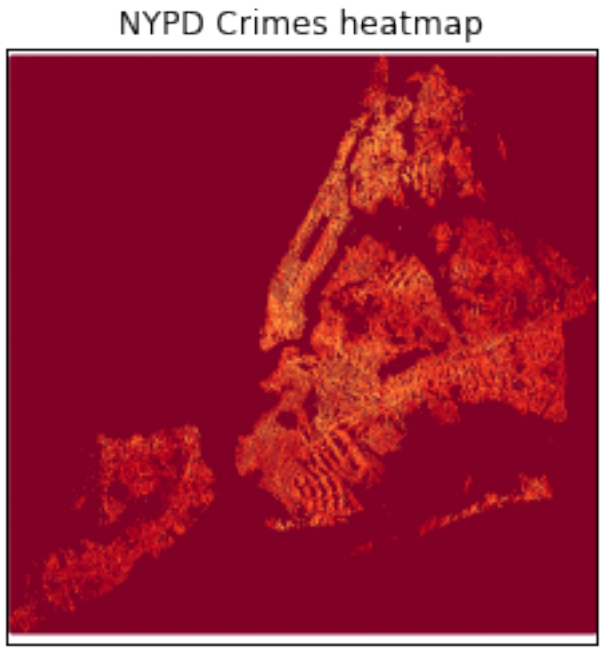
```
In [2]: # You should have installed mpl_toolkits.basemap first. If not, try conda install basema
        # or use anaconda prompt 'conda install -c conda-forge basemap-data-hires=1.0.8.dev0'

        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from matplotlib import cm
        from mpl_toolkits.basemap import Basemap
```

```
In [3]: df = pd.read_csv("/Users/xinyan/Downloads/NYPD_Complaint_Data_Historic.csv")
```

```
/Users/xinyan/.local/lib/python3.5/site-packages/IPython/core/interactiveshell.py:2717: DtypeWar
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [5]: """
        Heatmap of geolocated collisions in New York City area of a selected year.
        This map only plot data with location information, i.e. latitude and longitude.
        Data without location will be ignored.

        Baesd on tweets heatmap by Kelsey Jordahl, Enthought in Scipy 2013 geospatial tutorial.
        See more in github page: https://github.com/kjordahl/SciPy2013.

        """
        def heatmap(df):
            """
            The function turns input dataframe into heatmap. Input should contain feature
            'LATITUDE' and 'LONGITUDE', otherwise it will not be plotted.
            """
            west, south, east, north = -74.26, 40.49, -73.70, 40.92 # NYC
            a = np.array(df['Latitude'].dropna())
            b = np.array(df['Longitude'].dropna())
            N = len(a)

            fig = plt.figure()
            m = Basemap(projection='merc', llcrnrlat=south, urcrnrlat=north,
                        llcrnrlon=west, urcrnrlon=east, lat_ts=south, resolution='i')
            x, y = m(b, a)
            m.hexbin(x, y, gridsize=650, bins='log', cmap=cm.YlOrRd_r)
            plt.title("NYPD Crimes heatmap")
            plt.show()
```

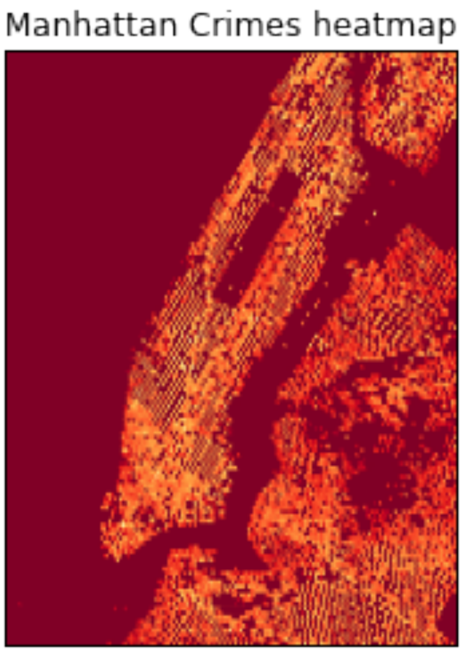
```
In [10]: heatmap(df)
```



0.1.2 Map of 5 boroughs

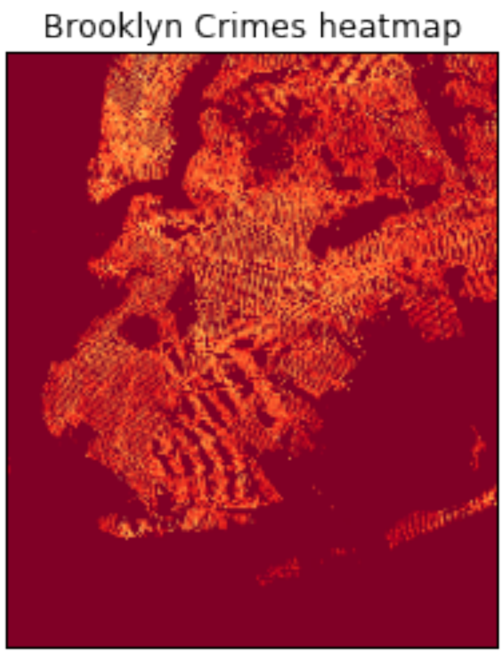
```
In [11]: def heatmap_mht(df):
             """
             The function turns input dataframe into heatmap. Input should contain feature
             'LATITUDE' and 'LONGITUDE', otherwise it will not be plotted.
             """
             west, south, east, north = -74.05, 40.68, -73.90, 40.83 # Manhattan, gridsize=650
             a = np.array(df['Latitude'].dropna())
             b = np.array(df['Longitude'].dropna())
             N = len(a)

             fig = plt.figure()
```

```
        m = Basemap(projection='merc', llcrnrlat=south, urcrnrlat=north,
                    llcrnrlon=west, urcrnrlon=east, lat_ts=south, resolution='i')
        x, y = m(b, a)
        m.hexbin(x, y, gridsize=650, bins='log', cmap=cm.YlOrRd_r)
        plt.title("Manhattan Crimes heatmap")
        plt.show()

    heatmap_mht(df)
```



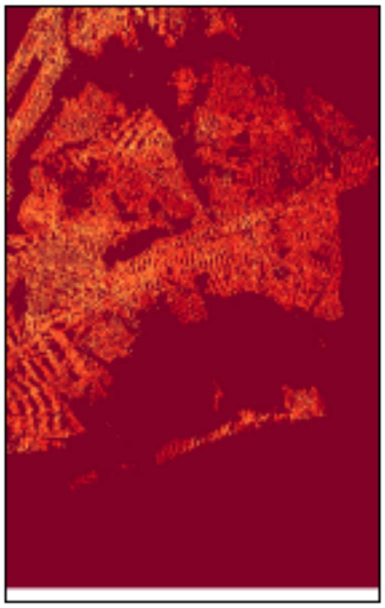
```
In [12]: def heatmap_bk(df):
             """
             The function turns input dataframe into heatmap. Input should contain feature
             'LATITUDE' and 'LONGITUDE', otherwise it will not be plotted.
             """

             a = np.array(df['Latitude'].dropna())
             b = np.array(df['Longitude'].dropna())
             N = len(a)
             west, south, east, north = -74.06, 40.53, -73.81, 40.76 # BK
             fig = plt.figure()
             m = Basemap(projection='merc', llcrnrlat=south, urcrnrlat=north,
                         llcrnrlon=west, urcrnrlon=east, lat_ts=south, resolution='i')
             x, y = m(b, a)
             m.hexbin(x, y, gridsize=650, bins='log', cmap=cm.YlOrRd_r)
             plt.title("Brooklyn Crimes heatmap")
             plt.show()
```

```
heatmap_bk(df)
```



```
In [13]: def heatmap_q(df):
             """
             The function turns input dataframe into heatmap. Input should contain feature
             'LATITUDE' and 'LONGITUDE', otherwise it will not be plotted.
             """
             west, south, east, north = -73.98, 40.49, -73.70, 40.83 # QS
             a = np.array(df['Latitude'].dropna())
             b = np.array(df['Longitude'].dropna())
             N = len(a)

             fig = plt.figure()
             m = Basemap(projection='merc', llcrnrlat=south, urcrnrlat=north,
                         llcrnrlon=west, urcrnrlon=east, lat_ts=south, resolution='i')
             x, y = m(b, a)
             m.hexbin(x, y, gridsize=650, bins='log', cmap=cm.YlOrRd_r)
             plt.title("Queens Crimes heatmap")
             plt.show()

         heatmap_q(df)
```

Queens Crimes heatmap



```
In [14]: def heatmap_si(df):
             """
             The function turns input dataframe into heatmap. Input should contain feature
             'LATITUDE' and 'LONGITUDE', otherwise it will not be plotted.
             """
             west, south, east, north = -74.26, 40.49, -74.03, 40.68 # SI
             a = np.array(df['Latitude'].dropna())
             b = np.array(df['Longitude'].dropna())
             N = len(a)

             fig = plt.figure()
             m = Basemap(projection='merc', llcrnrlat=south, urcrnrlat=north,
                         llcrnrlon=west, urcrnrlon=east, lat_ts=south, resolution='i')
             x, y = m(b, a)
             m.hexbin(x, y, gridsize=650, bins='log', cmap=cm.YlOrRd_r)
             plt.title("Staten Island Crimes heatmap")
             plt.show()

         heatmap_si(df)
```

Staten Island Crimes heatmap