PLOT -1

Description:

We have plotted the various event types against their location(latitudes and longitudes). Since there we 35 event types, we have grouped them into 5 categories which are ‘accidents’, ‘traffic’, ‘events’, ‘construction’, ‘natural’. Based on their names each event was allotted one category out of the five.

We have used matplotlib’s basemap to plot the event types. We have 5 output maps.

Map1: Depicts all the events that fall under traffic category. The events include delay due to traffic, traffic congestion, etc.

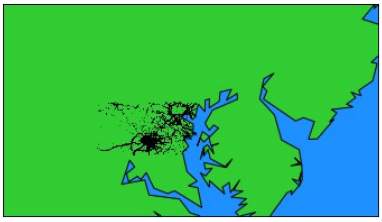
Map2: Depicts all the events that fall under events category that is special events, sporting events, etc

Map3: Depicts all the events that fall under natural category, showing natural events like precipitation, winds, visibility and humidity, etc.

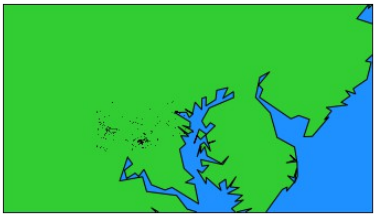
Map4: Depicts all the events that fall under construction category. The construction category has construction related events like road works, water works, etc.

Map1: Depicts all the events that fall under accident category. Events such as accidents, incidents, abandoned vehicle, etc. fall under this category.

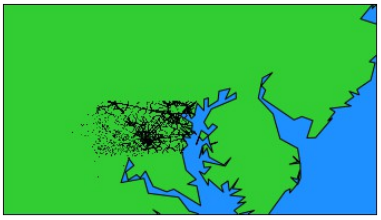
traffic



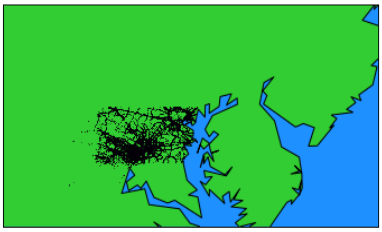
events



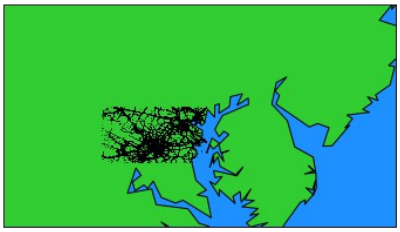
natural



Construction



accidents



Inference:

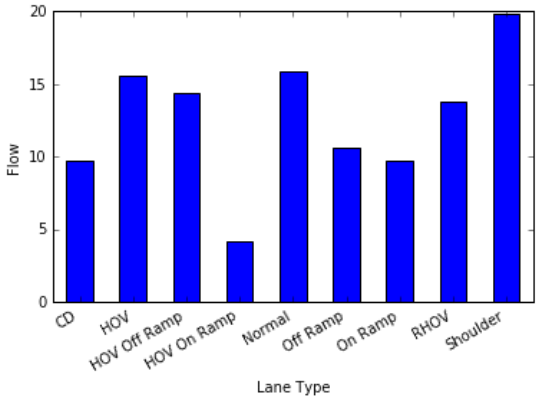
From the above plots we infer that the majority of events in the data are a result of construction work and accidents. This is closely followed by traffic related activities and natural events. Very few special events occur in the given location.

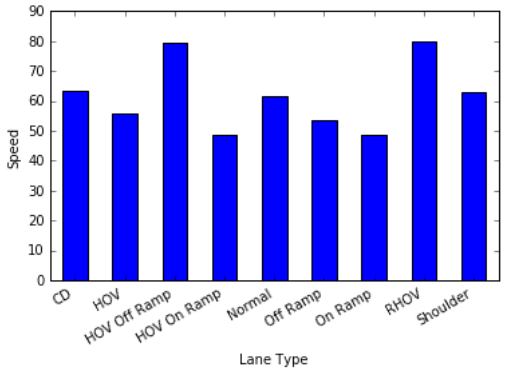
We can also see the distribution of these events across the locations. For instance the event type traffic is concentrated to a few locations while accidents and constructions are spread almost uniformly across all the locations.

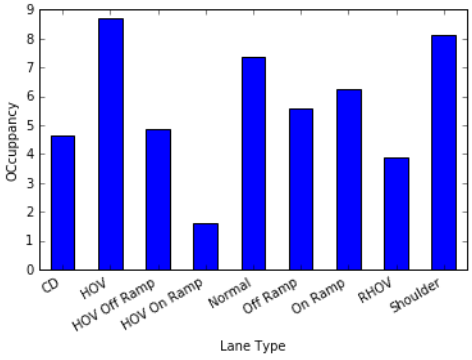
PLOT-2

Description:

We have plotted the speed, flow and measurement VS the Lane Type in three different plots. Firstly, we find out all the unique lane types from the file. Then we have grouped the lanes for each of the speed, flow and measurement taking mean of each. These averaged values are then plotted.



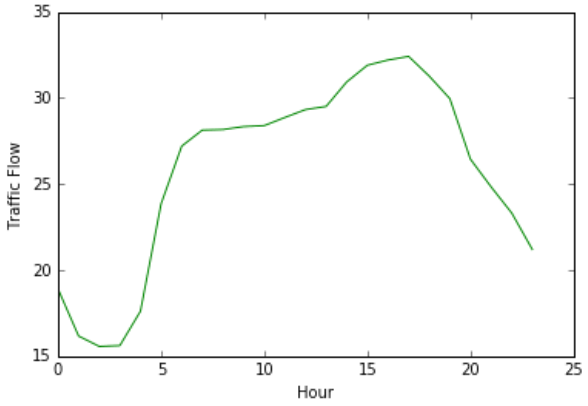


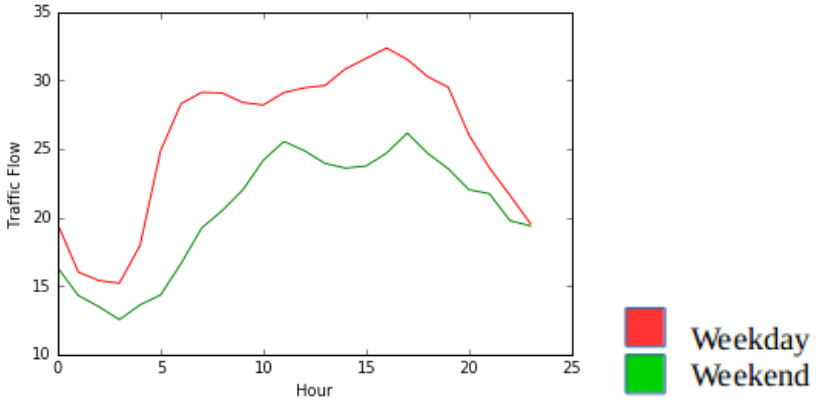


Inference:
The above graphs show that how speed, flow and occupancy of traffic vary with the lane type. We can see that the flow of traffic on the shoulder lane type is much higher than other lanes, but for the speed there is not much variation between the lane types. The occupancy of traffic graph again shows a lot of variations for different lane types. These graphs can be used to analyse how the lane types affect flow, speed and occupancy of traffic, furthermore they will also tell us whether all three are inter-related or not.

PLOT -3

Description:
We plotted the average flow of traffic during each hour of the day. This was done by grouping the flow values in the file against their corresponding hours.





First graph depicts flow of traffic on various hours (combined all days of a week). Second graph consists of two line plots. One plot (Red Line) is the flow of traffic on different hours of weekdays. Other plot (green line) is the flow of traffic on different hours of weekends.

Inference:
When compared it shows that the trends in the flow of traffic take a similar course. Although there is more flow during weekdays than weekends, but the increase or decrease in the flow is similar. Also, at the last hour of the day the flow for both the type of days tends to converge. There are two peaks one for morning and the other for evening. There is one peak between the times of 5-10 and other between 15-20 (3-8).

Code 1

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
df = pd.read_csv("C:/Users/hp/Downloads/events_train_holdout.tsv",sep="\t",error_bad_lines=False)
df['event_type'].unique()
accidents = ['accidentsAndIncidents','accident','minor accident', 'incident','security incident','incidentResponseEquipment']
construction = ['obstruction','roadwork','work on underground services','road maintenance operations','water main work', 'pavementConditions', 'systemInformation']
events = ['specialEvents','major event','sportingEvents']
natural = ['obstruction','precipitation','hazardous material spill','overgrown grass','visibilityAndAirQuality', 'debris on roadway','winds','sign down','disasters','warningAdvice','fallen trees']
```

```
traffic = ['trafficConditions','DelayStatusCancellation', 'deviceStatus','traffic lights not working','disabled vehicle', 'disturbances','traffic congestion','abandoned vehicle']

df1 = df[df['event_type'].isin(traffic)]
df2 = df[df['event_type'].isin(events)]
df3 = df[df['event_type'].isin(natural)]
df4 = df[df['event_type'].isin(construction)]
df5 = df[df['event_type'].isin(accidents)]

df1 = df1[np.isfinite(df1['latitude']) & np.isfinite(df1['longitude'])]
df2 = df2[np.isfinite(df2['latitude']) & np.isfinite(df2['longitude'])]
df3 = df3[np.isfinite(df3['latitude']) & np.isfinite(df3['longitude'])]
df4 = df4[np.isfinite(df4['latitude']) & np.isfinite(df4['longitude'])]
df5 = df5[np.isfinite(df5['latitude']) & np.isfinite(df5['longitude'])]
lat  = df1['latitude']
lat1 = lat.tolist()
lon = df1['longitude']
lon1 = lon.tolist()
##traffic
themap = Basemap(projection='gall',
            llcrnrlon =  -78.8379,          # lower-left corner longitude
            llcrnrlat = 37.9269,           # lower-left corner latitude
            urcrnrlon = -74.0918,          # upper-right corner longitude
            urcrnrlat = 40.7140,             # upper-right corner latitude
            resolution = 'l',
            area_thresh = 100000.0,
            )

themap.drawcoastlines()
themap.drawcountries()
themap.fillcontinents(color = 'limegreen')
themap.drawmapboundary(fill_color='dodgerblue')
x, y = themap(lon1, lat1)
themap.plot(x, y,
```

```
        'o',                  # marker shape
        color='Indigo' ,    # marker colour
        markersize=0.25           # marker size
        )

plt.show()

lat = df2['latitude']
lat2 = lat.tolist()
lon = df2['longitude']
lon2 = lon.tolist()
## events -special events


themap = Basemap(projection='gall',
          llcrnrlon =  -78.8379,          # lower-left corner longitude
          llcrnrlat = 37.9269,           # lower-left corner latitude
          urcrnrlon = -74.0918,          # upper-right corner longitude
          urcrnrlat = 40.7140,             # upper-right corner latitude
          resolution = 'l',
          area_thresh = 100000.0,
          )

themap.drawcoastlines()
themap.drawcountries()
themap.fillcontinents(color = 'limegreen')
themap.drawmapboundary(fill_color='dodgerblue')
x, y = themap(lon2, lat2)
themap.plot(x, y,
        'o',                  # marker shape
        color='Indigo' ,    # marker colour
        markersize=0.25           # marker size
        )

plt.show()
```

```
lon = df3['longitude']
lon3 = lon.tolist()
lat = df3['latitude']
lat3 = lat.tolist()
##natural
themap = Basemap(projection='gall',
            llcrnrlon =  -78.8379,          # lower-left corner longitude
            llcrnrlat = 37.9269,           # lower-left corner latitude
            urcrnrlon = -74.0918,          # upper-right corner longitude
            urcrnrlat = 40.7140,             # upper-right corner latitude
            resolution = 'l',
            area_thresh = 100000.0,
            )
themap.drawcoastlines()
themap.drawcountries()
themap.fillcontinents(color = 'limegreen')
themap.drawmapboundary(fill_color='dodgerblue')
x, y = themap(lon3, lat3)
themap.plot(x, y,
          'o',                  # marker shape
          color='Indigo' ,    # marker colour
          markersize=0.25           # marker size
          )

plt.show()
lon = df4['longitude']
lon4 = lon.tolist()
lat = df4['latitude']
lat4 = lat.tolist()
##construction
themap = Basemap(projection='gall',
            llcrnrlon =  -78.8379,          # lower-left corner longitude
            llcrnrlat = 37.9269,           # lower-left corner latitude
            urcrnrlon = -74.0918,          # upper-right corner longitude
```

```
            urcrnrlat = 40.7140,             # upper-right corner latitude
            resolution = 'l',
            area_thresh = 100000.0,
            )

themap.drawcoastlines()
themap.drawcountries()
themap.fillcontinents(color = 'limegreen')
themap.drawmapboundary(fill_color='dodgerblue')
x, y = themap(lon4, lat4)
themap.plot(x, y,
          'o',                  # marker shape
          color='Indigo' ,    # marker colour
          markersize=0.25           # marker size
          )

plt.show()
lat = df5['latitude']
lat5 = lat.tolist()
lon = df5['longitude']
lon5 = lon.tolist()
##accidents
themap = Basemap(projection='gall',
            llcrnrlon =  -78.8379,          # lower-left corner longitude
            llcrnrlat = 37.9269,           # lower-left corner latitude
            urcrnrlon = -74.0918,         # upper-right corner longitude
            urcrnrlat = 40.7140,             # upper-right corner latitude
            resolution = 'l',
            area_thresh = 100000.0,
            )
themap.drawcoastlines()
themap.drawcountries()
themap.fillcontinents(color = 'limegreen')
themap.drawmapboundary(fill_color='dodgerblue')
x, y = themap(lon5, lat5)
```

```
themap.plot(x, y,
        'o',                # marker shape
        color='Indigo' ,    # marker colour
        markersize=0.25          # marker size
        )
plt.show()
```

Code -2

```
import pandas as pd
import numpy as np
fpath = "/home/swati/Downloads/detector_lane_inventory.tsv"
df = pd.read_csv(fpath, sep='\t')
df['lane_id'].unique()

df2 =
pd.read_csv("/home/swati/Downloads//cleaning_test_06_09.tsv",sep="\t",error_ba
d_lines=False)
for i in range (0,len(unique_lanes)) :
      df2.ix[df2['lane_id']==unique_lanes[i],'lane_type'] =\
      df1.ix[df1['lane_id']==unique_lanes[i],'lane_type'].values[0]

df2 = df2[(df2['occupancy']>0) & (df2['flow']>0) & (df2['speed']>0) ]
lane_grouped = df2.groupby(lambda x : df2['lane_type'][x])

average_flow = lane_grouped['flow'].mean()
average_speed = lane_grouped['speed'].mean()
average_occupancy = lane_grouped['occupancy'].mean()

x = lane_grouped.size().index
x
fig, ax = plt.subplots()
average_flow.plot(kind='bar',rot=1)
fig.autofmt_xdate()
ax.set_xlabel('Lane Type')
ax.set_ylabel('Flow')
plt.show
fig, ax = plt.subplots()
```

```
average_speed.plot(kind='bar',rot=1)
fig.autofmt_xdate()
plt.show
ax.set_xlabel('Lane Type')
ax.set_ylabel('Speed')
fig, ax = plt.subplots()
average_occupancy.plot(kind='bar',rot=1)
fig.autofmt_xdate()
plt.show
ax.set_xlabel('Lane Type')
ax.set_ylabel('OCcuppancy')
```

Code -3

```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import datetime
import pandas as pd
df =
pd.read_csv("/home/swati/Downloads/cleaning_test_06_09.tsv",sep="\t",error_bad
_lines=False)

df['measurement_start_date'], df['measurement_start_time'] =
df['measurement_start'].str.split('T', 1).str
df['measurement_hour'] = df['measurement_start_time'].str.split(':', 1)
df['measurement_hour'] = df['measurement_hour'].apply(lambda x: int(x[0]))
hour_grouped = df.groupby(lambda x : df['measurement_hour'][x])
average_flow = hour_grouped['flow'].mean()

fig, ax1 = plt.subplots()
x = hour_grouped.size().index
ax1.plot(x,average_flow,'g-')
ax1.set_xlabel('Hour')
ax1.set_ylabel('Traffic Flow')
def isWeekend(dat):
    try:
        year,month,day = (int(x) for x in dat.split('-'))
```

```
        ans = datetime.date(year, month, day)
        return ans.strftime("%A")
    except ValueError:
        return -1


subset_df = df[1:3141738]
subset_df['isWeekend'] = subset_df['measurement_start_date'].apply(lambda x : isWeekend(x))
subset_df1 = subset_df[subset_df['isWeekend'].isin(['Saturday','Sunday'])]
subset_df2 = subset_df[subset_df['isWeekend'].isin(['Monday','Tuesday','Wednesday', 'Thursday','Friday'])]

hour_grouped1 = subset_df1 .groupby(lambda x : df['measurement_hour'][x])
average_flow1 = hour_grouped1['flow'].mean()

hour_grouped2 = subset_df2 .groupby(lambda x : df['measurement_hour'][x])
average_flow2 = hour_grouped2['flow'].mean()

fig, ax1 = plt.subplots()
ax1.plot(x,average_flow1,'g-')
ax1.plot(x,average_flow2,'r-')
ax1.set_xlabel('Hour')
ax1.set_ylabel('Traffic Flow')
```