911 Calls - Descriptive Analytics Project

July 30, 2020

1 911 Calls - Descriptive Analytics Project

Description: Emergency (911) Calls: Fire, Traffic, EMS for Montgomery County, PA.

For this descriptive analytics project I will be analyzing **911 Calls** dataset from Kaggle.The 911 Calls data contains the following fields:

- lat : String variable, Latitude
- lng: String variable, Longitude
- desc: String variable, Description of the Emergency Call
- zip: String variable, Zipcode
- title: String variable, Title
- timeStamp: String variable, YYYY-MM-DD HH:MM:SS
- twp: String variable, Township
- addr: String variable, Address
- e: String variable, Dummy variable (always 1)

1.1 Importing Libraries and Loading Data

[1]:
```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
from matplotlib.dates import DateFormatter
%matplotlib inline
```

[106]:
```
# Loading 911 calls dataset
path = 'D:\Simon\___My Projects\Python - 911 Calls - DataViz'
path_to_file = (path + '\911.csv')
df = pd.read_csv(path_to_file)
```

1.2 Exploratory Analysis and Data Cleaning

First let's check the percentage of NAN values in all columns.

[4]:
```
round((df.isna().sum() / df.count()) * 100, 1)
```

[4]:
```
lat           0.0
lng           0.0
desc          0.0
zip          14.8
title         0.0
timeStamp     0.0
twp           0.0
addr          0.5
e             0.0
dtype: float64
```

Most of the columns have no NAN values, except for **'zip code'**, **'township'** and **'address'**. However, only **zip code** has a significant amount of NANs - 14.8% of all values.

[5]:
```
df.describe()
```

[5]:

	lat	lng	zip	e
count	99492.000000	99492.000000	86637.000000	99492.0
mean	40.159526	-75.317464	19237.658298	1.0
std	0.094446	0.174826	345.344914	0.0
min	30.333596	-95.595595	17752.000000	1.0
25%	40.100423	-75.392104	19038.000000	1.0
50%	40.145223	-75.304667	19401.000000	1.0
75%	40.229008	-75.212513	19446.000000	1.0
max	41.167156	-74.995041	77316.000000	1.0

Zip code is numeric type, we should convert it to strings. The dummy variable column 'e' is always 1 so we can remove it.

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99492 entries, 0 to 99491
Data columns (total 9 columns):
lat          99492 non-null float64
lng          99492 non-null float64
desc         99492 non-null object
zip          86637 non-null float64
title        99492 non-null object
timeStamp    99492 non-null object
twp          99449 non-null object
addr         98973 non-null object
e            99492 non-null int64
dtypes: float64(3), int64(1), object(5)
memory usage: 6.8+ MB
```

Timestamp column contains strings, we should convert it to datetime type. It will also be useful to create some more date columns from it: year, month, date, weekday, hour.

1.2.1 Results of Exploratory Analysis

1. Cleaning the dataset

- Remove dummy variable column 'e'
- Change type of timeStamp to date

2. Creating new variables

- Create type and subtype columns based on the title column
- Create Year, Month, Date and Hour columns from the timestemp column

[107]:
```
# Remove dummy variable 'e'
df = df.drop(['e'], axis = 1, errors = 'ignore')
# change type of timestamp
df['timeStamp'] = pd.to_datetime(df['timeStamp'])
# create type and subtype columns
df['Type'] = df['title'].apply(lambda x: x.split(':')[0].strip())
df['Subtype'] = df['title'].apply(lambda x: x.split(':')[1].strip())
# create Year, Month, Date, Day of Week and Hour columns
df['Year'] = df['timeStamp'].apply(lambda x: x.year)
df['Month'] = df['timeStamp'].apply(lambda x: x.month)
df['Date'] = df['timeStamp'].apply(lambda x: x.date())
days = {0:'Mon', 1:'Tue', 2:'Wed', 3:'Thu', 4:'Fri', 5:'Sat', 6:'Sun'}
df['Weekday'] = df['timeStamp'].apply(lambda x: days[x.dayofweek])
df['Hour'] = df['timeStamp'].apply(lambda x: x.hour)
```

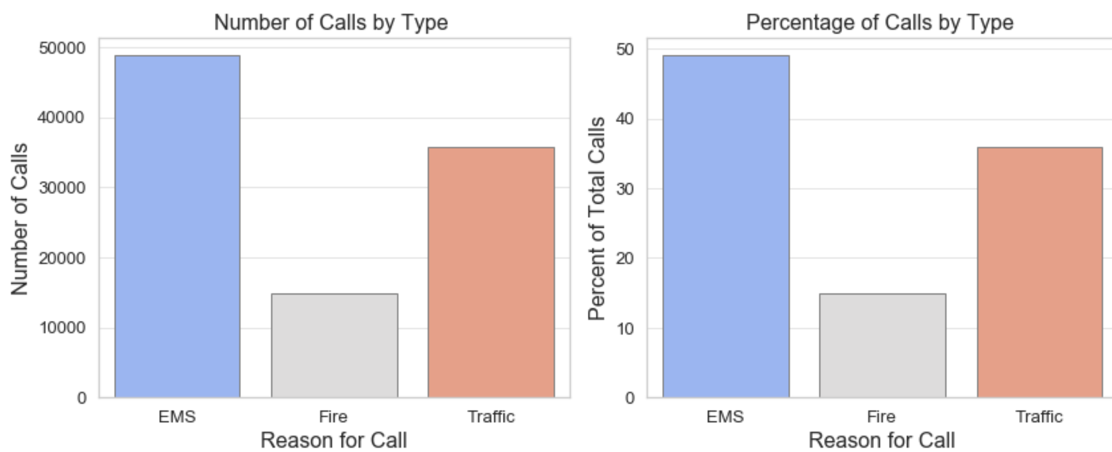
[108]:
```
# future analysis revealed that Traffic Subtypes have a hyphen in the end.␣
↪let's remove it.
# I have to use specifically .loc here, otherwise it's an error.
df.loc[df['Type'] == 'Traffic', 'Subtype'] = (df[df['Type'] ==␣
↪'Traffic']['Subtype'].apply(lambda x: x.split('-')[0].strip()))
```

1.3 Descriptive Analysis

1.3.1 911 Calls by Type

[8]:
```
# set plot size, gridline width, style, font scale, ...
sns.set(rc={'figure.figsize':(12,5), "grid.linewidth": 0.5}, style="whitegrid",␣
↪font_scale=1.2)
# plotting two graphs
fig, (ax1, ax2) = plt.subplots(ncols=2)
sns.countplot(x = 'Type', data = df, palette = 'coolwarm', lw = 1, edgecolor =␣
↪'gray', ax = ax1)
sns.barplot(x = "Type", y = "lat", data = df, estimator = lambda x: len(x) /␣
↪len(df) * 100, ax = ax2,
            lw = 1, edgecolor = 'gray', palette = 'coolwarm')

# set axis and title for ax1
```

```
ax1.set_xlabel("Reason for Call",fontsize=16)
ax1.set_ylabel("Number of Calls",fontsize=16)
ax1.set_title("Number of Calls by Type ",fontsize=16)

# set axis and title for ax2
ax2.set_xlabel("Reason for Call",fontsize=16)
ax2.set_ylabel("Percent of Total Calls",fontsize=16)
ax2.set_title("Percentage of Calls by Type ",fontsize=16)

plt.tight_layout()
```



- Almost 50% of all 911 Calls are Emergency calls.
- Trafic come second with over 35%.

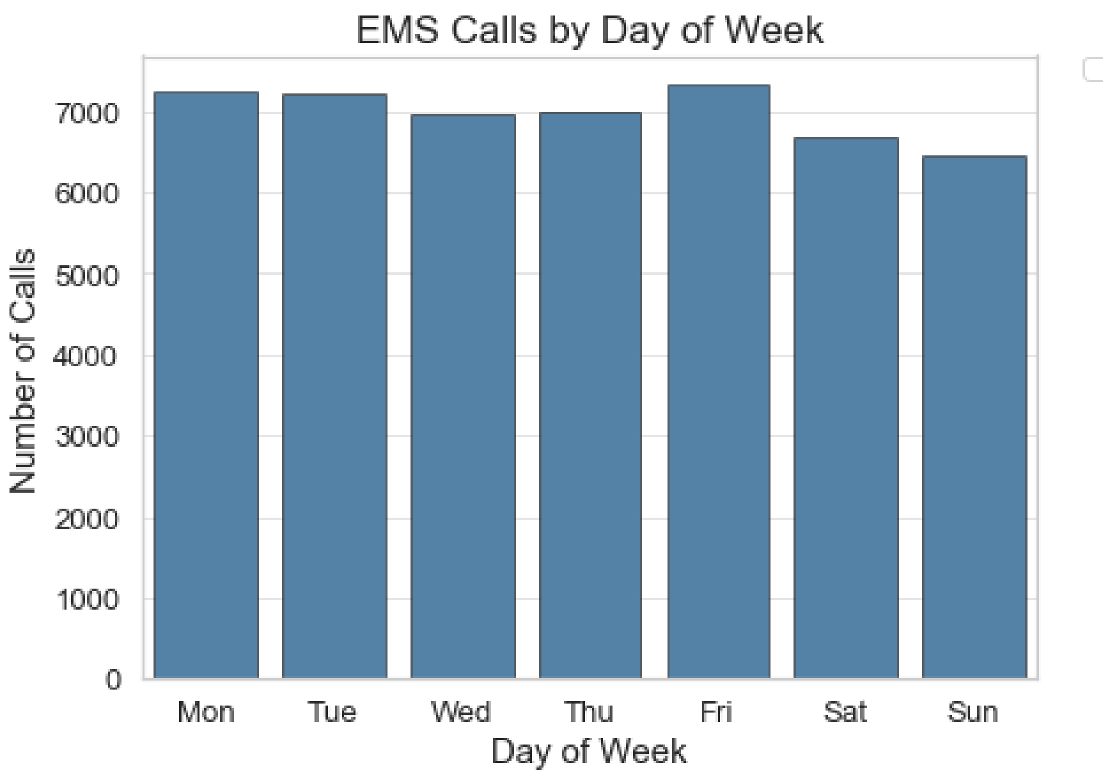
1.3.2 911 Calls by Day of Week

[124]:
```
sns.set(rc={'figure.figsize':(7,5), "grid.linewidth": 0.5}, style="whitegrid",␣
 ↪font_scale=1.2)

# Countplot + move legend + change: colors + borders + titles + figure size
order = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

#sns.set_style("whitegrid") # changing figure style
#sns.set_context('notebook', font_scale = 1.2)
sns.countplot(x = 'Weekday', data = df[df['Type'] == 'EMS'],
              lw = .5, edgecolor = 'black', order = order, color = 'steelblue')

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # moving legend␣
 ↪outsude of the plot
plt.ylabel('Number of Calls', fontsize=15)
plt.xlabel('Day of Week', fontsize=15)
```

```
plt.title('EMS Calls by Day of Week', fontsize = 17)
```

```
No handles with labels found to put in legend.
```

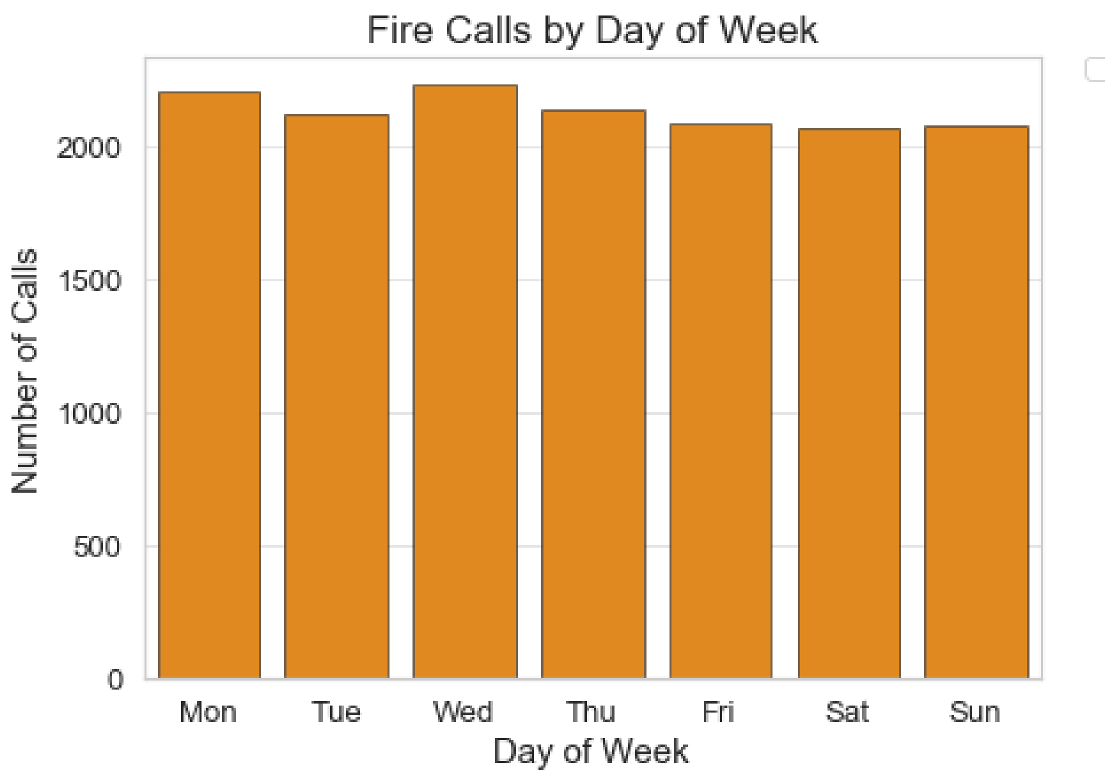
[124]:
```
Text(0.5, 1.0, 'EMS Calls by Day of Week')
```



[10]:
```
sns.set(rc={'figure.figsize':(7,5), "grid.linewidth": 0.5}, style="whitegrid",␣
↪font_scale=1.2)

# Countplot + move legend + change: colors + borders + titles + figure size
order = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

#sns.set_style("whitegrid") # changing figure style
#sns.set_context('notebook', font_scale = 1.2)
sns.countplot(x = 'Weekday', data = df[df['Type'] == 'Fire'],
              lw = .5, edgecolor = 'black', order = order, color = 'darkorange')

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # moving legend␣
↪outsude of the plot
plt.ylabel('Number of Calls', fontsize=15)
plt.xlabel('Day of Week', fontsize=15)
plt.title('Fire Calls by Day of Week', fontsize = 17)
```

```
No handles with labels found to put in legend.
```

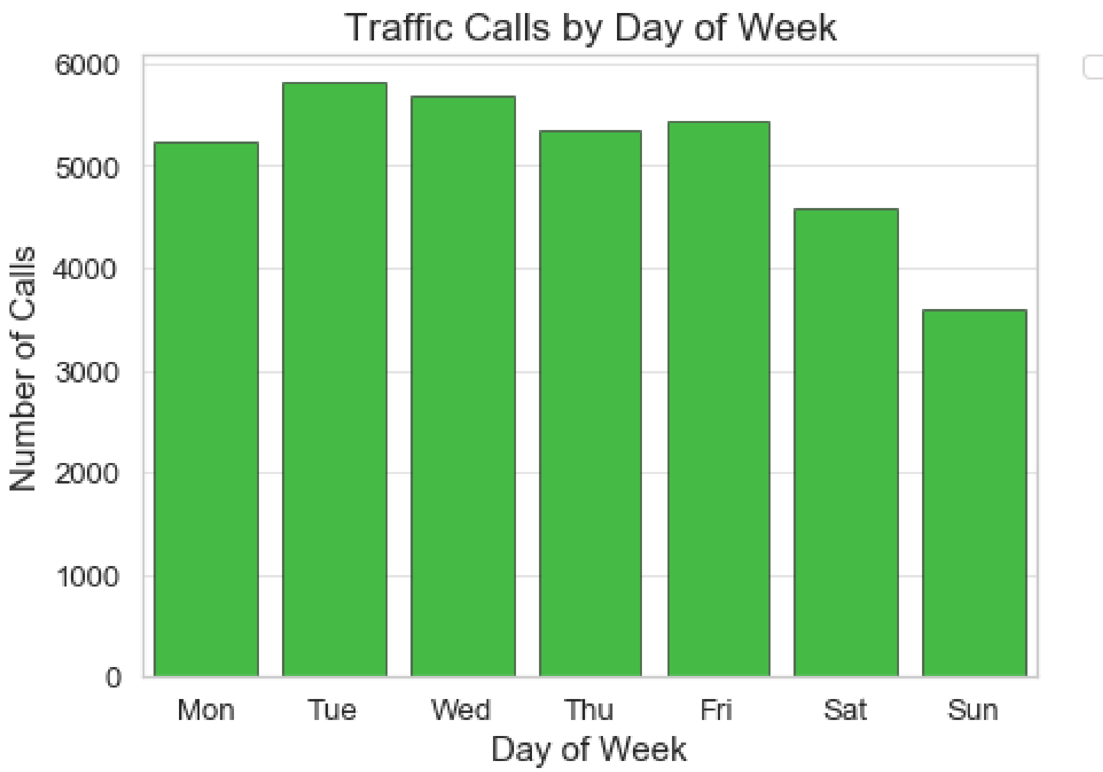
[10]:
```
Text(0.5, 1.0, 'Fire Calls by Day of Week')
```



[123]:
```
sns.set(rc={'figure.figsize':(7,5), "grid.linewidth": 0.5}, style="whitegrid",␣
↪font_scale=1.2)

# Countplot + move legend + change: colors + borders + titles + figure size
order = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

#sns.set_style("whitegrid") # changing figure style
#sns.set_context('notebook', font_scale = 1.2)
sns.countplot(x = 'Weekday', data = df[df['Type'] == 'Traffic'],
             lw = .5, edgecolor = 'black', order = order, color = 'limegreen')

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # moving legend␣
↪outsude of the plot
plt.ylabel('Number of Calls', fontsize=15)
plt.xlabel('Day of Week', fontsize=15)
plt.title('Traffic Calls by Day of Week', fontsize = 17)
```

```
No handles with labels found to put in legend.
```

[123]: Text(0.5, 1.0, 'Traffic Calls by Day of Week')

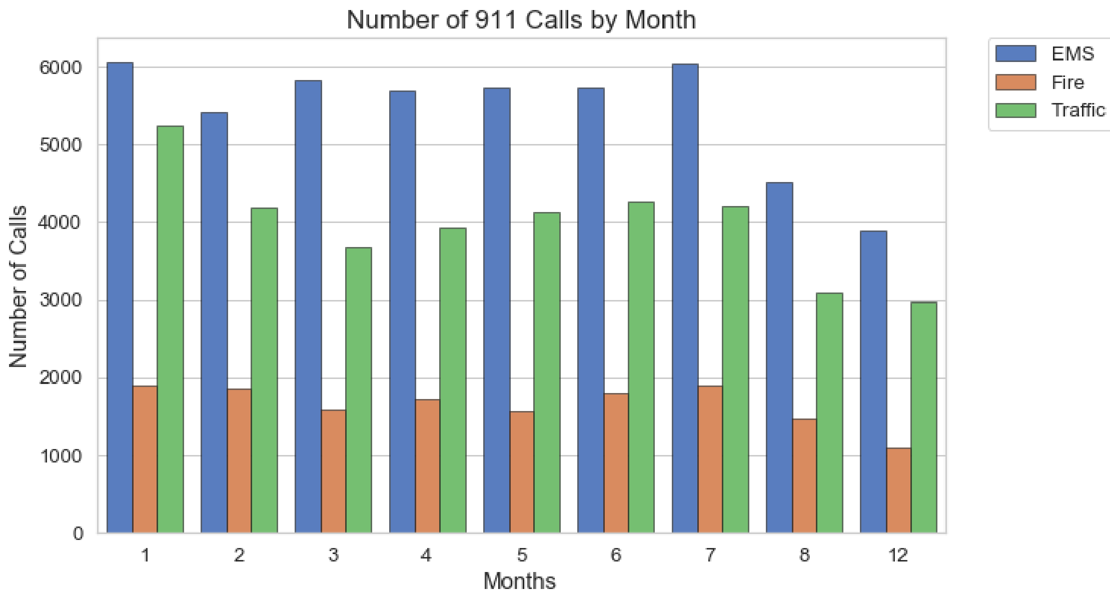


- Emergency and Traffic calls have a significant drop on weekends.
- Friday is the most busy day with Emergency calls.
- Most Traffic 911 calls happen on Tuesdays and Wednsdays.

1.3.3 911 Calls by Month

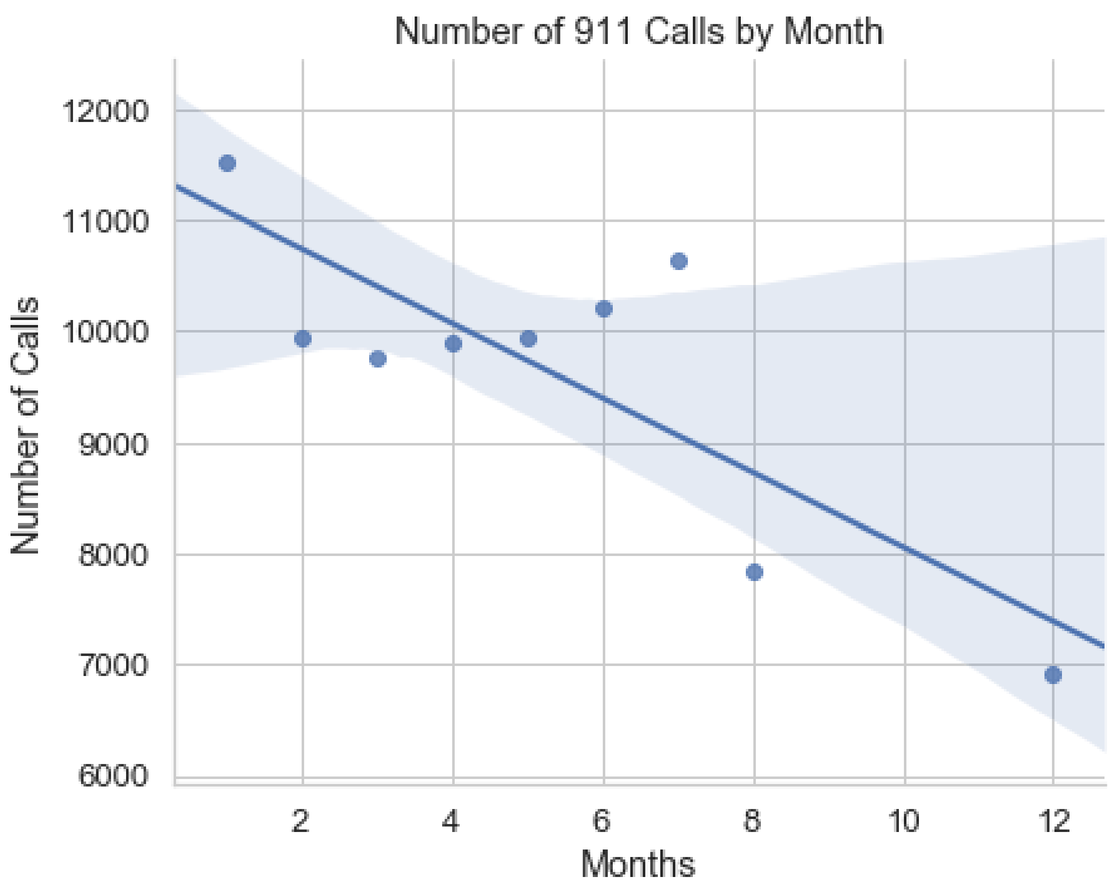
[122]:
```
# Countplot + move legend + change: colors + borders + titles + figure size
sns.set_style("whitegrid") # changing figure style
sns.set_context('notebook', font_scale = 1.2)
plt.figure(figsize = (10,6)) # changing size - has to be before plot function
sns.countplot(x = 'Month', data = df, hue = 'Type', palette = 'muted',
             lw=.5, edgecolor = 'black')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.) # moving legend␣
 ↪outsude of the plot
plt.ylabel('Number of Calls', fontsize=15)
plt.xlabel('Months', fontsize=15)
plt.title('Number of 911 Calls by Month', fontsize = 17)
```

[122]: Text(0.5, 1.0, 'Number of 911 Calls by Month')



[13]:
```
df_m = df.groupby('Month').count()
df_m.reset_index(inplace = True)
g = sns.lmplot(data = df_m, y = 'zip', x = 'Month')
g.fig.set_size_inches(7,5)
# labels
plt.xlabel('Months')
plt.ylabel('Number of Calls')
plt.title('Number of 911 Calls by Month')
```

[13]:
```
Text(0.5, 1, 'Number of 911 Calls by Month')
```



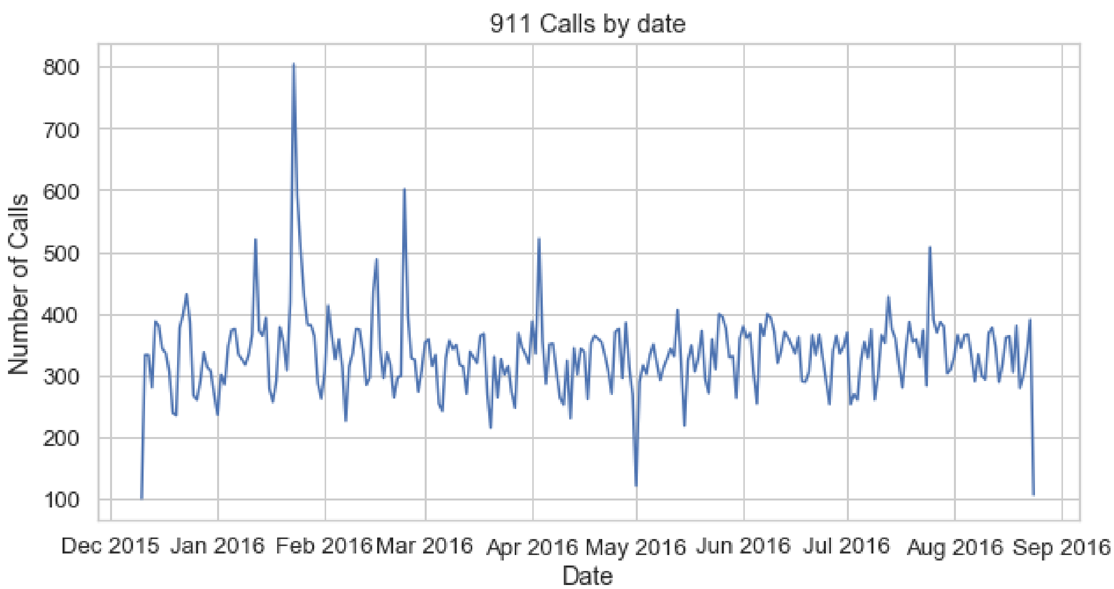
It seems like the number of calls goes down in the second half of the year, however there is not enough data to be confident about this assumption.

1.3.4 911 Calls by Date

[14]:
```
# preparing data
df_d = df.groupby('Date').count()
df_d.reset_index(inplace = True)
# plotting
fig, ax = plt.subplots(figsize = (10,5))
fig = sns.lineplot(data = df_d, x = 'Date', y = 'zip')
# set title and lables for axes
ax.set(xlabel="Date", ylabel="Number of Calls", title="911 Calls by date")
# set appropriate date format
date_form = DateFormatter("%b %Y")
ax.xaxis.set_major_formatter(date_form)
```

```
D:\Personal\Anaconda\lib\site-packages\pandas\plotting\_converter.py:129:
FutureWarning: Using an implicitly registered datetime converter for a
```

```
matplotlib plotting method. The converter was registered by pandas on import.
Future versions of pandas will require you to explicitly register matplotlib
converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```



[15]:
```
# plotting
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 12))
sns.lineplot(data = df[df['Type'] == 'EMS'], x = 'Date', y = 'lat', estimator =␣
 ↪lambda x: len(x), color = 'steelblue', ax = ax1)
sns.lineplot(data = df[df['Type'] == 'Traffic'], x = 'Date', y = 'lat',␣
 ↪estimator = lambda x: len(x), color = 'limegreen', ax = ax2)
sns.lineplot(data = df[df['Type'] == 'Fire'], x = 'Date', y = 'lat', estimator␣
 ↪= lambda x: len(x), color = 'darkorange', ax = ax3)
# axis
ax1.set(xlabel="", ylabel="Number of Calls", title="Emergency Calls")
ax2.set(xlabel="", ylabel="Number of Calls", title="Traffic Calls")
ax3.set(xlabel="", ylabel="Number of Calls", title="Fire Calls")
# set appropriate date format
date_form = DateFormatter("%b %Y")
ax1.xaxis.set_major_formatter(date_form)
ax2.xaxis.set_major_formatter(date_form)
ax3.xaxis.set_major_formatter(date_form)
plt.tight_layout()
```

- Total calls by each Type seem to have a significant random variaton during the year, with no meaninful trends.
- Emergency calls data has outlier dates with far less calls then usual.
- At the same time, Fire and Traffic data has noticeable ourliers with much higher volume of 911 calls.
- **The trajectory of Fire and Traffic Calls look very similar. It seems that the same events can cause both types of calls.**

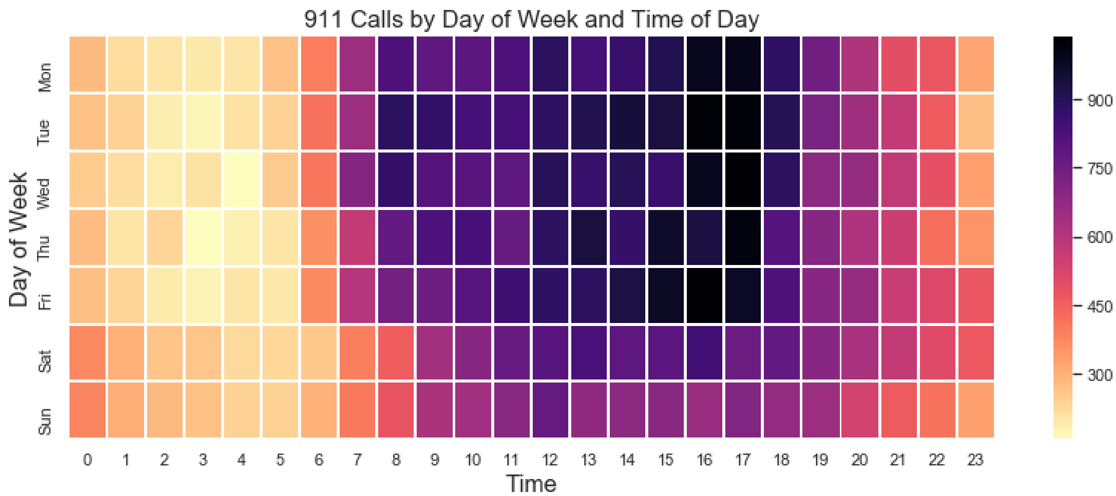
1.3.5 911 Calls by Hour and Day of Week

[144]:
```python
df_m = df_m.reindex(index = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
df_m
```

[144]:
```
Hour       0    1    2    3    4    5    6    7    8    9  ...   14   15  \
Weekday                                                    ...
Mon      282  221  201  194  204  267  397  653  819  786  ...  869  913
Tue      269  240  186  170  209  239  415  655  889  880  ...  943  938
Wed      250  216  189  209  156  255  410  701  875  808  ...  904  867
Thu      278  202  233  159  182  203  362  570  777  828  ...  876  969
Fri      275  235  191  175  201  194  372  598  742  752  ...  932  980
Sat      375  301  263  260  224  231  257  391  459  640  ...  789  796
Sun      383  306  286  268  242  240  300  402  483  620  ...  684  691

Hour       16    17   18   19   20   21   22   23
Weekday
Mon       989   997  885  746  613  497  472  325
Tue      1026  1019  905  731  647  571  462  274
Wed       990  1037  894  686  668  575  490  335
Thu       935  1013  810  698  617  553  424  354
Fri      1039   980  820  696  667  559  514  474
Sat       848   757  778  696  628  572  506  467
Sun       663   714  670  655  537  461  415  330

[7 rows x 24 columns]
```

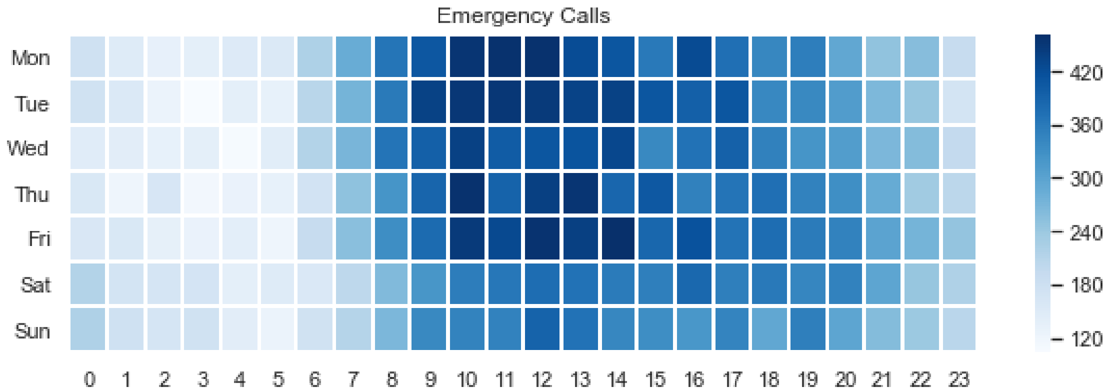
[145]:
```python
# creating pivot table for heatmap and setting right index order
df_m = df.pivot_table(index = 'Weekday', columns = 'Hour', values = 'lat',
                      aggfunc = 'count')
df_m = df_m.reindex(index = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
fig, ax = plt.subplots(figsize = (14,5))
fig = sns.heatmap(df_m, linecolor = 'white', lw = 1, cmap = 'magma_r')
ax.set_xlabel("Time",fontsize=16)
ax.set_ylabel("Day of Week",fontsize=16)
ax.set_title("911 Calls by Day of Week and Time of Day",fontsize=16)
#plt.savefig('calls_heatmap.png')
```

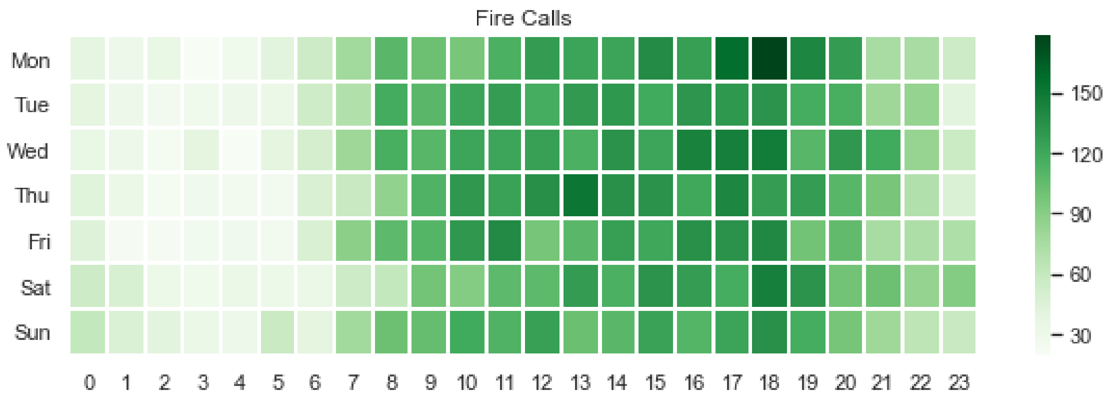
[145]:
```
Text(0.5, 1, '911 Calls by Day of Week and Time of Day')
```

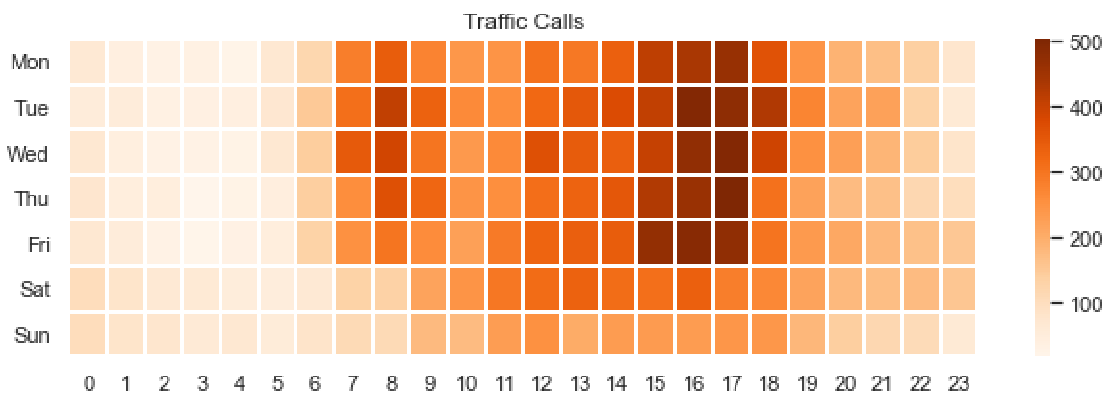


We can see that the highest concentration of 911 Calls happen between 3 to 5 pm. However, it may happen due to high volume of traffic caused by the end of working day. Let's look at the heatmap of each individual Type of Call.

[147]:
```
# plotting
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(9, 9))
# creating pivot tables and setting right index order
df_e = df[df['Type'] == 'EMS'].pivot_table(index = 'Weekday', columns = 'Hour',␣
↪values = 'lat', aggfunc = 'count')
df_f = df[df['Type'] == 'Fire'].pivot_table(index = 'Weekday', columns =␣
↪'Hour', values = 'lat', aggfunc = 'count')
df_t = df[df['Type'] == 'Traffic'].pivot_table(index = 'Weekday', columns =␣
↪'Hour', values = 'lat', aggfunc = 'count')
df_e = df_e.reindex(index = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
df_f = df_f.reindex(index = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
df_t = df_t.reindex(index = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun'])
# plots
sns.heatmap(df_e, linecolor = 'white', lw = 1, cmap = 'Blues', ax = ax1)
sns.heatmap(df_f, linecolor = 'white', lw = 1, cmap = 'Greens', ax = ax2)
sns.heatmap(df_t, linecolor = 'white', lw = 1, cmap = 'Oranges', ax = ax3)
# axis
ax1.set(xlabel="", ylabel="", title="Emergency Calls")
ax2.set(xlabel="", ylabel="", title="Fire Calls")
ax3.set(xlabel="", ylabel="", title="Traffic Calls")
plt.tight_layout()
plt.savefig('calls_heatmap_3.png')
```





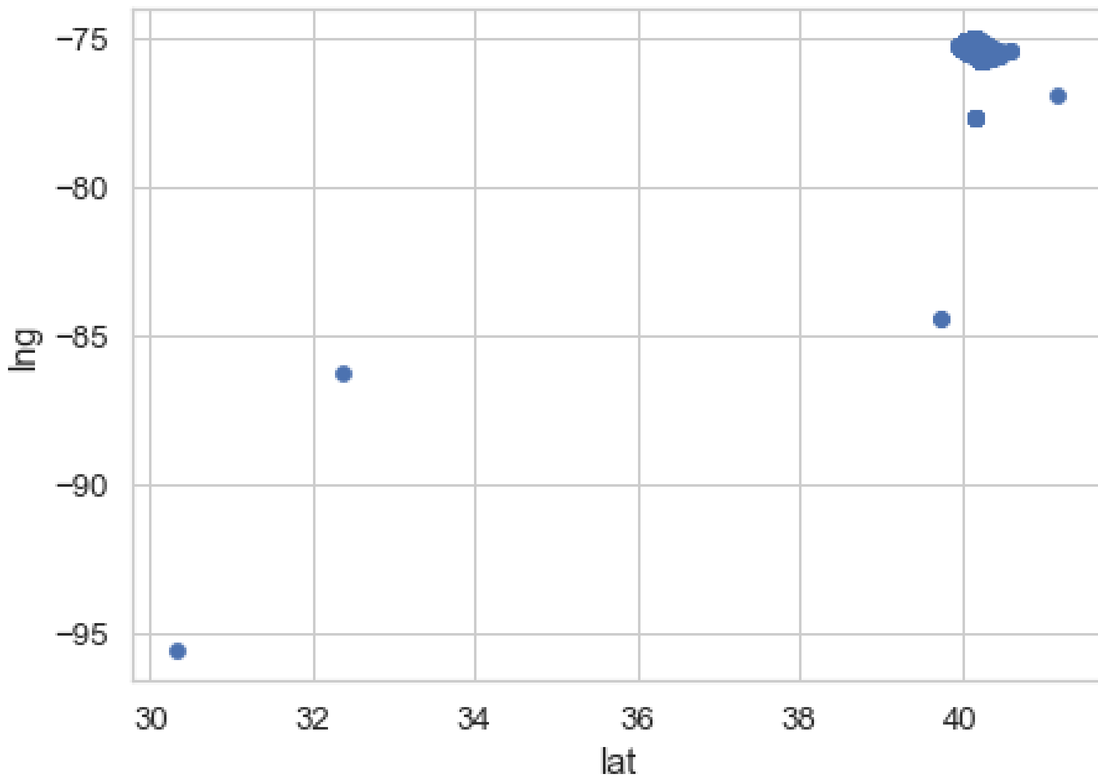


- Emergency calls happen more often during working hours (9am : 5pm) and are much less frequent during weekends.
- Fire calls seem to be consistent throughout the week and have similar density between 8am and 8pm.
- Traffic calls have the most destinct patterns with the majority happening during rush hours (4-5pm) and much smaller volume during the weekends.

1.3.6 Mapping 911 Calls

[17]:
```
sns.scatterplot(x = 'lat', y = 'lng', data = df, edgecolor = None)
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2731a451358>



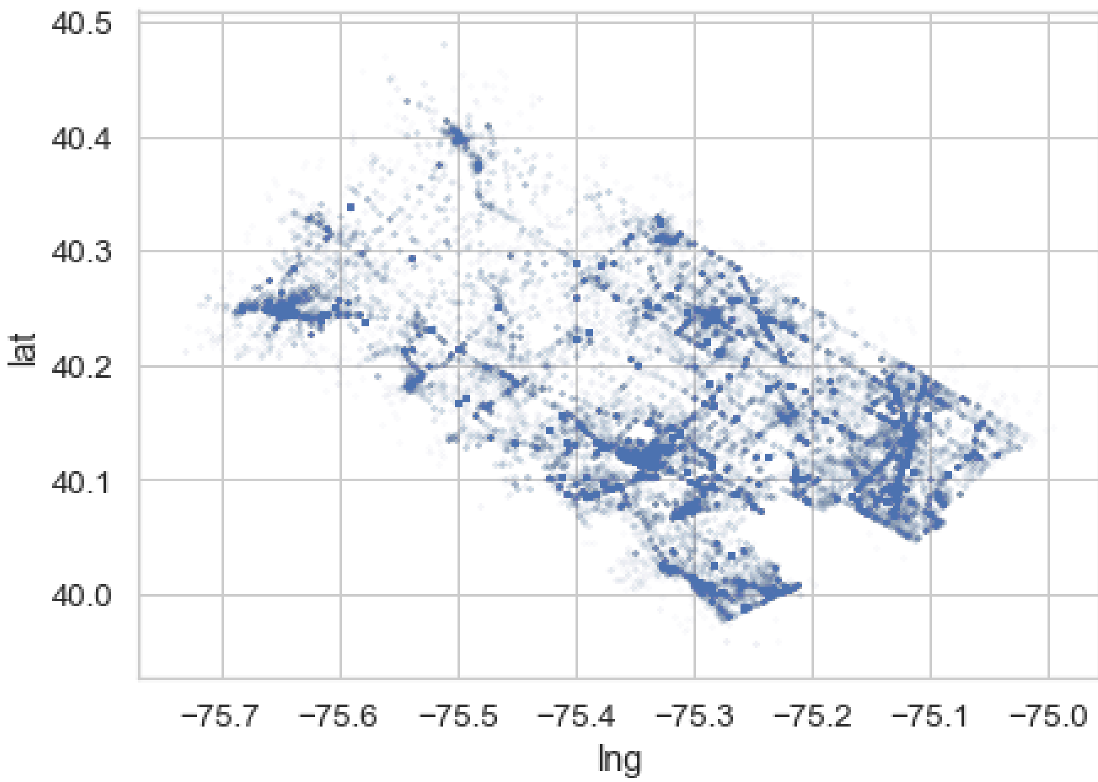
A few locations are outliers. By removing them we can take a better look at the majority of calls.

[18]:
```
df_map = df[(df['lat'] > 38) & (df['lng'] > -76) & (df['lat'] < 40.5)]
(len(df) - len(df_map)) / len(df) * 100
```

[18]: 0.019097012825151773

[19]:
```
df_map = df[(df['lat'] > 38) & (df['lng'] > -76) & (df['lat'] < 40.5)]
sns.scatterplot(x = 'lng', y = 'lat', data = df_map, edgecolor = None, alpha =␣
↪0.02, s = 3)
```

[19]: <matplotlib.axes._subplots.AxesSubplot at 0x2731a650da0>



By removing just 19 observations (less than 0.02% of the data) we now can take a much better look at the locations of 911 Calls. Now let's upload a map image and plot 911 calls on the map.

[20]:
```
# Uploading map image
path_to_file = (path + '\map.png')
image = plt.imread(path_to_file)
im_h = image.shape[0]
im_w = image.shape[1]
image.shape
```

[20]: (1116, 1199, 4)

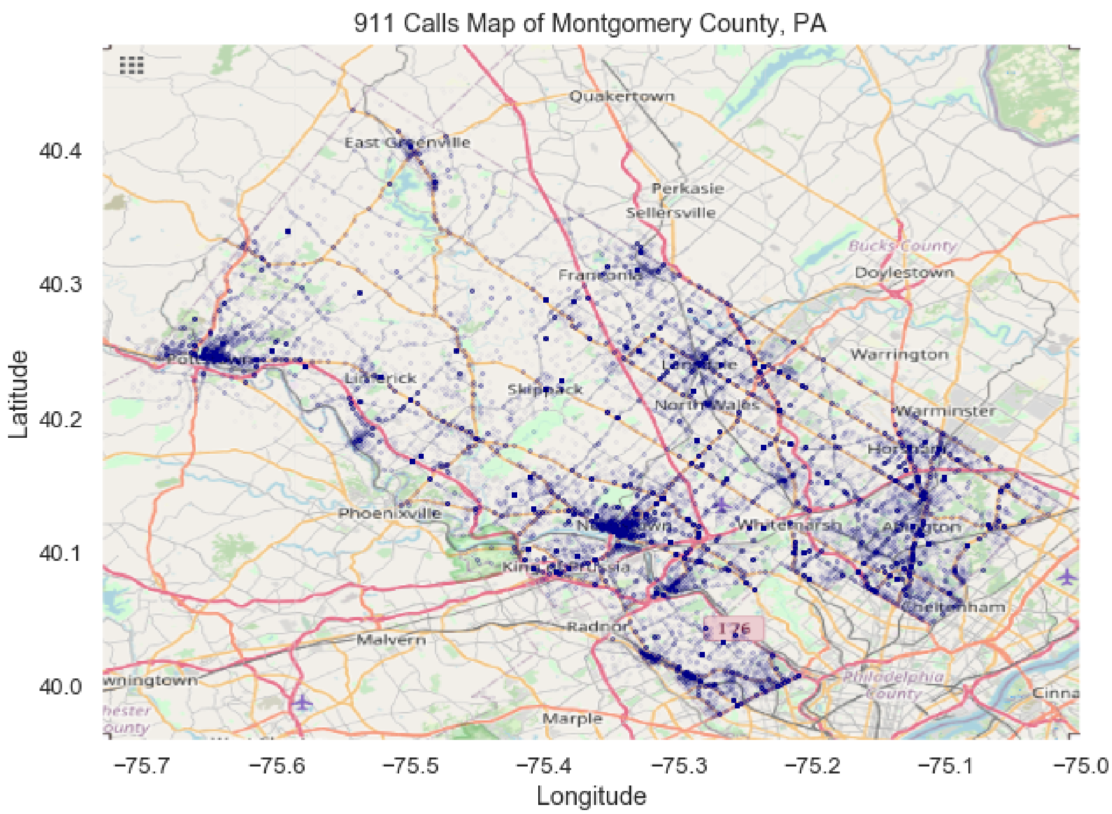
[21]:
```
# setting limits for the map
BBox = (round(df_map['lng'].min(),2), round(df_map['lng'].max(),2),
        round(df_map['lat'].min(),2), round(df_map['lat'].max(),2))
BBox
```

[21]: (-75.73, -75.0, 39.96, 40.48)

[74]:
```
#colors = {'EMS':'red', 'Fire':'blue', 'Traffic':'green'}
# plotting
fig, ax = plt.subplots(figsize = (im_w/120, im_h/120))
ax.scatter(df['lng'], df['lat'], zorder=1, alpha= 0.01, c = 'darkblue', s=5)
ax.grid(False)
ax.set_title('911 Calls Map of Montgomery County, PA')
```

```
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
ax.set_xlim(BBox[0],BBox[1])
ax.set_ylim(BBox[2],BBox[3])
ax.imshow(image, zorder=0, extent = BBox, aspect= 'equal')
#plt.savefig('calls_map.png')
```

[74]: <matplotlib.image.AxesImage at 0x27321e72358>



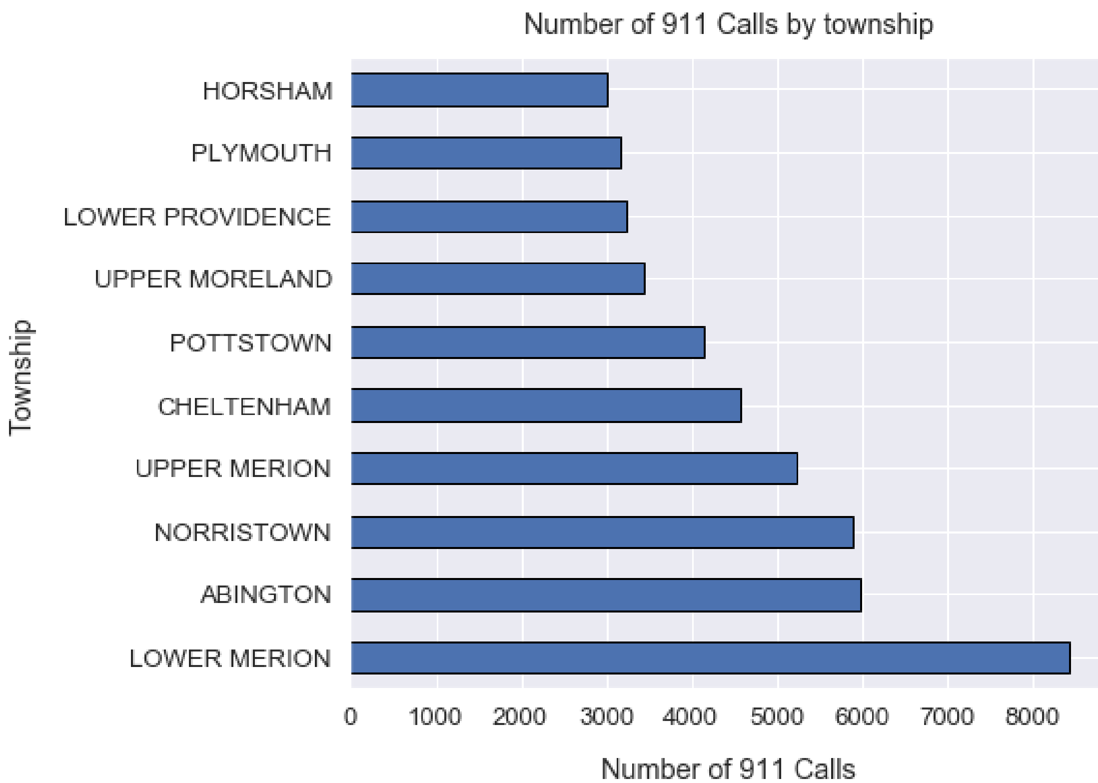
Judjing from the map the highest concentration of 911 Calls happens in the towns, like Norristown, Pottstown, Abington and Horsham.

Let's check this conclusion buy plotting the number of 911 Calls by Townships.

1.3.7 911 Calls across Townships

[119]:
```
sns.set(font_scale = 1.2)
df['twp'].value_counts().head(10).plot(kind='barh', figsize=(7, 6), rot=0,␣
 ↪edgecolor = 'black')
plt.xlabel("Number of 911 Calls", labelpad=14)
plt.ylabel("Township", labelpad=14)
plt.title("Number of 911 Calls by township", y=1.02)
```

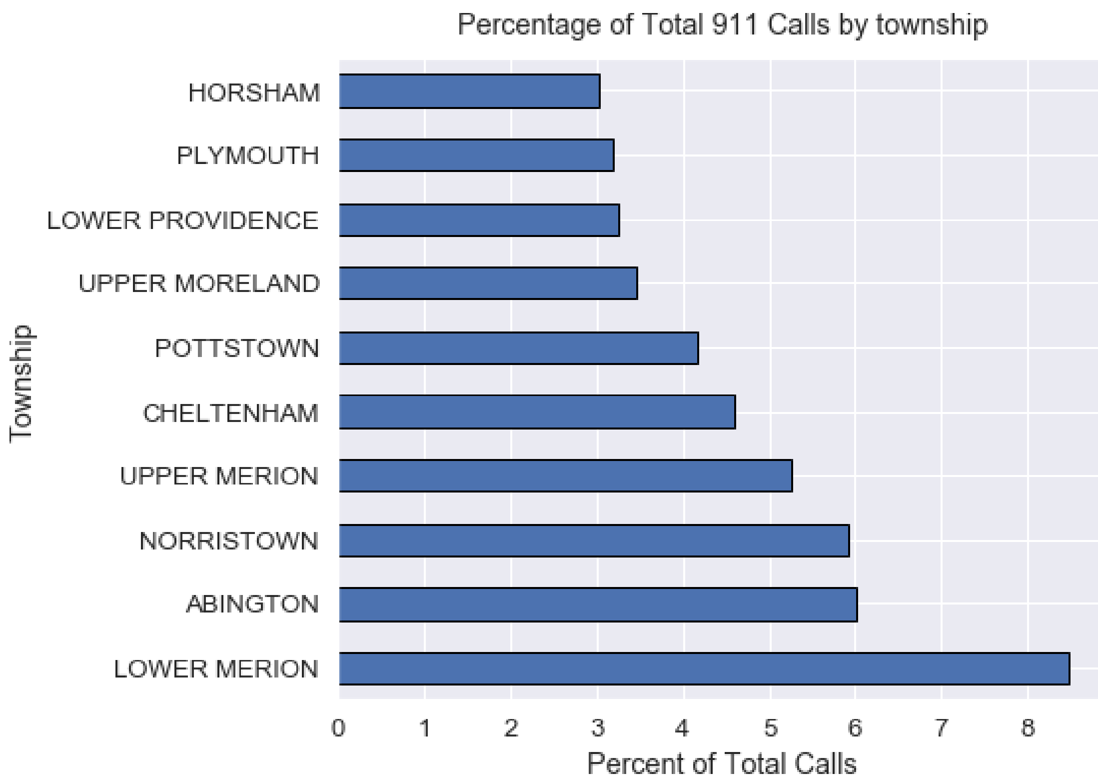
[119]: Text(0.5, 1.02, 'Number of 911 Calls by township')



- All 4 towns that seemed to have most 911 Calls from the map are actually in Top10 towns by the number of calls.

[120]:
```
df_temp = df['twp'].value_counts().head(10) / len(df) * 100
df_temp.plot(kind='barh', figsize=(7, 6), rot=0, edgecolor = 'black')
plt.xlabel("Percent of Total Calls", labelpad=5)
plt.ylabel("Township", labelpad=5)
plt.title("Percentage of Total 911 Calls by township", y=1.02)
```

[120]:
```
Text(0.5, 1.02, 'Percentage of Total 911 Calls by township')
```



[73]:
```
[sum(df['twp'].value_counts().head(10) / len(df)) * 100,
 10 / df['twp'].nunique() * 100]
```

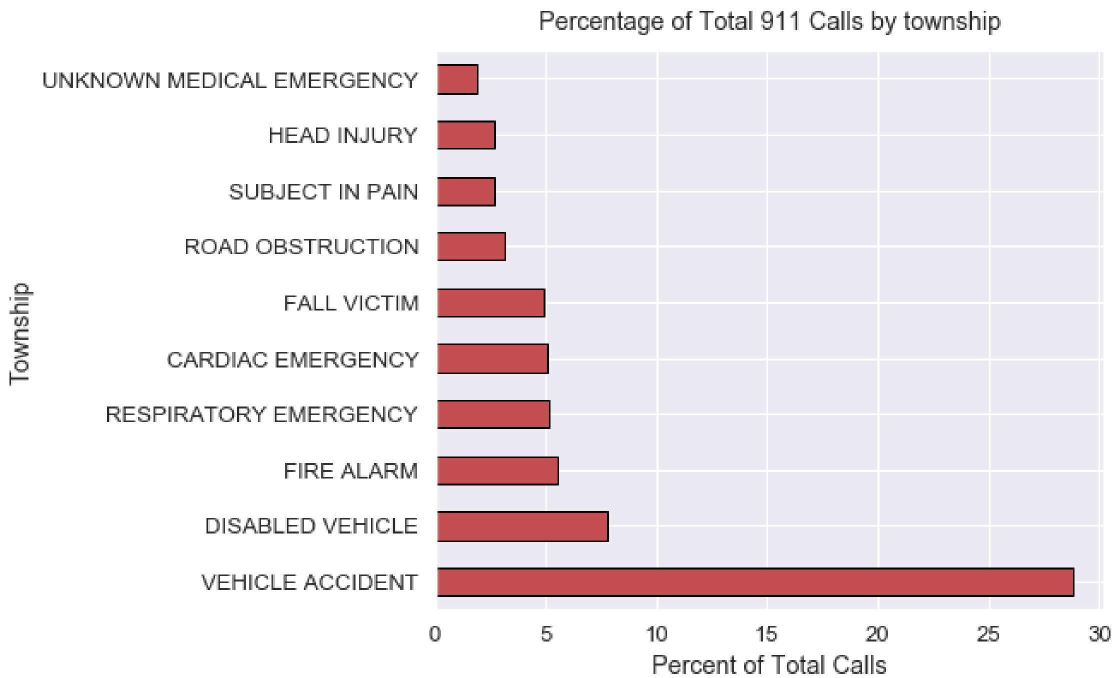
[73]:
```
[47.31837735697343, 14.705882352941178]
```

- 10 townships (15% of all townships in the county) make up 47% of Total 911 Calls in Mobtgomery County, PA.

1.3.8 Most popular Reasons for 911 Calls

[121]:
```
df_subt = df['Subtype'].value_counts().head(10) / len(df) * 100
df_subt.plot(kind='barh', figsize=(7, 6), rot=0, color = 'r', edgecolor =␣
 ↪'black')
plt.xlabel("Percent of Total Calls", labelpad=5)
plt.ylabel("Township", labelpad=5)
plt.title("Percentage of Total 911 Calls by township", y=1.02)
```

[121]:
```
Text(0.5, 1.02, 'Percentage of Total 911 Calls by township')
```



[126]:
```
[len(df[df['Subtype'] == 'VEHICLE ACCIDENT']) / len(df) * 100,
 sum(df['Subtype'].value_counts().head(10) / len(df)) * 100,
 10 / df['Subtype'].nunique() * 100]
```

[126]:
```
[28.785228963132713, 67.51799139629317, 13.157894736842104]
```

- Vehicle Accident Calls, most popular accident for 911 calls, make up almost 29% of all calls in Montgomery County
- 10 Call Subtypes (13% of all Subtypes) make up more than 67% of all calls.

1.4 Final Insights from the Analysis:

Total Calls:

- Almost 50% of all 911 Calls are Emergency calls. Trafic comes second with over 35%.

Variation over year:

- The number of total 911 calls (and also each separate type) does not seem to be dependent on the time of the year.
- The variation of **Fire and Traffic calls has a noticeable similar trend**. It seems the same events cause both types of calls.
- Unlike fire and traffic calls, emergency 911 calls didn't have outliers with much higher volume of inquiries.

Variation over weekends and time of day:

- Emergency and Traffic calls have a significant drop on weekends, but fire calls are consistent throughout the week.
- Most 911 calls happen during working hours, with traffic calls showing highest intensity right after working day 4-5pm.

Across towns:

- Lower Merion with more than 8000 calls and 8% of total in the analyzed period has the highest volume of 911 calls in Montgomery County.
- Top 10 townships (15% of all towns) make up 47% of Total 911 Calls in Montgomery County, PA.

By detailed reason for call:

- Vehicle Accident Calls, most popular accident for 911 calls, make up almost 29% of all calls in Montgomery County
- Top 10 Call Subtypes (reasons for call) make up more than 67% of all calls.