Scraping Data

The data related to murder rates in the US is scraped from the FBI website.
The code below uses the UCR url at https://ucr.fbi.gov/ucr-publication (https://ucr.fbi.gov/ucr-publication) and scrapes the data related to murders by Metrological Statistical Area (M.S.A)

Import required libraries

In [1]:
```
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import pandas as pd
import time
import seaborn as sns
import requests
from bs4 import BeautifulSoup
from IPython.display import IFrame, HTML
```

Function definitions

In [2]:
```
# Checks if words are part of any of the link titles a[title=?]
def filter_links_title (links, included_words):
    for link in links:
        if link.get("title"):
            if(included_words in link.get("title")):
                return link.get("href")

# Checks if a certain marker is part of the table row
def check_for_marker(els):
    return_value = False
    for el in els:
        if el.get("rowspan"):
            if int(str(el.get("rowspan"))) > 1:
                return_value = True
                break
    return return_value

# Returns true if a string is numeric
def is_number(s):
    try:
        float(s)
        return True
```

```
    except ValueError:
        return False

# Extracts population data from a row
def get_pop(els):
    for el in els:
        if is_number(el.get_text().replace(",","")):
            return int(el.get_text().replace(",",""))

# Imputes the data in a row using knn with 2 neighbors - Requires edits
def knn_impute_row(row, n):
    row = row.copy(deep=True)
    numbers = []
    for i,col in enumerate(row):
        if is_number(col):
            if not np.isnan(col):
                numbers.append(i)
    for i,col in enumerate(row):
        if(is_number(col)):
            if(np.isnan(col)):
                if(len(numbers) == 1):
                    row[i] = row[numbers[0]]
                else:
                    distance = [np.abs(index - i) for index in numbers]
                    knn_1 = row[numbers[np.argmin(distance)]]
                    distance[np.argmin(distance)] = 9999
                    knn_2 = row[numbers[np.argmin(distance)]]
                    row[i] = (knn_1 + knn_2)/2
    return row

# Shorthand function to create axs of a certain size and ravel for easier grap
hing
def get_axs(rows, columns, fig_size_width, fig_size_height):
    dims = (fig_size_width, fig_size_height)
    fig, axs = plt.subplots(rows, columns, figsize=dims)
    if(rows*columns>1):
         axs = axs.ravel()
    return axs
```

Initial Scraping

In [3]:
```
url = "https://ucr.fbi.gov/ucr-publications"
```

In [4]:
```
req = requests.get(url)
page = req.text
soup = BeautifulSoup(page, 'html.parser')
```

In [5]:
```
years = range(2006,2017)
```

In [6]:
```
year_links = {}
for year in years:
    year_links[year] = soup.find_all("a", string=year)[0]["href"]
```

Extract links for further scraping

In [7]:
```
for year in years:
    url = year_links[year]
    req = requests.get(url)
    page = req.text
    soup = BeautifulSoup(page, 'html.parser')
    if soup.find("a", string = "Violent Crime"):
        url = url.replace("/" + url.split("/")[-1], "")
        link_add = str(soup.find("a", string = "Violent Crime")["href"])
        if "http" not in link_add:
            year_links[year] = url + "/" + link_add
        else:
            year_links[year] = link_add
    else:
        print("Couldnt find link for year: " + str(year))
```

```
Couldnt find link for year: 2009
```

Link for 2009 manually entered

In [17]:
```
year_links[2009] = "https://www2.fbi.gov/ucr/cius2009/offenses/violent_crime/index.html"
```

Navigation to murder data by MSA

In [9]:
```
for year in years:
    url = year_links[year]
    req = requests.get(url)
    page = req.text
    soup = BeautifulSoup(page, 'html.parser')
    link_add = str(soup.find("a", string="Murder")["href"])
    if link_add:
       if "http" in link_add:
           year_links[year] = link_add
       else:
           year_links[year] = url.replace("/" + url.split("/")[-1], "") + "/"
+link_add
```

In [10]:
```
for year in years:
    url = year_links[year]
    req = requests.get(url)
    page = req.text
    soup = BeautifulSoup(page, 'html.parser')
    link_add = filter_links_title(soup.find_all("a"), "Metropolitan Statistica
l Area")
    if "http" in link_add:
        year_links[year] = link_add
    else:
        n_times = link_add.count('../')
        for i in range(0,n_times):
            url = url.replace("/" + url.split("/")[-1], "")
        year_links[year] = url.replace("/" + url.split("/")[-1], "") + "/" + l
ink_add.replace("../","")
```

Parsing data into Dataframe

In [11]:
```
MSA_INDEX = 0
POP_INDEX = 2
NAME_INDEX = 0
RATE_INDEX = 1
DATA_INDEX = 3
```

In [12]:
```
## Downloads MSA name, population and murder rate
murder_data = {}
for year in years:
    url = year_links[year]
    req = requests.get(url)
    page = req.text
    soup = BeautifulSoup(page, 'html.parser')
    x = soup.find("table", "data")
    rows = x.find("tbody").find_all("tr")
    data = []
    row_data = []
    for row in rows:
        elements = row.find_all(["th","td"])
        new_msa = check_for_marker(elements)
        if new_msa:
            if len(row_data)==2:
                data.append(row_data)
            row_data = []
            text = str(elements[MSA_INDEX].get_text().split("M.S.A")[0])
            row_data.append(text.split(",")[0].strip()+", "+ text.split(",")[1
].strip())

        if("Rate per 100,000 inhabitants" in elements[NAME_INDEX].get_text()):
            row_data.append(float(elements[DATA_INDEX].get_text()))
    df = pd.DataFrame(data, columns=["msa", str(year)]).sort_values("msa", asc
ending=[0])
    murder_data[year] = df.copy(deep=True)
```

Combining to create single Dataframe

In [13]:
```
result = murder_data[2006]
years = range(2007,2017)
for year in years:
    result = pd.merge(result, murder_data[year], left_on='msa', right_on='msa'
, how='outer')
years = range(2006,2017)
```

In [14]:
```
result = result.sort_values(["msa"], ascending = [1])
```

Exporting rows with missing data for imputation/manual fill

In [ ]:
```
missing_data = result[result.notnull().all(axis=1)]
missing_data.to_csv("murder_data_missing.csv")
missing_data.shape
```

In [15]:
```
non_missing_data = result[result.notnull().all(axis=1)]
non_missing_data.shape
```

Out[15]:
```
(168, 12)
```

Import dataframe for final imputation

In [76]:
```
murder_data = pd.read_csv("murder_data_full.csv", index_col=0)
murder_data.msa = murder_data.msa.str.replace(' M.D.','')
murder_data.msa = murder_data.msa.str.replace('\d+', '')
murder_data.index = murder_data.msa
murder_data.index = range(0, murder_data.shape[0])

msa_name = [msa.split(',')[0].strip() for msa in murder_data.msa]
msa_state = [msa.split(',')[1].strip() for msa in murder_data.msa]
murder_data['msa_name'] = msa_name
murder_data['msa_state'] = msa_state
murder_data.msa_state = murder_data.msa_state.str.replace('-',',')
murder_data.to_csv('FBI_Murder_Data_by_MSA.csv')
```

In [88]:
```
num_rows = murder_data.shape[0]
```

In [89]:
```
for row_index in range(0, num_rows):
    row = murder_data.iloc[row_index]
    if murder_data.iloc[row_index].isnull().any():
        imputed_row = knn_impute_row(row, 2)
        murder_data.iloc[row_index] = imputed_row
```