DC Crime Map

September 24, 2020

1 Folium Interactive Crime Map_DC

1.0.1 Data Cleaning and Processing

- Data was retreived from Metropolitan Police Department from Aug.31st, 2020
  - url link: https://dcatlas.dcgis.dc.gov/crimecards/
- More open data at: https://opendata.dc.gov/search?q=crime%20incidents

[1]:
```
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np

import geopandas as gpd
import branca.colormap as cm

from shapely.geometry import Point

import warnings
warnings.simplefilter(action = 'ignore', category = FutureWarning)
```

[2]:
```
# Download data from link
url = 'https://datagate.dc.gov/search/open/crimes?
 ↪daterange=2years&details=true&format=csv'
df = pd.read_csv(url)
df.head()
```

[2]:

	NEIGHBORHOOD_CLUSTER	CENSUS_TRACT	offensegroup	LONGITUDE	\
0	cluster 17	1804.0	property	-77.027957	
1	cluster 8	4702.0	property	-77.015679	
2	cluster 22	9102.0	property	-76.993150	
3	cluster 6	10700.0	property	-77.039451	
4	cluster 5	5600.0	property	-77.054266	

	END_DATE	offense-text	SHIFT	YBLOCK	DISTRICT	WARD	\
0	NaN	theft/other	evening	143785.0	4.0	4	
1	NaN	theft/other	day	137185.0	1.0	6	
2	2019-06-08T10:18:57.000	theft/other	day	139371.0	5.0	5	

```
3  2019-06-08T11:11:11.000   theft/other      day  137120.0       2.0     2
4  2019-06-08T10:50:33.000  theft f/auto      day  137321.0       2.0     2

   …                                        BLOCK               START_DATE  \
0  …     5900 - 5999 block of georgia avenue nw  2019-06-03T17:12:18.000
1  …             300 - 399 block of k street nw  2019-06-08T08:05:00.000
2  …  900 - 999 block of rhode island avenue ne  2019-06-08T10:13:43.000
3  …          900 - 999 block of 17th street nw  2019-06-08T10:30:50.000
4  …           2500 - 2599 block of l street nw  2019-06-07T21:30:31.000

        CCN       OFFENSE  OCTO_RECORD_ID  ANC               REPORT_DAT  \
0  19096357   theft/other     19096357-01   4A  2019-06-03T21:28:45.000Z
1  19099579   theft/other     19099579-01   6E  2019-06-08T13:12:10.000Z
2  19099605   theft/other     19099605-01   5C  2019-06-08T15:01:51.000Z
3  19099615   theft/other     19099615-01   2B  2019-06-08T15:11:04.000Z
4  19099628  theft f/auto     19099628-01   2A  2019-06-08T17:14:21.000Z

   METHOD                                location   LATITUDE
0  others  38.961978891670213,-77.027959395242561  38.961971
1  others  38.902526359929666,-77.015681184476662  38.902519
2  others  38.922219443509491,-76.993152403339593  38.922212
3  others  38.901935207057228,-77.039452838989632  38.901927
4  others  38.903739943954207,-77.054268346692766  38.903732

[5 rows x 29 columns]
```

[3]:
```
# Check data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63962 entries, 0 to 63961
Data columns (total 29 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   NEIGHBORHOOD_CLUSTER  63394 non-null  object
 1   CENSUS_TRACT          63882 non-null  float64
 2   offensegroup          63962 non-null  object
 3   LONGITUDE             63962 non-null  float64
 4   END_DATE              54957 non-null  object
 5   offense-text          63962 non-null  object
 6   SHIFT                 63962 non-null  object
 7   YBLOCK                63962 non-null  float64
 8   DISTRICT              63955 non-null  float64
 9   WARD                  63962 non-null  int64
 10  YEAR                  63962 non-null  int64
 11  offensekey            63962 non-null  object
 12  BID                   12273 non-null  object
```

```
 13  sector                63955 non-null  object
 14  PSA                   63955 non-null  float64
 15  ucr-rank              63962 non-null  int64
 16  BLOCK_GROUP           63882 non-null  object
 17  VOTING_PRECINCT       63962 non-null  object
 18  XBLOCK                63962 non-null  float64
 19  BLOCK                 63962 non-null  object
 20  START_DATE            63962 non-null  object
 21  CCN                   63962 non-null  int64
 22  OFFENSE               63962 non-null  object
 23  OCTO_RECORD_ID        63962 non-null  object
 24  ANC                   63962 non-null  object
 25  REPORT_DAT            63962 non-null  object
 26  METHOD                63962 non-null  object
 27  location              63962 non-null  object
 28  LATITUDE              63962 non-null  float64
dtypes: float64(7), int64(4), object(18)
memory usage: 14.2+ MB
```

[4]:
```
# Check common column names
def common_member(list1, list2):

    list1_as_set = set(list1)
    intersection = list1_as_set.intersection(list2)

    intersection_as_list = list(intersection)

    return(intersection_as_list)
```

[5]:
```
# Get value counts for interested variables
# Get tempary dataframe with object-format columns
def dataCheck(df):
    # Variables of interest
    obj_intr = ['NEIGHBORHOOD_CLUSTER', 'offensegroup', 'offense-text',␣
 ↪'SHIFT', 'DISTRICT', 'YEAR']
    col_names = df.columns.values

    #obj_df = temp_col_df.select_dtypes(include = ['object'])
    temp_num_names = common_member(obj_intr, col_names)
    if temp_num_names:

        for col in temp_num_names:
            print("\n\nAttribute Frequency for", col,␣
 ↪"\n------------------------------")
            display(df[col].value_counts())
```

[6]:
```
dataCheck(df)
```

```
Attribute Frequency for offense-text
------------------------------

theft/other                   27787
theft f/auto                  20352
motor vehicle theft            4960
robbery                        4124
assault w/dangerous weapon     3211
burglary                       2783
sex abuse                       370
homicide                        353
arson                            22
Name: offense-text, dtype: int64



Attribute Frequency for YEAR
------------------------------

2019    33909
2020    18677
2018    11376
Name: YEAR, dtype: int64



Attribute Frequency for NEIGHBORHOOD_CLUSTER
------------------------------

cluster 8     4564
cluster 2     4495
cluster 25    4034
cluster 6     3526
cluster 3     3244
cluster 23    2760
cluster 26    2703
cluster 18    2699
cluster 21    2388
cluster 22    2387
cluster 7     2381
cluster 4     2056
cluster 39    1975
cluster 17    1717
cluster 34    1697
cluster 31    1616
cluster 33    1518
```

```
cluster 32    1429
cluster 1     1327
cluster 11    1218
cluster 9     1217
cluster 24    1106
cluster 5     1062
cluster 19    1030
cluster 30     971
cluster 15     828
cluster 38     826
cluster 20     738
cluster 35     720
cluster 28     689
cluster 27     637
cluster 10     562
cluster 37     552
cluster 14     551
cluster 36     541
cluster 13     507
cluster 16     486
cluster 12     461
cluster 29     131
cluster 45      18
cluster 43      11
cluster 44       9
cluster 46       4
cluster 40       2
cluster 41       1
Name: NEIGHBORHOOD_CLUSTER, dtype: int64



Attribute Frequency for offensegroup
------------------------------

property    55904
violent      8058
Name: offensegroup, dtype: int64



Attribute Frequency for DISTRICT
------------------------------

2.0    13142
3.0    11602
5.0     9748
1.0     8828
```

```
6.0     8191
4.0     7803
7.0     4641
Name: DISTRICT, dtype: int64



Attribute Frequency for SHIFT
------------------------------

evening     27828
day         22256
midnight    13878
Name: SHIFT, dtype: int64
```

[7]:
```
# Create date for different years
dc_crime18 = df[df['YEAR'] == 2018]
dc_crime19 = df[df['YEAR'] == 2019]
dc_crime20 = df[df['YEAR'] == 2020]
```

1.0.2 Crime Data Visualization

[8]:
```
import folium
import matplotlib.pyplot as plt
import seaborn as sns

import folium.plugins
from folium.plugins import MarkerCluster
from branca.element import Template, MacroElement
```

1.0.3 Attributes need for the map

LATITUDE, LONGITUDE, offense-text, METHOD, BLOCK_GROUP, SHIFT, START_DATE

crime_type = [theft/other, theft f/auto, motor vehicle theft, robbery, assault w/dangerous weapon, burglary, sex abuse, homicide] ### Set Colors **darkred:** THEFT/OTHER

orange: THEFT F/AUTO

lightblue: MOTOR VEHICLE THEFT

gray: ROBBERY

cadetblue: ASSAULT W/DANGEROUS WEAPON

blue: BURGLARY

green: SEX ABUSE

purple: HOMICIDE

black: others

[9]:
```
# Crime data: set color for each crime type and get index used during for␣
 ↪visualization
def crime_color_idx(df):
        # Calculate the average lat and long used as center of the map:
    ave_lat = sum(df.LATITUDE)/len(df.LATITUDE)
    ave_long = sum(df.LONGITUDE)/len(df.LONGITUDE)

    crime_type = ['theft/other', 'theft f/auto', 'motor vehicle theft',␣
 ↪'robbery',
              'assault w/dangerous weapon', 'burglary', 'sex abuse', 'homicide']

    # Set color list
    color_list = []
    off_txt_idx = df.columns.get_loc('offense-text')

    for i in range(len(df)):
        if df.iloc[i][off_txt_idx] == 'theft/other':
            color = "darkred"
        elif df.iloc[i][off_txt_idx] == 'theft f/auto':
            color = "orange"
        elif df.iloc[i][off_txt_idx] == 'motor vehicle theft':
            color = "lightblue"
        elif df.iloc[i][off_txt_idx] == 'robbery':
            color = "gray"
        elif df.iloc[i][off_txt_idx] == 'assault w/dangerous weapon':
            color = "cadetblue"
        elif df.iloc[i][off_txt_idx] == 'burglary':
            color = "blue"
        elif df.iloc[i][off_txt_idx] == 'sex abuse':
            color = "green"
        elif df.iloc[i][off_txt_idx] == 'homicide':
            color = "purple"
        else:
            color = "black"
        color_list.append(color)

    df['color'] = pd.Series(color_list).values

    # Get column indices
    lat_idx = df.columns.get_loc('LATITUDE')
    long_idx = df.columns.get_loc('LONGITUDE')
    color_idx = df.columns.get_loc('color')
```

```
    method_idx = df.columns.get_loc('METHOD')
    block_gp_idx = df.columns.get_loc('BLOCK_GROUP')
    shift_idx = df.columns.get_loc('SHIFT')
    str_date_idx = df.columns.get_loc('START_DATE')

    return off_txt_idx, lat_idx, long_idx, color_idx, method_idx, block_gp_idx,␣
↪shift_idx, str_date_idx
```

[10]:
```
def crime_cluster_map(df, mapobj, title):

    off_txt_idx, lat_idx, long_idx, color_idx, method_idx, block_gp_idx,␣
↪shift_idx, str_date_idx = crime_color_idx(df)

    marker_cluster = MarkerCluster().add_to(mapobj)

    # Set map popup info box layout
    for i in range(len(df['LATITUDE'])):
        lat = df.iloc[i][lat_idx]
        long = df.iloc[i][long_idx]
        color = df.iloc[i][color_idx]
        popup_text = """ <b>Offense</b>: {}<br>
                         <b>Methd</b>: {}<br>
                         <b>Block-Group</b>: {}<br>
                         <b>Shift</b>: {}<br>
                         <b>Start Date</b>: {}<br>"""

        popup_text = popup_text.format(df.iloc[i][off_txt_idx],
                                       df.iloc[i][method_idx],
                                       df.iloc[i][block_gp_idx],
                                       df.iloc[i][shift_idx],
                                       df.iloc[i][str_date_idx]
                                      )
        folium.Marker(location = [lat, long], popup = popup_text,
                      icon = folium.Icon(color=color, icon =␣
↪'exclamation-triangle', prefix='fa')).add_to(marker_cluster)


    # Add the legend to the map
    template = """
    {% macro html(this, kwargs) %}

    <!doctype html>
    <html lang="en">
    <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <title>jQuery UI Draggable - Default functionality</title>
```

```
    <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/
↪jquery-ui.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
↪font-awesome/4.7.0/css/font-awesome.min.css">

    <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
    <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

    <script>
    $( function() {
      $( "#maplegend" ).draggable({
                      start: function (event, ui) {
                          $(this).css({
                              right: "auto",
                              top: "auto",
                              bottom: "auto"
                          });
                      }
                  });
  });

    </script>
  </head>
  <body>


  <div id='maplegend' class='maplegend'
      style='position: absolute; z-index:9999; border:2px solid grey;␣
↪background-color:rgba(255, 255, 255, 0.8);
       border-radius:6px; padding: 10px; font-size:14px; right: 20px; bottom:␣
↪20px;'>

  <div class='legend-title'>Offenses</div>
  <div class='legend-scale'>
    <ul class='legend-labels'>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: darkred"␣
↪></i>  THEFT/OTHER</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: orange"␣
↪></i>  THEFT F/AUTO</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color:␣
↪lightblue" ></i>  MOTOR VEHICLE THEFT</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: gray" ></
↪i>  ROBBERY</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color:␣
↪cadetblue" ></i>  ASSAULT W/DANGEROUS WEAPON</li>
```

```
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: blue" ></
↪i>  BURGLARY</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: green" ></
↪i>  SEX ABUSE</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: purple"␣
↪></i>  HOMICIDE</li>
      <li><i class="fa fa-exclamation-triangle fa-lg" style="color: black" ></
↪i>  OTHERS</li>

    </ul>
  </div>
  </div>

  </body>
  </html>

  <style type='text/css'>
    .maplegend .legend-title {
      text-align: left;
      margin-bottom: 5px;
      font-weight: bold;
      font-size: 120%;
      }
    .maplegend .legend-scale ul {
      margin: 0;
      margin-bottom: 3px;
      padding: 0;
      float: left;
      list-style: none;
      }
    .maplegend .legend-scale ul li {
      font-size: 90%;
      list-style: none;
      margin-left: 2px;
      margin-right: 2px;
      line-height: 23px;
      margin-bottom: 3px;
      }
    .maplegend ul.legend-labels li span {
      display: block;
      float: left;
      height: 18px;
      width: 30px;
      margin-right: 5px;
      margin-left: 0;
      border: 1px solid #999;
      }
```

```
      .maplegend .legend-source {
        font-size: 80%;
        color: #777;
        clear: both;
        }
      .maplegend a {
        color: #777;
        }
    </style>
    {% endmacro %}"""

    macro = MacroElement()
    macro._template = Template(template)
    mapobj.get_root().add_child(macro)

    mapobj.save(title + '.html')

    return mapobj
```

[11]:
```
def default_map(lat = 38.9072, long = -77.0369):
    # Set DC map default layout
    ch_map = folium.Map(location = [lat, long],
                        zoom_start = 12,
                        tiles = "cartodbpositron",
                        control_scale = True) # openstreetmap option available
    folium.TileLayer('openstreetmap').add_to(ch_map)

    return ch_map
```

[12]:
```
ch_map = default_map()

DC_Crime_Cluster18 = crime_cluster_map(dc_crime18, ch_map, 'DC_Crime_Cluster18')
DC_Crime_Cluster18
```

[12]:
```
<folium.folium.Map at 0x1a20cda898>
```

[13]:
```
DC_Crime_Cluster19 = crime_cluster_map(dc_crime19, ch_map, 'DC_Crime_Cluster19')
DC_Crime_Cluster20 = crime_cluster_map(dc_crime20, ch_map, 'DC_Crime_Cluster20')
```

[ ]:
```

```

1.0.4 Folium Choropleth Map

Useful resources: 1. Interactive choropleth with Python and Folium (and some tips) - https://vverde.github.io/blob/interactivechoropleth.html 2. Python's Folium to create choropleth maps - https://www.nagarajbhat.com/post/folium-visualization/ 3. Creating Web Maps in

Python with GeoPandas and Folium - http://andrewgaidus.com/leaflet_webmaps_python/ 4. Your Guide to Getting Started with Geospatial Analysis using Folium (with multiple case studies) - https://www.analyticsvidhya.com/blog/2020/06/guide-geospatial-analysis-folium-python/ 5. Colormaps on Different Layers - https://nbviewer.jupyter.org/gist/BibMartin/f153aa957ddc5fadc64929abdee9ff2e 6. Examples of plugins usage in folium - https://nbviewer.jupyter.org/github/python-visualization/folium/blob/master/examples/Plugins.ipynb#Sub-categories 7. Color Brewer (color advice for cartography) - https://colorbrewer2.org/#type=diverging&scheme=RdYlBu&n=8

Data Used
+ **2020 Shapefile**: Data was retreived from Metropolitan Police Department <Preliminary 2020 Census Tract> from Aug.31st, 2020 + url link: https://opendata.dc.gov/datasets/preliminary-2020-census-tract-1?geometry=-77.910%2C38.707%2C-76.119%2C39.081 + **2019 Shapefile**: Data was retreived from Metropolitan Police Department from Sep. 1st, 2020
+ url link: https://catalog.data.gov/dataset/tiger-line-shapefile-2019-state-district-of-columbia-current-census-tract-state-based + **2018 Shapefile**: Data was retreived from Metropolitan Police Department from Sep. 1st, 2020
+ url link: https://catalog.data.gov/dataset/tiger-line-shapefile-2018-county-district-of-columbia-dc-topological-faces-polygons-with-all-ge + **ACS 2018 Population Variables Tract**: Data was retreived from Metropolitan Police Department <Preliminary 2020 Census Tract> from Sep. 1st, 2020
+ url link: https://opendata.dc.gov/datasets/acs-2018-population-variables-tract?geometry=-77.793%2C38.712%2C-76.237%2C39.086

- **ACS 2018 Median Household Income Variables Tract**: Data was retreived from Metropolitan Police Department <Preliminary 2020 Census Tract> from Sep. 1st, 2020
  - url link: https://opendata.dc.gov/datasets/acs-2018-median-household-income-variables-tract?geometry=-77.793%2C38.712%2C-76.237%2C39.086
- **ACS 2018 Employment Status Variables Tract**: Data was retreived from Metropolitan Police Department <Preliminary 2020 Census Tract> from Sep. 1st, 2020
  - url link: https://opendata.dc.gov/datasets/acs-2018-employment-status-variables-tract/data?geometry=-77.793%2C38.712%2C-76.237%2C39.086&orderBy=B23025_007E&orderByAsc=false&selectedAttribute=ALAND

[14]:
```
import geopandas as gpd
import branca.colormap as cm

from folium import plugins



from shapely.geometry import Point
```

[15]:
```
# Check common column names
def uncommon_member(list1, list2):

    list1_as_set = set(list1)
    intersection = list1_as_set.difference(list2)
```

```
    intersection_as_list = list(intersection)

    return(intersection_as_list)
# Check differences between two sets
'''
uncommon_member(crime_ct19['TRACTCE'], shape_file19['TRACTCE'])
uncommon_member(shape_file19['TRACTCE'], crime_ct19['TRACTCE'])
'''
```

[15]:
```
"\nuncommon_member(crime_ct19['TRACTCE'],
shape_file19['TRACTCE'])\nuncommon_member(shape_file19['TRACTCE'],
crime_ct19['TRACTCE'])\n"
```

[16]:
```
# Style plot if data contains null values
def style_zero_function(feature):
    default_style = {
        'fillOpacity': 0.1,
        'color': 'gray',
        'weight': 0.0001
    }

    default_style['fillPattern'] = plugins.pattern.StripePattern(angle = -45)

    return default_style
```

Possible colormaps palettes:

YlGnBu_09, viridis, YlOrRd_04, Pastel1_09, PuBuGn_03, RdYlBu_11, PuBuGn_03, Paired_03, RdPu_05, Paired_03

[17]:
```
def set_colormap(gdf, value_field, cmap_caption, cmap_color, num_classes):
    # Set color brackets
    if value_field == 'CRIME_CT':
        _, threshold_scale = pd.qcut(gdf[value_field], num_classes, retbins =␣
↪True)
    else:
        threshold_scale = np.linspace(gdf[value_field].min(), gdf[value_field].
↪max(), num_classes, dtype = int).tolist()

    # Set options for different colormaps
    if cmap_color == 'YlGnBu_09':
        colormap = cm.linear.YlGnBu_09.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'Greys_03':
        colormap = cm.linear.Greys_03.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
```

```
        colormap.caption = cmap_caption
    elif cmap_color == 'YlOrRd_04':
        colormap = cm.linear.YlOrRd_04.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'Spectral_07':
        colormap = cm.linear.Spectral_07.to_step(data = gdf[value_field], index␣
↪= threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'PuBuGn_03':
        colormap = cm.linear.PuBuGn_03.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'RdYlBu_11':
        colormap = cm.linear.RdYlBu_11.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'PuBuGn_03':
        colormap = cm.linear.PuBuGn_03.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'YlGn_03':
        colormap = cm.linear.YlGn_03.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    elif cmap_color == 'RdPu_05':
        colormap = cm.linear.RdPu_05.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption
    else:
        colormap = cm.linear.PuBu_06.to_step(data = gdf[value_field], index =␣
↪threshold_scale)
        colormap.caption = cmap_caption

    return colormap
```

[18]:
```
# add_choropleth layer to map
def add_choropleth_layer(mapobj, gdf, value_field, cmap_caption, popup_fields,␣
↪popup_captions, cmap_color, show = False, num_classes = 6):
    # get colormap
    colormap = set_colormap(gdf, value_field, cmap_caption, cmap_color,␣
↪num_classes)

    # Select non-null dataset
    nonzero_df = gdf[gdf[value_field].notnull()]
```

```
    # Set colormap and highlight color
    style_function = lambda x: {'weight': 0.5,
                                'color': 'black',
                                'fillColor':␣
↪colormap(x['properties'][value_field]),
                                'fillOpacity': 0.55}
    highlight_function = lambda x: {'fillColor': '#000000',
                                    'color': '#000000',
                                    'fillOpacity': 0.50,
                                    'weight': 0.1}


    # Create layer & set popup box info
    GDF = folium.features.GeoJson(
                            nonzero_df,
                            style_function = style_function,
                            name = cmap_caption,
                            control = True,
                            show = show,
                            highlight_function = highlight_function,
                            tooltip = folium.features.GeoJsonTooltip(fields =␣
↪popup_fields,
                                    aliases = popup_captions,
                                    style=("background-color: white; color:␣
↪#333333; font-family: arial; font-size: 12px; padding: 10px;"),
                                    sticky = True
                                    )
                            )

    # Set layout for null dataframe
    if gdf[value_field].isna().sum() != 0:
        zero_df = gdf[gdf[value_field].isnull()].fillna(0)

        folium.GeoJson(
                zero_df,
                style_function = style_zero_function,
                name = cmap_caption,
                control = True,
                #overlay = False,
                tooltip = folium.GeoJsonTooltip(
                                fields = [value_field],
                                aliases = [cmap_caption],
                                localize = False
                                )).add_to(GDF)
        '''
        folium.GeoJson(
```

```
                zero_df,
                style_function = lambda x: {
                    'color': 'black',
                    'weight': 0.25,
                    'fillOpacity': 0
                },
                name = cmap_caption).add_to(GDF)
        '''

    '''
    colormap.add_to(choropleth_lyr)
    choropleth_lyr.add_child(GDF)
    mapobj.add_child(choropleth_lyr)
    '''
    if value_field == 'Total Population' or value_field == 'CRIME_CT':

        colormap.add_to(mapobj)

    mapobj.add_child(GDF)


    return mapobj
```

[19]:
```
# GeoPandas load DC CENSUS TRACT shapefiles, need to contain 'TRACTCE' column
def prepare_choromap_data(gdf, df):
    gdf['TRACTCE'] = gdf['TRACTCE'].astype(float)

    # Check uncommon TRACTCE
    '''
    uncommon_member(df['TRACTCE'], gdf['TRACTCE'])
    uncommon_member(gdf['TRACTCE'], df['TRACTCE'])
    '''

    # Generate crime counts daraframe
    crime_ct = df['CENSUS_TRACT'].value_counts().reset_index()
    crime_ct.columns = ['TRACTCE', 'CRIME_CT']

    # Merge shapefile and crime file
    dc = gdf.merge(crime_ct, on = 'TRACTCE', how = "left")


    # Preliminary_2020_Census_Tract.shp contains population,
    # and return the geodataframe with only the fields of our interest

    if 'POP10' in dc.columns:
        dc['CRIME_RATE'] = round(dc['CRIME_CT'] / dc['POP10'] * 100, 2)
```

```
        # Datasets for choropleth
        dc_merge = dc[['TRACTCE', 'TRACTID', 'CRIME_CT', 'POP10', 'CRIME_RATE',␣
↪'geometry']]
        dc_merge.columns = [['TRACTCE', 'GEOID', 'CRIME_CT', 'POP10',␣
↪'CRIME_RATE', 'geometry']]


    else:
        dc_merge = dc[['TRACTCE', 'GEOID', 'CRIME_CT', 'geometry']]

    return dc_merge
```

[20]:
```
# Load shapefiles
shape_file20 = gpd.read_file('./Data/Preliminary_2020_Census_Tract-shp/
↪Preliminary_2020_Census_Tract.shp')
dc20_merge = prepare_choromap_data(shape_file20, dc_crime20)
#dc20_merge.head()
```

[21]:
```
shape_file19 = gpd.read_file('./Data/tl_2019_11_tract/tl_2019_11_tract.shp')
dc19_merge = prepare_choromap_data(shape_file19, dc_crime19)
#dc19_merge.head()
```

[22]:
```
shape_file18 = gpd.read_file('./Data/tl_2018_11_tract/tl_2018_11_tract.shp')
dc18_merge = prepare_choromap_data(shape_file18, dc_crime18)
#dc18_merge.head()
```

1.0.5 Data Processing Demographic Shapefiles for DC 2018

- ACS 2018 Employment Status Variables Tract
- ACS 2018 Median Household Income Variables Tract
- ACS 2018 Population Variables Tract

[23]:
```
# Load demographics shapefiles for 2018
employ18 = gpd.read_file('./Data/ACS_2018_Employment_Status_Variables_Tract-shp/
↪957c168f-c798-4686-a8da-d1320ac069e0202041-1-vtq3ml.feh1p.shp')
household_inc18 =gpd.read_file('./Data/
↪ACS_2018_Median_Household_Income_Variables_Tract-shp/
↪c9ca5f40-0f43-4de6-a527-28440f3bdf132020330-1-9idood.na0x.shp')
pop18 =gpd.read_file('./Data/ACS_2018_Population_Variables_Tract-shp/
↪1a06e536-b186-4e78-bab7-63836dce84f82020328-1-r1rbgx.oico.shp')
```

[24]:
```
'''
# Percent Unemployed
employ18['B23025_c_2']
# Percent Not in Labor Force
employ18['B23025_ca1']
```

```
# Mean Usual Hours Worked of Population Age 16 to 64
employ18['B23020_001']
# Mean Usual Hours Worked of Men Age 16 to 64
employ18['B23020_002']
# Mean Usual Hours Worked of Women Age 16 to 64
employ18['B23020_003']
'''
# Select attributes of interests
employ18_select = employ18[['GEOID', 'B23025_c_2', 'B23025_cal', 'B23020_001',␣
 ↪'B23020_002', 'B23020_003']]
employ18_select.columns = ['GEOID', 'Percent Unemployed', 'Percent Not in Labor␣
 ↪Force', 'Mean Usual Hours Worked of Population Age 16 to 64',
                'Mean Usual Hours Worked of Men Age 16 to 64', 'Mean Usual␣
 ↪Hours Worked of Women Age 16 to 64']

employ18_group = employ18_select.groupby('GEOID', as_index = False).mean()
```

[25]:
```
household_inc18_select = household_inc18[['GEOID', 'B19049_001', 'B19013B_00',
                                          'B19013D_00', 'B19013H_00',
                                          'B19013I_00', 'B19053_001']]
household_inc18_select.columns = ['GEOID', 'Median Household Income', 'Median␣
 ↪Household Income (Black or African American Householder)',
                                  'Median Household Income (Asian␣
 ↪Householder)', 'Median Household Income (Non-Hispanic White Householder)',
                                  'Median Household Income (Hispanic or Latino␣
 ↪Householder)', 'Total Households']

household_inc18_group = household_inc18_select.groupby('GEOID', as_index =␣
 ↪False).mean()
```

[26]:
```
pop18_select = pop18[['GEOID', 'B01001_001', 'B01001_cal',
                      'B01001_c_4', 'B01001_c_8']]
pop18_select.columns = ['GEOID', 'Total Population', 'Ratio of Males to␣
 ↪Females',
                        'Percent of Population Less Than 18 Years', 'Percent of␣
 ↪Population 65 Years and Over']

pop18_group = pop18_select.groupby('GEOID', as_index = False).sum()
```

[27]:
```
# Merge cleaned demographic shapefiles
from functools import reduce
dfs18 = [employ18_group, household_inc18_group, pop18_group]
df_final18 = reduce(lambda left, right: pd.merge(left, right, on = 'GEOID'),␣
 ↪dfs18)
df_final18
```

[27]:

	GEOID	Percent Unemployed	Percent Not in Labor Force	\
0	11001000100	2.4	22.4	
1	11001000201	5.5	61.9	
2	11001000202	3.5	35.0	
3	11001000300	2.6	21.6	
4	11001000400	0.7	37.1	
..	…	…	…	
174	11001010700	4.1	25.6	
175	11001010800	8.0	60.6	
176	11001010900	14.1	35.7	
177	11001011000	5.3	27.0	
178	11001011100	11.0	38.7	

	Mean Usual Hours Worked of Population Age 16 to 64	\
0	44.7	
1	22.1	
2	34.6	
3	39.4	
4	45.5	
..	…	
174	40.9	
175	25.3	
176	37.3	
177	42.0	
178	40.2	

	Mean Usual Hours Worked of Men Age 16 to 64	\
0	45.9	
1	24.6	
2	37.0	
3	39.6	
4	44.6	
..	…	
174	39.6	
175	26.5	
176	37.0	
177	43.5	
178	41.4	

	Mean Usual Hours Worked of Women Age 16 to 64	Median Household Income	\
0	43.4	191146.0	
1	19.7	NaN	
2	32.2	170987.0	
3	39.3	152120.0	
4	46.1	126731.0	
..	…	…	
174	42.4	73688.0	

```
175                                          24.3                  45278.0
176                                          37.4                  41660.0
177                                          40.8                  84375.0
178                                          38.8                  79628.0

     Median Household Income (Black or African American Householder)  \
0                                                    NaN
1                                                    NaN
2                                                    NaN
3                                                    NaN
4                                                    NaN
..                                                   …
174                                                  NaN
175                                                  NaN
176                                              42015.0
177                                              45148.0
178                                              63929.0

     Median Household Income (Asian Householder)  \
0                                       234083.0
1                                            NaN
2                                       188611.0
3                                       161800.0
4                                         2499.0
..                                           …
174                                      30962.0
175                                          NaN
176                                          NaN
177                                     110714.0
178                                     223750.0

     Median Household Income (Non-Hispanic White Householder)  \
0                                               195859.0
1                                                    NaN
2                                               182972.0
3                                               144359.0
4                                               125673.0
..                                                   …
174                                              78375.0
175                                              47792.0
176                                                  NaN
177                                              96500.0
178                                             117857.0

     Median Household Income (Hispanic or Latino Householder)  \
0                                                83601.0
1                                                    NaN
```

```
2                                          150278.0
3                                          229853.0
4                                          203393.0
..                                              …
174                                         78472.0
175                                             NaN
176                                             NaN
177                                        133445.0
178                                             NaN

     Total Households  Total Population  Ratio of Males to Females  \
0                2351              5160                       0.89
1                   0              3817                       0.88
2                1563              4541                       1.03
3                2455              6334                       0.81
4                 618              1428                       0.74
..                …                 …                          …
174              1101              1756                       1.25
175               782              6356                       0.79
176              1263              3819                       0.96
177              2645              4188                       0.75
178              1741              5571                       1.02

     Percent of Population Less Than 18 Years  \
0                                        17.2
1                                         1.9
2                                         8.1
3                                        15.0
4                                        18.6
..                                        …
174                                       1.2
175                                       0.7
176                                      29.2
177                                       6.3
178                                      17.2

     Percent of Population 65 Years and Over
0                                       16.7
1                                        0.0
2                                       19.1
3                                        7.7
4                                       22.4
..                                       …
174                                     11.2
175                                      0.8
176                                      6.1
177                                     25.4
```

```
178                                    20.0

[179 rows x 16 columns]
```

[28]:
```
# Merge cleaned demographic shapefiles with cleaned crime shapefiles together␣
 ↪on GEOID ro prepare final data for 2018
dc18 = dc18_merge.merge(df_final18, on = 'GEOID', how = "outer")

# Generate new features, 'crime count' and change 'median household income'␣
 ↪datatype
dc18['CRIME_RATE'] = round(dc18['CRIME_CT'] / dc18['Total Population']* 100, 2)
dc18['Median Household Income'] = dc18['Median Household Income'].astype(float)
#dc18 = dc18.fillna("NA")

#dc18.head()
```

1.0.6 Generate Multi-layer Demographics Choropleth Map

[29]:
```
######## Demographic Choropleth Map (Population, Household Income, Unemployment␣
 ↪Rate, Crime Rate) ############
demographic_map = default_map()


demographic_map = add_choropleth_layer(demographic_map, dc18, 'Total␣
 ↪Population', 'Total Population',
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
 ↪Population Less Than 18 Years', 'Percent of Population 65 Years and Over',␣
 ↪'CRIME_RATE'],
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
 ↪Population Less Than 18 Years %', 'Percent of Population 65 Years and Over␣
 ↪%', 'Crime Rate %'],
             'YlGnBu_09', show = True)
demographic_map = add_choropleth_layer(demographic_map, dc18, 'Median Household␣
 ↪Income', 'Households Income',
            ['Total Households', 'Median Household Income', 'Median Household␣
 ↪Income (Black or African American Householder)', 'Median Household Income␣
 ↪(Asian Householder)',
             'Median Household Income (Non-Hispanic White Householder)',␣
 ↪'Median Household Income (Hispanic or Latino Householder)'],
            ['Total Households', 'Median Household Income', 'Median Household␣
 ↪Income (Black or African American Householder)', 'Median Household Income␣
 ↪(Asian Householder)',
             'Median Household Income (Non-Hispanic White Householder)',␣
 ↪'Median Household Income (Hispanic or Latino Householder)'],
             'RdPu_05')
```

```
demographic_map = add_choropleth_layer(demographic_map, dc18, 'Percent␣
↪Unemployed', 'Unemployment Rate %',
            ['Percent Unemployed', 'Percent Not in Labor Force', 'Mean Usual␣
↪Hours Worked of Population Age 16 to 64',
             'Mean Usual Hours Worked of Men Age 16 to 64', 'Mean Usual Hours␣
↪Worked of Women Age 16 to 64'],
            ['Unemployment rate %', 'Percent Not in Labor Force %', 'Mean Usual␣
↪Hours Worked of Population Age 16 to 64',
             'Mean Usual Hours Worked of Men Age 16 to 64', 'Mean Usual Hours␣
↪Worked of Women Age 16 to 64'],
             'Greys_03')
demographic_map = add_choropleth_layer(demographic_map, dc18, 'CRIME_RATE',␣
↪'Crime Rate %',
            ['CRIME_RATE', 'CRIME_CT', 'Total Population'],
            ['Crime Rate %', 'Total Crime', 'Total Population'],
             'YlOrRd_04', num_classes = 30)

demographic_map = add_choropleth_layer(demographic_map, dc18, 'CRIME_CT',␣
↪'Total Crime',
            ['CRIME_RATE', 'CRIME_CT', 'Total Population'],
            ['Crime Rate %', 'Total Crime', 'Total Population'],
             'YlOrRd_04')

folium.LayerControl(collapsed = False).add_to(demographic_map)
demographic_map
```

[29]:
```
<folium.folium.Map at 0x1a48215828>
```

[30]:
```
demographic_map.save('DC_demographics18.html')
```

1.0.7 Load recreation facilities

Data Used

+ **Recreation Facilities**: Data was retreived from Metropolitan Police Department from Aug.31st, 2020 + url link: https://opendata.dc.gov/datasets/7122c1c815314588abe5c1864da8a355_3

[31]:
```
# Load recreation facilities
recre_faci = pd.read_csv('./Data/Recreation_Facilities.csv')
#recre_faci.head()
```

[32]:
```
# Set customized marker icon for different facility types
def icon(value_field):
    if value_field == 'RECREATION CENTER':
        icon = ["orange", 'dribbble']
    elif value_field == 'SPRAY PARK':
        icon = ["red", 'universal-access']
```

```
    elif value_field == 'POOL':
        icon = ["blue", 'tint']
    elif value_field == 'AQUATIC CENTER':
        icon = ["darkblue", 'anchor']
    elif value_field == 'OFFICE':
        icon = ["gray", 'building']
    else:
        icon = ["green", 'child']
    return icon
```

[33]:
```
# Get facilities data index
def demographics_marker_idx(df):
    # Get column indices
    name_idx = df.columns.get_loc('NAME')
    add_idx = df.columns.get_loc('ADDRESS')
    type_idx = df.columns.get_loc('USE_TYPE')
    pool_idx = df.columns.get_loc('POOL')
    web_idx = df.columns.get_loc('WEB_URL')
    pho_idx = df.columns.get_loc('PHONE')
    fit_idx = df.columns.get_loc('FITNESS_CENTER')

    return name_idx, add_idx, type_idx, pool_idx, web_idx, pho_idx, fit_idx
```

[34]:
```
def add_demographic_markers(df, mapobj, layer_name, show = False):
    # Get columns of interest idx
    name_idx, add_idx, type_idx, pool_idx, web_idx, pho_idx, fit_idx =␣
 ↪demographics_marker_idx(recre_faci)

    #Create a Folium feature group for this layer, since we will be displaying␣
 ↪multiple layers
    recre_faci_lyr = folium.FeatureGroup(name = layer_name,
                                         show = show)

    # Set popup message format
    for i in range(0, len(df)):
        lat = df.iloc[i]['Y']
        long = df.iloc[i]['X']
        popup_text = """ <b>Name</b>: {}<br>
                         <b>Address</b>: {}<br>
                         <b>Type</b>: {}<br>
                         <b>Pool</b>: {}<br>
                         <b>Website</b>: {}<br>
                         <b>Phone</b>: {}<br>
                         <b>Fitness Center</b>: {}<br>"""
        popup_text = popup_text.format(df.iloc[i][name_idx],
                                 df.iloc[i][add_idx],
                                 df.iloc[i][type_idx].title(),
```

```
                                df.iloc[i][pool_idx],
                                df.iloc[i][web_idx],
                                df.iloc[i][pho_idx],
                                df.iloc[i][fit_idx])

        tooltip_text = """ <b>{}</b><br>"""
        tooltip_text = tooltip_text.format(df.iloc[i][name_idx])


        folium.Marker([lat, long],
                      name = layer_name,
                      icon = folium.Icon(color = icon(df.
↪iloc[i]['USE_TYPE'])[0],
                                         icon = icon(df.
↪iloc[i]['USE_TYPE'])[1], prefix='fa'),
                      tooltip = tooltip_text,
                      control = True,
                      popup = popup_text
                      ).add_to(recre_faci_lyr)

    mapobj.add_child(recre_faci_lyr)

    return mapobj
```

[35]:
```
# Recreational facilities legend template
def facility_legend_template():
    template = """
        {% macro html(this, kwargs) %}

        <!doctype html>
        <html lang="en">
        <head>
          <meta charset="utf-8">
          <meta name="viewport" content="width=device-width, initial-scale=1">
          <title>jQuery UI Draggable - Default functionality</title>
          <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/
↪jquery-ui.css">
          <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
↪font-awesome/4.7.0/css/font-awesome.min.css">

          <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
          <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

          <script>
          $( function() {
            $( "#maplegend" ).draggable({
                            start: function (event, ui) {
```

```
                                $(this).css({
                                    right: "auto",
                                    top: "auto",
                                    bottom: "auto"
                                });
                            }
                        });
        });

          </script>
        </head>
        <body>


        <div id='maplegend' class='maplegend'
            style='position: absolute; z-index:9999; border:2px solid grey;␣
↪background-color:rgba(255, 255, 255, 0.8);
             border-radius:6px; padding: 10px; font-size:14px; left: 20px;␣
↪bottom: 20px;'>

        <div class='legend-title'>Type of Recreational Facilities</div>
        <div class='legend-scale'>
          <ul class='legend-labels'>
            <li><i class="fa fa-dribbble fa-lg" style="color: orange" ></
↪i>  Recreation Center</li>
            <li><i class="fa fa-universal-access fa-lg" style="color: red" ></
↪i>  Spray Park</li>
            <li> <i class="fa fa-tint fa-lg" style="color: cadetblue" ></
↪i>   Pool</li>
            <li><i class="fa fa-anchor fa-lg" style="color: darkblue" ></
↪i>  Aquatic Center</li>
            <li><i class="fa fa-building fa-lg" style="color: gray" ></i> 
↪ Office</li>
            <li><i class="fa fa-child fa-lg" style="color: green" ></i> 
↪ Helping Resources</li>

          </ul>
        </div>
        </div>

        </body>
        </html>

        <style type='text/css'>
          .maplegend .legend-title {
            text-align: left;
```

```
            margin-bottom: 5px;
            font-weight: bold;
            font-size: 120%;
            }
          .maplegend .legend-scale ul {
            margin: 0;
            margin-bottom: 3px;
            padding: 0;
            float: left;
            list-style: none;
            }
          .maplegend .legend-scale ul li {
            font-size: 90%;
            list-style: none;
            margin-left: 2px;
            margin-right: 2px;
            line-height: 23px;
            margin-bottom: 3px;
            }
          .maplegend ul.legend-labels li span {
            display: block;
            float: left;
            height: 18px;
            width: 30px;
            margin-right: 5px;
            margin-left: 0;
            border: 1px solid #999;
            }
          .maplegend .legend-source {
            font-size: 80%;
            color: #777;
            clear: both;
            }
          .maplegend a {
            color: #777;
            }
        </style>
        {% endmacro %}"""
    return template
```

1.0.8 Generate Demographics with Recreational Facilities Map

[36]:
```
mymap = default_map()

mymap = add_choropleth_layer(mymap, dc18, 'Total Population', 'Total␣
↪Population',
```

```
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
 ↪Population Less Than 18 Years', 'Percent of Population 65 Years and Over',␣
 ↪'CRIME_RATE'],
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
 ↪Population Less Than 18 Years %', 'Percent of Population 65 Years and Over␣
 ↪%', 'Crime Rate %'],
             'YlGnBu_09', show = True)
mymap = add_choropleth_layer(mymap, dc18, 'CRIME_RATE', 'Crime Rate %',
            ['CRIME_RATE', 'CRIME_CT', 'Total Population'],
            ['Crime Rate %', 'Total Crime', 'Total Population'],
             'YlOrRd_04', num_classes = 30)

mymap = add_demographic_markers(recre_faci, mymap, 'All Recreational␣
 ↪Facilities', show = True)
mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] ==␣
 ↪'RECREATION CENTER'], mymap, 'Recreation Center')
mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] == 'SPRAY␣
 ↪PARK'], mymap, 'Spray Park')
mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] == 'POOL'],␣
 ↪mymap, 'Pool')
mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] == 'AQUATIC␣
 ↪CENTER'], mymap, 'Aquatic Center')
mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] == 'OFFICE'],␣
 ↪mymap, 'Office')

resource = recre_faci[(recre_faci['USE_TYPE'] != 'RECREATION CENTER') &␣
 ↪(recre_faci['USE_TYPE'] != 'SPRAY PARK') & (recre_faci['USE_TYPE'] !=␣
 ↪'POOL') & (recre_faci['USE_TYPE'] != 'AQUATIC CENTER') &␣
 ↪(recre_faci['USE_TYPE'] != 'OFFICE')]
mymap = add_demographic_markers(resource, mymap, 'Helping Resources')

facility_template = facility_legend_template()

macro = MacroElement()
macro._template = Template(facility_template)
mymap.get_root().add_child(macro)



folium.LayerControl(collapsed = False).add_to(mymap)
plugins.LocateControl().add_to(mymap)

mymap
```

[36]: <folium.folium.Map at 0x1a47c5b0b8>

[ ]:

[37]:
```
mymap.save('DC_Demographics_Recreation18.html')
```

[ ]:

1.0.9 Generate Crime Clusters with Choropleth

[38]:
```
# Create add crime cluster function, need crime incidents data
def add_crime_clusters(df, mapobj, layer_name, show = False):

    off_txt_idx, lat_idx, long_idx, color_idx, method_idx, block_gp_idx,␣
↪shift_idx, str_date_idx = crime_color_idx(df)


    #Create a Folium feature group for this layer, since we will be displaying␣
↪multiple layers
    crime_lyr = folium.FeatureGroup(name = layer_name, show = show)

    marker_cluster = MarkerCluster().add_to(crime_lyr)


    # Set map popup info box layout
    for i in range(len(df['LATITUDE'])):
        lat = df.iloc[i][lat_idx]
        long = df.iloc[i][long_idx]
        color = df.iloc[i][color_idx]
        popup_text = """ <b>Offense</b>: {}<br>
                         <b>Methd</b>: {}<br>
                         <b>Block-Group</b>: {}<br>
                         <b>Shift</b>: {}<br>
                         <b>Start Date</b>: {}<br>"""

        popup_text = popup_text.format(df.iloc[i][off_txt_idx],
                                       df.iloc[i][method_idx],
                                       df.iloc[i][block_gp_idx],
                                       df.iloc[i][shift_idx],
                                       df.iloc[i][str_date_idx]
                                      )
        folium.Marker(location = [lat, long], popup = popup_text,
                      icon = folium.Icon(color = color, icon =␣
↪'exclamation-triangle', prefix='fa')).add_to(marker_cluster)


    mapobj.add_child(crime_lyr)
```

```
    return mapobj
```

[39]:
```
def crime_legend_template():
    # Add the legend to the map
    template = """
    {% macro html(this, kwargs) %}

    <!doctype html>
    <html lang="en">
    <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <title>jQuery UI Draggable - Default functionality</title>
      <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/
↪jquery-ui.css">
      <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
↪font-awesome/4.7.0/css/font-awesome.min.css">

      <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
      <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

      <script>
      $( function() {
        $( "#maplegend" ).draggable({
                        start: function (event, ui) {
                            $(this).css({
                                right: "auto",
                                top: "auto",
                                bottom: "auto"
                            });
                        }
                    });
    });

      </script>
    </head>
    <body>


    <div id='maplegend' class='maplegend'
        style='position: absolute; z-index:9999; border:2px solid grey;␣
↪background-color:rgba(255, 255, 255, 0.8);
         border-radius:6px; padding: 10px; font-size:14px; left: 20px; bottom:␣
↪20px;'>
```

```
   <div class='legend-title'>Offenses</div>
   <div class='legend-scale'>
     <ul class='legend-labels'>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: darkred"␣
↪></i>  THEFT/OTHER</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: orange"␣
↪></i>  THEFT F/AUTO</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color:␣
↪lightblue" ></i>  MOTOR VEHICLE THEFT</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: gray" ></
↪i>  ROBBERY</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color:␣
↪cadetblue" ></i>  ASSAULT W/DANGEROUS WEAPON</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: blue" ></
↪i>  BURGLARY</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: green" ></
↪i>  SEX ABUSE</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: purple"␣
↪></i>  HOMICIDE</li>
       <li><i class="fa fa-exclamation-triangle fa-lg" style="color: black" ></
↪i>  OTHERS</li>

     </ul>
   </div>
   </div>

   </body>
   </html>

   <style type='text/css'>
     .maplegend .legend-title {
       text-align: left;
       margin-bottom: 5px;
       font-weight: bold;
       font-size: 120%;
       }
     .maplegend .legend-scale ul {
       margin: 0;
       margin-bottom: 3px;
       padding: 0;
       float: left;
       list-style: none;
       }
     .maplegend .legend-scale ul li {
       font-size: 90%;
       list-style: none;
```

```
        margin-left: 2px;
        margin-right: 2px;
        line-height: 23px;
        margin-bottom: 3px;
        }
      .maplegend ul.legend-labels li span {
        display: block;
        float: left;
        height: 18px;
        width: 30px;
        margin-right: 5px;
        margin-left: 0;
        border: 1px solid #999;
        }
      .maplegend .legend-source {
        font-size: 80%;
        color: #777;
        clear: both;
        }
      .maplegend a {
        color: #777;
        }
    </style>
    {% endmacro %}"""

    return template
```

[40]:
```
crime_clusters = default_map()

crime_clusters = add_choropleth_layer(crime_clusters, dc18, 'Total Population',␣
↪'Total Population',
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
↪Population Less Than 18 Years', 'Percent of Population 65 Years and Over',␣
↪'CRIME_RATE'],
            ['Total Population', 'Ratio of Males to Females', 'Percent of␣
↪Population Less Than 18 Years %', 'Percent of Population 65 Years and Over␣
↪%', 'Crime Rate %'],
             'YlGnBu_09', show = True)
crime_clusters = add_choropleth_layer(crime_clusters, dc18, 'CRIME_RATE',␣
↪'Crime Rate %',
            ['CRIME_RATE', 'CRIME_CT', 'Total Population'],
            ['Crime Rate %', 'Total Crime', 'Total Population'],
             'YlOrRd_04', num_classes = 30)

crime_clusters = add_crime_clusters(dc_crime18, crime_clusters, 'Total Crimes',␣
↪show = True)
```

```
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'theft/other'], crime_clusters, 'Theft/Other')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'theft f/auto'], crime_clusters, 'Theft f/auto')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'motor vehicle theft'], crime_clusters, 'Motor Vehicle Theft')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'robbery'], crime_clusters, 'Robbery')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'assault w/dangerous weapon'], crime_clusters, 'Assault w/Dangerous Weapon')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'burglary'], crime_clusters, 'Burglary')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'sex abuse'], crime_clusters, 'Sex Abuse')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'homicide'], crime_clusters, 'Homicide')
crime_clusters = add_crime_clusters(dc_crime18[dc_crime18['offense-text'] ==␣
↪'arson'], crime_clusters, 'Other')

#mymap = add_demographic_markers(recre_faci[recre_faci['USE_TYPE'] == 'POOL'],␣
↪mymap, 'Pool')

crime_template = crime_legend_template()

macro = MacroElement()
macro._template = Template(crime_template)
crime_clusters.get_root().add_child(macro)



folium.LayerControl(collapsed = False).add_to(crime_clusters)
plugins.LocateControl().add_to(crime_clusters)

crime_clusters
```

[40]:
```
<folium.folium.Map at 0x1a4714c860>
```

[41]:
```
crime_clusters.save('Crime_Clusters_w_Choropleth18.html')
```

[ ]:
```

```

1.0.10 DualMap

[42]:
```
m = plugins.DualMap(location=(38.9072, -77.0369), tiles=None, zoom_start=8)
'''
folium.TileLayer('cartodbpositron').add_to(m.m2)
folium.TileLayer('openstreetmap').add_to(m)

fg_both = folium.FeatureGroup(name='markers_both').add_to(m)
fg_1 = folium.FeatureGroup(name='markers_1').add_to(m.m1)
fg_2 = folium.FeatureGroup(name='markers_2').add_to(m.m2)

icon_red = folium.Icon(color='red')
folium.Marker((52, 5), tooltip='both', icon=icon_red).add_to(fg_both)
'''
mymap.add_to(m.m1)
crime_clusters.add_to(m.m2)

folium.LayerControl(collapsed = False).add_to(m)
m.save("Dual_Map18.html")
```

[ ]:
```

```