Assignment 2

Before working on this assignment please read these instructions fully. In the submission area, you will notice that you can click the link to **Preview the Grading** for each step of the assignment. This is the criteria that will be used for peer grading. Please familiarize yourself with the criteria before beginning the assignment.

An NOAA dataset has been stored in the file `data/C2A2_data/BinnedCsvs_d400/fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv`. This is the dataset to use for this assignment. Note: The data for this assignment comes from a subset of The National Centers for Environmental Information (NCEI) Daily Global Historical Climatology Network (https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt) (GHCN-Daily). The GHCN-Daily is comprised of daily climate records from thousands of land surface stations across the globe.

Each row in the assignment datafile corresponds to a single observation.

The following variables are provided to you:

- **id** : station identification code
- **date** : date in YYYY-MM-DD format (e.g. 2012-01-24 = January 24, 2012)
- **element** : indicator of element type
  - TMAX : Maximum temperature (tenths of degrees C)
  - TMIN : Minimum temperature (tenths of degrees C)
- **value** : data value for element (tenths of degrees C)

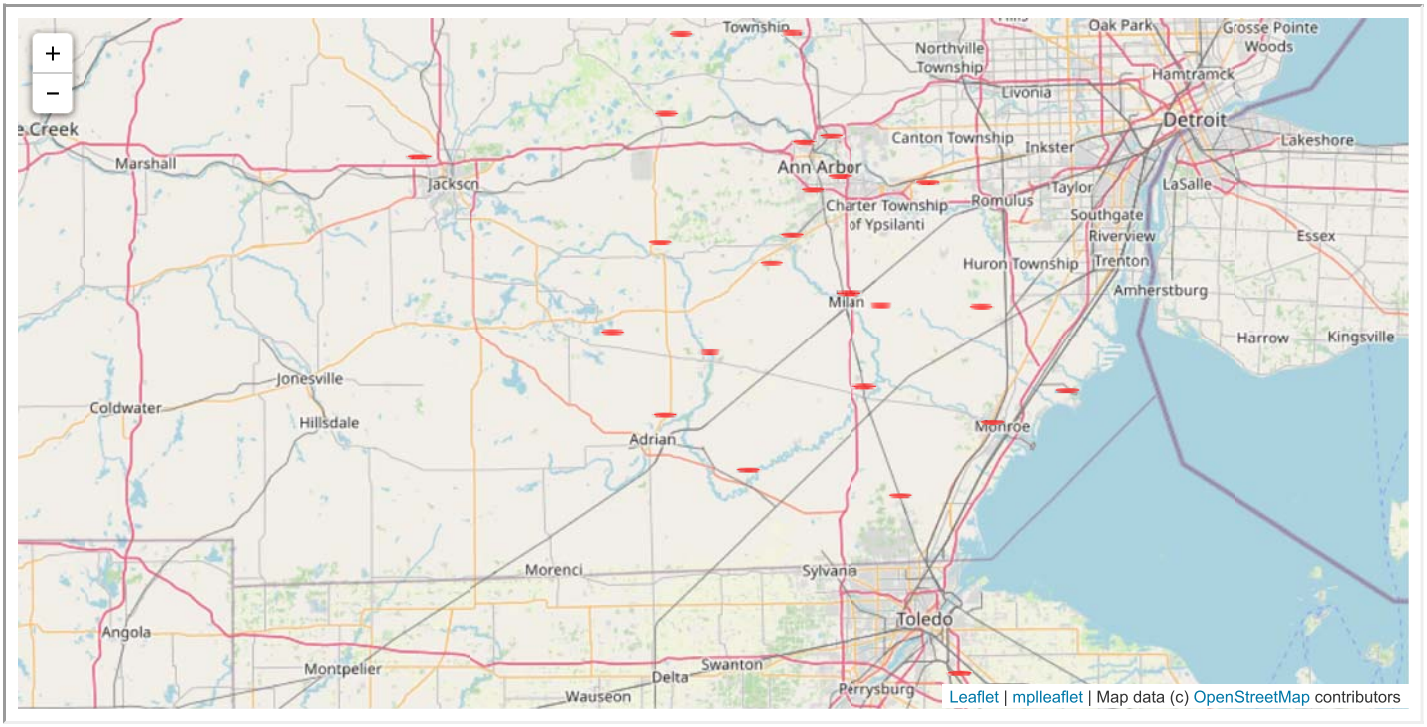
For this assignment, you must:

1. Read the documentation and familiarize yourself with the dataset, then write some python code which returns a line graph of the record high and record low temperatures by day of the year over the period 2005-2014. The area between the record high and record low temperatures for each day should be shaded.
2. Overlay a scatter of the 2015 data for any points (highs and lows) for which the ten year record (2005-2014) record high or record low was broken in 2015.
3. Watch out for leap days (i.e. February 29th), it is reasonable to remove these points from the dataset for the purpose of this visualization.
4. Make the visual nice! Leverage principles from the first module in this course when developing your solution. Consider issues such as legends, labels, and chart junk.

The data you have been given is near **Ann Arbor, Michigan, United States**, and the stations the data comes from are shown on the map below.

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

def leaflet_plot_stations(binsize, hashid):

    df = pd.read_csv('data/C2A2_data/BinSize_d{}.csv'.format(binsize))

    station_locations_by_hash = df[df['hash'] == hashid]

    lons = station_locations_by_hash['LONGITUDE'].tolist()
    lats = station_locations_by_hash['LATITUDE'].tolist()

    plt.figure(figsize=(8,8))

    plt.scatter(lons, lats, c='r', alpha=0.7, s=200)

    return mplleaflet.display()

leaflet_plot_stations(400,'fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89')
```

Out[1]:



Import Libraries

In [2]:
```
'''
@author:  Steven Ponce
Date:     01 May 2021
'''
# Importing important libraries
import sys
import pandas as pd
import numpy as np
from datetime import datetime as dt

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.dates as dates
from matplotlib.font_manager import FontProperties

# Hide warnings
import warnings
warnings.filterwarnings('ignore')

print('You\'re running python %s' % sys.version.split(' ')[0])
print('You\'re running matplotlib: {}'.format(matplotlib.__version__))

# The following lines adjust the granularity of reporting
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.1f}'.format
```

```
You're running python 3.6.2
You're running matplotlib: 2.0.0
```

Loading the data

In [3]:
```
# Loading the data
df = pd.read_csv('./data/C2A2_data/BinnedCsvs_d400/fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv',
                 index_col='ID',
                 squeeze=True)
df.head()
```

Out[3]:

	Date	Element	Data_Value
ID			
USW00094889	2014-11-12	TMAX	22
USC00208972	2009-04-29	TMIN	56
USC00200032	2008-05-26	TMAX	278
USC00205563	2005-11-11	TMAX	139
USC00200230	2014-02-27	TMAX	-106

Examining and cleaning the data

In [4]:
```
'''
@author:  Steven Ponce
Date:     01 May 2021
'''
def quick_analysis(df):

    print('\n 1. Dataset Information:')
    print('-'*40)
    print(df.info())

    print('\n 2. Number of Rows and Columns:', df.shape)
    print('-'*40)

    print('\n 3. Null Values:')
    print('-'*40)
    print(df.apply(lambda x: sum(x.isnull()) / len(df)))

    return quick_analysis
    raise NotImplementedError()
```

In [5]:
```
quick_analysis(df);
```

```
 1. Dataset Information:
---------------------------------------
<class 'pandas.core.frame.DataFrame'>
Index: 165085 entries, USW00094889 to USC00207312
Data columns (total 3 columns):
Date          165085 non-null object
Element       165085 non-null object
Data_Value    165085 non-null int64
dtypes: int64(1), object(2)
memory usage: 5.0+ MB
None

 2. Number of Rows and Columns: (165085, 3)
---------------------------------------

 3. Null Values:
---------------------------------------
Date         0.0
Element      0.0
Data_Value   0.0
dtype: float64
```

In [6]:
```
# Convert temperature from tenths ºC to ºC
df['Data_Value'] = df['Data_Value'] * 0.1

# Change date to datetime formats and add a few columns
df['Date'] = pd.to_datetime(df['Date'])

df['Year'] = df['Date'].dt.year

df['Month-Day'] = df['Date'].dt.strftime('%m-%d')
df['Month-Year'] = df['Date'].dt.strftime('%Y-%m')

# Removing leap days (i.e. February 29th)
df = df[df['Month-Day'] != '02-29']

# Lets sort by Date
df = df.sort_values(by='Date', ascending=True)
df.head()
```

Out[6]:

	Date	Element	Data_Value	Year	Month-Day	Month-Year
ID						
USW00014853	2005-01-01	TMAX	5.6	2005	01-01	2005-01
USC00201502	2005-01-01	TMIN	-3.9	2005	01-01	2005-01
USC00200228	2005-01-01	TMAX	15.0	2005	01-01	2005-01
USC00207320	2005-01-01	TMAX	15.0	2005	01-01	2005-01
USC00200228	2005-01-01	TMIN	-3.9	2005	01-01	2005-01

In [7]:
```
# Getting min & max temp for 2005 - 2014
min_temp = df[(df.Year != 2015) & (df['Element'] == 'TMIN')].groupby(['Month-Day'])['Data_Value'].min()
max_temp = df[(df.Year != 2015) & (df['Element'] == 'TMAX')].groupby(['Month-Day'])['Data_Value'].max()

# Getting min & max temp for 2015
min_temp_2015 = df[(df.Year == 2015) & (df['Element'] == 'TMIN')].groupby(['Month-Day'])['Data_Value'].min()
max_temp_2015 = df[(df.Year == 2015) & (df['Element'] == 'TMAX')].groupby(['Month-Day'])['Data_Value'].max()

df = df.merge(min_temp.reset_index(drop=False).rename(columns={'Data_Value':'Min_Temp'}), on='Month-Day', how='left')
df = df.merge(max_temp.reset_index(drop=False).rename(columns={'Data_Value':'Max_Temp'}), on='Month-Day', how='left')

df.head()
```

Out[7]:

	Date	Element	Data_Value	Year	Month-Day	Month-Year	Min_Temp	Max_Temp
0	2005-01-01	TMAX	5.6	2005	01-01	2005-01	-16.0	15.6
1	2005-01-01	TMIN	-3.9	2005	01-01	2005-01	-16.0	15.6
2	2005-01-01	TMAX	15.0	2005	01-01	2005-01	-16.0	15.6
3	2005-01-01	TMAX	15.0	2005	01-01	2005-01	-16.0	15.6
4	2005-01-01	TMIN	-3.9	2005	01-01	2005-01	-16.0	15.6

In [8]:
```
# Record lox a& high temps for 2015
record_low = df[(df.Year==2015) & (df.Data_Value < df.Min_Temp)]
record_high = df[(df.Year==2015) & (df.Data_Value > df.Max_Temp)]
```

In [9]:
```
'''
@author:  Steven Ponce
Date:     03 May 2021
'''
def create_fig(df):

    # Figure size
    plt.figure(figsize=(15,10))
    plt.subplots_adjust(left=0.5, bottom=0.5, right=1, top=1, wspace=0, hspace=0)

    # x-axis (year 2015)
    date_index = np.arange('2015-01-01','2016-01-01', dtype='datetime64[D]')

    # Upper and lower plot lines (max & min temp for 2005 - 2014)
    plt.plot(date_index,min_temp,color='gray', linestyle='-',linewidth=1, alpha=0.8)
    plt.plot(date_index,max_temp,color='gray', linestyle='-',linewidth=1, alpha=0.8)

    # Scatter plots for record low and high temps for 2015
    plt.scatter(record_high.Date.values, record_high.Data_Value.values, s=25, color='red', marker='o', alpha=0.5)
    plt.scatter(record_low.Date.values, record_low.Data_Value.values, s=25, color='blue', marker='o', alpha=0.5)

    # x, y axis
    ax = plt.gca()
    ax.axis(['2015/01/01','2015/12/31',-40,50])

    # Fill colour between highs and lows:
    ax.fill_between(date_index, max_temp, min_temp, facecolor='gray', alpha=0.35)

    # Remove top and right spines
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    # format spine color
    ax.spines['left'].set_color('dimgray')
    ax.spines['bottom'].set_color('dimgray')

    # Set axis labels and title:
    plt.xlabel('Months', fontsize=14, color='dimgray')
    plt.ylabel('Temp (°C)', fontsize=14, color='dimgray')
    plt.title('TEMPERATURE IN ANN ARBOR, MI (2005-2015)\n', fontsize=18, loc='left', alpha=0.65, fontweight ="bold")

    # x ticks
    xticks = (pd.date_range('2015-01-01','2016-01-01', freq = 'M') - 1 + pd.Timedelta('1D')).strftime('%-j').astype(int)
    xticks_labels = pd.to_datetime(xticks, format = '%j').strftime('%b')
    ax.set_xticklabels(xticks_labels)

    # Format axis
    plt.tick_params(bottom='off', left='off', labelsize=14, labelcolor='dimgray')

    # text strings for direct labels
    textstr1 = 'Max Temp (2005-2014)'
    textstr2 = 'Min Temp (2005-2014)'

    # Custom Legend
    string1 = "Record Low 2015"
    ax.text(0.16, 0.05, string1, transform=ax.transAxes,fontsize=12, color="blue")
    string2 = "Record High 2015"
    ax.text(0.57, 0.87, string2, transform=ax.transAxes,fontsize=12, color="red")

    string3 = "Min Temp (2005-2014)"
    ax.text(1.01, 0.3, string3, transform=ax.transAxes,fontsize=12, color="dimgray")
    string4 = "Max Temp (2005-2014)"
    ax.text(1.01, 0.62, string4, transform=ax.transAxes,fontsize=12, color="dimgray")

    plt.show();

    return create_fig
```

In [10]:
```
create_fig(df);
```