# Assignment2

December 20, 2017

## 1 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/aefc0d2e401a89908` The data for this assignment comes from a subset of The National Centers for Environmental Information (NCEI) Daily Global Historical Climatology Network (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 **Santa Cruz, California, 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,'aefc0d2e401a89908467da05ac7e23d5e317bd07f1fc0b3c
```

Out[1]:
```
<IPython.core.display.HTML object>
```

In [2]:
```
## Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

In [3]:
```
%matplotlib notebook
```

In [4]:
```
import matplotlib as mpl
mpl.get_backend()
```

Out[4]:
```
'nbAgg'
```

In [5]:
```
## Load the data
df = pd.read_csv('data/C2A2_data/BinnedCsvs_d400/aefc0d2e401a89908467da05ac
                 parse_dates = [1], infer_datetime_format = True) \
    .sort('Date')
```

```
/opt/conda/lib/python3.5/site-packages/ipykernel/__main__.py:4: FutureWarning: sort
```

In [6]:
```
## Add year, month and day / Drop 2/29
df['Year'] = df['Date'].map(lambda d: d.year)
df['Month'] = df['Date'].map(lambda d: d.month)
df['Day'] = df['Date'].map(lambda d: d.day)
df = df[(df['Month'] != 2) | (df['Day'] != 29)]
```

In [34]:
```
## Min temp
df_min = df[df['Element'] == 'TMIN']
t_min = df[df['Year'] < 2015].groupby(['Month', 'Day']) \
    .agg({'Data_Value': 'min'}) \
    .reset_index()
t_min['Date'] = t_min.apply(lambda r: pd.to_datetime(str(r['Month']) + '/'
                                                     format = '%m/%d'),
                            axis = 1)
t_min['Data_Value'] = t_min['Data_Value'] / 10
t_min = t_min[['Date', 'Data_Value']]
t_min.columns = ['Date', 'Temp']

## Max temp
df_max = df[df['Element'] == 'TMAX']
t_max = df[df['Year'] < 2015].groupby(['Month', 'Day']) \
    .agg({'Data_Value': 'max'}) \
    .reset_index()
t_max['Date'] = t_max.apply(lambda r: pd.to_datetime(str(r['Month']) + '/'
                                                     format = '%m/%d'),
                            axis = 1)
t_max['Data_Value'] = t_max['Data_Value'] / 10
t_max = t_max[['Date', 'Data_Value']]
t_max.columns = ['Date', 'Temp']

## Combine
t_all = t_min.copy()
t_all.columns = ['Date', 'Min']
t_all['Max'] = t_max['Temp']

## Records broken in 2015
# Min
day_min = df_min.loc[df_min.groupby(['Month', 'Day'])['Data_Value'].idxmin
day_min = day_min[day_min['Year'] == 2015]
day_min['Date'] = day_min.apply(lambda r: pd.to_datetime(str(r['Month']) +
                                                         format = '%m/%d')
                                axis = 1)
day_min = day_min[['Date', 'Data_Value']]
day_min['Data_Value'] = day_min['Data_Value'] / 10
day_min.columns = ['Date', 'Record Low Broken in 2015']
# Max
day_max = df_max.loc[df_max.groupby(['Month', 'Day'])['Data_Value'].idxmax
day_max = day_max[day_max['Year'] == 2015]
day_max['Date'] = day_max.apply(lambda r: pd.to_datetime(str(r['Month']) +
                                                         format = '%m/%d')
                                axis = 1)
day_max = day_max[['Date', 'Data_Value']]
day_max['Data_Value'] = day_max['Data_Value'] / 10
day_max.columns = ['Date', 'Record High Broken in 2015']
```

In [40]:
```
## Plot the temps
# Dates and Dates to show
t_all['Show'] = t_all['Date'].map(lambda d: True if d.day == 1 else False)
dates = list(t_all.Date)
dates_show = list(t_all[t_all['Show']].Date)
# Line
fig = plt.figure()
ax = fig.add_subplot(111)
t_max.columns = ['Date', '2005-2014 Record High']
plt.plot(dates, t_max['2005-2014 Record High'], '-', c = 'red', alpha = 0.
t_min.columns = ['Date', '2005-2014 Record Low']
plt.plot(dates, t_min['2005-2014 Record Low'], '-', c = 'blue', alpha = 0.
n_bins = 1000
b = np.array(mpl.colors.to_rgb('blue'))
r = np.array(mpl.colors.to_rgb('red'))
temp_max = max(t_all['Max'])
temp_min = min(t_all['Min'])
d = (temp_max - temp_min) / n_bins
for i in range(n_bins):
    col = r * (n_bins - i) / n_bins + b * i / n_bins
    ax.fill_between(dates,
                    t_all.apply(lambda t: min(max(temp_max - d * (i + 1),
                    t_all.apply(lambda t: min(max(temp_max - d * i, t['Mir
                    facecolor = col,
                    alpha = 0.25)

# Rotate the tick labels for the x axis
d_fmt = mdates.DateFormatter('%b %d')
x = ax.xaxis
x.set_major_formatter(d_fmt)
x.set_ticks(dates_show)
for item in x.get_ticklabels():
    item.set_rotation(45)

# Scatter
plt.scatter(list(day_min['Date']), day_min['Record Low Broken in 2015'],
            s = 50, c = 'darkblue', marker = 'x')
plt.scatter(list(day_max['Date']), day_max['Record High Broken in 2015'],
            s = 50, c = 'darkred', marker = 'x')

# Set the lims
plt.ylim(ymax = 65)

# Add a label to the y axis
plt.ylabel('Temperature, C')

# Add a title
plt.title('Record High and Low Temperatures in 2005-2014, Santa Cruz, CA')
```

```
        # Legend
        plt.legend(loc = 2, frameon = False)

        # Remove part of the frame
        ax.spines['right'].set_visible(False)
        ax.spines['top'].set_visible(False)

        # Save
        plt.savefig('ha2_sc.pdf')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```