Exploring Weather Trends with SQL and Python

Submitted by: John Meehan

Date: 4/21/2019

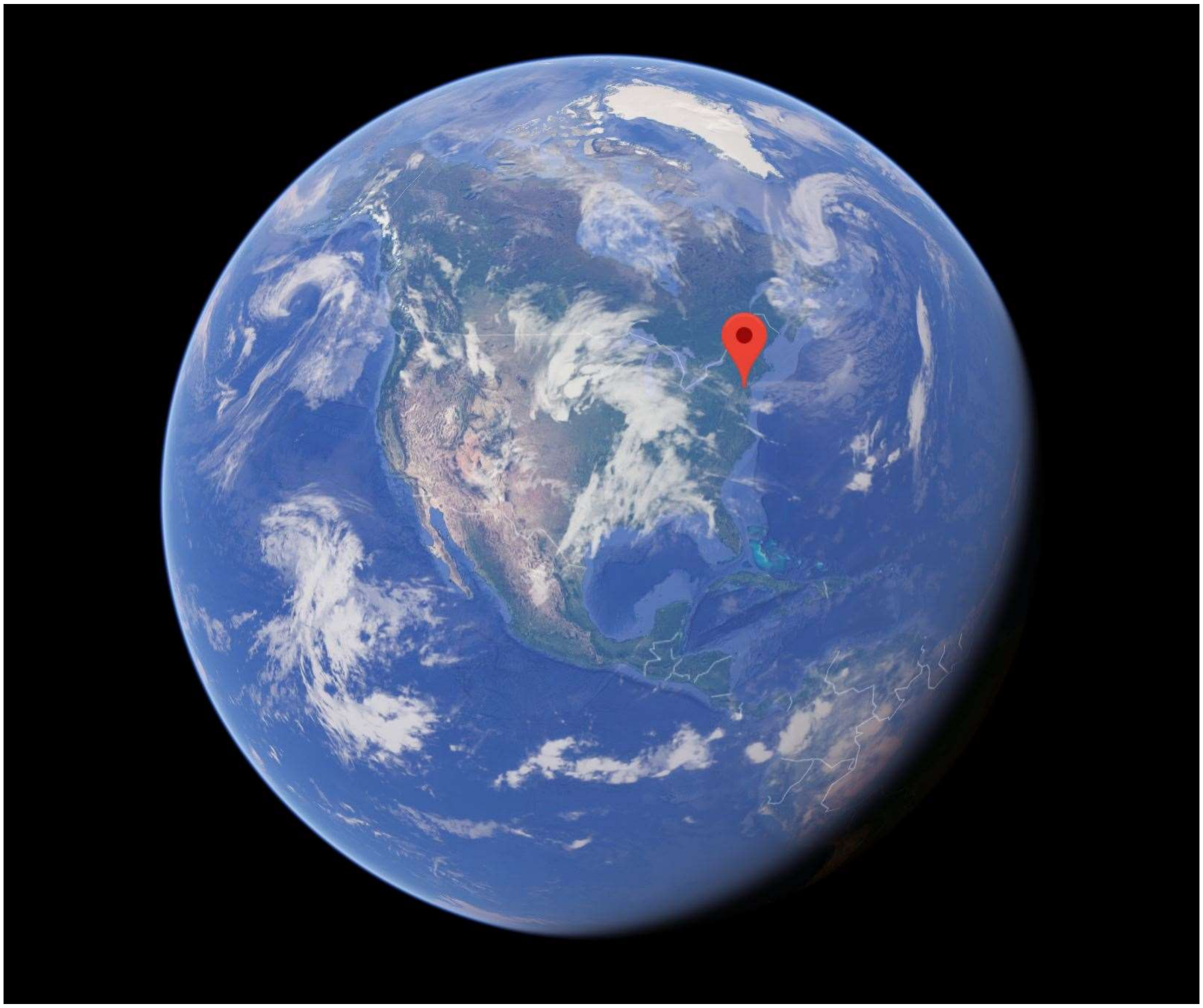
Summary ¶

In this project, I analyze local and global temperature data and compare the temperature trends in Philadelphia to overall global temperature trends.

1. Extract the Data from the Database

City chosen: Philadelphia, PA, USA

In [1]:
```
from IPython.display import display, Image
display(Image('philadelphia_globe.jpg', width=750))
```



Preview

```
SELECT * FROM city_data WHERE city = 'Philadelphia'
LIMIT 10
```

year	city	country	avg_temp
1743	Philadelphia	United States	6.07
1744	Philadelphia	United States	13.74
1745	Philadelphia	United States	3.96
1746	Philadelphia	United States	
1747	Philadelphia	United States	
1748	Philadelphia	United States	
1749	Philadelphia	United States	
1750	Philadelphia	United States	12.36
1751	Philadelphia	United States	13.05
1752	Philadelphia	United States	5.64

Query

```
SELECT * FROM city_data WHERE city = 'Philadelphia'
```

Global data

Preview

```
SELECT * FROM global_data
LIMIT 10
```

year	avg_temp
1750	8.72
1751	7.98
1752	5.78
1753	8.39
1754	8.47
1755	8.36
1756	8.85
1757	9.02
1758	6.74
1759	7.99

Query

```
SELECT * FROM global_data
```

2. Data Manipulation Using Python

Below I have written a script to manipulate and plot the imported query data.

First, I define a function `fit_func` , which will be used later to define a best-fit curve to the rolling-average temperature data. I chose a sigmoid function (specifically the logistic function) to make a best-fit curve, under the assumption that long-term temperature change is driven by a consistent input (man-made greenhouse gas emmissions), which will achieve a new steady-state value after some period of adjustment.

The logistic function is defined as:

```
In [2]: from IPython.display import Math
        Math(r'f(x) = {L \over {1+e^{-k(x-x_0)}}}')
```

Out[2]:
$$f(x) = \frac{L}{1+e^{-k(x-x_0)}}$$

However, for curve-fitting purposes, I define a set of constants, `[a, b, c, d]` , whose values will define the closest-approximation fit of the logistic function to the temperature data.

- `a` is added as a vertical offset
- `b` replaces `L` above as the 'maximum value' of the function
- `c` replaces `k` above as the 'growth factor' of the function
- `d` replaced `x0` above as the 'initial value' or 'mid-point' of the function

The logistic function is now defined in `fit_func` as:

```
In [3]: Math(r'f(x) = {a + {b \over {1+e^{-c(x-d)}}}}')
```

Out[3]:
$$f(x) = a + \frac{b}{1+e^{-c(x-d)}}$$

Then I define `rolling_window` , a variable which will allow me to easily play with the window on which my rolling average is based on.

The data is imported using Pandas; I directly import the local and global data from a CSV. Then I add a column to my dataframe in which I convert the yearly temperature data into a rolling average, using the `pd.rolling().mean()` function.

Because the rolling window will generate `NA` values for the first `rolling_window` number of rows, I then drop the rows of the dataframe where any of the columns are missing values. This is important to avoid incorporating false information into the temperature averages or curve fits later on.

Then I convert the `year` and `rolling_avg` dataframes to Numpy arrays and define them as the input and output values for both the local `[x1, y1]` and global `[x2, y2]` datasets to be used for curve-fitting and plotting.

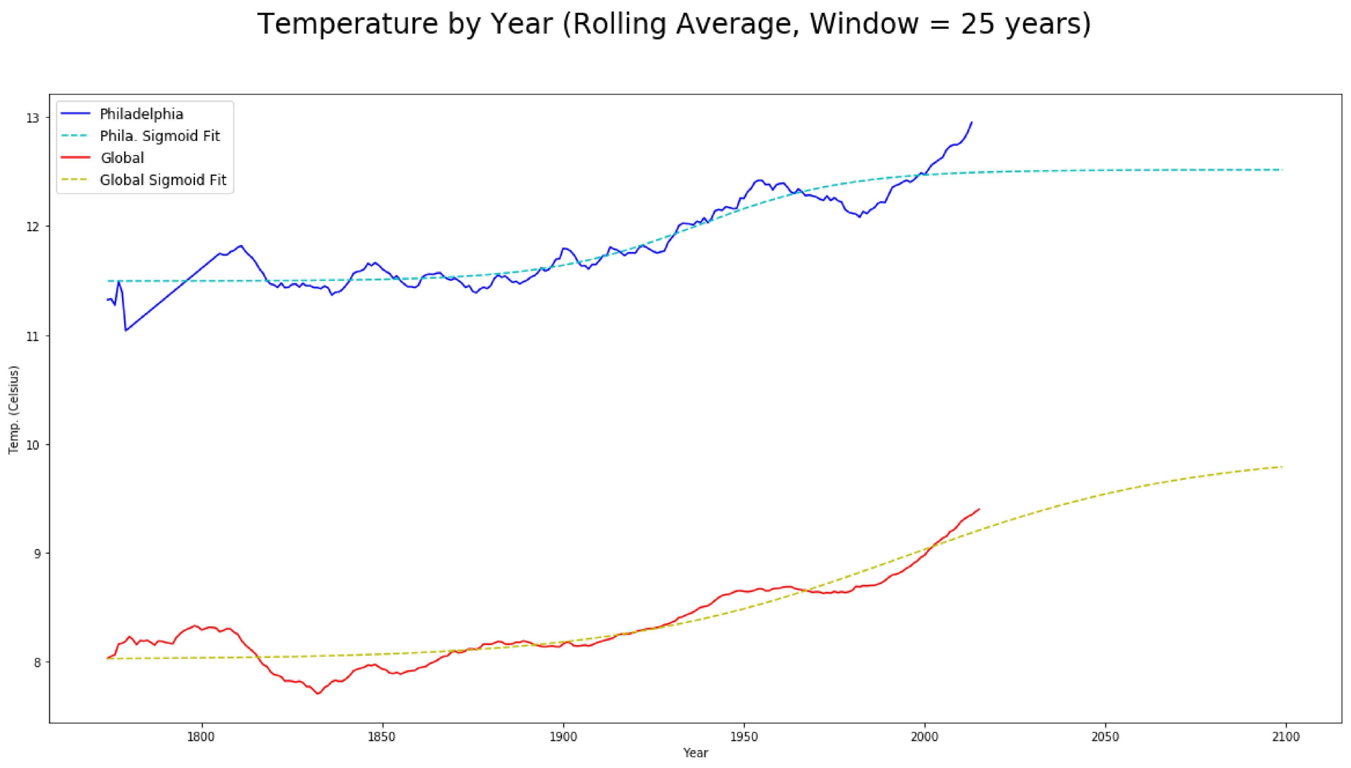
```
In [4]: %%writefile weather_trends.py

        import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        from scipy import optimize


        # Define a sigmoid function to fit the temperature data to
        def fit_func(x, a, b, c, d):
            # a: vertical offset
            # b: max value
            # c: growth factor
            # d: s-curve mid-point
            return (a+b/(1+np.exp(-c*(x-d))))

        # define rolling average window size
        rolling_window = 25

        # load city data into dataframe
        city_data = pd.read_csv('city_data.csv')

        # add rolling average dataframe with the pre-defined window size
        city_data['rolling_avg'] = city_data['avg_temp'].rolling(window=rolling_window).mean()

        # drop rows with empty values
        city_data = city_data.dropna()

        # define local temperature input/output datasets
        x1 = city_data['year'].values
        y1 = city_data['rolling_avg'].values

        # load global data into dataframe
        global_data = pd.read_csv('global_data.csv')

        # add rolling average dataframe with the pre-defined window size
        global_data['rolling_avg'] = global_data['avg_temp'].rolling(window=rolling_window).mean
        ()

        # drop rows with empty values
        global_data = global_data.dropna()

        # define global temperature input/ouput datasets
        x2 = global_data['year'].values
        y2 = global_data['rolling_avg'].values

        # fit local and global datasets to sigmoid curve with set of beginning guesses as define
        d
        popt1, pcov1 = optimize.curve_fit(fit_func, x1, y1, p0=[5,10,0.001,1800])
        popt2, pcov2 = optimize.curve_fit(fit_func, x2, y2, p0=[5,10,0.001,1800])

        # inner join the datasets where yearly temperature data is available for both
        result = pd.merge(city_data, global_data, how='inner', on='year', suffixes=('_local','_g
        lobal'))
        # determine correlation coefficent of merged dataset
        corr_coef = np.corrcoef(result['rolling_avg_local'].values, result['rolling_avg_globa
        l'].values, rowvar=False)

        plt.figure(figsize=(20,10))

        # plot local raw data in blue
```

```
plt.plot(x1, y1, 'b-',label="Philadelphia")
# plot fit data in cyan; extend projection into future
plt.plot(range(x1[0],2100,1), fit_func(range(x1[0],2100,1), *popt1), 'c--',label="Phila.
Sigmoid Fit")

# plot global raw data in red
plt.plot(x2, y2, 'r-',label="Global")
# plot fit data in yellow; extend projection into future
plt.plot(range(x2[0],2100,1), fit_func(range(x2[0],2100,1), *popt2), 'y--',label="Global
Sigmoid Fit")
plt.legend(fontsize=12)

plt.ylabel('Temp. (Celsius)')
plt.xlabel('Year')

plt.suptitle('Temperature by Year (Rolling Average, Window = {} years)'.format(rolling_w
indow),fontsize=24)
plt.show
```

```
Overwriting weather_trends.py
```

3. Data Visualization

In [11]:
```
%run weather_trends.py
```



4. Data Interpretation

Philadelphia is consistently warmer than the global average

It is clearly observable from the plot above that temperatures in Philadelphia are always higher than the global average. There is a fairly consistent offset, the average of which is calculated below to be about 3.53 °C.

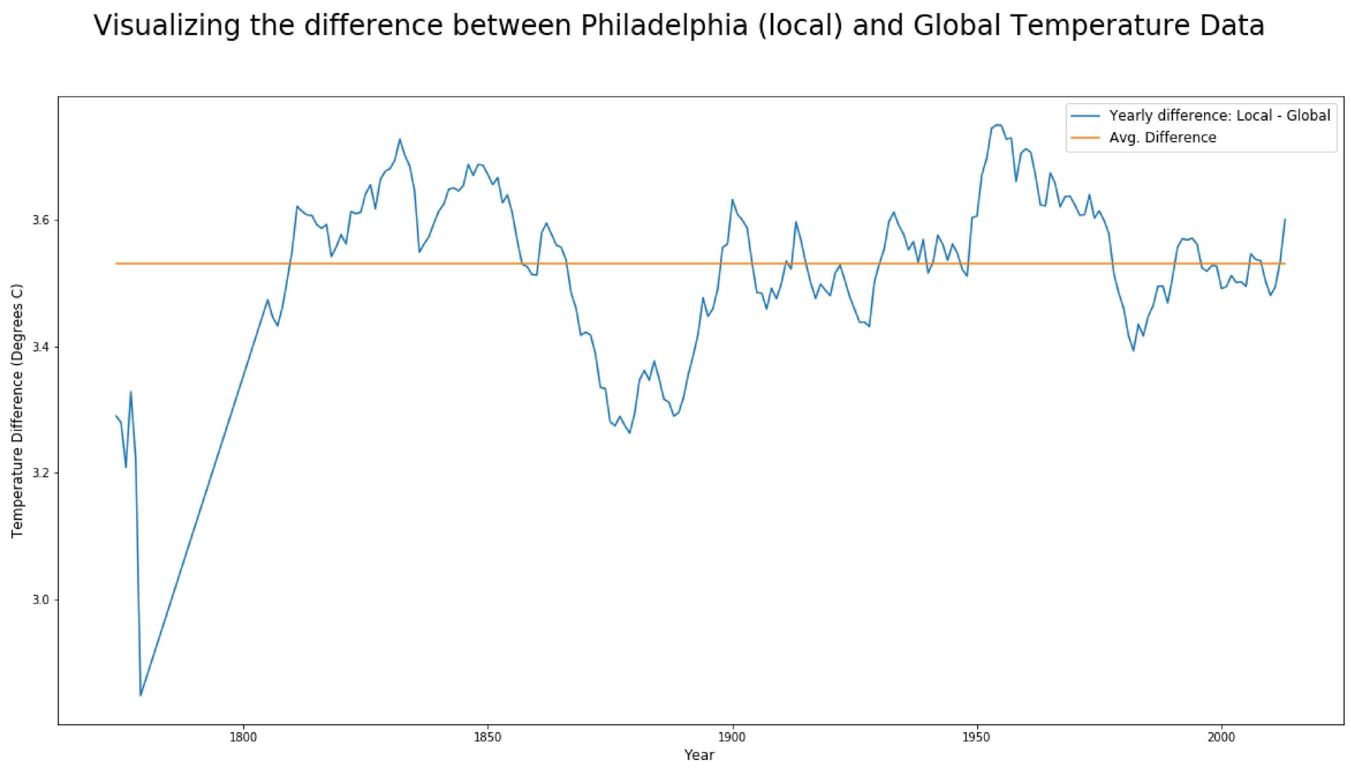
To visualize the difference, the plot below shows only the difference between the two sets of data over time, as well as the average over the entire dataset. One can see that the difference between local and global data after 1800 typically does not dip below 3.2 °C or exceed 3.7 °C.

In [6]:
```
local_avg = city_data['rolling_avg'].mean()
global_avg = global_data['rolling_avg'].mean()
print('\nLocal temperature is an average of %.2f degrees C higher than global temperatur
e.\n' % (local_avg-global_avg))
```

```
Local temperature is an average of 3.53 degrees C higher than global temperature.
```

In [7]:
```
plt.figure(figsize=(20,10))
plt.plot(result['year'],(result['rolling_avg_local']-result['rolling_avg_global']),label
='Yearly difference: Local - Global')
plt.plot(result['year'],[(result['rolling_avg_local']-result['rolling_avg_global']).mean
()]*len(result['year']),label='Avg. Difference')
plt.xlabel('Year',fontsize=12)
plt.ylabel('Temperature Difference (Degrees C)',fontsize=12)
plt.legend(fontsize=12)
plt.suptitle('Visualizing the difference between Philadelphia (local) and Global Tempera
ture Data', fontsize=24)
plt.show
```

Out[7]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



Both local and global temperatures have increased over the last century

Based on the data, temperature held a fairly steady-state value both locally (~11.5 °C) and globally (~8 °C) leading up to the turn of the 20th century. Starting just prior to 1900, temperature values begin to increase at an accelerating rate.

The most recent rolling average (window = 25 years) temperature values are below:

In [8]:
```
print(u'Philadelphia Rolling Avg. Temp 2013: %.2f\u00B0C' % result[result['year']==2013]
['rolling_avg_local'].values)
print(u'Global Rolling Avg. Temp 2013: %.2f\u00B0C' % result[result['year']==2013]['roll
ing_avg_global'].values)
```

```
Philadelphia Rolling Avg. Temp 2013: 12.95°C
Global Rolling Avg. Temp 2013: 9.35°C
```

These represent an increase of about 1.45°C and 1.35°C respectively.

Trend in temperature change is likely to continue into the near future

The trend of increasing temperature persists for the subsequent century. For forecasting, I make the assumption that temperature change is the result of a consistent input (man-made greenhouse gas emissions), and that eventually temperatures will reach a new steady-state value.

By looking at the logistic curve-fit values determined earlier, I determined a predicted long-term steady-state value for local and global average temperatures.

In [9]:
```
print('Predicted long-term average temperatures:\n')
print(u'Philadelphia - %.2f\u00B0C \n' % (popt1[0]+popt1[1]))
print(u'Global - %.2f\u00B0C \n' % (popt2[0]+popt2[1]))
```

```
Predicted long-term average temperatures:

Philadelphia - 12.52°C

Global - 9.92°C
```

Philadelphia temperature data is highly correlated with global data

By looking at the long-term rolling temperature data above, there are clear similarities between the two datasets. Although, as mentioned earlier, the Philadelphia data is offset from the global data, the offset is fairly consistent. The two data sets also share a similar long-term trend. Further, the two datasets share local minima and maxima close to the same years.

An objective measure of the statistical relationship between two variables is the correlation coefficient, which was calculated above as part of Section 2. The corrleation coefficent can between -1 and 1; a value of 0 means the two variables are uncorrelated, -1 would be perfectly inversely correlated, and 1 would be perfectly correlated. Typically, a correlation coefficient above 0.5 indicates a significantly correlated relationship.

In [10]:
```
print('The correlation coefficient between local and global temperature data is: %.2f' %
corr_coef[0][1])
```

```
The correlation coefficient between local and global temperature data is: 0.95
```

With a correlation coefficient calculated to be 0.95, the two datasets are very strongly correlated.

Conclusions

Through this project, I:

- Imported database information using an SQL query
- Converted the raw data into Pandas dataframes
- Manipulated the data using Numpy and SciPy to:
  1. Calculate a rolling average
  2. Determine a curve fit
  3. Determine the correlation between two datasets
- Created a visualization using PyPlot of the manipulated data
- Established the following observations
  1. Philadelphia is warmer than the global average
  2. Local and global temperatures have increased over the past century
  3. Temperature increases are likely to continue in the near future
  4. Local and global datasets are highly correlated

Rubric

Criteria	Meets Specifications
Student is able to extract data from a database using SQL.	- The SQL query used to extract the data is included. - The query runs without error and pulls the intended data.
Student is able to manipulate data in a spreadsheet or similar tool.	Moving averages are calculated to be used in the line chart.
Student is able to create a clear data visualization.	- A line chart is included in the submission. - The chart and its axes have titles, and there's a clear legend (if applicable).
Student is able to interpret a data visualization.	- The student includes four observations about their provided data visualization. - The four observations are accurate.

Suggestions to make your project stand out:

Think about other ways to compare and find insights from this data beyond interpreting the chart. Here are a few ideas:

- What's the correlation coefficient?
- Can you estimate the average temperature in your city based on the average global temperature?
- Multiple cities - Add your favorite cities from around the globe to your visualization. What do you learn about them?