Explore Weather Trends

Lucas Belpaire

October 11, 2018

1 Preparing data

1.1 Accessing Data With SQL

The first step in this project is extracting the data out of the temperatures database using SQL statements.

The following SQL statement extracts the attributes: year, city, country and average temperature of each record from the city_data table.

```
SELECT year, city, country, avg\_temp FROM city_data WHERE city='Brussels'
```

The next SQL statement extracts the attributes: year, average temperature of each record from the global_data table.

```
SELECT year, avg\_temp FROM global_data
```

The data collected with these statements are stored as .csv files. We will use python to process these files.

1.2 Python

Organizing the data will be done by using Python. The NumPy library is used for calculations and the Matplotlib is used to plot the line graph. The libraries need to be imported which can be seen in the following python code.

```
import numpy as np
import unicodecsv
import matplotlib.pyplot as plt
```

The first step is loading the data from the .csv files into variables we can use.

```
def get_csv(filename):
    with open(filename, 'rb') as file:
        reader = unicodecsv.DictReader(file)
        return list(reader)

temperature_data_brussels = get_csv('belgiumTrends.csv')
temperature_data_global = get_csv('globalTrends.csv')
```

Because not all data is necessarily in the correct type, we will convert all data to the types we need.

```
def correct_data_type(data):
        for data_entry in data:
                try:
                data_entry['avg_temp'] = float(data_entry['avg_temp'])
                except ValueError:
                        data_entry['avg_temp'] = None
                try:
                        data_entry['year'] = float(data_entry['year'])
                except ValueError:
```

```
                data_entry['year'] = None
        return data

corrected_data_brussels = correct_data_type(temperature_data_brussels)
corrected_data_global = correct_data_type(temperature_data_global)
```

The next step is calculating the moving average. The function 'get_moving_average' can calculate the moving average for different amount of years. In this example the amount is 10.

```
def get_moving_average(data, length_of_average):

        last_n_years = np.zeros(length_of_average)

        for data_entry in data:
                temperature = data_entry['avg_temp']
                try:
                        temperature = float(temperature)
                        last_n_years = np.insert(last_n_years, 0, temperature)[:-1]
                except ValueError:
                        last_n_years = np.insert(last_n_years, 0, 0)[:-1]
                except TypeError:
                        last_n_years = np.insert(last_n_years, 0, 0)[:-1]
                if not 0 in last_n_years:
                        data_entry['moving_average_temp'] = float(np.average(last_n_years))
                else:
                        data_entry['moving_average_temp'] = None
        return data

temperature_moving_average_brussels = get_moving_average(corrected_data_brussels, 10)
temperature_moving_average_global = get_moving_average(corrected_data_global, 10)
```

After calculating all necessary values the line charts can be constructed. Which is done in the following code.

```
def get_temperatures_and_years(data, moving):
        years = []
        temperatures = []
        for data_entry in data:
                years.append(data_entry['year'])
                if moving:
                        temperatures.append(data_entry['moving_average_temp'])
                else:
                        temperatures.append(data_entry['avg_temp'])
        return [years, temperatures]

def plot_graph(data_set_1, data_set_2, x_label, y_label, moving, label_1, label_2):
        years_and_temperatures = get_temperatures_and_years(data_set_1, moving)
        years = years_and_temperatures[0]
        temperatures = years_and_temperatures[1]
        years_and_temperatures = get_temperatures_and_years(data_set_2, moving)
        years_2 = years_and_temperatures[0]
        temperatures_2 = years_and_temperatures[1]
        x = years
        y = temperatures
```

```
        x2 = years_2
        y2 = temperatures_2
        plt.plot(x, y, label=label_1)
        plt.plot(x2, y2, label=label_2)
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.yticks(np.arange(5, 16, 1.0))
        plt.legend()
        plt.show()

plot_graph(temperature_moving_average_global, temperature_moving_average_brussels,"Year"
                , "Temperature (C)", True, "Global", "Brussels")
plot_graph(corrected_data_global, corrected_data_brussels,"Year", "Temperature (C)"
                , False, "Global", "Brussels")
```

Extra The function 'get_temperatures' returns a NumPy list with all temperatures of a given list of data. This temperature list can be used by the function 'summarize_data' to return the mean, standard deviation, minimum and maximum of that list of temperatures.

```
def get_temperatures(data):
        temperatures = []
        for data_entry in data:
                try:
                        temperatures.append(float(data_entry['avg_temp']))
                except:
                        pass
        return np.array(temperatures)

def summarize_data(data):
        print ('Mean:', np.mean(data))
        print ('Standard_deviation:', np.std(data))
        print ('Minimum:', np.min(data))
        print ('Maximum:', np.max(data))

summarize_data(get_temperatures(temperature_moving_average_global))
summarize_data(get_temperatures(temperature_moving_average_brussels))
```

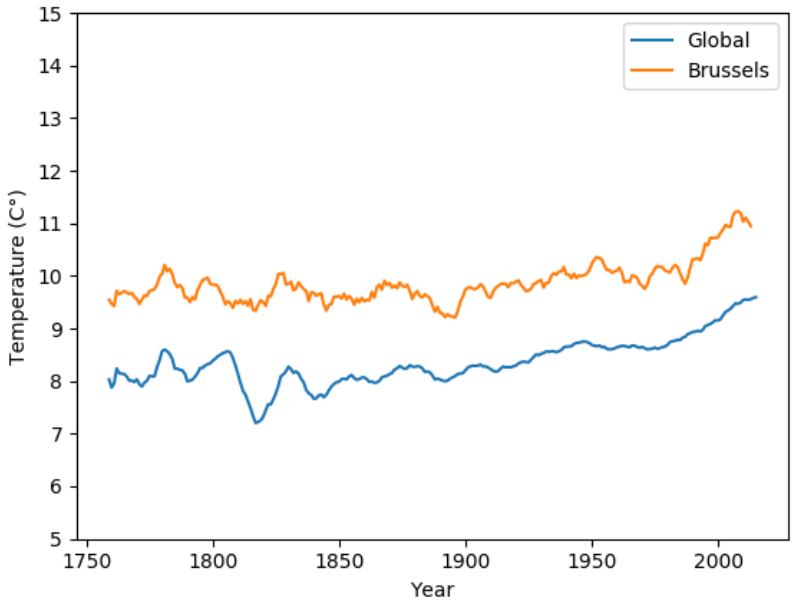
To calculate the correlation coefficient the following method will be used. Please note that NumPy has its own function to calculate this called 'corrcoef'. But I have chosen to implement the function to better understand this coefficient.

```
def correlation_coefficient(data):

        temperatures = []
        years = []

        for data_entry in data:
                try:
                        temperature = float(data_entry['avg_temp'])
                        year = float(data_entry['year'])
                        temperatures.append(temperature)
                        years.append(year)
```

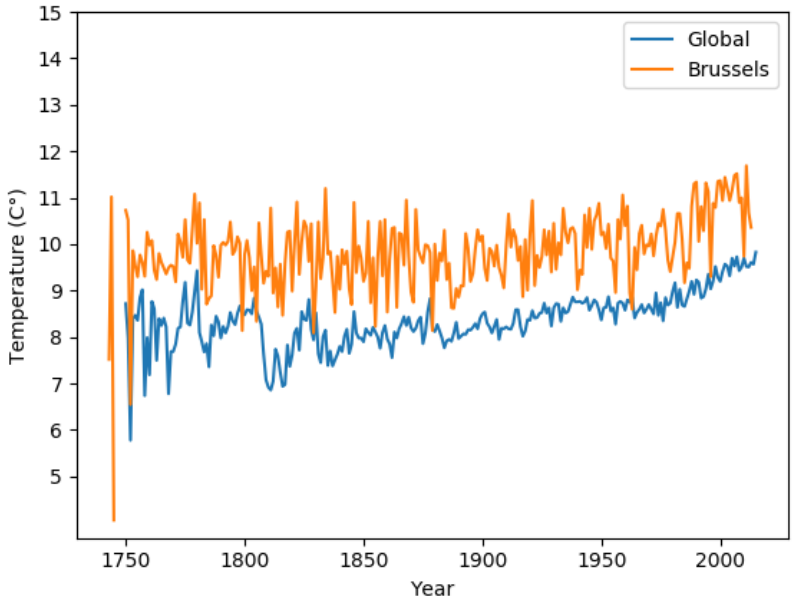
```
                except:
                        pass
        temperatures = np.array(temperatures)
        years = np.array(years)
        sum_temperatures = temperatures.sum()
        sum_years = years.sum()
        sum_years_times_temperatures = (years*temperatures).sum()
        sum_power_of_years = (years*years).sum()
        sum_power_of_temperatures = (temperatures*temperatures).sum()
        n = len(temperatures)

        return ((n * sum_years_times_temperatures) - (sum_temperatures * sum_years)) /
                np.sqrt((n*sum_power_of_years - sum_years**2)*
                (n*sum_power_of_temperatures - sum_temperatures**2))

print(correlation_coefficient(temperature_moving_average_global))
print(correlation_coefficient(temperature_moving_average_brussels))
```

2 Line Charts

The following line charts are produces by the python code.



(a) Moving Average Temperature (10 years)



(b) Average Temperature

Figure 1: Average Temperature Line Charts

3 Observations

Executing the summarize_data function and correlation_coefficient function results in the following values:

```
Global temperature:
Mean: 8.369473684210526
Standard deviation: 0.5836472255514844
Minimum: 5.78
Maximum: 9.83
Correlation coefficient: 0.6227745062665663

Brussels temperature:
Mean: 9.85059925093633
Standard deviation: 0.8252501700366419
Minimum: 4.06
Maximum: 11.69
Correlation coefficient: 0.3821422210006082
```

It is easy to conclude that the mean temperature in Brussels is higher than the global mean. The minimum temperature of Brussels is also lower than the global minimum, the highest temperature of Brussel is also higher than the global highest temperature.

That the difference in minimum and maximum temperature in Brussel is more extreme than the difference between global minimum and maximum can also be seen by Brussels' standard deviation, which is higher than the global one.

The correlation coefficient of the global average graph is 0.62. This means there is a positive correlation between year and temperature. Meaning the temperature will rise as the years increase. The correlation coefficient of the Brussels average graph is 0.38. which also means a positive correlation between year and temperature. The coefficient is lower than the coefficient of the global average, which means that the temperature will also rise as the years increase, but less quickly.

When looking at figure 1.a it is clear that the average temperature is higher in Brussels than globally. This also corresponds with the higher mean temperature in Brussels compared to the global mean temperature.

In figure 1 Brussels' temperature curve is smoother than the global curve. The most probable reason is that the global temperature will not be affected by local outliers, which can be the case in a city like Brussels.

Comparing figure 1 and figure 2 clearly shows the difference between a moving average and an average. The line charts in figure 2 are a lot less smooth than the ones in figure 1. It is a lot harder to read trends in figure 2 which is why a moving average can be very effective.