Exploring Weather Trends

by
John
Larson

Submitted to:
Data Analyst Nanodegree Program
Udacity

Date Submitted:
November 13, 2018

Table of Contents

Abstract

The goal of this project is to analyze local and global data in order to make observations about the similarities and differences of temperature trends. Exporting from a SQL database, calculating a rolling mean, and creating a helpful visualization are necessary steps to accomplish this goal.

SQL Query

The following query selects columns from the "global_data" and "city_data" databases[1] and joins them on a shared "year" column. To distinguish from average city temp and average global temp, "avg_temp" from the "global_data" database was renamed to "gbl_avg_temp"

```
SELECT
global_data.year, city_data.city,
city_data.country, city_data.avg_temp,
global_data.avg_temp AS gbl_avg_temp
FROM global_data JOIN city_data
ON global_data.year = city_data.year;
```

Figure 1. SQL query for CSV export

Evaluate the Data

The csv file that was exported using the SQL query in Figure 1 can be seen in Figure 2.

	A	B	C	D	E
1	year	city	country	avg_temp	gbl_avg_temp
2	1849	Abidjan	CÃ´te D'Ivo	25.58	7.98
3	1850	Abidjan	CÃ´te D'Ivo	25.52	7.9
4	1851	Abidjan	CÃ´te D'Ivo	25.67	8.18
5	1852	Abidjan	CÃ´te D'Ivoire		8.1

Figure 2. CSV export viewed as Excel file

This was imported into a Jupyter Notebook and merged with city latitude data.[2] Because different cities can share a name (e.g. Santiago, Chile vs. Santiago, Philippines), a unique identifier column was created. The result can be seen in Figure 3.

[1] https://classroom.udacity.com/nanodegrees/nd002/parts/93426fc7-0e68-4957-b16b-9fde38776c26/modules/e8455c07-092a-4b76-ba12-018cb53d0526/lessons/d551938c-d004-4801-a269-4b8dd784cc3b/concepts/530f21c0-2f37-4390-aaab-3ce440e56d80#

[2] https://simplemaps.com/data/world-cities

```
In [13]: merge.head()
```

Out[13]:

	year	city	country	avg_temp	gbl_avg_temp	lat	ID
0	1849	Abidjan	Côte D'Ivoire	25.58	7.98	NaN	AbidjanCôte D'Ivoire
1	1850	Abidjan	Côte D'Ivoire	25.52	7.90	NaN	AbidjanCôte D'Ivoire
2	1851	Abidjan	Côte D'Ivoire	25.67	8.18	NaN	AbidjanCôte D'Ivoire
3	1852	Abidjan	Côte D'Ivoire	NaN	8.10	NaN	AbidjanCôte D'Ivoire
4	1853	Abidjan	Côte D'Ivoire	NaN	8.04	NaN	AbidjanCôte D'Ivoire

Figure 3. “merge” DataFrame

As seen in Figure 3, some city latitude data was missing due to differences in city names from the two sources. This was resolved with a Google search of missing latitude info and mapping the values into the appropriate spots using a dictionary.

Missing temperature data or seemingly inexplicable anomalies on a year-to-year basis were replaced using the assumptions described in Figure 4.

In [24]:
```
# Create empty IDdf to populate
IDdf = pd.DataFrame()

# Unit algorithm for cities:
for ID in merge['ID'].unique():

    # Create city specific df. Must be explicit copy of original
    df = pd.DataFrame(merge[merge.loc[:,'ID'] == ID], copy=True)

    # Calculate mean and stddev of each city
    mean = df['avg_temp'].mean()
    std = df['avg_temp'].std()

    # Determine outliers that are outside three stddevs
    maxo = mean + 3*std
    mino = mean - 3*std

    # Edge cases: Should be addressed first so that looping
    # equations work properly
    if df.iloc[0,3] > maxo or df.iloc[0,3] < mino:
        df.iloc[0,3] = mean

    # Unit algorithm for filling outliers
    for x in range(1,len(df)):
        if df.iloc[x,3] > maxo or df.iloc[x,3] < mino:
            df.iloc[x,3] = df.iloc[x-1,3]

    # Ffill nulls before appending to avoid overlap between cities
    df['avg_temp'] = df['avg_temp'].fillna(method='ffill')
    IDdf = IDdf.append(df)

# Bfill remaining nulls (Should only be cases where first value is null)
IDdf['avg_temp'] = IDdf['avg_temp'].fillna(method='bfill')

IDdf.head()
```

Out[24]:

	year	city	country	avg_temp	gbl_avg_temp	lat	ID
0	1849	Abidjan	Côte D'Ivoire	25.58	7.98	5.3	AbidjanCôte D'Ivoire
1	1850	Abidjan	Côte D'Ivoire	25.52	7.90	5.3	AbidjanCôte D'Ivoire
2	1851	Abidjan	Côte D'Ivoire	25.67	8.18	5.3	AbidjanCôte D'Ivoire
3	1852	Abidjan	Côte D'Ivoire	25.67	8.10	5.3	AbidjanCôte D'Ivoire
4	1853	Abidjan	Côte D'Ivoire	25.67	8.04	5.3	AbidjanCôte D'Ivoire

Figure 4. If a temperature for a given year is outside three standard deviations from the mean, replace the temperature with that of the previous year. Forward fill temperatures for missing years. Backfill remaining missing values after appending city specific “df.”

Next, temperature values were converted to Fahrenheit as seen in Figure 5.

```
# Convert Celsius to Fahrenheit and create new columns
IDdf['avg_temp_f'] = IDdf['avg_temp'] * 1.8 + 32
IDdf['gbl_avg_temp_f'] = IDdf['gbl_avg_temp'] * 1.8 + 32
```

Figure 5. Convert Celsius to Fahrenheit

Finally, a 10-year rolling mean was calculated for individual cities as well as the global average. These calculations can be seen in Figure 6.

```
In [32]: # Calculate 10-year rolling mean for individual cities
         IDdf['r_mean'] = IDdf.groupby('ID')['avg_temp_f'].rolling(10).mean().reset_index(0,drop=True)

In [33]: # Calculate 10-year rolling mean globally
         gbldf = IDdf[['year','gbl_avg_temp_f']].sort_values('year').drop_duplicates().reset_index(0,drop=True)
         gbldf['gbl_r_mean'] = gbldf['gbl_avg_temp_f'].rolling(10).mean().reset_index(0,drop=True)

In [34]: # Merge back with IDdf
         totdf = IDdf.merge(gbldf.drop_duplicates(subset=['year','gbl_avg_temp_f']),how='left',on=['year','gbl_avg_temp_f'])
         totdf.head()
```

Out[34]:

	year	city	country	avg_temp	gbl_avg_temp	lat	ID	avg_temp_f	gbl_avg_temp_f	r_mean	gbl_r_mean
0	1849	Abidjan	Côte D'Ivoire	25.58	7.98	5.3	AbidjanCôte D'Ivoire	78.044	46.364	NaN	46.3604
1	1850	Abidjan	Côte D'Ivoire	25.52	7.90	5.3	AbidjanCôte D'Ivoire	77.936	46.220	NaN	46.3784
2	1851	Abidjan	Côte D'Ivoire	25.67	8.18	5.3	AbidjanCôte D'Ivoire	78.206	46.724	NaN	46.4666
3	1852	Abidjan	Côte D'Ivoire	25.67	8.10	5.3	AbidjanCôte D'Ivoire	78.206	46.580	NaN	46.4810
4	1853	Abidjan	Côte D'Ivoire	25.67	8.04	5.3	AbidjanCôte D'Ivoire	78.206	46.472	NaN	46.4576

Figure 6. Rolling mean calculations

Visualize the Data

Plotly was used to create line charts comparing global temperature data to city temperature data. A line chart was chosen because it effectively visualizes changes over time. Figure 7 shows global temperature data compared to Portland from 1900 – 2013.

In general, Portland's temperature trends mimicked those of the global average. In this timeframe, both Portland and the world experienced about a two degrees Fahrenheit increase in average temperature. From about 1940 – 1980, Portland and the world's temperature stayed about the same. Portland has more visible fluctuations in this timeframe which makes sense considering the global average sample size is much larger. A noticeable difference between the two datasets is in the last 20 years, where global temperature has increased at a steady rate whereas Portland's has remained relatively stagnant.

```
trace_gbl = go.Scatter(
    x = totdf.groupby(['year','gbl_r_mean']).size().reset_index(name='count')['year'],
    y = totdf.groupby(['year','gbl_r_mean']).size().reset_index(name='count')['gbl_r_mean'],
    name = 'Global')

trace_city = go.Scatter(
    x = totdf[totdf['city'] == 'Portland'].sort_values(['year'])['year'],
    y = totdf[totdf['city'] == 'Portland'].sort_values(['year'])['r_mean'],
    name = 'Portland')

data = [trace_city,trace_gbl]

layout = go.Layout(
    title = 'Global Temperature',
    xaxis = dict(
        title = 'Year',
        dtick = 10,
        range = [1900, 2013]),
    yaxis = dict(
        title = 'Temperature (°F)',
        dtick = 1))

fig = {'data':data, 'layout':layout}
iplot(fig)
```

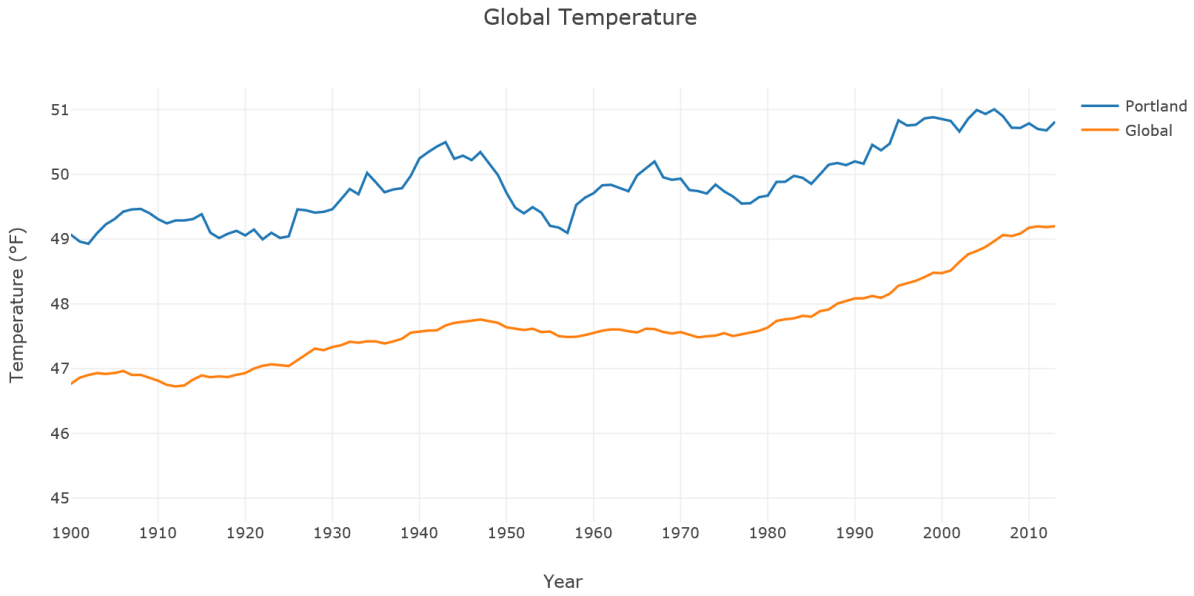

Figure 7. Code for line chart comparing Portland and global temperatures from 1900 – 2013.

Latitude data was imported out of curiosity as to what latitude matches global average temperature changes. The code found in Figure 8 demonstrates how cities that closely matched global data were identified.

```
In [98]: # Calculate the difference between average city temp and average global temp
         totdf['city_gbl_diff'] = abs(totdf['r_mean'] - totdf['gbl_r_mean'])

In [99]: # Filter totdf for period between 1963 - 2013 where a city's avg temp
         # was within 0.1 degrees of the global average temp
         last_50_years = totdf[(totdf['year'] > 1963) & (totdf['year'] < 2013)
                               & (totdf['city_gbl_diff'] < 0.1)]

         # Number of times a city's avg temp was within 0.1 degrees of the global
         # avg temp from 1963 - 2013
         last_50_years['city'].value_counts()

Out[99]: Tbilisi            14
         Yerevan            14
         Sofia              13
         Hamburg            12
         Belfast            11
         Dublin             10
         Prague             10
         Detroit            10
         Chisinau            3
         La Paz              1
         Denver              1
         Colorado Springs    1
         Name: city, dtype: int64
```

Figure 8. A column calculating the difference between city and global temperature was added. The output shows the number of times a city's temperature was within a tenth of a degree of the global temperature in the 50-year period that's being considered.

The lone Southern Hemisphere city (La Paz) was eliminated from the average latitude calculation as to not skew the data. The vast majority of world population and major cities are in the Northern Hemisphere, so average latitude given this dataset will be north of the equator even though there would be a corresponding value south of the equator. Figure 9 shows that the global average temperature is similar to city temperatures along the 46.7th parallel.

```
In [162]: last_50_years[last_50_years['lat'] > 0]['lat'].mean()

Out[162]: 46.68787878787869
```

Figure 9. Calculation of average latitude that corresponds to global temperature changes.

Visualizing the cities from the list in Figure 8 help paint a picture of what cities best matched global temperature for different time frames. Plotly's interactivity makes it easy to hide certain traces, as seen in Figure 10.

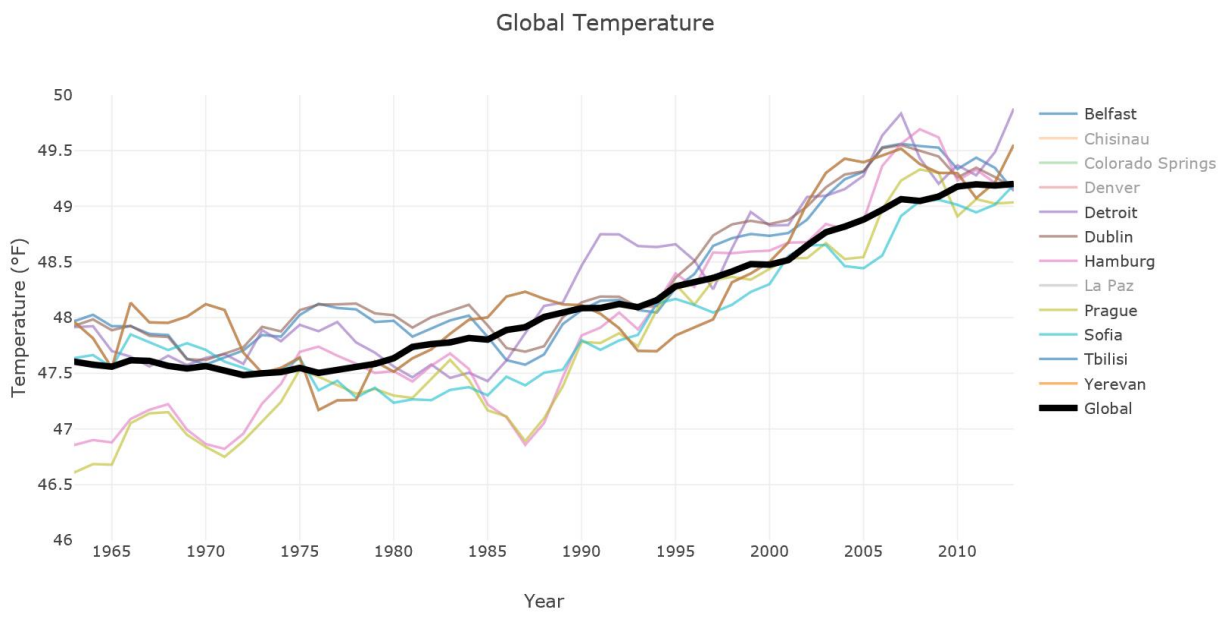

Figure 10. Cities closely matching global temperature data. Traces that have less that ten yearly temperatures within a tenth of the global average for the given 50-year period are hidden.

Relatively speaking, Hamburg and Prague are lower than global temperatures in the 1960s and Detroit is noticeably higher than the global average in the early 1990s. As suspected based on the list in Figure 8, Tbilisi, Yerevan, and Sofia are the best matches to global averages.

In observing these close matches, it was noted that Tbilisi and Yerevan share the same trace and therefore temperature data. This brings to light a question regarding the reliability of the data for at least these two cities which may have not been realized unless a visualization was created to expose this anomaly.

Conclusion

From the SQL query to interacting with Plotly's chart of close temperature matches, understanding each step of the process was crucial in order to make informed observations from the given datasets. Once data was joined and exported from SQL, pandas and numpy made it easy to pull in additional latitude data as well as find and replace null values and outliers. A rolling mean was then calculated to reveal trends in city and global data. Finally, interactive visualizations were created using Plotly to compare Portland and global temperature data and reveal what cities and latitude closely matched global data.