title: "PIC16B HW1:Data Wrangling and Visualization" author: "Zhe Shen" date: "2023-1-28" categories: [PIC16B HW]

HW1:Data Wrangling and Visualization

In this assignment, we will use NOAA climate data to create several interesting interactive data graphs and discover several interesting phenomena.

1. Create a Database

First, we need to create a database with three tables: **temperatures**, **stations**, and **countries** and keep these as three separate tables in your database.This step will help us extract or find data faster and more efficiently by storing it in a database.

In [1]:

```
import pandas as pd
from plotly import express as px
import sqlite3
```

In [2]:

```
conn = sqlite3.connect("temps.db")
```

In [3]:

```
df = pd.read_csv("temps_stacked.csv")
df.to_sql("temperatures", conn, index=False, if_exists='replace')
```

In [4]:

```
df = pd.read_csv("countries.csv")
df.to_sql("countries", conn, index=False, if_exists='replace')
```

```
C:\Users\shenz\anaconda3\lib\site-packages\pandas\core\generic.py:2779: UserWarning: The spaces in these column names will not
be changed. In pandas versions < 0.14, spaces were converted to underscores.
  sql.to_sql(
```

In [5]:

```
df = pd.read_csv("station-metadata.csv")
df.to_sql("stations", conn, index=False, if_exists='replace')
```

2. Write a Query Function

In second part，In order to check the temperature of a particular country for a particular month at a particular time， we will write a Python function called **query_climate_database()** which accepts four arguments:

- country, a string giving the name of a **country** for which we specified.
- **year_begin** and **year_end**, two integers giving the earliest and latest years for which should be returned.
- **month**, an integer giving the month of the year for which we specified.

The return value of this function would be a Pandas dataframe of temperature readings for the specified country, in the specified year range, in the specified month of the year. This dataframe include columns as follows:

- The station name.
- The latitude and longitude of the station.
- Thethe country name in which the station is located.
- The year which we specified.
- The month which we specified.
- The average temperature at the specified station during the specified year and month.

In [6]:

```
def query_climate_database(country, year_begin, year_end, month):
    cmd = \
        """
        SELECT S.NAME, S.LATITUDE, S.LONGITUDE, C.Name AS Country, T.Year, T.Month, T.Temp
        FROM temperatures T
        INNER JOIN stations S ON T.id = S.id
        INNER JOIN countries C ON SUBSTRING(S.id,1,2) = C.'FIPS 10-4'
        WHERE C.Name='{}' AND T.Year>={} AND T.Year<={} AND T.Month={}
        """.format(country, year_begin, year_end, month)

    dread = pd.read_sql_query(cmd, conn)
    return dread
```

Test Case: For example, we can access the temperature of India in January of each year between 1970 and 2020.

In [7]:

```
country_temps = query_climate_database(country="India", year_begin=1980, year_end=2020, month=1)
country_temps
```

Out[7]:

	NAME	LATITUDE	LONGITUDE	Country	Year	Month	Temp
0	PBO_ANANTAPUR	14.583	77.633	India	1980	1	23.48
1	PBO_ANANTAPUR	14.583	77.633	India	1981	1	24.57
2	PBO_ANANTAPUR	14.583	77.633	India	1982	1	24.19
3	PBO_ANANTAPUR	14.583	77.633	India	1983	1	23.51
4	PBO_ANANTAPUR	14.583	77.633	India	1984	1	24.81
...	...	...	...	...	...	...	...
3147	DARJEELING	27.050	88.270	India	1983	1	5.10
3148	DARJEELING	27.050	88.270	India	1986	1	6.90
3149	DARJEELING	27.050	88.270	India	1994	1	8.10
3150	DARJEELING	27.050	88.270	India	1995	1	5.60
3151	DARJEELING	27.050	88.270	India	1997	1	5.70

3152 rows × 7 columns

3. Write a Geographic Scatter Function for Yearly Temperature Increases

In this part, we will explore a more complex issue by create a function to create visualizations:

How does the average yearly change in temperature vary within a given country?

We create and call this function as **temperature_coefficient_plot()**. This function will accept five explicit arguments, and an undetermined number of keyword arguments. Especially worth mentioning are the following arguments:

- **min_obs**, the minimum required number of years of data for any given station.
- ****kwargs**, additional keyword arguments passed to px.scatter_mapbox(). These can be used to control the colormap used, the mapbox style, etc.

The output of this function would be an interactive geographic scatterplot, constructed using Plotly Express. We can see a point for each station, such that the color of the point reflects an estimate of the yearly change in temperature during the specified month and time period at that station. This change value is calculated by a linear regression model.

In [19]:

```
import sqlite3 as sl
import pandas as pd
import numpy as np

import calendar

import plotly.graph_objects as go
from plotly import express as px
from plotly.subplots import make_subplots

from sklearn.linear_model import LinearRegression
def temperature_coefficient_plot(country, year_begin, year_end, month, zoom, mapbox_style, min_obs=10, **kwargs):
    cmd = \
        """
        SELECT S.NAME, S.LATITUDE, S.LONGITUDE, C.Name AS Country, T.Year, T.temp
        FROM temperatures T
        INNER JOIN stations S ON T.id = S.id
        INNER JOIN countries C ON SUBSTRING(S.id,1,2) = C.'FIPS 10-4'
        WHERE C.Name='{}' AND T.Year>={} AND T.Year<={} AND T.Month={}
        """.format(country, year_begin, year_end, month)

    def get_k_from_linreg(sample):

        '''
        Fits and applies linear regression to sample (frame) of data.

        '''


        linreg = LinearRegression()
        linreg.fit(np.array(sample['Year']).reshape(-1, 1), sample['Temp'])
        increasing_temp_rate = linreg.coef_[0] # k-value from kx+b mathematical function

        return increasing_temp_rate

    data = query_climate_database(country, year_begin, year_end, month)
    filtered_data = data[data['obs'] >= min_obs].reset_index(drop=True, inplace=False)
    filtered_data = filtered_data.merge(filtered_data.groupby(['NAME', 'Month']
                                                              ).apply(get_k_from_linreg).reset_index(),
                                        how='left',
                                        on=['NAME', 'Month'])
    #df = pd.read_sql_query(cmd, conn)
    #df['count'] = df.groupby('NAME')['Year'].transform('count')
    #df = df[(df['count'] >= min_obs)]
    #df['Estimated Yearly Increase(° C)'] = df.groupby('NAME')['Temp'].transform('var')
    #print(df)


    fig = px.scatter_mapbox(filtered_data,
                            lat = "LATITUDE",
                            lon = "LONGITUDE",
                            hover_name = "NAME",
                            color = "Estimated Yearly Increase(° C)",
                            title = 'Estimates of yearly increase in temperature in January for stations in India, years 1980 - 2020',
                            zoom = zoom,
                            opacity = 0.5,
                            height = 500,
                            mapbox_style = mapbox_style)

    fig.update_layout(margin={"r":0,"t":30,"l":0,"b":0})
    return fig


color_map = px.colors.diverging.RdGy_r # choose a colormap

fig = temperature_coefficient_plot("India", 1980, 2020, 1,
                                   min_obs = 10,
                                   zoom = 2,
                                   mapbox_style="carto-positron",
                                   color_continuous_scale=color_map)

fig.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3079             try:
-> 3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'obs'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
<ipython-input-19-c6c164a20e3b> in <module>
     63 color_map = px.colors.diverging.RdGy_r # choose a colormap
     64
---> 65 fig = temperature_coefficient_plot("India", 1980, 2020, 1,
     66                                    min_obs = 10,
     67                                    zoom = 2,

<ipython-input-19-c6c164a20e3b> in temperature_coefficient_plot(country, year_begin, year_end, month, zoom, mapbox
_style, min_obs, **kwargs)
     34
     35     data = query_climate_database(country, year_begin, year_end, month)
---> 36     filtered_data = data[data['obs'] >= min_obs].reset_index(drop=True, inplace=False)
     37     filtered_data = filtered_data.merge(filtered_data.groupby(['NAME', 'Month']
     38                                                               ).apply(get_k_from_linreg).reset_index(),

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3022             if self.columns.nlevels > 1:
   3023                 return self._getitem_multilevel(key)
-> 3024             indexer = self.columns.get_loc(key)
   3025             if is_integer(indexer):
   3026                 indexer = [indexer]

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3080                 return self._engine.get_loc(casted_key)
   3081             except KeyError as err:
-> 3082                 raise KeyError(key) from err
   3083
   3084         if tolerance is not None:

KeyError: 'obs'
```