HW4 (Score: 18.5 / 20.0)

1. Task (Score: 7.0 / 8.0)
2. Comment
3. Task (Score: 11.5 / 12.0)

DATA 601: HW4

Fall 2019

Due: Wed. Oct. 9, 2019 (by 23:55)

Learning Objectives

- Work with more than one tabular dataset to compute quantities.
- Wrangle and organize data for the purpose of geographic visualization.
- Produce interactive charts and plots.

This is an individual homework assignment.

Please complete this homework assignment within the Jupypter notebook environment, by inserting Markdown and Code cells to properly format your answers. Please do not alter the question cells.

Your completed Jupyter notebook is to be submitted via the HW4 dropbox on D2L.

In [1]:

```
# Import packages

import numpy as np
import pandas as pd
import datetime as dt
import geopandas as gpd
import shapely.geometry as sg

import matplotlib as mpl
import matplotlib.pyplot as plt

import plotly as plotly
import plotly.offline as py
import plotly.graph_objs as go
import plotly.express as px

py.init_notebook_mode(connected=True)

#%matplotlib inline
mpl.rcParams['figure.dpi'] = 96
mpl.style.use('ggplot')
```

(Top)

Question 1: Scatter Plot of Rainfall Channels

This question builds upon HW3. Please ask for assistance if you did not manage to complete the portion of HW3 related to this question.

Use the provided Calgary rainfall dataset as well as the dataset containing the locations of the rainfall channels to produce a visualization that:

- shows on a map scatter plot, the total amount of rainfall recorded by each channel for the *2018* rainfall season;
- provides a hover interaction for each channel that shows as text the following: total rainfall for the season, the number of rainy days for the season, and the mean rainfall per day for all the days it rained.

Comments:
You assume that if a day is listed in the dataset, it has rain. That's true of this dataset, but dangerous to assume in general. The default zoom level is too close, some datapoints are well off-screen and could easily be missed. Good of you to notice that a rain gauge had an invalid lat/long, but your solution is very ad-hoc; a proper bounds test is much more robust.
Good use of comments, however.

In [2]:

```
# Import data as 'rainfall'
rainfall = pd.read_csv("Historical_Rainfall.csv")

# Convert TIMESTAMP to datetime object using the format '%Y/%m/%d %H:%M:%S %p'
rainfall['TIMESTAMP'] = pd.to_datetime(rainfall['TIMESTAMP'], format='%Y/%m/%d %H:%M:%S %p')

# Drop columns using pandas.DataFrame.drop
rainfall_dropped = rainfall.drop(['ID', 'RG_ACTIVE'], axis = 1)

# Create DAY, MONTH and YEAR columns
rainfall_dropped['YEAR'] = rainfall_dropped['TIMESTAMP'].dt.year
rainfall_dropped['MONTH'] = rainfall_dropped['TIMESTAMP'].dt.month
rainfall_dropped['DAY'] = rainfall_dropped['TIMESTAMP'].dt.day

# Create dataframe with channel locations
clocs = pd.read_csv('Rain_Gauge_locations.csv',usecols=["CHANNEL", "Northing", "Easting"])
clocs.set_index('CHANNEL', inplace=True)
clocs.drop_duplicates(inplace=True)
clocs.sort_index(inplace=True)
clocs['Northing'] = clocs['Northing'].astype(float)
clocs['Easting'] = clocs['Easting'].astype(float)
clocs.drop(47, inplace=True) # This channel had a 0,0 coordinate which affected the map, this deletes it
```

In [3]:

```
# Create a dataframe that includes:
    # The number of rainy days per channel in 2018
    # Total rainfall per channel in 2018
    # Average rainfall per day per channel in 2018
    # Locations

## Create daily rainfall total object for the 2018 season
rainfall_daily_total_2018 = rainfall_dropped.groupby(['YEAR', 'MONTH', 'DAY', 'CHANNEL']).sum().loc[2018]

## Reset index in order to properly groupby CHANNEL
reset_rainfall_2018 = rainfall_daily_total_2018.reset_index()

## Drop unnecessary columns
rainfall_2018_dropped = reset_rainfall_2018.drop(['MONTH', 'DAY'], axis = 1)

### Group by channel and count the number of days of rain recorded by each channel
rainfall_days_2018_per_channel = rainfall_2018_dropped.groupby(['CHANNEL']).count()
rainfall_days_2018_per_channel = rainfall_days_2018_per_channel.rename(columns={'RAINFALL':'Days of Rain'})

### Group by channel and sum the rainfall recorded by each channel
rainfall_total_2018_per_channel = rainfall_2018_dropped.groupby(['CHANNEL']).sum().round(2)
rainfall_total_2018_per_channel = rainfall_total_2018_per_channel.rename(columns={'RAINFALL':'2018 Rainfall (mm)'})

### Merge data frames and create new column with mean rainfall per day
rainfall_2018 = pd.merge(rainfall_days_2018_per_channel, rainfall_total_2018_per_channel, on = ['CHANNEL'])
rainfall_2018['Average Daily Rainfall (mm)'] = round(rainfall_2018['2018 Rainfall (mm)'] / rainfall_2018['Days of R
ain'], 2)

# Merge dataframes with 2018 rainfall data and channel locations
rainfall_2018_with_locs = pd.merge(rainfall_2018, clocs, on = ['CHANNEL'], how = 'inner')
```

In [4]:

```
# I used a personal token from mapbox. Please let me know if you need me to provide this. Thanks!
with open('mapboxtoken.txt', 'r') as file:
    token = file.read().replace('\n', '')
px.set_mapbox_access_token(token)
fig = px.scatter_mapbox(rainfall_2018_with_locs,
                        lat="Northing",
                        lon="Easting",
                        size='Average Daily Rainfall (mm)',
                        zoom=10,
                        color="2018 Rainfall (mm)", # Visualises distribution of total rainfall with a color scale
                                                    # This also allows for 2018 rainfall to show in the text box
                        color_continuous_scale=px.colors.sequential.Blues,
                        hover_data=['Days of Rain'], # This allows the user to see the days of rain in the text box
                        width=1000,
                        height=800)

fig.show()
```

(Top)

Question 2: Life Expectancy Index

The Life Expectancy Index (LEI) is used as part of the Human Development Index (https://en.wikipedia.org/wiki/Human_Development_Index) which ranks countries into different tiers of human development. The LEI for a country is defined as:

$$\mathrm{LEI} = \frac{\mathrm{LE} - 20}{85 - 20},$$

where LE is the life expectancy at birth.

Using the provided datasets only, perform the following tasks:

- Produce a choropleth world map visualization showing the LEI for the year 2017. Colour map the LEI values according to the following bins: $[0.5, 0.6)$, $[0.6, 0.7)$, $[0.7, 0.8)$, $[0.8, 0.9)$ and $[0.9, 1)$, and use a visually distinct colour to identify countries for which the data is not available.
- Please browse the UNDP Human Development Reports (http://hdr.undp.org/en/data). This task asks you to produce a visualization that is inspired by the UNDP's visualization of the LEI. Compute the LEI per country for all the years for which the data is available. Use k-means clustering to cluster the LEI series' into *four* clusters. Produce a visualization showing the LEI series' as lines for all the countries and colour the lines according to their cluster membership.

For bonus marks, please include informative non-default hover interactions in your visualizations to facilitate data exploration and comparison.

In [5]:

```
# Create dataframe with LEI for each Country in 2017
LEI_2017_raw = pd.read_excel('Life expectancy at birth (years).xlsx', skiprows=1, usecols=[0, 1, 29])

# Convert 2017 column to a string in order to properly index this column
LEI_2017 = LEI_2017_raw.rename(columns={2017:'2017'})

# Convert values in 2017 column to numeric because they are strings for some reason?
LEI_2017['2017'] = pd.to_numeric(LEI_2017['2017'], errors='coerce')

# Create LEI column that converts LE to LEI
LEI_2017['LEI'] = round((LEI_2017['2017'] - 20) / (85 - 20), 3)

# Create a column for the bin that the LEI falls into
# This is done by taking the LEI column and dropping all decimal places past the first decimal place
# using a string index. For example, "0.1948329"[:3] = "0.1" using this code.
LEI_2017['LEI bin'] = ((LEI_2017['2017'] - 20) / (85 - 20)).map(lambda x: float(str(x)[:3]))
```

In [6]:

```
## Define color scale to bin the LEI.
# This purposely uses a darker color for the low end of the scale so that countries that do not
# have data available are visually distinct. If the color scale was lighter at the low end of the scale,
# the countries that do not have data would not be as visually distinct.
my_colorsc=[[0, 'rgb(250,159,181)'],
            [0.2, 'rgb(250,159,181)'],
            [0.2, 'rgb(247,104,161)'],
            [0.4, 'rgb(247,104,161)'],
            [0.4, 'rgb(221,52,151)'],
            [0.6, 'rgb(221,52,151)'],
            [0.6, 'rgb(174,1,126)'],
            [0.8, 'rgb(174,1,126)'],
            [0.8, 'rgb(122,1,119)'],
            [1, 'rgb(122,1,119)']]

# Create Choropleth map using Plotly
fig = go.Figure(data=go.Choropleth(
    locations=LEI_2017['Country'],
    customdata = LEI_2017['LEI'], # Includes the precise LEI data for the hover text
    z = LEI_2017['LEI bin'], # Uses the 'LEI bin' column to assign the LEI to specific color bins
    locationmode = 'country names', # Instead of using the "country_codes.csv", this argument works just as well
    colorscale = my_colorsc, # Use the custom color scale I created above
    colorbar_title = "LEI",
    zmin = 0.5, # Sets the minimum of the color scale
    zmax = 1, # Sets the maximum of the color scale
    # "hovertemplate" overrides the 'hoverinfo' argument to include the 'LEI' column and the 'Country' column
    # The <extra></extra> removes the secondary box
    hovertemplate = "Country: %{location} | LEI: %{customdata:.3f} <extra></extra>"
))

fig.update_layout(
    title_text = 'Life Expectancy Index - 2017'
)

fig.show()
```

In [7]:

```
# Load dataframe with raw LE data
LE_raw = pd.read_excel('Life expectancy at birth (years).xlsx', skiprows=1)

# Create new empty dataframe for LEI data
LEI = pd.DataFrame()

# Loop over the columns in 'LE_raw' to convert LE to LEI in order to create the LEI dataframe
for i in range(1990, 2018, 1):
    LEI[i] = round((pd.to_numeric(LE_raw[i], errors='coerce') - 20) / (85 - 20), 3)

# Drop the countries that are missing data. They will be excluded from this visualization
LEI = LEI.dropna()

# Perform k means clustering with 4 clusters
from sklearn.cluster import KMeans
kmeansLEI = KMeans(n_clusters=4, random_state=0).fit(LEI)

# Add in Country and Cluster label
LEI['Country'] = LE_raw['Country']
LEI['Cluster'] = kmeansLEI.labels_

# Melt dataframe for use in plot
LEI_flat = LEI.melt(id_vars = ['Country', 'Cluster'], value_name = "LEI", var_name = "Year")
```

In [8]:

```
import plotly.express as px
fig = px.line(LEI_flat, x="Year", y="LEI", color="Cluster", line_group="Country", hover_name="Country",
        line_shape="spline", render_mode="svg")
fig.show()
```

In [ ]:

```

```