Programming-Assignment-4

July 3, 2020

1 Assignment 4

Before working on this assignment please read these instructions fully. In the submission area, you will notice that you can click the link to **Preview the Grading** for each step of the assignment. This is the criteria that will be used for peer grading. Please familiarize yourself with the criteria before beginning the assignment.

This assignment requires that you to find **at least** two datasets on the web which are related, and that you visualize these datasets to answer a question with the broad topic of **religious events or traditions** (see below) for the region of **Daegu, Daegu, South Korea**, or **South Korea** more broadly.

You can merge these datasets with data from different regions if you like! For instance, you might want to compare **Daegu, Daegu, South Korea** to Ann Arbor, USA. In that case at least one source file must be about **Daegu, Daegu, South Korea**.

You are welcome to choose datasets at your discretion, but keep in mind **they will be shared with your peers**, so choose appropriate datasets. Sensitive, confidential, illicit, and proprietary materials are not good choices for datasets for this assignment. You are welcome to upload datasets of your own as well, and link to them using a third party repository such as github, bitbucket, pastebin, etc. Please be aware of the Coursera terms of service with respect to intellectual property.

Also, you are welcome to preserve data in its original language, but for the purposes of grading you should provide english translations. You are welcome to provide multiple visuals in different languages if you would like!

As this assignment is for the whole course, you must incorporate principles discussed in the first week, such as having as high data-ink ratio (Tufte) and aligning with Cairo's principles of truth, beauty, function, and insight.

Here are the assignment instructions:

- State the region and the domain category that your data sets are about (e.g., **Daegu, Daegu, South Korea** and **religious events or traditions**).
- You must state a question about the domain category and region that you identified as being interesting.
- You must provide at least two links to available datasets. These could be links to files such as CSV or Excel files, or links to websites which might have data in tabular form, such as Wikipedia pages.
- You must upload an image which addresses the research question you stated. In addition to addressing the question, this visual should follow Cairo's principles of truthfulness, functionality, beauty, and insightfulness.

- You must contribute a short (1-2 paragraph) written justification of how your visualization addresses your stated research question.

What do we mean by **religious events or traditions**? For this category you might consider calendar events, demographic data about religion in the region and neighboring regions, participation in religious events, or how religious events relate to political events, social movements, or historical events.

1.1 Tips

- Wikipedia is an excellent source of data, and I strongly encourage you to explore it for new data sources.
- Many governments run open data initiatives at the city, region, and country levels, and these are wonderful resources for localized data sources.
- Several international agencies, such as the United Nations, the World Bank, the Global Open Data Index are other great places to look for data.
- This assignment requires you to convert and clean datafiles. Check out the discussion forums for tips on how to do this from various sources, and share your successes with your fellow students!

1.2 Example

Looking for an example? Here's what our course assistant put together for the **Ann Arbor, MI, USA** area using **sports and athletics** as the topic. Example Solution File

[1]:
```
import math
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

[2]:
```
def read_data():
    df = pd.read_csv('data/C2A2_data/BinnedCsvs_d400/
 ↪fb441e62df2d58994928907a91895ec62c2c42e6cd075c2700843b89.csv')
    df['Date'] = pd.to_datetime(df['Date'])
    df['Year'] = df['Date'].map(lambda d: d.year)

    df_agg = (df[['Year', 'Data_Value']]
           .groupby('Year')
           .apply(lambda sdf: pd.Series({'mean': sdf['Data_Value'].mean(),
                                         'err': 2 * 1.98 * sdf['Data_Value'].
 ↪std() / math.sqrt(sdf.shape[0])}))
           .sort_index()
           .copy(True)
          )
    return df, df_agg
df, mi = read_data()
```

[3]:
```
gl = pd.Series([0.69, 0.63, 0.66, 0.54, 0.64, 0.71, 0.6, 0.63, 0.65, 0.74, 0.
 ↪86], index=mi.index)
```

[4]:
```
fig, ax1 = plt.subplots(figsize=(12, 8))

plt.title('Ann Arbor temperature in comparison with Global Warming trend',
          size=18, alpha=0.8)

line1 = ax1.errorbar(mi.index, mi['mean'], yerr=mi['err'], color = 'red', lw=3,␣
 ↪alpha=0.6, elinewidth=2, capsize=5, capthick=3)

temp_anno_pos = [80, 85, 90, 95, 100, 105, 110]
temp_anno_label = list(map(lambda t: '{}$^{{\circ}}$C'.format(int(t/10)),␣
 ↪temp_anno_pos))
ax1.hlines(temp_anno_pos, *ax1.get_xlim(), color='k', linestyles='--', lw=0.3,␣
 ↪alpha=0.2)
for pos, text in zip(temp_anno_pos, temp_anno_label):
    ax1.annotate(s=text, xy =(2004.5, pos), xycoords='data', alpha=0.8, size=12,
                 verticalalignment='center', horizontalalignment='right' ,␣
 ↪rotation=0)



year_anno_pos = np.arange(2005, 2016, 1)
year_anno_label = list(map(str, year_anno_pos))
ax1.vlines(year_anno_pos, *ax1.get_ylim(), color='k', linestyles='--', lw=0.3,␣
 ↪alpha=0.2)
for pos, text in zip(year_anno_pos, year_anno_label):
    ax1.annotate(s=text, xy =(pos, 73), xycoords='data', alpha=0.8, size=12,
                 verticalalignment='top', horizontalalignment='center' ,␣
 ↪rotation=0)

ax2 = ax1.twinx()
line2 = ax2.plot(mi.index, gl, ls='--', lw=3, alpha=0.6, color='black')
yl, yu = ax2.get_ylim()
y_adj = .06
ax2.set_ylim((yl - y_adj, yu - y_adj))

lines = [line1.lines[0]] + line2
labels = ['Annual Mean of Ann Arbor Temperature' , 'Annual Mean of Global␣
 ↪Land-Ocean Temperature Index ']
leg = ax1.legend(lines, labels, bbox_to_anchor=(.1, .98), loc=2)


ax1.tick_params(top='off', bottom='off', left='off', right='off',␣
 ↪labelleft='off', labelbottom='off')
ax2.tick_params(top='off', bottom='off', left='off', right='off',␣
 ↪labelright='off', labelbottom='off')
for spine in ax1.spines.values():
    spine.set_visible(False)
```

```
for spine in ax2.spines.values():
    spine.set_visible(False)
```