MCIS6273 Data Mining (Prof. Maull) / Fall 2017 / HW4

This assignment is worth up to 20 POINTS to your grade total if you complete it on time.

Points Possible	Due Date	Time Commitment (estimated)
20	Thursday, Dec 7 @ Midnight	*up to* 3 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by Univerisity or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- BUILD BASIC VISUALIZATIONS IN Plot.ly: Work with Plot.ly Python API to build a distribution plot
- BUILD MAPS IN MATPLOTLIB: Work with Matplotlib `Basemap` to make a map
- BUILD BASIC VISUALIZATIONS IN Matplotlib: Build a bubble plot
- BUILD BASIC VISUALIZATIONS IN Bokeh: Build a Chord Diagram

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, clone the course repository and modify the `hw4.ipynb` file in the `homework/hw4` directory. If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using Github and cloning repositories.

Turn in a copy of a `.ipynb` file, a PDF or Word Document to Blackboard with the code or answers to the questions /tasks labeled with the § sign.

ASSIGNMENT TASKS

(25%) BUILD BASIC VISUALIZATIONS IN Plot.ly: Work with Plot.ly Python API to build a distribution plot

Much of your energy as a data scientist will be in pre-processing data and presenting that data in ways that are useful and informative to others, often not involved in the data analytics process. We talked briefly about plot.ly in the lecture and now we're going to use it.

We have a large dataset of wine ratings from Wine Magazine on Kaggle. We would like to visualize the distribution of ratings for wines based on the following countries: *Spain, Chile, Argentina, Austria*. What we are after is to understand **visually** which of these countries has the best distribution of high rated wines.

§ Use the distplot tool to display the distribution of points with the following 5 countries: Spain, Chile, Argentina, Portugal and Austria. You can use the boilerplate code provided here

```
%matplotlib inline
import pandas as pd
import numpy as np
import plotly
```

```
import plotly.plotly as py
import plotly.figure_factory as ff
import plotly.graph_objs as go

# required to make offline plots plotly.offline.init_notebook_mode(connected=True)
df = pd.read_csv("./data/winemag-data_first150k.csv")
df = df.drop('Unnamed: 0', axis=1)

country_list = ['Spain', 'Chile', 'Argentina', 'Portugal', 'Austria']
points_data = []

## TWO LINE FOR LOOP HERE ##

plotly.offline.iplot(ff.create_distplot(points_data,
                                        country_list,
                                        curve_type='kde',
                                        bin_size=1),
                             filename='distplot with pandas')
```

You will need to add only 2 lines of code to loop over the `country_list` to append the points to that. The loop just iterates over all the countries in `country_list` and then appends the list of points for that country to `points_data`. In other words, `points_data` will be a list of lists (e.g. `[[90,91,87],[98,96,84]]`) where `points_data[0]` contains the list of numeric point scores for the country in `country_list[0]`. You can get the `points` list with `df[df.country==country_in_country_list].points.tolist()`.

§ Visually inspect the `distplot` and look at the peaks in the distribution. Which country has the highest number of high scoring wines?

§ Make sure to turn in the full notebook code with your submission!

(25%) BUILD MAPS IN MATPLOTLIB: Work with Matplotlib `Basemap` to make a map

Sometimes you might be asked to plot data on a map. We're going to have some fun here and work with another really fun dataset from Kaggle. This time we're going to look at tens of thousands UFO sightings data over the past century. We've been tasked with just looking at specific UFO types and a specific date range.

We want to plot on the map only those sightings which

- were documented between **January 1, 1970** and **January 1, 1980**,
- the UFO sightings where the `shape` was listed as `formation`,
- sightings ONLY in the United States.

The file for this part is in `data/scrubbed.csv` and is easiest to load as the Kaggle version requires logging in to obtain the full version of the same file. You will need to restrict the data to just those parameters to complete the parts below.

You will only need to load the appropriate data into a Dataframe called `df_us` and plug in the code provided below:

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np
import pandas as pd
```

```
## LOAD THE DATA INTO A DATAFRAME CALLED df_us  ##
## EVERYTHING ELSE SHOULD PLOT CORRECTLY        ##

scale = 1

map = Basemap(width=10000000,height=6000000,projection='lcc',
            resolution=None,lat_1=45.,lat_2=55,lat_0=50,lon_0=-107.)
plt.figure(figsize=(19,20))
map.bluemarble()

## NOTE: df_us!
for i in df_us.itertuples(index=False):
    map.plot(*map(i.longitude, i.latitude), marker='o', color='Red', markersize=5)

plt.show()
```

Study the code to understand what is going on.

§ Look at the map visually. What pattern are you seeing in the data for this time period?

§ What would you suggest to improve this visualization?

§ Make sure to turn in the full notebook code with your submission!

(25%) BUILD BASIC VISUALIZATIONS IN Matplotlib: Build a bubble plot

Bubble plots are valuable visualizations for understanding the proportion of data relative to two variables. We're going to take an inspiration from *Visualize This!* (Yau, 2011) and plot some data along these dimensions. In this case, we are going to use it to plot 3 dimensions with 2 dimensions using a scatterplot, but using the *bubble* size as the third dimension.

I have created a unified dataset from three different datasets, so this is work you don't have to do (this time :smile:).

- US unemployment rate **by city** as of October 2017 from the Bureau of Labor Statistics,
- US educational attainment **by state** from Wikipedia but sourced from the US Dept of Education, and
- US Crime index **by city** provided by Numbeo but sourced from the US FBI crime statistics dataset.

What we want to do is make a bubble plot. In the file `data/bubble_plot_data.csv` you will find the data you need. It isn't large, so there is not a lot to play with there. Use the boilerplate code below and fill in what is required so that you can read the file and display the data.

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import math

### LOAD THE BUBBLE DATAFILE INTO df ##

# create data
x = df.pct_ba
y = df.crime_index
z = df.unemployment
z = 50*((z-2.3)*(z-2.3))+1

plt.figure(figsize=(20,10))
```

```
# Change color with c and alpha. I map the color to the X axis value.
plt.scatter(x, y, s=z,
            c=z*.05,
            cmap="Reds",
            alpha=0.4,
            edgecolors="grey",
            linewidth=2)

# Add titles (main and on axis)
plt.xlabel(x.name)
plt.ylabel(y.name)
plt.title("Bubble plot of Crime Index, Education and Unemployment")

# thank you Scott Boston: https://stackoverflow.com/questions/46027653/adding-labels-in-x-y-scatter-plo
def label_point(x, y, val, ax):
    a = pd.concat({'x': x, 'y': y, 'val': val}, axis=1)
    for i, point in a.iterrows():
        ax.text(point['x']+.001, point['y'], str(point['val']))

label_point(x, y, df.city, plt.gca())

plt.show()
```

§ Recall the size and color of the bubble indicates relative unemployment (larger unemployment, larger bubble. Do you notice any patter about umemployment relative to crime and education with the cities we've explored here?

§ Make sure to turn in the full notebook code with your submission!

(25%) BUILD BASIC VISUALIZATIONS IN Bokeh: Build a Chord Diagram

Chord diagrams provide a circular visual tool for see connections between objects that you would not ordinarily be able to see in the same amount of space. You can read a bit more about them here. Though there is a debate on whether Chord diagrams are too visually dense and thus not as valuable as people may think, they are nonetheless valuable and we will see why.

For social networks and data where there are clear connections between objects, they **can** be valuable if not overdone and if the amount of data in them is not so dense as to make them unreadable.

We're going to practice Chord diagrams using the very nice Bokeh library. Spend some time browsing this very useful library – it has a lot of really nice features to offer and has some really great features for real-time display of data in web contexts.

We're going to borrow data from a co-author network that can be found on Github that was used to build a much larger social network analysis of academic authors. We've sliced the data up and made it work for our needs – we're only going to look at the top co-authors in this network and build a Chord graph accordingly.

The data file master/author-edges_ox-cam.csv has been pre-processed and made available to you so that it is ready to be displayed using Bokeh. Explore the file and notice some of the structure of it so if you ever want to do another on your own, you have a template to work with.

There are only a few short tasks to perform so you can display the graph. You may need to install Bokeh, so try :

```
conda install Bokeh
```

from the command line if you are getting errors.

You will need to use the boilerplate code below to perform the tasks for the assignment:

```
import pandas as pd
from bokeh.charts import output_file, Chord, output_notebook
from bokeh.io import show
from bokeh.sampledata.les_mis import data
output_notebook()

### LOAD THE AUTHOR DATASET PROVIDED INTO df_coauthor ###

chord_from_df = Chord(df_coauthor, source="name_x", target="name_y", value="value")
output_file('co_author_chordgraph_bokeh.html', mode="inline")
show(chord_from_df)
```

§ Load the dataset into a dataframe called `df_coauthor` and run the boilerplate code. Submit your notebook showing the completed chord graph. The labels on the outer circle are author identifiers.

§ **Reduce** the dataset to just those with `df_coauthor.value>50` and re-run the boilerplate code. Submit the notebook showing the new chord graph.

§ What are your observed differences in the two parts? Which seems more valuable?

§ Using the graph from the second part, which pair of authors (identifiers) have the highest number of connections?