MCIS6273 Data Mining (Prof. Maull) / Fall 2018 / HW1

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	Sunday, Oct 21 @ Midnight	*up to* 12 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

- improve on your homework 0 assignment, if necessary
- work with core Pandas concepts for loading data, working with data types, selecting, filtering and representing and plotting data
- understand and use Pearson's correlation to explore the relationships between variables

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hw1`. Put all of your files in that directory. Then zip that directory, rename it with your name as the first part of the filename (e.g. `maull_hw1_files.zip`), then download it to your local machine, then upload the `.zip` to Blackboard.

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 zip in Linux.

ASSIGNMENT TASKS

(70%) work with core Pandas concepts for loading data, working with data types, selecting, filtering and representing and plotting data

Pandas, as we will learn, is a powerful tool that allows for a great deal of data engineering, preliminary analysis and exploration data analysis activities. We will use it in this part of the homework to explore a specific dataset. Denver, Colorado is called the "mile high" city because it is around 5280 feet above sea level. Upon flying into Denver you may hear on the train to the terminal that Denver boasts nearly 300 days of sunshine annually. This claim is widespread among people who live in and around Denver, and while it is very sunny year round in Denver, this claim may be called into question and may vary greatly by the definition of "sunny day".

Being data scientists, we're going to demonstrate how we could begin the process of verifying this claim and put shape to the reality of sunshine in Denver (and perhaps other places).

DATA

The National Oceanic and Atompsheric Agency (NOAA) is a US goverment agency responsible for many data collection and dissemination activities around the weather (both weahter that is normal and extreme). Weather, of course, has been the subject of much observation by humans over the millenia, but weather

data collection activities have become more widespread, consistent and prolific with advances in technology, engineering and science.

Unsurprisingly, data about sunshine is widely collected today. What is quite surprising, however, is that data for sunshine in the US goes back to the late 1800s. We will not try to deconstruct the definition of "sunshine", since that definition has changed somewhat over the years with more consistent data collection tools and methodologies. We're primarily concerned only that the dataset exists and is available to us.

We're going to use Pandas to load the dataset, and analyze it to build an argument with data that demonstrates the reality of sunshine in Denver (and maybe a few other places, just for fun).

§ READ THE ARTICLE "Throwing Shade on Denver's Claim of 300 Days of Sunshine a Year" [Chris Bianchi, 07/23/2018, Westword.com Magazine] AND ANSWER THE FOLLOWING QUESTION:

- What, according to the article, is the definition of a "sunny day"?

§ LOAD THE DATASET

We've learned Pandas is adept at grabbing data from a variety of sources and we've already worked with CSV files, as it is a common file format that is durable, easy to store, compress, transmit and edit. Another common file format (similar to CSV, but more common in older datasets) is what is called a 'fixed width' dataset, that is each column of data resides with a certain range of row widths.

The dataset we will be working with is maintained by NOAA and contains a large number of data

We will be working with a large text file file for this part of the assignment. You will need to first pull the file to your local file system any way you see fit: you can use the terminal in you Lab environment and perform `wget`, you can download the file to your local system and then upload it to your lab environment, etc.

In whatever way you'd like (e.g. curl, wget, `requests`), pull the following dataset to your local filespace:

- Steurer P M; Karl T R (2018): Historical Sunshine and Cloud Data in the United States (revised 1991) (NDP-021). Carbon Dioxide Informational Analysis Center (CDIAC), Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States). doi:10.3334/CDIAC/CLI.NDP021
- You will need the file:
  - `https://cdiac.ucsd.edu/ftp/ndp021r1.f10`
  - it would also be a good idea if you took a look at the `README` to understand these and other files in the CDIAC repository `https://cdiac.ucsd.edu/ftp/ndp021/`

§ PLOT THE MEAN ANNUAL SUNSHINE DATA FOR DENVER, CO OVER THE ENTIRE DATA AVAILABLE

This plot should have the years as the x-axis and the sunshine hours for that year on the y-axis. To complete this part, you will need to explore the `DataFrame.plot()` method.

- Please write your rationale *using the data in your analysis* for the support (or lack of support) for the claim that Denver gets 300 days of sunshine.
- What is the **actual** number of days of sunshine *according to your sunshine hours* analysis (you can simply multiply your calculation by 365)?

§ WRITE A FUNCTION THAT COMPUTES THE PERCENT ANNUAL SUNSHINE FOR A GIVEN STATION

To complete this part, you will need to use a second dataset, which contains the *potential* sunshine hours for a given city. This data will can be found at the following location:

- `https://cdiac.ucsd.edu/ftp/ndp021r1.f11`

Your function will look something like this:

```
def compute_sunshine_hours(station_id):
    # your code goes here

    return # the calculated sunshine hours
```

Remember, all you need is the mean sunshine hours over *all* the data and potential sunshine hours. The calculation is simply

$$\rho_s = \frac{\mu_s}{\tau_s}$$

where μ_s is the mean sunshine hours, and τ_s the potential hours. You will, of course, need to pull the data for potential hours into its own DataFrame.

§ USING THE FUNCTION FROM THE PREVIOUS PART, PLEASE GIVE THE MEAN ANNUAL SUNSHINE HOURS FOR THE FOLLOWING STATIONS

- station #23042 (Lubbock, TX)
- station #25309 (Anchorage, AK)
- station #21504 (Hilo, HI)
- Which of these is most surprising and why?

§ SORT ALL THE STATIONS BY THE TOTAL ANNUAL SUNSHINE HOURS AND SORT BY THE TOP 25 CITIES

- make sure the DataFrame for the sorted top25 is called `sorted_top25`, you will need to use your function from above;
- according to this data, which would be your choice (based on percent sunshine hours): live in **(a)** San Diego or **(b)** Dodge City, KS? Why?

(30%) understand and use Pearson's correlation to explore the relationships between variables

In the prior part we computed the annual sunshine for the cities in our dataset. It should be obvious upon looking at the results why certain places receive more sunshine than others. And eventhough we can eyeball the answer to what the common features are of the cities in the top 10 of the list, we must rely on data and techniques for reasoning about data to derive and guide our conclusions.

We talked in lecture and read in our assigned readings about Pearson's correlation or otherwise known as Pearsons-r. This technique provides us a tool to determine *correlation* between variables – that is to determine the relationships between how variables vary with one another.

Recall, that we said there are positive, negative and no correlations, and that there are strong and weak variations on either side of positive and negative correlation.

In this part, we are going to continue with our investigation of sunshine and explore a variable along side it: precipitation.

NOAA maintains an online API, Climate Data Online, that is free to use and exposes access to a tremendous amount of current and historical observational data. Since CDO doesn't readily have precipitation data for the stations we're interested in, we're going to derive it on our own.

§ GET AN API TOKEN

Get a fast (instantaneous), free API token to use the CDO API (get it here: https://www.ncdc.noaa.gov/cdo-web/token). You should use your `saumag.edu` email address – you will not be able to complete the assignment without this token.

§ COMPUTE THE AVERAGE ANNUAL PRECIPITATION FOR A RANDOM SAMPLE OF 30 STATIONS

CDO provides end users with an endpoint to query historical data. Much of the work you will need for this part is done for you in this Python function:

```
def get_avg_decadal_precipitation(wban="23062", start_decade=1980, token=''):
    import pandas as pd

    headers = {'token': token}
    url = "https://www.ncdc.noaa.gov/cdo-web/api/v2/data?datasetid=GSOY&datatype=PRCP&stationid=GHCND:US

    # get the initial request to know how many pages to look for
    r = requests.get( "{}&limit=1".format(url) , headers=headers)

    data = []
    try:
        if (r.status_code==200):
            result = r.json()
            count = result['metadata']['resultset']['count']

            for offset in range(0, (int)(1.+(count/100))):
                r = requests.get( "{}&limit=100&offset={}".format(url, offset*100) , headers=headers)
                result = r.json()

                for r in result['results']:
                    if r['datatype']=='PRCP':
                        data.append(r)

                print(".", end="")
        else:
            print(r.status_code)
            print(r.text)
    except Exception as e:
        pass # print(e)

    return pd.DataFrame(data).drop(columns=['attributes', 'datatype']).value.describe()['mean']*0.039370
```

- You will need to use the DataFrame.sample(n=30) method on the DataFrame from the first part. With that sample, use the `WBAN` attribute and pass it as a parameter to the function along with your token (e.g. `get_avg_decadal_precipitation(wban=WBAN, token=TOKEN)`). You might also optionally explore the DataFrame.apply method.
- HINT: We are going to develop a dataset of precipation averages from 1940-2010, so this snippet of code may be of use to you:

```
d = [get_avg_decadal_precipitation(start_decade=dec) for dec in range(1940,2010)]
```

```
import statistics
statistics.mean(d)
```

another way to do the same thing with `Series` objects:

```
import pandas as pd
pd.Series(d).describe()['mean']
```

§ FIND THE CORRELATION BETWEEN SUNSHINE HOURS AND PRECIPITATION

- From the previous result, you will need to take the sunshine hours and precipitation and either use the DataFrame.corr() method or the SciPy `scipy.stats.pearson()` method to compute the Pearsons-r.
- Explain in your own words what the correlation is indicating. Is the outcome reasonable? Explain why or why not.