Python Workshop - Part 2

Part 2 - NumPy

Exercise 1.

Find indices of non-zero elements from the following vector: $[1, 2, 0, 0, 4, 0]$.

Exercise 2.

Create a 3×3 identity matrix.

Exercise 3.

Normalize a 5×5 random matrix.

Exercise 4.

Let's examine the following vectors:

$$X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \; Y = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Compute the inner product between X and Y.

Exercise 5.

Given two random arrays, return only the common values between the two arrays.

Exercise 6.

Given two random arrays, how do we know if they are equal? What if we want to know if the values are close enough?

Exercise 7.

Consider a random 10×2 matrix representing cartesian coordinates, convert them to polar coordinates.

Exercise 8.

Using the same 10×2 matrix of Cartesian coordinates from the previous problem, find the point-by-point distances.

Exercise 9.

The `np` function `np.sign` takes an array and returns an array of the same size, but with -1's and 1's to represent the sign of the different elements. Without using the `np.sign` function, how would you do this?

Exercise 10.

Return the diagonal elements of a dot product between two arbitrary 5×5 matrices.

Exercise 11.

Write a function to calculate a moving average using a sliding window over an array. The function should take in an array and the length of the window as inputs.

Exercise 12.

Find the most frequent value in an array. (Hint: the function `np.bincount` may be helpful here.)

Exercise 13.

Consider a one-dimensional simple random walk. The random walk begins at 0 and can take a step forward ($+1$), or step backward (-1), with equal probability. Implement a simple random walk with 1000 steps. First do this using a for-loop, and then write a version without using any for-loops.

Hint: For the second part, it may be helpful to think about what each step of the random walk consists of. You may find the `.cumsum` function useful.

Finally, plot the random walk to visualize. (Use the library `matplotlib.plot` for plotting. Link to documentation.)

Exercise 14.

Once you have done this, compute useful statistics, such as the minimum and maximum values of the walk. Additionally, something we often care about is something known as the first crossing time, which is the step at which the random walk reaches a particular distance away from the origin for the first time. Write a function that will return such this value. (For example, if we care about the value of 10, we want to find the first time the walk exceeds either 10 or -10.)

Hint: it may be helpful to remember that when dealing with booleans, `TRUE` values are represented as larger than `FALSE` values.

Exercise 15.

Simulate 5000 random walks all at once (all of 1000 steps). Calculate the max and min values obtained over all the walks, as well as the average minimum crossing time to 30 or -30. Plot a histogram of the minimum crossing times.

Part 3 - Pandas

Exercise 1.

Consider the following dictionary:

```
import numpy as np
import pandas as pd
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog',
                   'cat', 'snake', 'cat', 'dog', 'dog'],
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no',
'no']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

a. Create a DataFrame from this dictionary `data`, with the index as `labels`.

b. Then display summary of the basic information about the DataFrame and the data.

c. Return the first 3 rows of the DataFrame.

d. Select just the `animal` and `age` columns of the DataFrame.

e. Select only the rows where the number of visits is greater than 3.

f. Select the rows where the age is missing, i.e. is `NaN`.

g. Select the rows where the animal is a cat and the age is less than 3.

h. Calculate the sum of all visits (the total number of visits).

i. Calculate the mean age for each different animal in `df`. (Hint: use `groupby`)

Exercise 2.

Consider the following DataFrame:

```
df = pd.DataFrame({'From_To': ['LoNDon_paris', 'MAdrid_miLAN',
'londON_StockhOlm',
                               'Budapest_PaRis', 'Brussels_londOn'],
              'FlightNumber': [10045, np.nan, 10065, np.nan, 10085],
              'RecentDelays': [[23, 47], [], [24, 43, 87], [13], [67, 32]],
                   'Airline': ['KLM(!)', '<Air France> (12)', '(British Airways.
)',
                               '12. Air France', '"Swiss Air"']})
```

Part of data analysis is understanding how to clean data to get it in a form that you can use.

a. Interpolate the missing values under the `FlightNumber` column. (Hint: use the `interpolate` function.)

b. Split the `From_To column` into two separate columns (`From` and `To`). Split each string on the underscore delimiter `_` to give a new temporary DataFrame with the correct values. Assign the correct column names to this temporary DataFrame.

c. Standardize the strings in the city names so that the first letter is uppercase and rest are lower case (i.e., "MAdrid" should be "Madrid").

Exercise 3.

Pandas also allows you to do cool things like import data from Yahoo Finance.

Here is some example code:

```
import pandas.io.data as web

all_data = {}
for ticker in ['AAPL', 'IBM', 'MSFT', 'GOOG']:
  all_data[ticker] = web.get_data_yahoo(ticker, '1/1/2000', '1/1/2010')

price = DataFrame({tic: data['Adj Close'] for tic, data in all_data.iteritems()})
volume = DataFrame({tic: data['Volume'] for tic, data in all_data.iteritems()})
```

Pick some stocks of your choice and calculate the following.

a. Compute the percent changes of the prices.

b. Calculate the covariance and correlation matrices of these stocks.