hw6

December 26, 2019

1 STATS 507 HW6 Pandas and Matplotlib

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

1.0.1 Problem 1 Warmup: constructing pandas objects (2 points)

In this problem, you will create two simple pandas objects. 1. Create a pandas Series object with indices given by the first 10 letters of the English alphabet and values given by the first 10 primes.

```
In [2]: index = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
        prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
        pd.Series(prime, index)
```

```
Out[2]: A     2
        B     3
        C     5
        D     7
        E    11
        F    13
        G    17
        H    19
        I    23
        J    29
        dtype: int64
```

2. Below is a table that might arise in a genetics experiment. Reconstruct this as a pandas DataFrame.

```
In [3]: arrs = [['goat', 'goat', 'goat', 'goat', 'bird', 'bird', 'bird', 'bird', 'llama', 'llar
                ['A', 'A', 'a', 'a', 'A', 'A', 'a', 'a', 'A', 'A', 'a', 'a'],
                ['A', 'a', 'A', 'a', 'A', 'a', 'A', 'a', 'A', 'a', 'A', 'a']]
        index = pd.MultiIndex.from_arrays(arrs, names = ['animal', 'parent1', 'parent2'])
        df = pd.DataFrame({'score1': [1,2,3,4,5,6,7,8,9,10,11,12],
                           'score2': [2,4,4,6,6,8,8,10,10,12,12,14]}, index = index)
        df
```

Out[3]:

			score1	score2
animal	parent1	parent2		
goat	A	A	1	2
		a	2	4
	a	A	3	4
		a	4	6
bird	A	A	5	6
		a	6	8
	a	A	7	8
		a	8	10
llama	A	A	9	10
		a	10	12
	a	A	11	12
		a	12	14

1.0.2 Problem 2 Working with pandas DataFrames (3 points)

In this problem, you'll get practice working with pandas DataFrames, reading them into and out of memory, changing their contents and performing aggregation operations. For this problem, you'll need to download the celebrated iris data set, available as a .csv file from my website: www-personal.umich.edu/~klevin/teaching/Winter2018/ STATS701/iris.csv Note: for the sake of consistency, please use this version of the CSV, and not one from elsewhere.

1. Download the iris data set from the link above. Please include this file in your submission. Read iris.csv into Python as a pandas DataFrame. Note that the CSV file includes column headers.

```
In [4]: iris = pd.read_csv('iris.csv', sep = ',')
        iris.head()
```

Out[4]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Answer: 1. How many data points are there in this data set? 150 2. What are the data types of the columns? float64 3. What are the column names? The column names correspond to flower species names, as well as four basic measurements one can make of a flower: the width and length of its petals and the width and length of its sepal (the part of the pant that supports and protects the flower itself). sepal length, sepal width, petal length, petal width, species 4. How many species of flower are included in the data? ['setosa', 'versicolor', 'virginica']

```
In [5]: iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
```

```
Sepal.Length    150 non-null float64
Sepal.Width     150 non-null float64
Petal.Length    150 non-null float64
Petal.Width     150 non-null float64
Species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

```
In [6]: iris['Species'].unique()

Out[6]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

2. The data that I uploaded to my website, which you have downloaded, is based on the data initially uploaded to the UC Irvine machine learning repository. It is now known that this data contains errors in two of its rows (see the documentation at https://archive.ics.uci.edu/ml/datasets/Iris).

   Using 1-indexing, these errors are in the 35th and 38th rows. The 35th row should read 4.9, 3.1, 1.5, 0.2,"Irissetosa", where the fourth feature is incorrect as it appears in the file, and the 38th row should read 4.9, 3.6, 1.4, 0.1, "Iris-setosa", where the second and third features are incorrect as they appear in the file. Correct these entries of your DataFrame.

```
In [7]: iris.iloc[34:38,]

Out[7]:     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
        34           4.9          3.1           1.5          0.1  setosa
        35           5.0          3.2           1.2          0.2  setosa
        36           5.5          3.5           1.3          0.2  setosa
        37           4.9          3.1           1.5          0.1  setosa

In [8]: iris.iloc[34,3] = 0.2
        iris.iloc[37,1] = 3.6
        iris.iloc[37,2] = 1.4

In [9]: iris.iloc[34:38,]

Out[9]:     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
        34           4.9          3.1           1.5          0.2  setosa
        35           5.0          3.2           1.2          0.2  setosa
        36           5.5          3.5           1.3          0.2  setosa
        37           4.9          3.6           1.4          0.1  setosa
```

3. The iris dataset is commonly used in machine learning as a proving ground for clustering and classification algorithms. Some researchers have found it useful to use two additional features, called Petal ratio and Sepal ratio, defined as the ratio of the petal length to petal width and the ratio of the sepal length to sepal width, respectively. Add two columns to you DataFrame corresponding to these two new features. Name these columns Petal.Ratio and Sepal.Ratio, respectively.

```
In [10]: iris['Petal.Ratio'] = iris['Petal.Length']/iris['Petal.Width']
         iris['Sepal.Ratio'] = iris['Sepal.Length']/iris['Sepal.Width']
         iris.head()
```

```
Out[10]:    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species  Petal.Ratio  \
         0           5.1          3.5           1.4          0.2  setosa          7.0
         1           4.9          3.0           1.4          0.2  setosa          7.0
         2           4.7          3.2           1.3          0.2  setosa          6.5
         3           4.6          3.1           1.5          0.2  setosa          7.5
         4           5.0          3.6           1.4          0.2  setosa          7.0

            Sepal.Ratio
         0     1.457143
         1     1.633333
         2     1.468750
         3     1.483871
         4     1.388889
```

4. Save your corrected and extended iris DataFrame to a csv file called iris_corrected.csv. Please include this file in your submission.

```
In [11]: iris.to_csv('iris_corrected.csv')
```

5. Use a pandas aggregate operation to determine the mean, median, minimum, maximum and standard deviation of the petal and sepal ratio for each of the three species in the data set.

   Note: you should be able to get all of these numbers in a single table (indeed, in a single line of code) using a well-chosen group-by or aggregate operation.

```
In [12]: iris.groupby('Species').agg({'Petal.Ratio': ['mean', 'median', 'min', 'max', 'std'],
                                      'Sepal.Ratio': ['mean', 'median', 'min', 'max', 'std']})
```

```
Out[12]:            Petal.Ratio                                          Sepal.Ratio  \
                          mean    median       min   max       std        mean
         Species
         setosa       6.908000  7.000000  2.666667  15.0  2.854545    1.470188
         versicolor   3.242837  3.240385  2.666667   4.1  0.312456    2.160402
         virginica    2.780662  2.666667  2.125000   4.0  0.407367    2.230453

                       median       min       max       std
         Species
         setosa      1.463063  1.268293  1.956522  0.118750
         versicolor  2.161290  1.764706  2.818182  0.228658
         virginica   2.169540  1.823529  2.961538  0.246992
```

1.0.3 Problem 3 Plotting Dataframes: Major League Baseball (5 points)

In this problem, you'll get more practice working with pandas data frames and perform some basic plotting. We'll work with a data set consisting of all the baseball

games from the 2018 Major League Baseball (MLB) regular season, compiled by retrosheet.org. Don't worry– you don't need to know anything about baseball to complete this assignment! You can download the relevant CSV file either from the course web page at www.personal.umich.edu/~klevin/teaching/Winter2019/STATS507/mlb2018.zip or directly from the original source at https://www.retrosheet.org/gamelogs/gl2018. zip.

Note: even though the zipped file is named as a .txt file, it is in fact a CSV file, which pandas will still be able to read.

Requisite legal boilerplate: The information used here was obtained free of charge from and is copyrighted by Retrosheet. Interested parties may contact Retrosheet at "www.retrosheet.org".

1. Read the data into a table called mlb_df. Each row of the table represents the outcome of a single game from the 2018 MLB season. Take note that the file does not have columns names; see the header keyword to the pandas.read_csv function. The columns are explained in a .txt file which you can download from https://www.retrosheet.org/gamelogs/glfields.txt, but we will only make use of a few of them in this problem.

   The 10-th and 11-th columns (using 1-indexing) are the scores of the visiting and home teams, respectively. Rename these columns v_score and h_score, respectively. MLB comprises two leagues, the American League and the National League, encoded as AL and NL in the table. The 5-th and 8-th columns (also 1-indexed) are the league affiliations of the visiting and home team, respectively. Rename these columns v_league and h_league.

```
In [13]: league = pd.read_csv('GL2018.TXT', header = None)
         league.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9	...	151
0	20180329	0	Thu	COL	NL	1	ARI	NL	1	2	...	Nick Ahmed
1	20180329	0	Thu	PHI	NL	1	ATL	NL	1	5	...	Dansby Swanson
2	20180329	0	Thu	SFN	NL	1	LAN	NL	1	1	...	Yasmani Grandal
3	20180329	0	Thu	CHN	NL	1	MIA	NL	1	8	...	Miguel Rojas
4	20180329	0	Thu	SLN	NL	1	NYN	NL	1	4	...	Kevin Plawecki

	152	153	154	155	156	157	158	159
0	6	dysoj001	Jarrod Dyson	9	corbp001	Patrick Corbin	1	NaN
1	6	flahr001	Ryan Flaherty	5	tehej001	Julio Teheran	1	NaN
2	2	forsl001	Logan Forsythe	5	kersc001	Clayton Kershaw	1	NaN
3	6	wallc001	Chad Wallach	2	urenj001	Jose Urena	1	NaN
4	2	syndn001	Noah Syndergaard	1	rosaa003	Amed Rosario	6	NaN

	160
0	Y
1	Y
2	Y
3	Y
4	Y

[5 rows x 161 columns]

```
In [14]: league = league.rename(columns = {9: 'v_score', 10: 'h_score',4: 'v_league', 7: 'h_lea
```

```
In [15]: league[['v_score', 'h_score', 'v_league', 'h_league']].head()
```

```
Out[15]:    v_score  h_score v_league h_league
         0        2        8       NL       NL
         1        5        8       NL       NL
         2        1        0       NL       NL
         3        8        4       NL       NL
         4        4        9       NL       NL
```

2. Create a plot with two subplots, placed side-by-side. Each subplot should be a scatter plot in which the x- and y-axes correspond to the home and visitor scores, respectively, and in which each point corresponds to a game from the season.

   In the left-hand plot, include all games in which both teams were in the NL, and in the right-hand plot, include all games in which both teams were in the AL. Games in which the teams were from different leagues should be ignored.

   Specify the transparency (cf. the alpha parameter in the matplotlib documentation) so that scores that occur more often will be shaded darker than rare scores.

   Color the points in the scatter plot according to the league affiliation of the two teams as follows: games between two teams both in the AL should be rendered as 'red' points in the scatter plot. Games between two teams both in the NL should be rendered as 'blue' points in the scatter plot.

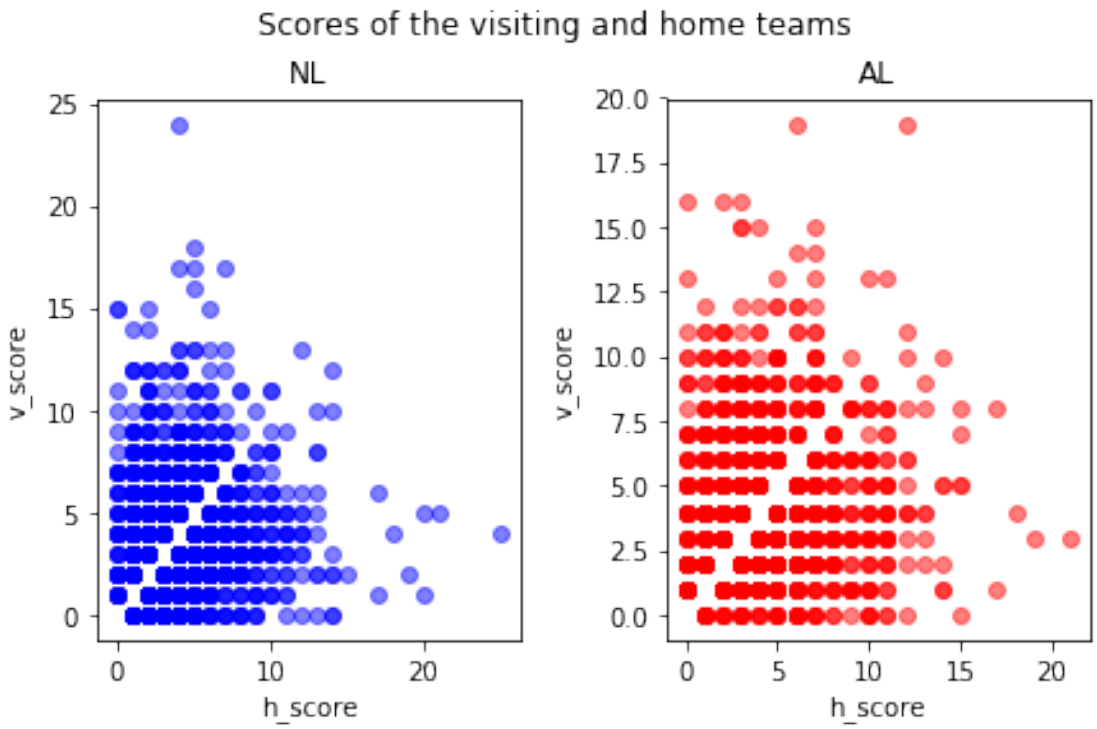
   Label your axes and provide an appropriate title for your plot as well as its subplots.
   Note: you may find it useful to create an extra column in the data frame encoding whether a given game is AL vs AL, NL vs NL or mixed.

```
In [16]: left = league[['v_score', 'h_score', 'v_league', 'h_league']][(league['v_league'] ==
         right = league[['v_score', 'h_score', 'v_league', 'h_league']][(league['v_league'] ==
```

```
In [17]: fig, axs = plt.subplots(1, 2, constrained_layout=True)
         axs[0].scatter(left['h_score'], left['v_score'], color = 'blue', alpha = 0.5)
         axs[0].set_title('NL')
         axs[0].set_xlabel('h_score')
         axs[0].set_ylabel('v_score')
         fig.suptitle('Scores of the visiting and home teams')

         axs[1].scatter(right['h_score'], right['v_score'], color = 'red', alpha = 0.5)
         axs[1].set_title('AL')
         axs[1].set_xlabel('h_score')
         axs[1].set_ylabel('v_score')
```

```
Out[17]: Text(0, 0.5, 'v_score')
```

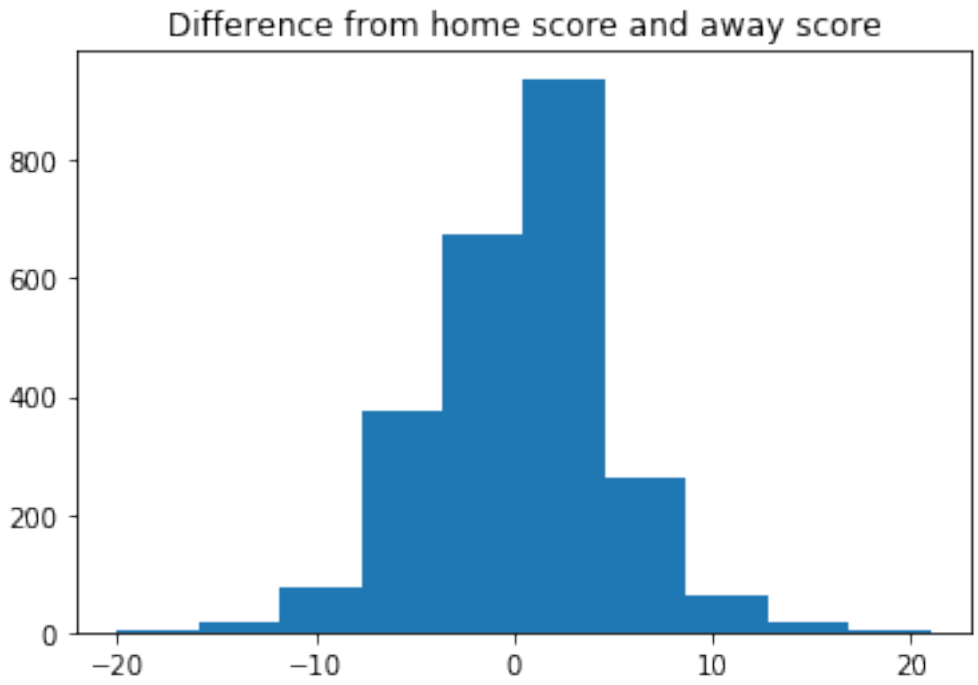


3. The Skellam distribution (https://en.wikipedia.org/wiki/Skellam_distribution) is the distribution that results from taking the difference between two Poisson random variables. It is often suggested as a model for the difference between scores in sports games, particularly baseball. Add a new column to the data frame called score_diff, given by the home score minus the away score. Make a histogram of this score difference and give the plot an appropriate title.

```
In [18]: league['score_diff'] = league['h_score'] - league['v_score']

         plt.hist(league['score_diff'])
         plt.title('Difference from home score and away score')

Out[18]: Text(0.5, 1.0, 'Difference from home score and away score')
```



4. Read the documentation about the scipy implementation of the Skellam distribution at https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.skellam.html. If $_H$ and $_V$ are the means of two independent Poisson random variables K_H and K_V , respectively, then the Skellam distribution that describes the difference $K_H K_V$ has parameters $_H$ and $_V$.

   We will assume (perhaps incorrectly) that the location parameter of the Skellam distribution is 0. To fit a Skellam distribution to the data, we will first fit Poisson distributions to the home and away teams.

   Estimate parameters $\hat{}_H$ and $\hat{}_V$ as the means of the home and visitor scores, respectively.

   Use scipy to run a Kolmogorov-Smirnov test assessing whether or not the Skellam distribution with parameters $(_1, _2) = (\hat{}_H, \hat{}_V)$ and location parameter 0 is a good fit for the score differences.

   Hint: see the documentation at https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.h to see how to perform such a test. Is the Skellam distribution a reasonable model to use? What might we do to build a more accurate model?

```
In [19]: from scipy.stats import skellam
         from scipy import stats
         lambda_h = league['h_score'].mean()
         lambda_v = league['v_score'].mean()
```

```
In [20]: # mu1, mu2 = 4.526, 4.372
         mu1 = lambda_h
         mu2 = lambda_v
         # mean, var, skew, kurt = skellam.stats(mu1, mu2, location = 1, moments='mvsk')
         rvs = skellam.rvs(mu1, mu2, size=len(league['score_diff']))
```

```
In [21]: stats.kstest(league['score_diff'], lambda x: skellam.cdf(x, mu1, mu2, loc=0))
```

```
Out[21]: KstestResult(statistic=0.2052406383420739, pvalue=2.827215852687608e-90)
```

Since $p < 0.05$, skellam distribution is not a good fit for this dataset.