Homework 6: `pandas`

Due October 31, 11:59 pm

Worth 15 points

Read this first. A few things to bring to your attention:

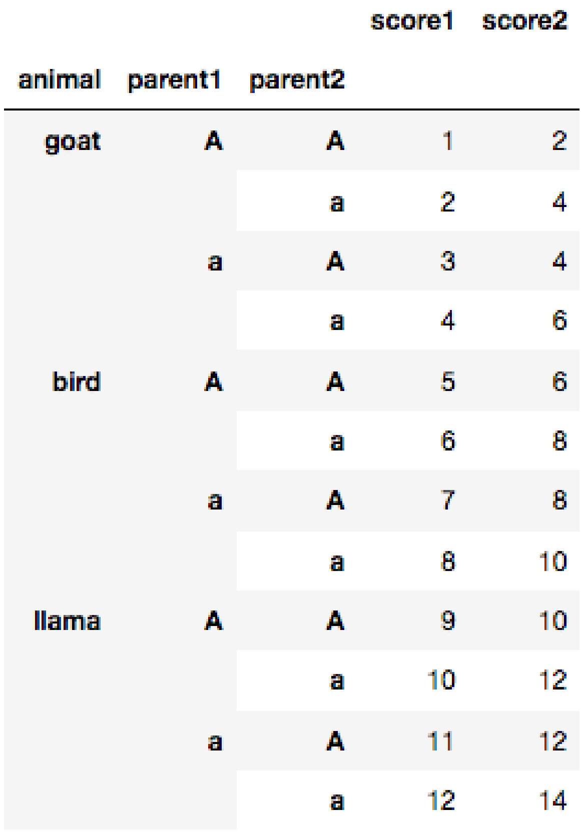
1. Start early! If you run into trouble installing things or importing packages, it's best to find those problems well in advance so we can help you.
2. **Make sure you back up your work!** I recommend, at a minimum, doing your work in a Dropbox folder or, better yet, using `git`.
3. **A note on grading:** overly complicated solutions or solutions that suggest an incomplete grasp of key concepts from lecture will not receive full credit.

Instructions on writing and submitting your homework can be found at `http://www-personal.umich.edu/~klevin/teaching/Fall2019/STATS507/hw_instructions.html`. Failure to follow these instructions will result in lost points. Please direct any questions to either the instructor or your GSI.

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. Assign this object to a variable called `alphaprimes`.
2. Below is a table that might arise in a genetics experiment. Reconstruct this as a `pandas` DataFrame and assign it to a variable called `animals`.



			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

2 Working with `pandas` DataFrames (5 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/Fall2019/STATS507/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. How many data points are there in this data set? What are the data types of the columns? 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). How many species of flower are included in the data?

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,"setosa"`, 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,"setosa"`, where the second and third features are incorrect as they appear in the file. Correct these entries of your DataFrame.

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 your DataFrame corresponding to these two new features. Name these columns `Petal.Ratio` and `Sepal.Ratio`, respectively.

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

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 five numbers in a single table (indeed, in a single line of code) using a well-chosen group-by or aggregate operation.

3 Plotting Dataframes: Major League Baseball (8 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/Fall2019/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`.

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. Save your plot in a pdf file called `home_vs_away_scores.pdf` and include it in your submission. **Note:** you may find it useful to create an extra column in the encoding whether a given game is AL vs AL, NL vs NL or mixed.

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.

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 λ_H and λ_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 λ_H and λ_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 $\hat{\lambda}_H$ and $\hat{\lambda}_V$ as the means of the home and visitor scores, respectively. Save your estimates in variables `lambda_home` and `lambda_visitor`, respectively.

5. Now let's run a goodness-of-fit test to see how well the Skellam distribution matches our data. There are several ways to do this, but we will use the χ^2 test, which is available in `scipy` as `scipy.stats.chisquare` (documentation here: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html`). There are two problems we must address. First, the Skellam distribution predicts that sometimes $K_H = K_V$, which cannot happen in our data because ties are not allowed in baseball (play continues until a team wins `https://en.wikipedia.org/wiki/Longest_professional_baseball_game`), so our model should really be the Skellam distribution conditioned on the fact that $K_H \neq K_V$. Second, the χ^2 test as implemented in `scipy` requires that the observed and expected frequencies be finite-length arrays, but the Skellam distribution gives non-zero probability to every possible integer value. To account for these two issues, we will model the score difference as a Skellam distribution conditioned on the event that $K_H \neq K_V$ and $|K_H - K_V| \leq 30$ (36 is the most runs ever scored by one team in a baseball game, so 30 is a reasonable upper bound for our purposes). Use the `scipy` implementation of the χ^2 test to assess the fit of our model to the data (use the default delta degrees of freedom, `ddof=0`). Save the resulting p-value and the test statistic in variables named `chi2pvalue` and `chi2stat`, respectively. Does our model agree with the data? What changes might we consider to improve the fit of our model (a few sentences will suffice here— no need to implement any of these improvements unless you really want to)?