Jeopardy Assignment

[ 100 points - due by 11:59 pm, Sunday, ]

Submit these files to the CS submission system at the usual place by 11:59. You may work on your own or with 1-2 partners on the programming portions of this assignment. (The reading/response is individual only.) Groups larger than 3, please split into smaller groups! Remember that partners need to work in the same physical location, share composition time equally (or each compose on their own machines) and be fully equal owners and producers of their work. *Have fun experimenting!*  [cs35 homepage](https://www.cs.hmc.edu/~dodds/cs35/)

**Downloads**

Here is the starter file: <https://drive.google.com/open?id=1xAfLTmkD-YKOpOd38H2u7dOb3aMgKzfd>

Here is the dataset you’ll be working with:

<https://drive.google.com/open?id=1NKEMspc-OY8G-5OMCeCz8UOA_oA_Bwj7>

**Submission**

Again we ask you to submit all of your files at once (much smoother grading -- thank you!) GradeScope allows a *"select-all-and-upload"* which is easiest (a zip archive is ok, too). This week we ask for at least these files:

hw11pr0

jeopardystarter.py

This week, please submit your reading response, hw11pr0.txt, in its own spot at the [submission site](http://cicero.cs.hmc.edu/).

**[Reading] Problem 0**:[5 pts]

If you have followed Jeopardy, James Holzhauer is one of the new Jeopardy Giants, with the highest one day winning record of $131,127 and a 32-game streak that was about $50,000 shy of beating Ken Jenning’s record of the most money won in non-tournament games. In total over 33 games, James won $2,464,216.

Here is the master’s thesis of the woman who beat him just before he could overtake Ken’s record. Emma Boettcher wrote a master’s thesis about Jeopardy questions and their level of difficulty and used some machine learning. Certainly, you do not have to read all of it, but skim over it and write about what interests you in her work with machine learning.

<https://cdr.lib.unc.edu/concern/parent/tb09j9193/file_sets/bz60d095h>

As with each week's reading, responses should carefully considered, but need not be very long (4-5 sentences is wonderful). Submit **hw3pr0.txt** in the usual way on GradeScope (it's separate from the Python portion).

**Problem 1:** [71 pts]

* This problem asks you to write all of your code in the **jeopardystarter.py** file.
* In addition, be sure to submit three screenshot files: one of the bar graph, pie graph, and line graph.

This assignment integrates creating csv files, matplotlib visualizations, and natural language processing.

* Warning: running on the whole 319K+ rows file on the jeopardy() function can have a runtime of 4+ minutes. Unfortunately, this dataset does not include James Holzhauer’s run :(. We suggest you try on a portion of the dataset while you still write your code. However, at the end, when you make your visualizations and give your text-based answers, these must be based on the whole dataset, so we can grade it correctly.
* Preliminary stuff: [5 points]
* Make 2 new csv files, one for categories and finaljeopardy categories. [2 points]
* Above the matplotlib visualizations, please write in hashtagged comments the most asked category, the most asked final jeopardy category, and the most asked generalized jeopardy category. The general categories will already be provided in the starter code. [3 points]
* We’re using general categories or else there would be too many different bars or slices in a pieplot. Make sure to exclude the “other” category, since this doesn’t give very useful information. It is pretty hard to generalize the tens of thousands of Jeopardy categories.

Plotting: [26 points]

* Make sure to use the xkcd format for all the graphs.
* Create a **barplot** for general categories with their respective frequencies, **labels for the x and y axes**, and a **title**. Give the barplot some color and **edgecolor**. [5 points]
* Create a **pieplot** with labels for each general category, each slice being a different **color** and listing its **percentage**. Have the **starting angle be from 90 degrees** on a unit circle and have the most asked general category **explode** out a little. Make the plot colorful! [5 points]
* Create a **line** graph with a **title**, **legend**, and **labels for each question number and money winnings** for the **3 different players**. Another way you can do this is **create another csv file with just data from 1 game and make the same line graphs** [8 points total]
  + ex: Game 7973 <http://www.j-archive.com/showscores.php?game_id=6266>
  + Make sure to account for when there are **negative money values**.
  + First, create a **spreadsheet with the table data** and download it as a csv file, and then work with it. Another way to do the same thing is just make a new csv file with only the data from one game, which is given by the show’s date.
  + Make a line graph tracking each player’s Coryat score and your own if you try playing along. [8 points]
* Natural Language Processing: the appropriate libraries will be imported in the starter code. [40 points] 8 Functions
  + Using the statistics library, write a function that
    - Returns the average length of a clue by how many words are in the clue and prints its median, mode, and standard deviation. Do the same for the average length of an answer to a clue. [12 points]
      * Create both of the same functions but have it take an argument rownumber so that you find the average length for a specific row. Only have it print the median.
    - Returns the average length of words (how many letters in each word) in a clue and prints its median, mode, standard deviation, and variance. Do the same for the average length of words to a clue. [8 points]
      * Create both of the same functions but have it take an argument rownumber so that you find the average length for a specific row.
  + Returns the shortest and longest clues. Do the same for the shortest and longest answers. [4 points]
  + Returns the number of a specific character in a specific clue. This function should take an argument character and argument rownumber. [4 points]
  + Make a matplotlib graph that relates the results of one of these functions with each row (clue). (This would probably be a scatterplot.) [5 points]
  + Find where daily doubles are most commonly found by checking the clue location. Note that the first number is the x coordinate and the second number is the y coordinate, where J\_1\_3 would be the first column and the third-to-last row on the bottom, like in Cartesian coordinates. Make sure to make a deepcopy of the list of all clue locations, reverse the list to show the most common locations of daily doubles in descending order, and make another list recording the indices of where the daily doubles in descending order are according to the indices of the original list. [5 points]
    - Make a heatmap based off this data through matplotlib. Have x and y labels, a title, and a color legend. The x-axis labels should be rotated 45 degrees. [4 points]
  + Find which month tends to have the highest and lowest average and median winnings. [5 points]
  + Find which players’ first names tends to win most. Do this and divide by how common that player’s name is. [ points]
  + Find which questions are hardest by finding which questions have the most -1.0’s for contestant responses, and find which questions contestants did not answer at all. AKA Triple Stumpers. Have each function print the number of triple stumpers and return the list of all triple stumper questions. [10 points]
  + Find how many total ties there are in the game and by printing the number and returning the tied amount of money. Do the same function for those tying for first place. [10 points]
  + Create a heatmap for the average value of each square, including its regular label (whether people won or lost on that square +/-) and how much people bet on it for daily doubles. Normalize for each square.
  + Make a function that prints out extreme statistics for winnings, like the lowest and highest winning and losing scores. [5 points]
    - There’s redundant data for clue location, so make another csv file that gets rid of the separate x and y coordinate data
  + **#Create a simulator that predicts how well you would do on each category [5 points]**
  + **For fun:** Run a function that guesses a new clue/answer in typical Jeopardy style. Most of the code will be given in the starter code. You must make a new csv file that only includes the Jeopardy clues. Also, make a csv file that only includes the clue locations of the daily doubles. Then, create a text file from that csv file by using the csv\_to\_txt() function that's given later in this script. In a comment, put what your function predicted.

**Problem 2:** [14 pts]

Towards the bottom, there are 7 questions about betting strategies for different Final Jeopardy situations. Answer them in the file between the triple quotes or hashtagged comments.