Programming Assignment 5

Getting started

Review class handouts and examples, complete the reading and practice assignments posted on the course schedule. This assignment is designed to perform data analysis tasks using Pandas.

Programming Project: Recommend worth: 25 points

Rating-based movie recommendation.

Data and program overview

In this assignment you will be working with data on movies and people's ratings of these movies. The task will be to create movie recommendations for a person, based on the person's and critics' ratings of the movies. The following data will be provided using csv files:

- A table with movie information (*IMDB.csv*); we will call this the *movies data.*
- A table with ratings of all movies listed in the movies data, by 100 critics (*ratings.csv*); let's call this the *critics data.* The column names in the critics data correspond to the name of each critic.
- A table with one person's ratings of a subset of the movies in the movies data set (*pX.csv*), the *person data,* where X is a number*.* The column name in the file indicates the name of the person.

All personal ratings are integer numbers in the 1..10 range.

I provide two data sets for this assignment, in zip files called *data.zip* and *data-tiny.zip*. Download and unzip the files in your project folder. Unzipping should result in two folders added to your project folder: *data* and *data-tiny*. You must review the data files to familiarize yourself with their content and structure.

The program that you write must work as follows.

1. Ask the user to specify the subfolder in the current working directory, where the files are stored, along with the names of the critics, person and movies data files.
2. Determine and output the names of three critics, whose ratings of the movies are closest to the person's ratings based on the Euclidean distance metric (described later).
3. Use the ratings by the critics identified in item 2 to determine which movies to recommend. Display information about recommended movies as described below and illustrated by the sample interactions below.
    a. The movie recommendations must consist of the top-rated movies in each movie genre, based on the average ratings of movies by the three critics identified in step 2 above.
    b. Movie genre is determined by the `Genre1` column of the movies data.
    c. Recommendations must be listed in alphabetical order by genre.
    d. Missing data (e.g. running time) should not be included.

The sample interactions below demonstrate the running of the program.

Sample interactions

First, let's use the tiny data set:

```
Please enter the name of the folder with files, the name of movies file,
the name of critics file,  the name of personal ratings file, separated by spaces:
data-tiny  tinyIMDB.csv tinyratings.csv tinyp.csv

['Aldbridge', 'Benris', 'Whitelea']

Recommendations for Kimberwick:
"127 Hours"  (Adventure), rating: 8.67, 2010, runs 94 min
"About Time" (Comedy), rating: 8.0, 2013, runs 123 min
```

The next interaction shows the output given the larger data set

```
Please enter the name of the folder with files, the name of movies file,
the name of critics file, the name of personal ratings file, separated by spaces:
data  IMDB.csv ratings.csv   p8.csv

['Quartermaine', 'Arvon', 'Merrison']

Recommendations for Catulpa:
"Star Wars: The Force Awakens"    (Action), rating: 9.67, 2015, runs 136 min
"The Grand Budapest Hotel"        (Adventure), rating: 9.0, 2014, runs 99 min
"The Martian"                     (Adventure), rating: 9.0, 2015, runs 144 min
"How to Train Your Dragon"        (Animation), rating: 9.67, 2010
"Kubo and the Two Strings"        (Animation), rating: 9.67, 2016
"Hacksaw Ridge"                   (Biography), rating: 9.33, 2016, runs 139 min
"What We Do in the Shadows"       (Comedy), rating: 9.0, 2014
"Prisoners"                       (Crime), rating: 8.33, 2013, runs 153 min
"Spotlight"                       (Crime), rating: 8.33, 2015, runs 128 min
"The Perks of Being a Wallflower" (Drama), rating: 9.67, 2012, runs 102 min
"Shutter Island"                  (Mystery), rating: 8.33, 2010, runs 138 min
```

Note that in the above interaction there are sometimes more than one movie listed per genre. As, for instance, is the case with the two Adventure movies, both of them had the highest average rating, hence both are included in the list.

Important Notes and Requirements

In addition to the requirements stated so far, your code must satisfy the following to gain full credit:

- Your program should not use any global variables and should have no code outside of function definitions, except for a single call to `main`.
- All file related operations should use device-independent handling of paths (use `os.getcwd()` and `os.path.join()` functions to create paths, instead of hardcoding them).
- You must define and use functions specified below in the **Required Functions** section. You may and should define other methods as appropriate.
- You should use the `pandas` data structures effectively to efficiently achieve the goals of your functions and programs.
- The formatting of the recommendation printout should use the length of the longest movie in formatting the output in a way that aligns categories

Required Functions

You **must** define and use the following functions, plus define and use others as you see fit:

a. Function `findClosestCritics()` with two parameters of type DataFrame, the first one providing data about critics ratings, and the second – about personal ratings. The function should return a list of three critics, whose ratings of movies are *most similar* to those provided in the personal ratings data, based on *Euclidean distance.*

   Euclidean distance of two vectors x = (x_1, x_2, … ,x_n) and y = (y_1, y_2, … ,y_n) is computed as $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$. To compute how similar ratings of a critic are to the ratings of the person, we compute the distance between a vector, in which the coordinates are the *critic's* ratings of each movie, and the vector composed of the *person's* ratings. The lower the distance, the closer, thus more similar, the critic's ratings are to the person's.

   For example, if the personal data included three ratings $(4, 7, 6)$, where the critic rated **the same movies** as $(4, 5, 6)$, the Euclidean distance would equal $\sqrt{(4-4)^2 + (5-7)^2 + (6-6)^2} = 2.0$.

b. Function `recommendMovies ()` with four parameters: the critics and personal ratings data frames, the list of three critics most similar to the person, and the movie data frame. The function should determine out of the set of movies that are *not rated* in the personal data, but are *rated* by the critics, which movies have the highest average of the rating by the most similar critics in each movie genre (specified by the `Genre1` column of movie data). In other words, you need to compute the top-rated unwatched movies in each genre category, based on the average of the three critics' ratings.

   You may assume that the critics data will always be complete, i.e. will include ratings of all movies.

   The function must then (a) put together information about these top-rated movies sufficient to produce the printout, showing the details of each of the recommended movies as illustrated by the interactions, and (b) return it using some data structure of your choice.

c. Function `printRecommendations ()` with two parameters: the first containing information about the recommended movies, and the second – the name of the person, for whom the recommendation is made (the name is specified in the header of the personal ratings data file). The function must produce a printout of all recommendations passed in via the first parameter, in alphabetical order by the genre, as shown in the sample interactions. Make sure to examine the sample interactions and implement the details of the printout. The function should return no value.

d. Function main(), which will be called to start the program and works according to your design.

Hints

- Keeping your data frames indexed by the title should help in making joins easy. Note that the title can be both an index and one of the columns, if necessary.
- For some csv data, you may not need all of the columns. You can specify which columns to import into a data frame, or you could drop unnecessary ones to improve performance and simplify testing and debugging.

- Although I have provided a sample small data sets, for testing purposes, I encourage you to create your own one, for which you should know the result in advance.

Grading
The grading schema for this project is roughly as follows:

Five points will be awarded for the correct implementation of each of the four functions above (which may call other functions that you define), which uses data structures, methods and functions of the `pandas/numpy` package appropriately and effectively.

Three points will be awarded for making the code sufficiently general to handle different input files, i.e. not tied to the specific content of the files that you are given (though it might be somewhat dependent on the *structure* of those files, i.e. what is provided by the columns, rows, etc.)

Two points will be awarded for style, as defined by the guidelines in Handout 1.

Created by Tamara Babaian on November 13, 2018