STAT 29000 Project 1

Topics: python3

Motivation: Python is an interpreted programming language that is known for its simplicity, readability, and small learning curve. Python is one of the most popular programming languages used today. It is extremely versatile, and used by all of the "top" technology companies (google, facebook, microsoft, apple, netflix, etc.). Learning to use Python will help prepare you for working at a company who tends to use Python more often than R.

Context: We have used and become very familiar with R and various UNIX tools. We will now begin to learn and gain experience using Python to solve data-driven problems.

Scope: Python is an interpreted programming language known for its simplicity, readability, and small learning curve. It is one of the most popular tools for data analysis, scientific computing, and machine learning.

To work with Jupyter notebook on scholar, please read this.

A good resource for a summary of useful techniques for working in Jupyter Notebooks can be found here.

A good resource for a brief introduction to Python 3 can be found here. Please note that in The Data Mine we will focus only on Python 3.

First, we must install a missing package called block-timer. In scholar, open up a shell and type the following:

```
python3.6 -m pip install block-timer --user
```

For the following questions, please copy and paste the following code at the top of your Jupyter notebook. We are simply importing useful Python modules that are required for this project. For now, think of these statements as the `library()` function in R. We will address this in more detail at a later date. Be sure to replace "PURDUEALIAS" with your Purdue username. Note that if you get an error after running this chunk of code, you may need to click on Kernel->Restart.

```
import sys
sys.path.append("/home/PURDUEALIAS/.local/lib/python3.6/site-packages")
import csv
import requests
from block_timer.timer import Timer
from collections import defaultdict
```

The `/class/datamine/data/spring2020/stat29000project01examples.ipynb` notebook will provide you with useful examples that will help you solve problems given in this project.

Use the template found here or on scholar: `/class/datamine/data/spring2020/stat29000project01template.ipynb` to submit your solutions.

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms that will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online.

Question 1: reading data, if statement, for loop

1a. *(2 pts)* Use the csv module to read and print the csv header and every tenth line (i.e., 1st line after the header, 11th line after the header, . . .) of the csv file located at https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/books.csv.

We've shown how to read and write files locally in the example notebook, do the following to get the file from the link:

```python
# open a file from a link
file = requests.get("https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/books.csv").text

# strip whitespace and split line by line
file = file.strip().split('\n')

# file above now contains the equivalent of file shown below
# with open("/path/to/our.csv") as file:
    # use file...
```

Keywords: *csv, modulo operation, if statement, csv.reader(), print, for loop, enumerate*

1b. *(1 pts)* Modify the solution to (1a) to now print every line (except the header, use the keywords provided to learn how to skip this line) and print only the `original_title`, and `average_rating`. An example of the desired format would be: "The Hunger Games: 4.34" or "Harry Potter and the Philosopher's Stone: 4.44".

Hint: To avoid re-downloading the entire file, just refresh the reader: reader = csv.reader(file)

Keywords: *f-Strings, next function*

Question 2: for loop, lists, tuples, & sets

2a. *(1 pts)* In Python, a collection of values could be stored in a list, a tuple, or a set. Use your knowledge about these structures to create a grouping of (around 3-5) keywords called `keywords`. You will search for the keywords of your choice in book titles. You can use lists, tuples, and sets as you may need. You do not have to use them all. Print the header and every line where the keyword occurs within the title of the book. Note that if a keyword appears in the title as a part of a larger word, this counts. For example if a keyword is "fun" and a title contains "Fundamentals", we are going to let that count. What did you use to store your keywords?

Hint: We don't care about what case (upper/lower) the keywords appear in the titles. So if you had a keyword "harry", the Harry Potter books should be included in your output.

Keywords: *'in' keyword, string methods*

2b. *(1 pts)* Cycle through our data and store three subsets of the data containing only `authors`. Call one subset `good_books` and store data where the `average_rating` is greater than or equal to 4.02. Call the other subset `ehh_books` and store data where the `average_rating` is strictly less than 4.02. Call the last subset `books` and store the column `authors` regardless of the `average_rating`.

Hint: You will want to skip the header row.

Keywords: *float*

The following two problems are the same. One of our solutions will use loops, and the other will not. Although in this particular case, looping is not a good solution due to the slow quadratic runtime complexity, in general, looping is not something to avoid like in R where we want to take advantage of vectorized solutions.

2c. *(1 pts)* Find every author that has written an ehh_book and a good_book. Find every author who has written an ehh_book but not a good_book. Note for the sake of simplicity, there is no need to modify the `authors` column at all. If there is a situation where Bob wrote his own book, and also collaborated with Susan on a book, we will consider those unique authors. How many authors wrote both and ehh_book and a good_book? How many authors wrote an ehh_book but not a good_book?

Keywords: *list append, len*

2d. *(1 pts)* Given the `ehh_books`, `good_books`, and `books`, solve (2c) without using a single loop.

Hint: Google the topics of the question, one of them will help you here.

2e. (optional) Between problems (2c) and (2d), and the examples provided, you can start to see how in certain situations this technique could be very useful! Let's quickly explore how much faster (2d) was than (2c). Put your solutions to (2c) and (2d) in the code chunk below to see the time difference. Imagine how this could become even more out of hand given a larger dataset. In many situations, a solution that has quadratic runtime complexity would be considered unacceptable (imagine 1 billion lines instead of 10000.

```python
with Timer(title="2c"):
    # insert 2c solution here

with Timer(title="2d"):
    # insert 2d solution here
```

Question 3: dicts, comprehensions

3a. *(1 pts)* Modify question (2c) to use comprehensions rather than loops. In your opinion which is more readable?

Keywords: *list comprehension*

3b. *(1 pts)* Let's assume that the higher the quantity of ratings a book has, the more popular it is. Use dicts to sum the total number of ratings (`ratings_count`) by `language_code`. What are the top 3 most popular languages? Consider all language codes with "en" as English, and don't count blank language codes.

Keywords: *defaultdict, int, sorted, dict methods*

3c. *(1 pts)* Modify (1b)'s solution to maintain the same functionality, but include `authors` as well. If there are multiple listed authors, make sure they are separated with the word "and". Example output includes: "Harry Potter and the Philosopher's Stone by J.K. Rowling and Mary GrandPré: 4.44".

Keywords: *string methods: strip, split, join*

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/qvu-vdHk using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .ipynb and .html formats.