Problem Set 1

Due Wednesday Sep. 6, 10 am

Comments

- This covers material in Units 2 and 4 as well as practice with Quarto.
- It's due at 10 am (Pacific) on September 6, both submitted as a PDF to Gradescope as well as committed to your GitHub repository.
- Please note my comments in the syllabus about when to ask for help and about working together. **In particular, please give the names of any other students that you worked with on the problem set and indicate in the text or in code comments any specific ideas or code you borrowed from another student or any online reference (including ChatGPT or the like).**

Formatting requirements

1. Your electronic solution should be in the form of an Quarto file named ps1.qmd, with Python code chunks included in the file. Please see the Lab 1 and the dynamic documents tutorial for more information on how to do this.
2. Your PDF submission should be the PDF produced from your qmd. Your GitHub submission should include the qmd file, any Python code files containing chunks that you read into your qmd file, and the final PDF, all named according to the submission guidelines.
3. Your solution should not just be code - you should have text describing how you approached the problem and what the various steps were. Your code should have comments indicating what each function or block of code does, and for any lines of code or code constructs that may be hard to understand, a comment indicating what that code does.
4. You do not need to (and should not) show exhaustive output, but in general you should show short examples of what your code does to demonstrate its functionality. Please see the grading rubric, and note that the output should be produced as a result of the code chunks being run during the rendering process, not by copy-pasting of output from running the code separately.

Problems

1. Please read these lecture notes about how computers work, used in a class on statistical computing at CMU. Briefly (a few sentences) describe the difference between disk and memory based on that reference and/or other resources you find.

2. This problem uses the ideas and tools in Unit 2, Sections 1-3 to explore approaches to reading and writing data from files and to consider file sizes in ASCII plain text vs. binary formats in light of the fact that numbers are (generally) stored as 8 bytes per number in binary formats.

   a. Generate a numpy array (named `x`) of random numbers from a standard normal distribution with 10 columns and as many rows as needed that the data takes up about 100 Mb in size. As part of your answer, show the arithmetic (formatted using LaTeX math syntax) you did to determine the number of rows.

   b. Explain the sizes of the two files created below. In discussing the CSV text file, how many characters do you expect to be in the file (i.e., you should be able to estimate this very accurately from first principles without using `wc` or any explicit program that counts characters). Hint: what do we know about numbers drawn from a standard normal distribution?

```
import os
import pandas as pd
x = x.round(decimals = 10)

pd.DataFrame(x).to_csv('x.csv', header = False, index = False)
print(f"{str(os.path.getsize('x.csv')/1e6)} MB")

pd.DataFrame(x).to_pickle('x.pkl', compression = None)
print(f"{str(os.path.getsize('x.pkl')/1e6)} MB")
```

```
167.36151 MB
```

```
100.000572 MB
```

   Suppose we had rounded each number to three decimal places. Would using CSV have saved disk space relative to the pickle file?

   c. Now consider saving out the numbers one row per number in a CSV file. Given we no longer have to save all the commas, why is the file size unchanged?

   d. Read the CSV file into Python using `pandas.read_csv`. Compare the speed of reading with and without providing the `dtype` argument and using the `python` vs `c` engines. Repeat the timing of your first attempt (without `dtype` and with the default engine) a few times. In some cases you might find that the first time is slower; if so this has to do with the operating system caching the file in memory (we'll discuss this further in Unit 8).

   e. Finally, let's consider reading the CSV file in chunks as discussed in Unit 2. Time how long it takes to read the first 100,000 rows.

   f. Now experiment with the `skiprows` to see if you can read in a large chunk of data from the middle of the file as quickly as the same size chunk from the start of the file. What does this indicate regarding whether Pandas/Python has to read in all the data up to the point where the chunk in the middle starts or can skip over it in some fashion? Is there any savings relative to reading all the initial rows and the chunk in the middle all at once?

   g. Now read the data sequentially in equal-sized chunks and determine if reading in the large

chunk in the middle (after having already read the earlier chunks) takes the same amount of time as it did in part (e). Comment on what you've learned.

3. Please read Section 1 of Unit 4 on good programming/project practices and incorporate what you've learned from that reading into your solution for Problem 4. (You can skip the section on Assertions and Testing, as we'll cover that in Lab.) As your response to this question, briefly (a few sentences) note what you did in your code for Problem 4 that reflects what you read. Please also note anything in Unit 4 that you disagree with, if you have a different stylistic perspective.

4. We'll experiment with webscraping and manipulating HTML by getting song lyrics from the web. Go to http://mldb.org/search and (in the search bar in the middle, not at the left) enter the name of a song and choose to search by 'Title' and 'All words'. In some cases the search goes directly to the lyrics of the song (presumably when there is no ambiguity) and in others it goes to a table of potential songs with that or similar name. (For example, compare 'Dance in the Dark' (or 'Dancing in the Dark') to 'Leaving Las Vegas'.)

   a. Based on the GET request being sent to the MLDb server (in the cases like 'Dance in the Dark' where you get a table back rather than a single song's lyrics), determine how to programmatically search for a song by 'Title' and 'All words' using Python, based on our explorations in Unit 2. Side question: what does the *si* parameter control?

      Warning: It's possible that if you repeatedly query the site too quickly, it will start returning "503" errors because it detects automated usage (see problem 5 below). So, if you are going to run code from a script such that multiple queries would get done in quick succession, please put something like `time.sleep(2)` in between the calls that do the HTTP requests. Also when developing your code, once you have the code working to download the HTML, use the downloaded HTML to develop the remainder of your code that manipulates the HTML and don't repeatedly re-download the HTML as you work on the remainder of the code.

   b. Write an overall Python function (and modular helper functions to do particular pieces of the work needed) that takes as input a title and artist, searches by the title, and then (based on an exact match to the title and artist in the resulting set of song results) finds the URL of the page for the lyrics for that particular song. Then use that URL and return the lyrics, the artist, and the album(s). You can assume that the song you want is on the first page of results. If no exact match is found, just return `None`. Make sure to explain how your code extracts the HTML elements you need. Hint: you will need to use some string processing functions to do basic manipulations. We'll see this more in Unit 5, but for now, you can find information in the https://berkeley-scf.github.io/tutorial-string-processing/text-manipulation#2-basic-text-manipulation-in-python. You should NOT need to use regular expressions (which we'll cover in Units 3 and 5) or the `re` package, though you can if you want to.

   c. Modify your function so it works either when the lyrics are returned directly from the initial search or when multiple songs are returned. Include checks in your code so that it fails gracefully if the user provides invalid input or MLDb doesn't return a result.

   d. (Extra credit) Modify your code to handle cases (e.g., searching for "Dance with me") that return more than one page of results.

5. Look at the robots.txt file for MLDb and for Google Scholar (scholar.google.com) and the references in Unit 2 on the ethics of webscraping. Does it seem like it's ok to scrape data from MLDb? What about Google Scholar?