Problem Set 1

Due Wednesday Sep. 11, 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 11, 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, include a short initial section giving the names of any other students that you worked with on the problem set (or indicating you didn't work with anyone if that was the case) and then 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. Please see Lab 1 and the dynamic documents tutorial for more information on how to do this. You can put longer functions in a .py file and show the source code in your qmd file using `inspect.getsource()`.
2. Your PDF submission to Gradescope should be the PDF produced from your qmd. Your GitHub submission should include the qmd file, any Python code files (modules) that you use in your qmd file, your environment file (see problem 4d) 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 (and definitely not as screenshots).

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 20 columns and as many rows as needed so that the data take up about 16 MB (megabytes) 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 reasonably 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 = 12)

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")
```

```
30.778541 MB
16.000572 MB
```

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

   c. Now consider saving out the numbers one number per row 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 the CSV to reading the pickle file with `pandas.read_pickle`. Note that in some cases you might find that the first time you read a file is slower; if so this has to do with the operating system caching the file in memory (we'll discuss this further in Unit 7 when we talk about databases).

   e. Finally, in the next parts of the question, we'll consider reading the CSV file in chunks as discussed in Unit 2. First, time how long it takes to read the first 10,000 rows in a single chunk using `nrows`.

   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?

3. Please read the Code syntax and style section 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.) In particular, lint your code (e.g., using `ruff` or another tool of your choice as discussed in the Unit 4. (This is most straightforward for code in a .py file. If your code is directly in the qmd file, you probably need to copy-paste it into another file to lint it, unfortunately. If anyone figures out a good way to directly lint the code chunks, please post on Ed!)

   As your response to this question, very briefly (a few sentences) note what you did in your code for Problem 4 that reflects what you read. Please also note anything in what you read 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 publication information off of Google Scholar. Note that Google Scholar does not have an API, so we are forced to deconstruct the queries that are produced when we point and click on the website.

   Your functions may be short enough that it's ok to put a function directly within a code chunk in your qmd file. Or you might choose to put one or more functions into a .py file and use `inspect.getsource()` to show us the code. For this problem in which the focus of the work is the function and how it works, it will generally be best to show the function code as part of the problem solution rather than in an appendix.

   a. Go to Google Scholar and enter the name (including first name to help with disambiguation) for a researcher whose work interests you. (If you want to do the one that will match the problem set solutions, you can use "Michael Jordan", who is a well-known statistician and machine learning researcher here at Berkeley.) If you've entered the name of a researcher that Google Scholar recognizes as having a Google Scholar profile, you should see that the first item shown in the results page is a "User profile". Now, based on the information returned, show the HTML element containing the Google Scholar ID and determine the Google Scholar ID for the researcher. Ideally (see the extra credit part (e)) we would automate that process and write a Python function that returns the ID, but see Question 5 for why that seemingly would violate Google Scholar's terms of use.

   b. Create a function that constructs the http GET request (and submits that request) to get the citations for the scholar, taking the ID as the input argument and returning the HTML as a Python object.

      IMPORTANT: While running the query in an automated fashion is seemingly allowed (see Problem 5), Google may return "429" errors because it detects automated usage. Here are some things to do in that case:

      1. You can try to download the HTML file via the UNIX `curl` command, which you can run within Python as `subprocess.run(["curl", "-L", request_string], capture_output=True)`. If necessary for part (c), you can use `curl` or `wget` from the command line to separately download the file and then read that file into Python.

2. When developing your code, once you have the code in this part of the problem working to download the HTML, use the downloaded HTML to develop the remainder of your code for part (c) and don't keep re-downloading the HTML as you work on the remainder of the code.

For now, you can assume the user will provide a valid ID and that Google Scholar returns a result for the specified person. We'll deal with making the code more robust in PS2.

c. Now write a function that processes the HTML to return a Pandas (or Polars) data frame with the citation information (article title, authors, journal information, year of publication, and number of citations as five columns of information) for the researcher. Try your function on a second researcher to provide more confidence that your function is working properly. (We'll add unit tests in PS2). Given the comments in (b), ideally your function will work either if given (i) the name of a file that you've already downloaded or (ii) the HTML content produced by your function in (b).

Hint: a possibly useful argument for `find_all` is to request element(s) with certain attributes, e.g., `html.find("p", attrs = {'class': 'songtext'})` for finding a `p` element whose class is `songtext`.

d. Create a requirements file (based on either pip or Conda) that has the necessary information (in particular Python package versions) to reproduce the environment in which you ran your code. Include this file in your GitHub repository directory for this problem set.

e. (Extra credit) If you'd like extra practice, write a Python function that will return the Google Scholar ID when the function is provided an html file as its argument. The file would be the file that is returned by searching for a researcher name as the input at scholar.google.com. As discussed in (a), your function should not query Google Scholar using the `requests` package, but rather should manipulate an HTML file that you download after manually querying Google Scholar yourself.

f. (Extra credit) If you'd like extra practice, fix your function so that you get all of the results for a researcher and not just the first 20. E.g., for Michael Jordan there are several hundred.

5. Look at the `robots.txt` 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 Google Scholar? Hopefully this will make clear why our scraping in Problem 4 did not include programmatically obtaining the Google Scholar ID.