Statistics 507, Fall 2021 (./index.html)

Problem Set 2

Due Friday ~~September 24~~ October 1 by 5pm.

Instructions

Complete all 4 questions of the assignment below and submit to Canvas by the due date. Remember, if you are using late days you should submit a draft of the assignment by the due date and leave a comment indicating how many late days you want to use.

For this problem set, you should submit source code as plain text Python scripts (with extension `.py`) and an associated Jupyter notebook (with extension `.ipynb`). Use Jupytext (https://jupytext.readthedocs.io/en/latest/install.html) and (preferably) the *light* format to associate the two files.

Questions on this and future problem sets may ask you to use concepts or ideas that have not been discussed in class. One of the goals of these assignments is to help you learn to be independent, read documentation, and otherwise make reasonable decisions about how to analyze and present data, code, or other data science material.

You may discuss the problem set and its solution with your peers, but you are required to work independently on the files to be submitted and to submit your own original work. If you use or closely follow patterns or code from sources other than the course notes or texts you should cite the source.

In addition to the content of your submission, you will be graded on the quality of your source code and the professionalism of your notebook file.

Maintain a consistent and literate style in your Python script and work to make your notebook look professional and well polished. Follow all style rules from previous problem sets. Here are 3 additional style rules focused on spacing you should follow to recieve full credit:

1. Use consistent spacing around binary operators (including assignment).

i. Preferably use a space on either side of all binary operators.
ii. Do not use spaces around `=` for parameter assignments within functions.

1. Use consistent indentation throughout your code, preferably with 4 spaces for each level of indentation.
2. Use a space after a comma, *every comma*.

When asked for a nicely formatted table you should produce one using HTML (perhaps via Markdown) and it should follow standards suitable for publication, e.g. columns should be named in English rather than `code_speak`.

Question 0 - Code review warmup [10 points]

In this question, you will interpret a code snippet you did not write and then write a “code review” for that snippet.

Code Reviews

Many organizations that produce software utilize code reviews (https://en.wikipedia.org/wiki/Code_review) to help mantain a high quality code base. A good code review should address the following questions:

1. Does the code work? Does it do what it is supposed to?
2. How is the style of the code? Does it follow the style guidelines?
3. Is the code clearly structured and easy to understand?
4. Is the code efficient? If clarity is sacrificed for efficiency, are there comments that help to alleviate this?

In addition, a good code review is kind to the author. Comments are framed as suggestions to the author and written in a positive, respectful, tone. Comments address the good qualities of the code as well as any areas for improvement.

Finally, a good code review begins with the most essential aspects related to the four questions above. More minor items – called 'nitpicks' – are clearly labeled as such and kept to a minimum.

Here are two optional readings about code reviews:

1. Code Review for Data Science (https://medium.com/apteo/code-reviewing-data-science-work-774747248e33),
2. How to Make Good Code Reviews Better (https://stackoverflow.blog/2019/09/30/how-to-make-good-code-reviews-better/).

Code snippet

```
sample_list = [(1, 3, 5), (0, 1, 2), (1, 9, 8)]
op = []
for m in range(len(sample_list)):
    li = [sample_list[m]]
        for n in range(len(sample_list)):
            if (sample_list[m][0] == sample_list[n][0] and
                    sample_list[m][3] != sample_list[n][3]):
                li.append(sample_list[n])
        op.append(sorted(li, key=lambda dd: dd[3], reverse=True)[0])
res = list(set(op))
```

Instructions

a. Concisely describe what task the code above accomplishes. Say *what* it does (in total) and not *how* it accomplishes it. You may wish to understand the snippet step-by-step, but your description should *not* state each step individually.

b. Write a short code review that offers 3-5 (no more) concrete suggestions to make the snippent more efficient, literate (easier to read), or "pythonic". Focus your suggestions on concepts or principles that would help the author of this code snippet write better code in the future.

Question 1 - List of Tuples [20 points]

Write a function that uses NumPy and a list comprehension to generate a random list of `n` k-tuples containing integers ranging from `low` to `high`. Choose an appropriate name for your function, and reasonable default values for k, `low`, and `high`.

Use `assert` to test that your function returns a list of tuples.

Question 2 - Refactor the Snippet [40 points]

In this question, you will write functions to accomplish the goal you concisely described in part "a" of the warm up.

a. Encapsulate the code snippet from the warmup into a function that parameterizes the role of `0` and `3` and is otherwise unchanged. Choose appropriate names for these paramters.

b. Write an improved version of the function form part a that implements the suggestions from the code review you wrote in part b of the warmup.

c. Write a function from scratch to accomplish the same task as the previous two parts. Your solution should traverse the input list of tuples no more than twice. Hint: consider using a dictionary or a default dictionary in your solution.

d. Use the function you wrote in question 1 to generate a list of tuples as input(s), run and summarize a small Monte Carlo study comparing the execution times of the three functions above (a-c).

Question 3 - [30 points]

In this question you will use Pandas to read, clean, and append several data files from the National Health and Nutrition Examination Survey NHANES (https://www.cdc.gov/nchs/nhanes/index.htm). We will use the data you prepare in this question as the starting point for analyses in one or more future problem sets. For this problem, you should use the four cohorts spanning the years 2011-2018. You can find links to different NHANES cohorts here (https://wwwn.cdc.gov/nchs/nhanes/Default.aspx).

a. Use Python and Pandas to read and append the demographic datasets keeping only columns containing the unique ids (SEQN), age (RIDAGEYR), race and ethnicity (RIDRETH3), education (DMDEDUC2), and marital status (DMDMARTL), along with the following variables related to the survey weighting: (RIDSTATR, SDMVPSU, SDMVSTRA, WTMEC2YR, WTINT2YR). Add an additional column identifying to which cohort each case belongs. Rename the columns with literate variable names using all lower case and convert each column to an appropriate type. Finally, save the resulting data frame to a serialized "round-trip" format of your choosing (e.g. pickle, feather, or parquet).

b. Repeat part a for the oral health and dentition data (OHXDEN_*.XPT) retaining the following variables: SEQN, OHDDESTS, tooth counts (OHXxxTC), and coronal cavities (OHXxxCTC).

c. In your notebook, report the number of cases there are in the two datasets above.

Hint: Use `pd.read_sas()` to import files with the `.XPT` ("SAS transport") extension.