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

Problem Set 1

Due Friday September 17 by 5pm.

Instructions

Complete all four 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 those 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. Here are three style rules you must follow to recieve full credit:

1. No line should be more than 79 characters long,
2. Every function must have a docstring in the conventional format,
3. Module imports should appear at the start of the file.

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 - Markdown warmup [10 points]

Use markdown syntax to *exactly* reproduce the text below. In your paired notebook and script, in addition to regular (formatted) markdown, include the "raw" markdown by writing it between code fences "````". Use markdown and $\LaTeX$ only, no raw HTML. The code to be reproduced appears between horizontal rules – include these in your solution.

This is *question 0* for problem set 1 (https://jbhender.github.io/Stats507/F21/ps1.html) of Stats 507 (https://jbhender.github.io/Stats507/F21/).

> Question 0 is about Markdown.

The next question is about the **Fibonnaci sequence**, $F_n = F_{n-2} + F_{n-1}$. In part **a** we will define a Python function `fib_rec()`.

Below is a …

Level 3 Header

Next, we can make a bulleted list:

- Item 1
  - detail 1
  - detail 2
- Item 2

Finally, we can make an enumerated list:

a. Item 1
b. Item 2
c. Item 3

Question 1 - Fibonnaci Sequence [30]

The Fibonnaci seuqence (https://en.wikipedia.org/wiki/Fibonacci_number) F_n begins with the numbers $ $0, 1, 1, \ldots$ and continues so that each entry is the sum of adding its two immediate predecessors. The sequence can be defined as follows,

$$F_n = F_{n-2} + F_{n-1}, \qquad \text{with} \qquad F_0 = 0, F_1 = 1.$$

A common question in a code interview asks the interviewee to sketch a program generating Fibonnaci numbers at a board. In this quesiton, you will write and compare several such programs. After each function definition, write a test to ensure your function returns the following vlaues: $F_7 = 13, F_1 1 = 89, and F_1 3 = 233$.

a. Write a *recursive* function `fib_rec()` that takes a single input `n` and returns the value of F_n.

b. Write a function `fib_for()` with the same signature that computes F_n by summation using a `for` loop.

c. Write a function `fib_whl()` with the same signature that computes F_n by summation using a `while` loop.

d. Write a function `fib_rnd()` with the same signature that computes F_n using the rounding method described on the Wikipedia page linked above.

e. Write a function `fib_flr()` with the same signature that computes F_n using the truncation method described on the Wikipedia page linked above.

f. For a sequence of increasingly large values of `n` compare the median computation time of each of the functions above. Present your results in a nicely formatted table. (Point estimates are sufficient).

Question 2 - Pascal's Triangle [20]

Pascal's triangle (https://en.wikipedia.org/wiki/Pascal%27s_triangle) is a way to organize, compute, and present the Binomial coefficients $\binom{n}{k}$. The triangle can be constructed from the top starting with rows 0 [$\binom{0}{0} = 1$], and 1 [$\binom{1}{0} = 1$, $\binom{1}{1} = 1$]. From there, subsequent rows can be computed by adding adjacent entries from the previous row (implicitly appending zeros on the left and right).

a. Write a function to compute a specified row of Pascal's triangle. An arbitrary row of Pascal's triangle can be computed efficiently by starting with $\binom{n}{0} = 1$ and then applying the following recurrence relation among binomial coefficients,

$$\binom{n}{k} = \binom{n}{k-1} \times \frac{n+1-k}{k}.$$

b. Write a function for printing the first n rows of Pascal's triangle using the conventional spacing with the numbers in each row staggered relative to adjacent rows. Use your function to display a minimum of 10 rows in your notebook.

Question 3 - Statistics 101 [40]

In this question you will write functions to compute statistics providing point and interval estimates for common population parameters based on data. Parts a and below each ask you to write a function; each of those functions should default to returning a string of the form,

$$\hat{\theta}\ [XX\%CI : (\hat{\theta}_L, \hat{\theta}_U)],$$

where $\hat{\theta}$ is the point estimate, $(\hat{\theta}_L, \hat{\theta}_U)$ is a an XX% confidence interval, and the confidence level XX is configurable using an input parameter.

The format of this string should also be configurable using an input parameter. Define the function to return a dictionary with keys `est`, `lwr`, `upr`, and `level` when the function is called with the parameter controlling the format of the input string set to `None`.

Your functions should accept the data as a 1d Numpy array or any object coercable to such an array using `np.array()` and raise an informative exception if not.

In this question you may use any function from Numpy, but may only use Scipy for the distribution functions found in the `stats` library. Your functions should not rely on any other modules.

a. The standard point and interval estimate for the populaiton mean based on Normal theory takes the form $\bar{x} \pm z \times \mathrm{se}(x)$ where $\bar{x}$ is the mean, $\mathrm{se}(x)$ is the standard error, and z is a Gaussian multiplier that depends on the desired confidence level. Write a function to return a point and interval estimate for the mean based on these statistics.

b. There are a number of methods for computing a confidence interval (https://en.wikipedia.org/wiki/Binomial_proportion_confidence_interval) for a population proportion arising from a Binomial experiment consisting of n independent and identically distributed (iid) Bernoulli trials. Let x be the number of successes in thes trials. In this question you will write a function to return point and interval estimates based on several of these methods. Your function should have a parameter `method` that controls the method used. Include functionality for each of the following methods.

   i. The standard point and interval estimates for a population parameter based on the Normal approximation to the Binomial distribution takes the form $\hat{p} \pm z \times \sqrt{\hat{p}(1-\hat{p})}$ with $\hat{p}$ the sample proportion and z as in part a. The approximation is conventionally considered adequate when $n\hat{p} \wedge n(1-\hat{p}) > 12$. When this method is selected, your function should raise an informative warning if this condition is not satisfied.

   ii. The Clopper-Pearson interval for a population proportion can be expressed using quantiles from Beta distributions. Specifically, for a sample of size n with x successes and α 1 minus the confidence level the interval is,

$$\left(\hat{\theta}_L, \hat{\theta}_U\right) = \left(B\left(\frac{\alpha}{2}, x, n-x+1\right), B\left(1-\frac{\alpha}{2}, x+1, n-x\right)\right).$$

   iii. The Jeffrey's interval is a Bayesian credible interval with good frequentist properties. It is similar to the Clopper-Pearson interval in that it utilizes Beta quantiles, but is based on a so-called Jeffrey's prior of $B(p, 0.5, 0.5)$. Specifically, the Jeffrey's interval is $(0 \vee B(\alpha/2, x+0.5, n-x+0.5), B(1-\alpha/2, x+0.5, n-x+0.5) \wedge 1)$. (Use the sample proportion as the point estimate).

   iv. Finally, the Agresti-Coull interval arises from a notion "add 2 failures and 2 successes" as a means of regularization. More specifically, define $\tilde{n} = n + z^2$ and $\tilde{p} = (x + z^2/2)/\tilde{n}$. The Agresti-Coull interval is Normal approximation interval using $\tilde{p}$ in place of $\hat{p}$.

c. Create a 1d Numpy array with 42 ones and 48 zeros. Construct a nicely formatted table comparing 90, 95, and 99% confidence intervals using each of the methods above (including part a) on this data. Choose the number of decimals to display carefully to emphasize differences. For each confidence level, which method produces the interval with the smallest width?

Use numpy only except the last question; use sipy for the last one