Homework 6

PSTAT 5A: Spring 2023, with Ethan P. Marzban

Instructions

- Please submit your work to Gradescope by no later than **11:59pm on Tuesday, May 16**. As a reminder, late homework will not be accepted.
- Recall that you will be asked to upload a **single** PDF containing your work for *both* the programming and non-programming questions to Gradescope.
  - You can merge PDF files using either Adobe Acrobat, or using adobe's online PDF merger at this link.

Problem 1: Airplanes in the Night Sky

An aviation enthusiast is interested in determining the proportion of all US citizens that have never flown on a plane. She takes a representative sample of 210 US Citizens, and finds out that 18% of these people have never flown on a plane.

a. Is the value of 18% a population parameter or an observed instance of a sample statistic?
b. Construct a 95% confidence interval for the true proportion of US citizens that have never flown on a plane before, and interpret your interval in the context of the problem.
c. If we were to construct an 82% confidence interval for the true proportion of US citizens that have never flown on a plane before, would we expect this interval to be wider or narrower than the interval constructed in part (c)? **Do not construct the interval first**.
d. Now, construct an 82% confidence interval for the true proportion of US citizens that have never flown on a plane before. Comment on whether this agrees with your answer to part (b) or not.

Problem 2: Scores

In a particular iteration of PSTAT 5A, scores on the final exam had an average of 89 and a standard deviation of 40 The exact distribution of scores is, however, unknown. Suppose a representative sample of 100 students is taken, and the average final exam score of these 100 students is recorded.

a. Identify the population.
b. Identify the sample.
c. Define the parameter of interest. Use the notation discussed in Lecture 12.
d. Define the random variable of interest. Use the notation discussed in Lecture 12.
e. What is the sampling distribution of the random variable you defined in part (d) above? Be sure to check any conditions that might need to be checked!
f. What is the approximate probability that the average score of these 100 students lies within 5 points of the true average score of 89?

Problem 3: An Apple a Day

Quinn is interested in performing inference on the average weight of Granny Smith apples in the Santa Barbara location of *Bristol Farms*. To that end, he takes a representative sample of 52 apples; the mean weight of his sample was 83g and the standard deviation of weights in his sample was 17g.

a. Identify the population.
b. Identify the sample.
c. Define the parameter of interest. Use the notation discussed in Lecture 12.
d. Define the random variable of interest. Use the notation discussed in Lecture 12.
e. What distribution do we use to construct confidence intervals for the true average weight of a Granny Smith apple at the Santa Barbara location of *Bristol Farms*?
f. Construct a 95% confidence interval for the true average weight of a Granny Smith apple at the Santa Barbara location of *Bristol Farms*.

Problem 4: Programming

In this problem, we will build off of the material in Lab 06 to finally reproduce the demo performed in Lecture 10.

Part (a): Extending Lists

It is often necessary to be able *extend* a list of values. As an example, consider a variable `x = [1, 2, 3]`. To append a value of `4` to `x` (i.e. to redefine `x` as `[1, 2, 3, 4]`) we could simply redefine `x` to be `[1, 2, 3, 4]`, however for very large or otherwise complicated lists this may not be feasibe. Thankfully, Python has a built-in function to append to a list, called `append()`. To get a sense of how it works, let's do an example.

> **❗ Task 1**
>
> a. Define a variable `x` that is assigned a value of `[1, 2, 3, 4, 5]`.
> b. Run `x.append(6)` **exactly once**. Do **not** run this cell multiple times.
> c. Call `x`, and see if `x` has successfully been extended.

> **Caution**
>
> Using command `<list>.append()` function **mutates** the `<list>` object. That is, it physically changes the structure of `<list>` (specifically by adding an additional element at the end). What this means is that each time you run the command `<list>.append()`, a new element gets appended to `<list>`. This is why it is important you only run `append()` statements *once*.

For those of you who are curious, there is a *different* `append()` function (from the `numpy` module) that does *not* mutate the list/array being appended. But, for the purposes of this class, you do not need to know how to use it.

Part (b): Combining Loops and List Extensions

Let's combine our knowledge of `for` loops (from Lab 6) with our knowledge of list extensions (from part (a) above).

! Task 2

Consider the following game: roll a fair 12-sided die. If the number showing is between 1 and 4 inclusive we win a dollar; if the number showing is between 5 and 8 inclusive we win 2 dollars; if the number showing is between 9 and 12 inclusive we lose a dollar.

a. Simulate playing one round of this game. As a hint: you will need to use a conditional expression somewhere in your code.
b. Now, simulate playing 100 rounds of this game, and store the amount you win on each round in a list called `amount_won`. Here is a sample template you can use:

```
amount_won = []   # initialize a blank list to store the amount won

for <fill this in>:
  if <something>:
    amount_won.append(<something>)
  elif <something>:
    amount_won.append(<something>)
  else:
    amount_won.append(<something>)
```

Display the first 10 elements of the `amount_won` list.

Part (c): Creating a Population

We are almost in a position to fully understand the Lab 11 demo now! There is just perhaps one bit of code that we haven't explored fully yet:

! Task 3

Notice that in code cell 4 of the Lab 10 demo, the following code was run:

```
x = rnd.choices(['Positive', 'Negative'],
  weights = [0.035, 1 - 0.035], k = samp_size)
```

In a Markdown cell, explain *in words* what this code is doing. Specifically, think about how this code might be used to simulate drawing from a population of cats, some of which are FIV-positive. In your description, you should also mention what the population proportion (of FIV-positive cats) is in this particular simulation.

Part (d): Your Turn!

Alright, between Lab 6 and the above tasks we should have enough information to fully understand - and reproduce! - the Lab 10 demo.

> **! Task 4**
>
> In a particular country, 12% of residents live below the poverty line. Simulate taking 10,000 samples of size $n = 250$ each, computing the proportion of people in each sample that live below the poverty line, and storing these 10,000 sample proportions in a list. Plot a histogram of the list of observed sample proportions, and overlay the true density curve of $\widehat{P}$, the proportion of people in a representative sample of size $n = 250$ people that live below the poverty line.

Problem 5: One More Programming Question

I would also like to quickly introduce a function that might make our lives easier when it comes to computing the value of z_α in a confidence interval formula. Specifically, it is the `scipy.stats.norm.ppf()` function, which computes the percentiles of the normal distribution. For instance,

```python
import scipy.stats as sps
sps.norm.ppf(0.025)
```

```
-1.959963984540054
```

> **! Task 5**
>
> In Problem 1(d) above, you needed to find the value of z_α corresponding to an 82% confidence level. Use the `scipy.stats.ppf()` function to confirm the result you obtained from the standard normal table.

We can analogously find percentiles of the t_k distribution, using `scipy.stats.t.ppf(p, k)`.

> **! Task 6**
>
> a. Use Python to find the 2.5^{th} percentile of the t distribution with 20 degrees of freedom.
> b. Use Python to find the 2.5^{th} percentile of the t distribution with 10 degrees of freedom.
> c. Use Python to find the 2.5^{th} percentile of the t distribution with 1000 degrees of freedom.