Homework 6: Probability, Simulation, Estimation, and Assessing Models

Reading:

- Randomness (https://www.inferentialthinking.com/chapters/09/randomness.html)
- Sampling and Empirical Distributions (https://www.inferentialthinking.com/chapters/10/sampling-and-empirical-distributions.html)
- Testing Hypotheses (https://www.inferentialthinking.com/chapters/11/testing-hypotheses.html)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests. Each time you start your server, you will need to execute this cell again to load the tests.

Homework 6 is due Thursday, 3/18, at 11:59pm. Start early so that you can come to office hours if you're stuck. Check the website for the office hours schedule. Late work will not be accepted as per the policies of this course.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged. Refer to the policies page to learn more about how to learn cooperatively.

In [ ]:
```
# Don't change this cell; just run it.

import numpy as np
from datascience import *

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)
```

1. Probability

We will be testing some probability concepts that were introduced in lecture. For all of the following problems, we will introduce a problem statement and give you a proposed answer. Next, for each of the following questions, you must assign the provided variable to one of three integers. You are more than welcome to create more cells across this notebook to use for arithmetic operations, but be sure to assign the provided variable to 1, 2, or 3 in the end.

1. Assign the variable to 1 if you believe our proposed answer is too low.
2. Assign the variable to 2 if you believe our proposed answer is correct.
3. Assign the variable to 3 if you believe our proposed answer is too high.

Question 1.1. You roll a 6-sided die 10 times. What is the chance of getting 10 sixes?

Our proposed answer:

$$(\frac{1}{6})^{10}$$

Assign `ten_sixes` to either 1, 2, or 3 depending on if you think our answer is too low, correct, or too high.

In [ ]:
```
ten_sixes = ...
ten_sixes
```

Question 1.2. Take the same problem set-up as before, rolling a fair dice 10 times. What is the chance that every roll is less than or equal to 5?

Our proposed answer:

$$1 - (\frac{1}{6})^{10}$$

Assign `five_or_less` to either 1, 2, or 3.

In [ ]:
```
five_or_less = ...
five_or_less
```

Question 1.3. Assume we are picking a lottery ticket. We must choose three distinct numbers from 1 to 100 and write them on a ticket. Next, someone picks three numbers one by one, each time without putting the previous number back in. We win if our numbers are all called.

If we decide to play the game and pick our numbers as 12, 14, and 89, what is the chance that we win?

Our proposed answer:

$$(\frac{3}{100})^{3}$$

Assign `lottery` to either 1, 2, or 3.

In [ ]:
```
lottery = ...
```

Question 1.4. Assume we have two lists, list A and list B. List A contains the numbers [10,20,30], while list B contains the numbers [10,20,30,40]. We choose one number from list A randomly and one number from list B randomly. What is the chance that the number we drew from list A is larger than the number we drew from list B?

Our proposed solution:

$$1/4$$

Assign `list_chances` to either 1, 2, or 3.

In [ ]:
```
list_chances = ...
```

2. Monkeys Typing Shakespeare

(...or at least the string "datascience")

A monkey is banging repeatedly on the keys of a typewriter. Each time, the monkey is equally likely to hit any of the 26 lowercase letters of the English alphabet, regardless of what it has hit before. There are no other keys on the keyboard.

Question 2.1. Suppose the monkey hits the keyboard 11 times. Compute the chance that the monkey types the sequence `datascience` . (Call this `datascience_chance` .) Use algebra and type in an arithmetic equation that Python can evalute.

```
In [ ]: datascience_chance = ...
        datascience_chance
```

Question 2.2. Write a function called `simulate_key_strike` . It should take **no arguments**, and it should return a random one-character string that is equally likely to be any of the 26 lower-case English letters.

```
In [ ]: # We have provided the code below to compute a list called letters,
        # containing all the lower-case English letters.  Print it if you
        # want to verify what it contains.
        import string
        letters = list(string.ascii_lowercase)

        def simulate_key_strike():
            """Simulates one random key strike."""
            ...

        # An example call to your function:
        simulate_key_strike()
```

Question 2.3. Write a function called `simulate_several_key_strikes` . It should take one argument: an integer specifying the number of key strikes to simulate. It should return a string containing that many characters, each one obtained from simulating a key strike by the monkey.

Hint: If you make a list or array of the simulated key strikes, you can convert that to a string by calling `"".join(key_strikes_array)` (if your array is called `key_strikes_array`).

```
In [ ]: def simulate_several_key_strikes(num_strikes):
            # Fill in this function.  Our solution used several lines
            # of code.
            ...

        # An example call to your function:
        simulate_several_key_strikes(11)
```

Question 2.4. Use `simulate_several_key_strikes` 1000 times, each time simulating the monkey striking 11 keys. Compute the proportion of times the monkey types `"datascience"` , calling that proportion `datascience_proportion` .

In [ ]:
```
# Your solution may take more than one line.
...
datascience_proportion = ...
datascience_proportion
```

Question 2.5. Check the value your simulation computed for `datascience_proportion`. Is your simulation a good way to estimate the chance that the monkey types `"datascience"` in 11 strikes (the answer to question 1)? Why or why not?

Write your answer here, replacing this text.

Question 2.6. Compute the chance that the monkey types the letter `"e"` at least once in the 11 strikes. Call it `e_chance`. Use algebra and type in an arithmetic equation that Python can evalute.

In [ ]:
```
e_chance = ...
e_chance
```

Question 2.7. Do you think that a computer simulation is more or less effective to estimate `e_chance` compared to when we tried to estimate `datascience_chance` this way? Why or why not? (You don't need to write a simulation, but it is an interesting exercise.)

Write your answer here, replacing this text.

3. Sampling Basketball Players

This exercise uses salary data and game statistics for basketball players from the 2014-2015 NBA season. The data was collected from Basketball-Reference (http://www.basketball-reference.com) and Spotrac (http://www.spotrac.com).

Run the next cell to load the two datasets.

In [ ]:
```
player_data = Table.read_table('player_data.csv')
salary_data = Table.read_table('salary_data.csv')
player_data.show(3)
salary_data.show(3)
```

Question 3.1. We would like to relate players' game statistics to their salaries. Compute a table called `full_data` that includes one row for each player who is listed in both `player_data` and `salary_data`. It should include all the columns from `player_data` and `salary_data`, except the `"PlayerName"` column.

In [ ]:
```
full_data = ...
full_data
```

Basketball team managers would like to hire players who perform well but don't command high salaries. From this perspective, a very crude measure of a player's *value* to their team is the number of points the player scored in a season for every **$1000 of salary** (*Note*: the `Salary` column is in dollars, not thousands of dollars). For example, Al Horford scored 1156 points and has a salary of **$12 million.** This is equivalent to 12,000 thousands of dollars, so his value is $\frac{1156}{12000}$.

Question 3.2. Create a table called `full_data_with_value` that's a copy of `full_data`, with an extra column called `"Value"` containing each player's value (according to our crude measure). Then make a histogram of players' values. **Specify bins that make the histogram informative, and don't forget your units!** Remember that `hist()` takes in an optional third argument that allows you to specify the units!

Hint: Informative histograms contain a majority of the data and **exclude outliers**.

In [ ]:
```
full_data_with_value = ...
...
```

Now suppose we weren't able to find out every player's salary (perhaps it was too costly to interview each player). Instead, we have gathered a *simple random sample* of 100 players' salaries. The cell below loads those data.

In [ ]:
```
sample_salary_data = Table.read_table("sample_salary_data.csv")
sample_salary_data.show(3)
```

Question 3.3. Make a histogram of the values of the players in `sample_salary_data`, using the same method for measuring value we used in question 2. **Use the same bins, too.**

Hint: This will take several steps.

In [ ]:
```
# Use this cell to make your histogram.
```

Now let us summarize what we have seen. To guide you, we have written most of the summary already.

Question 3.4. Complete the statements below by filling in the [SQUARE BRACKETS].

Hint 1: For a refresher on distribution types, check out Section 10.1 (https://www.inferentialthinking.com/chapters/10/1/empirical-distributions.html)

Hint 2: The `hist()` table method ignores data points outside the range of its bins.

The plot in question 3.2 displayed a(n) [DISTRIBUTION TYPE] distribution of the population of [A NUMBER] players. The areas of the bars in the plot sum to [A NUMBER].

The plot in question 3.3 displayed a(n) [DISTRIBUTION TYPE] distribution of the sample of [A NUMBER] players. The areas of the bars in the plot sum to [A NUMBER].

4. Earthquakes

The next cell loads a table containing information about **every earthquake with a magnitude**

above 4.5 in 2017, compiled by the US Geological Survey. (source: https://earthquake.usgs.gov/earthquakes/search/ (https://earthquake.usgs.gov/earthquakes/search/))

In [ ]:
```
earthquakes = Table().read_table('earthquakes_2017.csv').select(['time', 'm
earthquakes
```

There are a several earthquakes that occurred in 2017 that we're interested in, and generally, we won't have access to this large population. Instead, if we sample correctly, we can take a small subsample of earthquakes in this year to get an idea about the distribution of magnitudes throughout the year!

Question 4.1. In the following lines of code, we take two different samples from the earthquake table, and calculate the mean of the magnitudes of these earthquakes. Are these samples representative of the population of earthquakes in the original table (that is, the should we expect the mean to be close to the population mean)?

Hint: Consider the ordering of the `earthquakes` table.

In [ ]:
```
sample1 = earthquakes.sort('mag', descending = True).take(np.arange(100))
sample1_magnitude_mean = np.mean(sample1.column('mag'))
sample2 = earthquakes.take(np.arange(100))
sample2_magnitude_mean = np.mean(sample2.column('mag'))
[sample1_magnitude_mean, sample2_magnitude_mean]
```

Write your answer here, replacing this text.

Question 4.2. Write code producing a sample that should represent the population of size 500 then take the mean of the magnitudes of the earthquakes in this sample. Assign these to `representative_sample` and `representative_mean` respectively.

Hint: In class, what sort of samples can properly represent the population?

In [ ]:
```
representative_sample = ...
representative_mean = ...
representative_mean
```

Question 4.3. Suppose we want to figure out what the biggest magnitude earthquake was in 2017, but we are tasked with doing this only with a sample of 500 from the earthquakes table.

To determine whether trying to find the biggest magnitude from a sample is a plausible idea, write code that simulates the maximum of a random sample of size 500 from the `earthquakes` table 5000 times. Assign your array of maximums to `maximums` .

In [ ]:
```
maximums = ...
for i in np.arange(5000):
    maximums = ...
```

In [ ]:
```
#Histogram of your maximums
Table().with_column('Largest magnitude in sample', maximums).hist('Largest
```

Question 4.4. We want to see if a random sample of size 500 is likely to help you determine the largest magnitude earthquake in the population. To help determine this, find the magnitude of the (actual) strongest earthquake in 2017.

In [ ]:
```
strongest_earthquake_magnitude = ...
strongest_earthquake_magnitude
```

Question 4.5. Explain whether you believe you can accurately use a sample size of 500 to determine the maximum. What is a specific con of using the maximum as your estimator? Use the histogram above to help answer.

Write your answer here, replacing this text.

5. Assessing Gary's Models

Games with Gary

Our friend Gary comes over and asks us to play a game with him. The game works like this:

> We will flip a fair coin 10 times, and if the number of heads is greater than or equal to 5, we win!
>
> Otherwise, Gary wins.

We play the game once and we lose, observing 1 head. We are angry and accuse Gary of cheating! Gary is adamant, however, that the coin is fair.

Gary's model claims that there is an equal chance of getting heads or tails, but we do not believe him. We believe that the coin is clearly rigged, with heads being less likely than tails.

Question 5.1. Assign `coin_model_probabilities` to a two-item array containing the chance of heads as the first element and the chance of tails as the second element under Gary's model. Make sure your values are between 0 and 1.

In [ ]:
```
coin_model_probabilities = ...
coin_model_probabilities
```

Question 5.2. We believe Gary's model is incorrect. In particular, we believe there to be a smaller chance of heads. Which of the following statistics can we use during our simulation to test between the model and our alternative? Assign `statistic_choice` to the correct answer.

1. The distance (absolute value) between the actual number of heads in 10 flips and the expected number of heads in 10 flips (5)
2. The expected number of heads in 10 flips
3. The actual number of heads we get in 10 flips

In [ ]:
```
statistic_choice = ...
statistic_choice
```

Question 5.3. Define the function `coin_simulation_and_statistic`, which, given a sample size and an array of model proportions (like the one you created in Q1), returns the number of heads in one simulation of flipping the coin under the model specified in `model_proportions`.

Hint: Think about how you can use the function `sample_proportions`.

In [ ]:
```
def coin_simulation_and_statistic(sample_size, model_proportions):
    ...

coin_simulation_and_statistic(10, coin_model_probabilities)
```

Question 5.4. Use your function from above to simulate the flipping of 10 coins 5000 times under the proportions that you specified in problem 5.1. Keep track of all of your statistics in `coin_statistics`.

In [ ]:
```
coin_statistics = ...
repetitions = ...

for ... in ...:
    ...

coin_statistics
```

Let's take a look at the distribution of statistics, using a histogram.

In [ ]:
```
#Draw a distribution of statistics
Table().with_column('Coin Statistics', coin_statistics).hist()
```

Question 5.5. Given your observed value, do you believe that Gary's model is reasonable, or is our alternative more likely? Explain your answer using the distribution drawn in the previous problem.

Write your answer here, replacing this text.

6. Submission

Once you're finished, submit your assignment as a .pdf (download as .html, then print to save as a .pdf) on Gradescope.

```
In [ ]:
```