Homework 5: Probability and Sampling

Please complete this notebook by filling in the cells provided. When you're done, follow the instructions in this short explainer video (https://www.youtube.com/watch?v=gMt_Rq43y_4&ab_channel=FahadKamran) to submit your homework.

If you cannot submit online, come to office hours for assistance. The office hours schedule appears on data8.org/fa16/weekly.html (http://data8.org/fa16/weekly.html).

This assignment is due Thursday, October 6 at 5PM. You will receive an early submission bonus point if you turn it in by Wednesday, October 5 at 5PM. Directly sharing answers is not okay, but discussing problems with course staff or with other students is encouraged.

Reading:

- Textbook chapters 8 (https://www.inferentialthinking.com/chapters/08/randomness.html) and 9 (https://www.inferentialthinking.com/chapters/09/empirical-distributions.html)

Run the cell below to prepare the notebook and the tests. **Passing the automatic tests does not guarantee full credit on any question.** The tests are provided to help catch some common errors, but it is *your* responsibility to answer the questions correctly.

In [ ]:
```
# Run this cell to set up the notebook, but please don't change 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)

from client.api.assignment import load_assignment
tests = load_assignment('hw05.ok')
```

1. How Many Devices?

When a company produces medical devices, it must be sure that its devices will not fail. Sampling is used ubiquitously in the medical device industry to test how well devices work.

Suppose you work at a company that produces syringes, and you are responsible for ensuring the syringes work well. After studying the manufacturing process for the syringes, you have a hunch that they have a 1% failure rate. That is, you suspect that 1% of the syringes won't work when a doctor uses them to inject a patient with medicine.

To test your hunch, you would like to find at least one faulty syringe. You hire an expert consultant who can test a syringe to check whether it is faulty. But the expert's time is expensive, so you need to avoid checking more syringes than you need to.

Important note: This exercise asks you to compute numbers that are related to probabilities. For all questions, you can calculate your answer using algebra, **or** you can write and run a simulation to compute an approximately-correct answer. (For practice, we suggest trying both.) An answer based on an appropriate simulation will receive full credit. If you simulate, use at least **5,000** simulations.

Question 1. Suppose there is indeed a 1% failure rate among all syringes. If you check 20 syringes chosen at random from among all syringes, what is the chance that you find at least 1 faulty syringe? (You may assume that syringes are chosen with replacement from a population in which 1% of syringes are faulty.) Name your answer `chance_to_find_syringe`.

```
In [ ]: # For your convenience, we have created a list containing
        # 99 copies of the number 0 (to represent good syringes)
        # and 1 copy of the number 1 (to represent a bad syringe).
        # This may be useful if you run a simulation.  Feel free
        # to delete it.
        faultiness = np.append(0*np.arange(99), 1)

        chance_to_find_syringe = ...
        chance_to_find_syringe
```

```
In [23]: _ = tests.grade('q1_1')
```

Question 2. Continue to assume that there really is a 1% failure rate. Find the smallest number of syringes you can check so that you have at least a 50% chance of finding a faulty syringe. (Your answer should be an integer.) Name that number `num_required_for_50_percent`. **It's okay if your answer is off by as many as 11.**

```
In [ ]: num_required_for_50_percent = ...
        num_required_for_50_percent
```

```
In [31]: _ = tests.grade('q1_2')
```

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 hits at other times. There are no other keys on the keyboard.

Question 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.

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

```
In [8]: _ = tests.grade('q2_1')
```

Question 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()
```

```
In [9]: _ = tests.grade('q2_2')
```

Question 3. Write a function called `simulate_several_key_strikes`. It should take one argument, an integer number of key strikes to simulate. It should return a *string* containing that many characters, each one a simulated 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)
```

```
In [10]: _ = tests.grade('q2_3')
```

Question 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 [ ]: # Our solution used several lines of code.
        ...
        datascience_proportion = ...
        datascience_proportion
```

```
In [11]: _ = tests.grade('q2_4')
```

Question 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 6. Compute the chance that the monkey types the letter `"e"` at least once in the 11 strikes. Call it `e_chance`.

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

```
In [ ]: _ = tests.grade('q2_6')
```

Question 7. In comparison to `datascience_chance`, do you think that a computer simulation would be a more or less effective way to estimate `e_chance`? 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 were 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 [3]:
```
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 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 [10]:
```
full_data = ...
full_data
```

In [11]:
```
_ = tests.grade('q3_1')
```

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 divided by their salary.

Question 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.**

In [12]:
```
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 [19]:
```
sample_salary_data = Table.read_table("sample_salary_data.csv")
sample_salary_data.show(3)
```

Question 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 [20]:
```
# 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 4. Complete the statements below by filling in the [SQUARE BRACKETS]:

The plot in question 2 displayed a(n) [TYPE OF PLOT] of the population of [A NUMBER] players. The sum of the areas of the bars in the plot was [A NUMBER].

The plot in question 3 displayed a(n) [TYPE OF PLOT] of the sample of [A NUMBER] players. The sum of the areas of the bars in the plot was [A NUMBER].

Question 5. Does the plot in question 3 accurately depict the proportion of players *in the population* whose value is between 0 and 0.0005? (Note the number of 0s.) What about players with value above 0.0005?

Write your answer here, replacing this text.

In [ ]:
```
# For your convenience, you can run this cell to run all the tests at on
ce!
import os
print("Running all tests...")
_ = [tests.grade(q[:-3]) for q in os.listdir("tests") if
q.startswith('q')]
print("Finished running all tests.")
```