hw06_student_solutions

December 10, 2019

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

Reading: * Randomness * Sampling and Empirical Distributions * Testing Hypotheses

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, 10/10 at 11:59pm. You will receive an early submission bonus point if you turn in your final submission by Wednesday, 10/9 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.

For all problems that you must write our explanations and sentences for, you **must** provide your answer in the designated space. Moreover, throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on.

[ ]:
```
# 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)

from client.api.notebook import Notebook
ok = Notebook('hw06.ok')
_ = ok.auth(inline=True)
```

1.1 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. You must assign the provided variable to one of the following three integers, depending on whether the proposed answer is too low, too high, or correct.

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.

You are more than welcome to create more cells across this notebook to use for arithmetic operations

Question 1. You roll a 6-sided die 10 times. What is the chance of getting 10 sixes?

Our proposed answer:

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

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

```
BEGIN QUESTION
name: q1_1
manual: false
```

[2]:
```
ten_sixes = 2 # SOLUTION
ten_sixes
```

[2]: 2

Question 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:

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

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

```
BEGIN QUESTION
name: q1_2
manual: false
```

[5]:
```
five_or_less = 3 # SOLUTION
five_or_less
```

[5]: 3

Question 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 from a bowl with numbers from 1 to 100 each time without putting the previous number back in. We win if our numbers are all called in order.

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:

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

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

```
BEGIN QUESTION
name: q1_3
manual: false
```

[8]:
```python
lottery = 3 # SOLUTION
```

Question 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.

```
BEGIN QUESTION
name: q1_4
manual: false
```

[11]:
```python
list_chances = 2 # SOLUTION
```

1.2 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 has hit before. There are no other keys on the keyboard.

This question is inspired by a mathematical theorem called the Infinite monkey theorem (https://en.wikipedia.org/wiki/Infinite_monkey_theorem), which postulates that if you put a monkey in the situation described above for an infinite time, they will eventually type out all of Shakespeare's works.

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 and type in an arithmetic equation that Python can evalute.

```
BEGIN QUESTION
name: q2_1
manual: false
```

[1]:
```python
datascience_chance = (1/26)**11 #SOLUTION
datascience_chance
```

[1]: 2.7245398995795435e-16

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.

BEGIN QUESTION
name: q2_2
manual: false

[6]:
```python
# 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."""
    return np.random.choice(letters) #SOLUTION

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

[6]: 'd'

Question 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 called `key_strikes_array`, you can convert that to a string by calling `"".join(key_strikes_array)`

BEGIN QUESTION
name: q2_3
manual: false

[10]:
```python
def simulate_several_key_strikes(num_strikes):
    # BEGIN SOLUTION
    """Simulates several random key strikes, returning them as a string."""
    strikes = make_array()
    for i in np.arange(num_strikes):
        one_strike = simulate_key_strike()
        strikes = np.append(strikes, one_strike)
    return "".join(strikes)
    # END SOLUTION

# An example call to your function:
```

```python
simulate_several_key_strikes(11)
```

[10]: 'zuovqqfrjdq'

Question 4. Call `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`.

```
BEGIN QUESTION
name: q2_4
manual: false
```

[14]:
```python
# BEGIN SOLUTION
num_simulations = 1000
num_datascience = 0
for i in np.arange(num_simulations):
    if simulate_several_key_strikes(11) == 'datascience':
        num_datascience = num_datascience + 1

datascience_proportion = num_datascience / num_simulations
# END SOLUTION
datascience_proportion
```

[14]: 0.0

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?

```
BEGIN QUESTION
name: q2_5
manual: true
```

SOLUTION: No, it is not a good way to estimate it. The monkey types `"datascience"` very rarely - roughly 1 in 27 quadrillion times. That usually won't happen even once in 1000 simulations, so our estimate will usually be 0. If it happened, our estimate would be at least .001, which would also be inaccurate! So we would need many more simulations (at least 27 quadrillion) to have any hope at a reasonable estimate. Algebra is more useful than a computer in this case.

Question 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.

```
BEGIN QUESTION
name: q2_6
manual: false
```

[16]:
```python
e_chance = 1 - (25/26)**11 #SOLUTION
e_chance
```

[16]: 0.35041906843673165

Question 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.)

```
BEGIN QUESTION
name: q2_7
manual: true
```

SOLUTION: Simulation would work better for estimating `e_chance`. The chance of typing 'datascience' was so small that we couldn't expect the event to happen with only 1000 iterations. But since the probability of `e_chance` is actually around 1/3, it will show up in our simulation as often as it should under its theoretical probability.

1.3 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 and Spotrac.

Run the next cell to load the two datasets.

[2]:
```
player_data = Table.read_table('player_data.csv')
salary_data = Table.read_table('salary_data.csv')
player_data.show(3)
salary_data.show(3)
```

```
<IPython.core.display.HTML object>


<IPython.core.display.HTML object>
```

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.

```
BEGIN QUESTION
name: q3_1
manual: false
```

[3]:
```
full_data = player_data.join('Name', salary_data, 'PlayerName') #SOLUTION
full_data
```

[3]:
```
Name             | Age  | Team | Games | Rebounds | Assists | Steals | Blocks |
Turnovers | Points | Salary
A.J. Price       | 28   | TOT  | 26    | 32       | 46      | 7      | 0      |
14        | 133    | 62552
Aaron Brooks     | 30   | CHI  | 82    | 166      | 261     | 54     | 15     |
157       | 954    | 1145685
Aaron Gordon     | 19   | ORL  | 47    | 169      | 33      | 21     | 22     |
```

```
38          | 243     | 3992040
Adreian Payne     | 23    | TOT   | 32      | 162        | 30        | 19        | 9         |
44          | 213     | 1855320
Al Horford        | 28    | ATL   | 76      | 544        | 244       | 68        | 98        |
100         | 1156    | 12000000
Al Jefferson      | 30    | CHO   | 65      | 548        | 113       | 47        | 84        |
68          | 1082    | 13666667
Al-Farouq Aminu | 24    | DAL   | 74      | 342        | 59        | 70        | 62        |
55          | 412     | 1100602
Alan Anderson     | 32    | BRK   | 74      | 204        | 83        | 56        | 5         |
60          | 545     | 1276061
Alec Burks        | 23    | UTA   | 27      | 114        | 82        | 17        | 5         |
52          | 374     | 3034356
Alex Kirk         | 23    | CLE   | 5       | 1          | 1         | 0         | 0         | 0
| 4        | 507336
… (482 rows omitted)
```

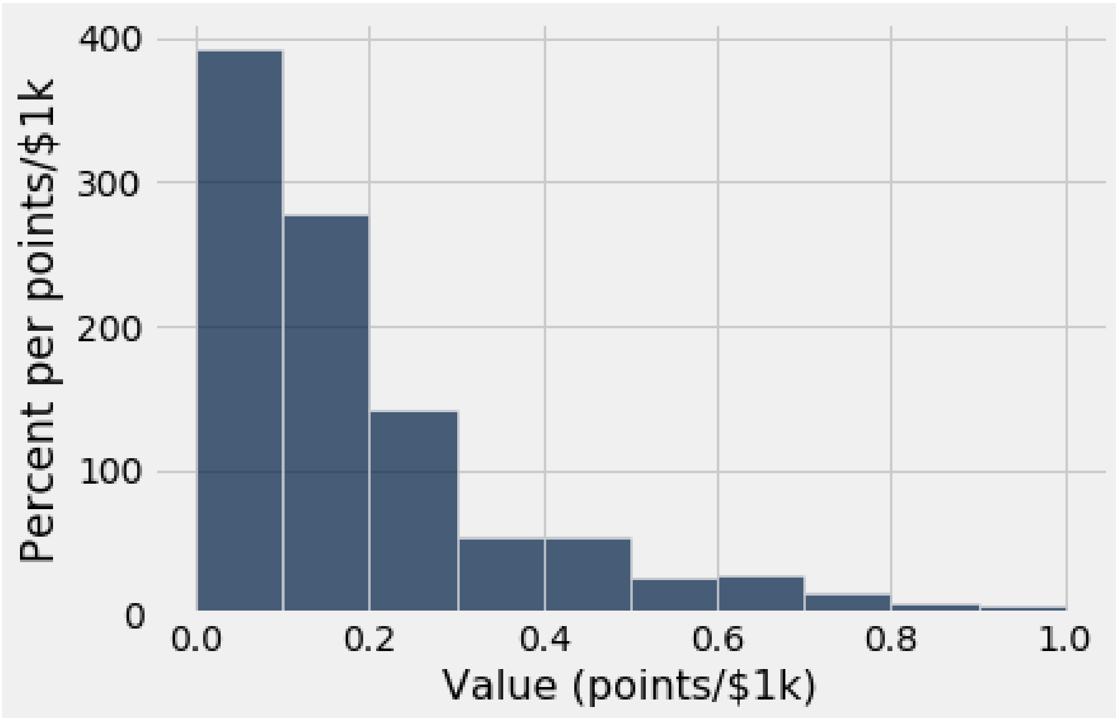
Basketball team managers would like to hire players who perform well but don't command high salaries. From this perspective, a very crude measure of a player's *value* to their team is the number of points the player scored in a season for every **\$1000 of salary** (*Note*: the `Salary` column is in 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 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**.

```
BEGIN QUESTION
name: q3_2
manual: true
```

[6]:
```
full_data_with_value = full_data.with_column("Value", full_data.
↪column("Points") / full_data.column("Salary") * 1000) #SOLUTION
full_data_with_value.hist("Value", bins=np.arange(0, 1.1, .1), unit="points/
↪$1k") #SOLUTION
```



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.

[7]:
```
sample_salary_data = Table.read_table("sample_salary_data.csv")
sample_salary_data.show(3)
```

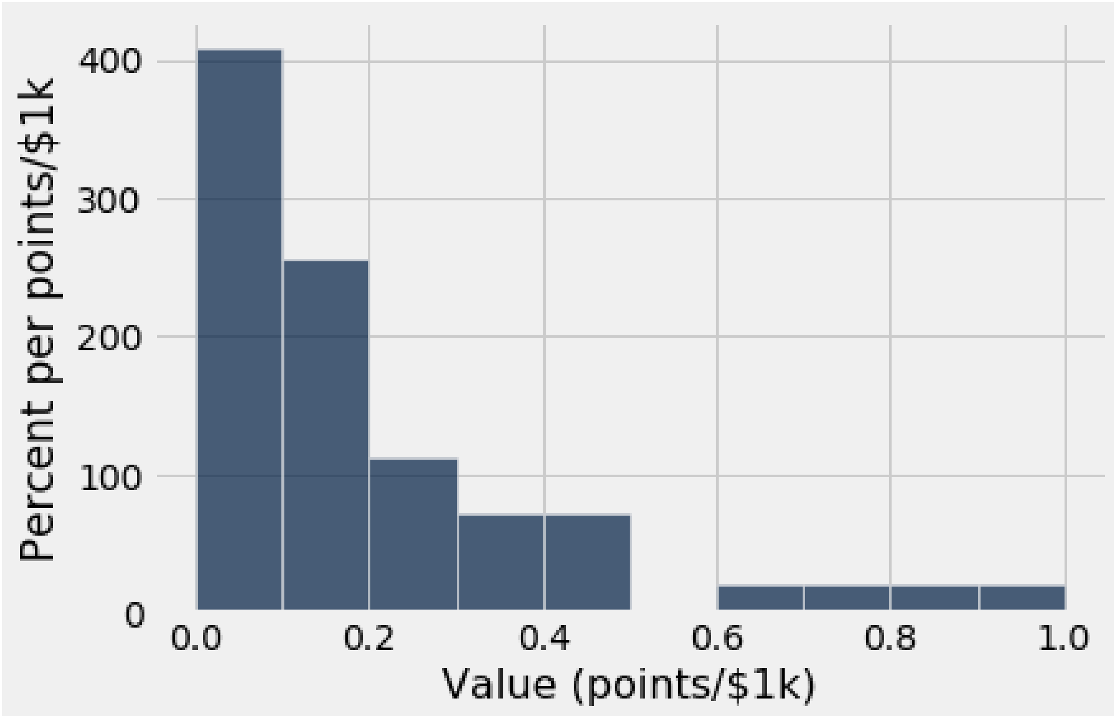
```
<IPython.core.display.HTML object>
```

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.

```
BEGIN QUESTION
name: q3_3
manual: true
```

[8]:
```
sample_data = player_data.join('Name', sample_salary_data, 'PlayerName')
sample_data_with_value = sample_data.with_column("Value", sample_data.
↪column("Points") / sample_data.column("Salary") * 1000) #SOLUTION
sample_data_with_value.hist("Value", bins=np.arange(0, 1.1, .1), unit="points/
↪$1k") #SOLUTION
```



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 setting each relevant variable name to the value that correctly fills the blank.

- The plot in question 2 displayed a(n) [`distribution_1`] distribution of the population of [`player_count_1`] players. The areas of the bars in the plot sum to [`area_total_1`].
- The plot in question 3 displayed a(n) [`distribution_2`] distribution of the sample of [`player_count_2`] players. The areas of the bars in the plot sum to [`area_total_2`].

`distribution_1` and `distribution_2` should be set to one of the following strings: `"empirical"` or `"probability"`.

`player_count_1`, `area_total_1`, `player_count_2`, and `area_total_2` should be set to integers.

Hint 1: For a refresher on distribution types, check out Section 10.1

Hint 2: The `hist()` table method ignores data points outside the range of its bins, but you may ignore this fact and calculate the areas of the bars using what you know about histograms from lecture.

BEGIN QUESTION
name: q3_4

[11]:
```
distribution_1 = "empirical" # SOLUTION
player_count_1 = 492 # SOLUTION
```

```python
area_total_1 = 100 # SOLUTION

distribution_2 = "empirical" # SOLUTION
player_count_2 = 100 # SOLUTION
area_total_2 = 100 # SOLUTION
```

Question 5. For which range of values does the plot in question 3 better depict the distribution of the **population's player values**: 0 to 0.5, or above 0.5? Explain your answer.

```
BEGIN QUESTION
name: q3_5
manual: true
```

SOLUTION: The sample histogram and population histogram look similar for values below 0.5. For values above 0.5, the sample histogram looks less accurate. The players in the population with values above 0.5 are rarer, so the sample gives us a worse estimate of that part of the distribution.

1.4 4. Earthquakes

The next cell loads a table containing information about **every earthquake with a magnitude above 4.5** in 2017 (smaller earthquakes are generally not felt, only recorded by very sensitive equipment), compiled by the US Geological Survey. (source: https://earthquake.usgs.gov/earthquakes/search/)

[2]:
```python
earthquakes = Table().read_table('earthquakes_2017.csv').select(['time', 'mag',␣
↪'place'])
earthquakes
```

[2]:
```
time                     | mag  | place
2017-12-31T23:48:50.980Z | 4.8  | 30km SSE of Pagan, Northern Mariana Islands
2017-12-31T20:59:02.500Z | 5.1  | Southern East Pacific Rise
2017-12-31T20:27:49.450Z | 5.2  | Chagos Archipelago region
2017-12-31T19:42:41.250Z | 4.6  | 18km NE of Hasaki, Japan
2017-12-31T16:02:59.920Z | 4.5  | Western Xizang
2017-12-31T15:50:22.510Z | 4.5  | 156km SSE of Longyearbyen, Svalbard and Jan
Mayen
2017-12-31T14:53:32.590Z | 5.1  | 41km S of Daliao, Philippines
2017-12-31T14:51:58.200Z | 5.1  | 132km SSW of Lata, Solomon Islands
2017-12-31T12:24:13.150Z | 4.6  | 79km SSW of Hirara, Japan
2017-12-31T04:02:18.500Z | 4.8  | 10km W of Korini, Greece
… (6350 rows omitted)
```

If we were studying all human-detectable 2017 earthquakes and had access to the above data, we'd be in good shape - however, if the USGS didn't publish the full data, we could still learn something about earthquakes from just a smaller subsample. If we gathered our sample correctly, we could use that subsample to get an idea about the distribution of magnitudes (above 4.5, of course) throughout the year!

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.

[3]:
```
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]
```

[3]:
```
[6.422999999999999, 4.774999999999995]
```

Question 1. 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.

```
BEGIN QUESTION
name: q4_1
manual: true
```

SOLUTION: These samples are deterministic samples, not random samples, so we have no reason to believe they will represent the population or have a statistic close to the population parameter. Sample 1 is especially bad, because we are taking the mean of the highest-magnitude earthquakes. Sample 2 might represent the population a little bit better if earthquakes are randomly distributed through time and there is nothing particularly unique about December earthquakes, but only sampling December earthquake still has its own deterministic bias.

Question 2. Write code to produce a sample of size 500 that is representative of the population. 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, we learned what kind of samples should be used to properly represent the population.

```
BEGIN QUESTION
name: q4_2
manual: false
```

[4]:
```
representative_sample = earthquakes.sample(500) #SOLUTION
representative_mean = np.mean(representative_sample.column('mag')) #SOLUTION
representative_mean
```

[4]:
```
4.8218000000000005
```

Question 3. Suppose we want to figure out what the biggest magnitude earthquake was in 2017, but we only have our representative sample of 500. Let's see if trying to find the biggest magnitude in the population from a random sample of 500 is a reasonable idea!

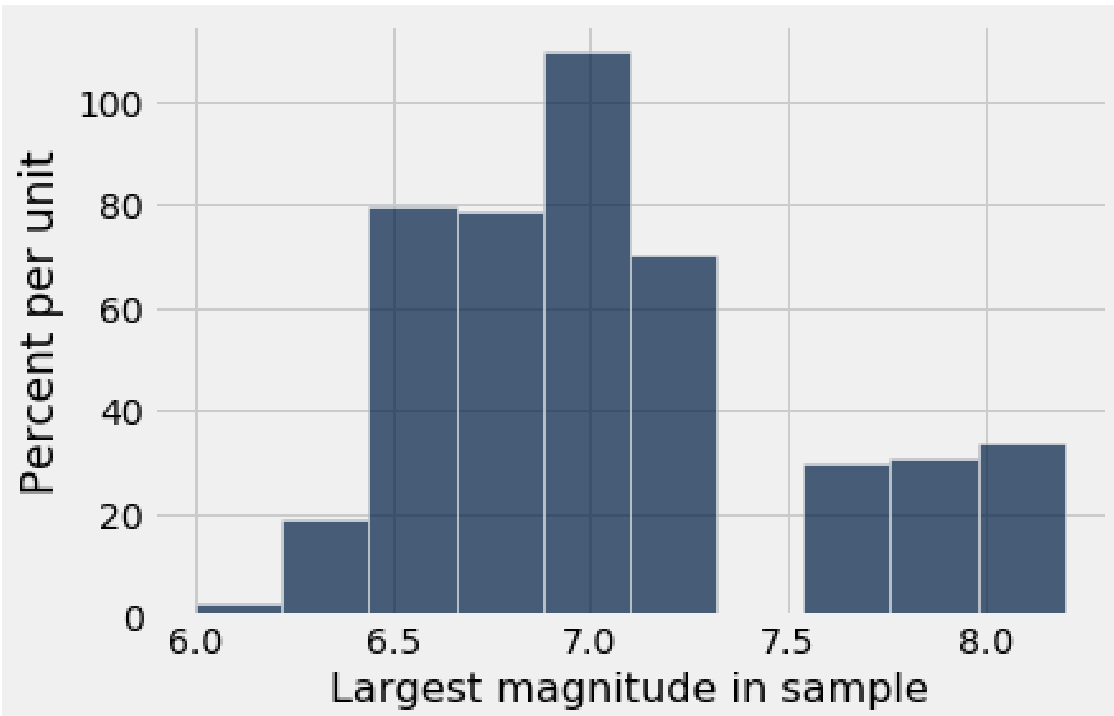
Write code that takes many random samples from the `earthquakes` table and finds the maximum of each sample. You should take a random sample of size 500 and do this 5000 times. Assign the array of maximum magnitudes you find to `maximums`.

```
BEGIN QUESTION
```

name: q4_3
manual: false

[8]:
```
maximums = make_array() #SOLUTION
for i in np.arange(5000):
    # BEGIN SOLUTION
    sample = earthquakes.sample(500)
    sample_max_magnitude = max(sample.column('mag'))
    maximums = np.append(maximums, sample_max_magnitude)
    # END SOLUTION
```

[12]:
```
#Histogram of your maximums
Table().with_column('Largest magnitude in sample', maximums).hist('Largest␣
↪magnitude in sample')
```



Question 4. Now find the magnitude of the actual strongest earthquake in 2017 (not the maximum of a sample). This will help us determine whether a random sample of size 500 is likely to help you determine the largest magnitude earthquake in the population.

BEGIN QUESTION
name: q4_4
manual: false

[13]:
```
strongest_earthquake_magnitude = max(earthquakes.column('mag')) #SOLUTION
strongest_earthquake_magnitude
```

[13]: 8.2

Question 5. Explain whether you believe you can accurately use a sample size of 500 to determine the maximum. What is one problem with using the maximum as your estimator? Use the histogram above to help answer.

```
BEGIN QUESTION
name: q4_5
manual: true
```

SOLUTION: While we get pretty close to the actual max in the histogram, we can probably not get the actual maximum using a sample size of 500. One con of this approach is that our estimate will always be less than or equal to the actual maximum.

1.5 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 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. Since we're working with probabilities, make sure your values are between 0 and 1.

```
BEGIN QUESTION
name: q5_1
manual: false
```

[2]:
```
coin_model_probabilities = make_array(.5,.5) #SOLUTION
coin_model_probabilities
```

[2]: array([0.5, 0.5])

Question 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

BEGIN QUESTION
name: q5_2
manual: false

[9]:
```
statistic_choice = 3 #SOLUTION
statistic_choice
```

[9]: 3

Question 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 Question 1), 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`.

BEGIN QUESTION
name: q5_3
manual: false

[14]:
```
def coin_simulation_and_statistic(sample_size, model_proportions):
    # BEGIN SOLUTION
    simulation = sample_proportions(sample_size, model_proportions)
    statistic = sample_size * simulation.item(0)
    return statistic
    # END SOLUTION

coin_simulation_and_statistic(10, coin_model_probabilities)
```

[14]: 5.0

Question 4

Use your function from above to simulate the flipping of 10 coins 5000 times under the proportions that you specified in Question 1. Keep track of all of your statistics in `coin_statistics`.

BEGIN QUESTION
name: q5_4
manual: false

[17]:
```
repetitions = 5000
# BEGIN SOLUTION
coin_statistics = make_array()

for i in np.arange(repetitions):
    one_coin_stat = coin_simulation_and_statistic(10, coin_model_probabilities)
    coin_statistics = np.append(coin_statistics, one_coin_stat)
```

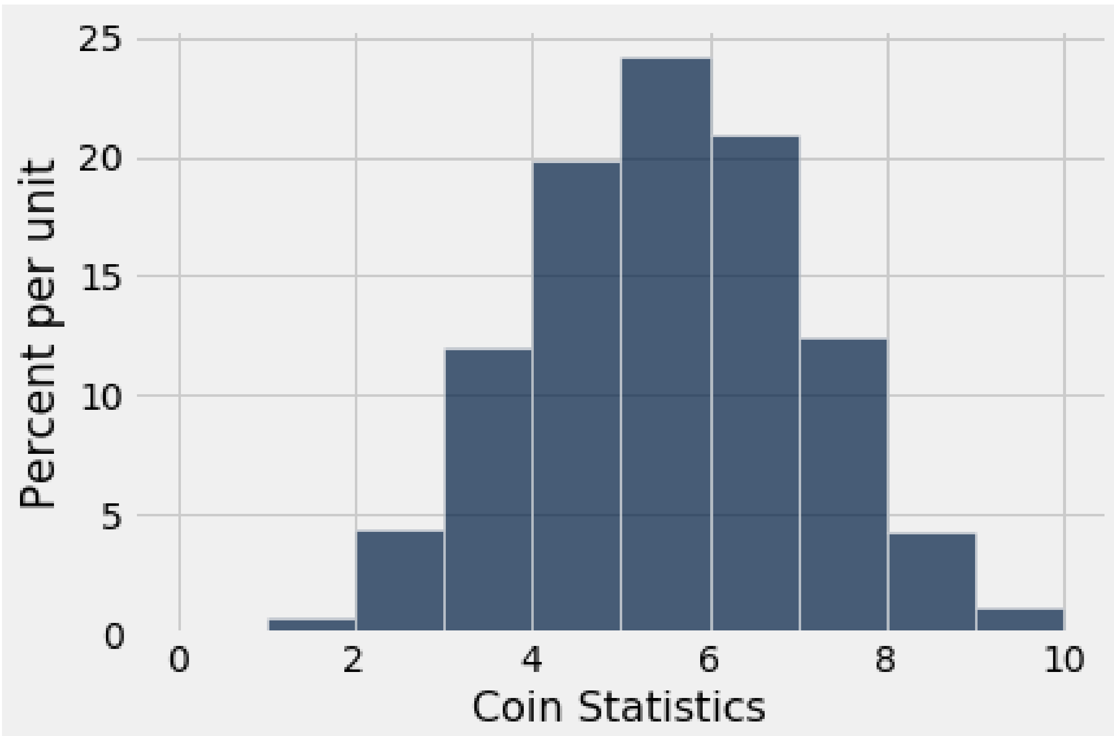
```
# END SOLUTION

coin_statistics
```

[17]:
```
array([5., 2., 5., …, 2., 4., 6.])
```

Let's take a look at the distribution of simulated statistics.

[20]:
```
#Draw a distribution of statistics
Table().with_column('Coin Statistics', coin_statistics).hist()
```



Question 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.

```
BEGIN QUESTION
name: q5_5
manual: true
```

SOLUTION: No; given Gary's model, 1 head is almost never appearing as a possibility under simulation. This points us to think that our alternative, that the probability of heads is less than 1/2, is more likely.

1.6 6. Submission

Once you're finished, select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission. If you mistakenly submit the wrong one, you can head to okpy.org and flag the correct version. To do so, go to the website, click on this assignment, and find the version you would like to have graded. There should be an option to flag that submission for grading!

[ ]:
```
_ = ok.submit()
```

[ ]:
```
# For your convenience, you can run this cell to run all the tests at once!
import os
print("Running all tests...")
_ = [ok.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q') and␣
↪len(q) <= 10]
print("Finished running all tests.")
```