hw06

November 20, 2018

1 Homework 6: Probability and Hypothesis Testing

1.1 Due Sunday November 18th, 11:59pm

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged.

You should start early so that you have time to get help if you're stuck.

In [190]:
```
#: Don't change this cell; just run it.

import numpy as np
from datascience import *

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

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

```
=====================================================================
Assignment: Homework 6: ...
OK, version v1.13.11
=====================================================================

Successfully logged in as wec149@ucsd.edu
```

Important: The `ok` tests don't usually tell you that your answer is correct. More often, they help catch careless mistakes. It's up to you to ensure that your answer is correct. If you're not sure, ask someone (not for the answer, but for some guidance about your approach).

Once you're finished, you must do two things:

1.1.1 a. Turn into OK

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.

```
In [191]: #: turn in your notebook
          _ = ok.submit()
```

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


<IPython.core.display.Javascript object>


Saving notebook... Saved 'hw06.ipynb'.
Submit... 100% complete
Submission successful for user: wec149@ucsd.edu
URL: https://okpy.org/ucsd/dsc10/fa18/hw06/submissions/wmovZR
```

1.1.2 b. Turn PDF into Gradescope

Select File > Download As > PDF via LaTeX in the File menu. Turn in this PDF file into the respective assignement at https://gradescope.com/. If you submit more than once before the deadline, we will only grade your final submission

1.2 1. Numbers in a Slot Machine

You are in front of a slot machine with three slots. Each slot in the slot machine has 10 possible outcomes: the numbers from 0-9. When you press the "Spin" button on the slot machine, each of the three slots spins independently and stops at a number. Assume that the slot machine always picks a number randomly.

Question 1. Suppose you win the jackpot if you are lucky enough to encounter the following sequence of spins, in order:

Spin 1: You see a 777 in the slot machine.

Spin 2: You see a 999 in the slot machine.

What is the probability that you win the jackpot if you press the "Spin" button twice? Assign your answer to `jackpot_chance`.

```
In [192]: jackpot_chance = 1/250
          jackpot_chance
```

```
Out[192]: 0.004
```

```
In [193]: #: grade 1.1
          _ = ok.grade('q1_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
```

```
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2. What is the probability that you see a number greater than 700 when you press "Spin" once? Assign your answer to `greater_than_700`.

```
In [194]: greater_than_700 = 299/1000
          greater_than_700

Out[194]: 0.299

In [195]: #: grade 1.2
          _ = ok.grade('q1_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 3. Write a function called `simulate_one_spin`. It should take **no arguments**, and it should return a random number that is equally-likely to come up in the slot-machine. Note that since it is a number, the leading zeros are ignored. For example, if the slot number spits out 009, then the corresponding return value of your function should be 9.

```
In [196]: # Place your answer here. It may contain several lines of code.
          def simulate_one_spin():
              chances = np.arange(0,10,1)
              jackpot_roll = np.random.choice(chances, 3)
              first_item = jackpot_roll.item(0)
              second_item = jackpot_roll.item(1)
              third_item = jackpot_roll.item(2)

              if first_item == 0 and second_item == 0:
                  string_spin = str(third_item)
              elif first_item == 0:
                  string_spin = str(second_item) + str(third_item)
              else:
                  string_spin = str(first_item) + str(second_item) + str(third_item)
```

```
            int_spin = int(string_spin)
            return int_spin

        simulate_one_spin()
```

```
Out[196]: 592
```

```
In [197]: #: grade 1.3
          _ = ok.grade('q1_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 4. Call the function `simulate_one_spin` 100,000 times. What proportion of times does the slot machine output 777? Assign your answer to `proportion_777`. Your solution may take more than one line.

```
In [198]: counter = 0
          for i in np.arange(100000):
              simulate_one_spin()
              if simulate_one_spin() == 777:
                  counter = counter + 1

          proportion_777 = counter / 100000
          proportion_777
```

```
Out[198]: 0.00077
```

```
In [199]: #: grade 1.4
          _ = ok.grade('q1_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 5. Compute the probability that at least one of the slots in the slot machine (out of the three) gives out a 7. You can write it as an expression which can be evaluated by Python. Assign your answer to `at_least_one_7`.

```
In [200]: # chances = np.arange(0,10,1)
          # jackpot_roll = np.random.choice(chances, 3)
          # first_item = jackpot_roll.item(0)
          # second_item = jackpot_roll.item(1)
          # third_item = jackpot_roll.item(2)

          # if first_item == 7 and second_item == 7 and third_item == 7:
          #     string_spin = str(third_item)
          # elif first_item == 0:
          #     string_spin = str(second_item) + str(third_item)
          # else:
          #     string_spin = str(first_item) + str(second_item) + str(third_item)

          at_least_one_7 = 1 - 0.9 ** 3
          at_least_one_7

Out[200]: 0.2709999999999999

In [201]: #: grade 1.5
          _ = ok.grade('q1_5')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

1.3 2. Apples and Oranges

Suppose you are given a huge farm that yields apples and oranges.

```
In [202]: #: Don't change this cell, just run it
          apples = ['Apple' for _  in range(400)]
          oranges = ['Orange' for _ in range(600)]
          farm_table = Table().with_column(
              'Fruit Type', apples + oranges
          )
          farm_table
```

```
Out[202]: Fruit Type
          Apple
          Apple
          Apple
          Apple
          Apple
          Apple
          Apple
          Apple
          Apple
          Apple
          ... (990 rows omitted)
```

Question 1. Because you like apples more, you're interested in the proportion of apples in the farm. Calculate the true proportion of apples in the farm. Store it in the variable `apples_true_prop`.

```
In [203]: apples_true_prop = farm_table.where('Fruit Type', are.equal_to('Apple')).num_rows / fa
          apples_true_prop

Out[203]: 0.4

In [204]: #: grade 2.1
          _ = ok.grade('q2_1')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 2. Which of the following would create a representative sample of fruits and why? Explain your answer.

1. `farm_table.take(np.arange(200))`
2. `farm_table.sample(200)`

farm_table.sample(200) works. Because farm_table.take(np.arange(200)) would only take the first 200 elements of the table and it would only pick the Apples in the table. Whereas farm_table.sample(200) picks random items across the table.

Question 3. Let's say we have a fruit basket that can contain at most 200 fruits. We pick 200 fruits (without replacement) from the farm and place it in our fruit basket using the sampling you chose in question 3 above. Write a function called `pick_200_fruits` that simulates this. Specifically, the function should take no arguments and should return an array of 200 fruits selected as per your choice in question 3.

```
In [205]: # Place your answer here. It may contain several lines of code.
          def pick_200_fruits():
              return farm_table.sample(200, with_replacement=False)
          pick_200_fruits()

Out[205]: Fruit Type
          Orange
          Apple
          Apple
          Apple
          Apple
          Orange
          Apple
          Orange
          Apple
          Apple
          ... (190 rows omitted)

In [206]: #: grade 2.3
          _ = ok.grade('q2_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 4. As we mentioned, we're interested in knowing the true proportion of apples in the farm. But we can pick only 200 fruits at a time in our fruit basket. Hence, we simulate this experiment in 500 trials. For each trial, we decide to calculate the proportion of apples in our basket. Simulate the experiment and store the *array* of proportions in the variable `apples_empirical_props`.

```
In [207]: # Place your answer here. It may contain several lines of code.
          apples_empirical_props = make_array()

          for i in np.arange(500):
              prop = pick_200_fruits().where('Fruit Type', are.equal_to('Apple')).num_rows / pic
              apples_empirical_props = np.append(apples_empirical_props, prop)

          apples_empirical_props

Out[207]: array([0.415, 0.465, 0.395, 0.38 , 0.46 , 0.39 , 0.39 , 0.455, 0.405,
                 0.415, 0.43 , 0.355, 0.365, 0.365, 0.455, 0.405, 0.38 , 0.38 ,
                 0.41 , 0.365, 0.365, 0.355, 0.395, 0.46 , 0.39 , 0.325, 0.41 ,
```

```
0.37 , 0.345, 0.395, 0.4  , 0.385, 0.44 , 0.435, 0.415, 0.38 ,
0.4  , 0.365, 0.405, 0.41 , 0.385, 0.42 , 0.415, 0.405, 0.42 ,
0.39 , 0.375, 0.36 , 0.48 , 0.4  , 0.43 , 0.34 , 0.405, 0.4  ,
0.375, 0.435, 0.385, 0.435, 0.375, 0.405, 0.38 , 0.37 , 0.41 ,
0.415, 0.365, 0.36 , 0.43 , 0.425, 0.405, 0.43 , 0.39 , 0.415,
0.435, 0.36 , 0.385, 0.44 , 0.39 , 0.34 , 0.375, 0.44 , 0.36 ,
0.41 , 0.405, 0.39 , 0.435, 0.42 , 0.38 , 0.395, 0.475, 0.375,
0.415, 0.38 , 0.415, 0.425, 0.42 , 0.395, 0.375, 0.375, 0.41 ,
0.36 , 0.41 , 0.375, 0.44 , 0.395, 0.435, 0.44 , 0.4  , 0.395,
0.42 , 0.385, 0.425, 0.445, 0.425, 0.385, 0.415, 0.39 , 0.335,
0.39 , 0.38 , 0.405, 0.46 , 0.48 , 0.405, 0.44 , 0.39 , 0.355,
0.44 , 0.425, 0.39 , 0.375, 0.395, 0.415, 0.38 , 0.41 , 0.365,
0.405, 0.405, 0.395, 0.405, 0.375, 0.43 , 0.38 , 0.355, 0.38 ,
0.355, 0.3  , 0.355, 0.41 , 0.365, 0.385, 0.4  , 0.4  , 0.385,
0.45 , 0.43 , 0.425, 0.36 , 0.39 , 0.41 , 0.425, 0.35 , 0.365,
0.38 , 0.42 , 0.34 , 0.365, 0.425, 0.4  , 0.39 , 0.395, 0.385,
0.375, 0.365, 0.38 , 0.42 , 0.41 , 0.415, 0.355, 0.36 , 0.35 ,
0.42 , 0.36 , 0.425, 0.425, 0.38 , 0.39 , 0.355, 0.375, 0.375,
0.34 , 0.465, 0.38 , 0.39 , 0.44 , 0.42 , 0.425, 0.375, 0.4  ,
0.415, 0.425, 0.385, 0.39 , 0.42 , 0.44 , 0.4  , 0.38 , 0.465,
0.395, 0.42 , 0.375, 0.45 , 0.385, 0.39 , 0.375, 0.34 , 0.415,
0.36 , 0.42 , 0.37 , 0.395, 0.415, 0.39 , 0.41 , 0.345, 0.415,
0.405, 0.415, 0.385, 0.4  , 0.425, 0.465, 0.415, 0.305, 0.375,
0.38 , 0.435, 0.445, 0.415, 0.4  , 0.405, 0.35 , 0.41 , 0.415,
0.395, 0.36 , 0.415, 0.465, 0.435, 0.395, 0.41 , 0.375, 0.37 ,
0.375, 0.405, 0.395, 0.405, 0.36 , 0.42 , 0.42 , 0.37 , 0.37 ,
0.415, 0.425, 0.325, 0.405, 0.38 , 0.36 , 0.35 , 0.385, 0.42 ,
0.42 , 0.39 , 0.4  , 0.375, 0.41 , 0.435, 0.4  , 0.45 , 0.405,
0.415, 0.425, 0.425, 0.38 , 0.39 , 0.41 , 0.415, 0.435, 0.405,
0.36 , 0.37 , 0.37 , 0.41 , 0.44 , 0.445, 0.415, 0.365, 0.435,
0.425, 0.355, 0.39 , 0.395, 0.42 , 0.35 , 0.415, 0.435, 0.455,
0.43 , 0.425, 0.39 , 0.42 , 0.415, 0.425, 0.445, 0.395, 0.42 ,
0.43 , 0.4  , 0.435, 0.425, 0.32 , 0.44 , 0.385, 0.405, 0.42 ,
0.445, 0.4  , 0.355, 0.425, 0.385, 0.405, 0.395, 0.4  , 0.425,
0.355, 0.395, 0.435, 0.375, 0.45 , 0.37 , 0.445, 0.41 , 0.36 ,
0.405, 0.39 , 0.415, 0.365, 0.415, 0.49 , 0.4  , 0.365, 0.405,
0.36 , 0.35 , 0.385, 0.42 , 0.39 , 0.38 , 0.37 , 0.4  , 0.395,
0.395, 0.435, 0.375, 0.445, 0.42 , 0.365, 0.41 , 0.38 , 0.375,
0.385, 0.395, 0.37 , 0.375, 0.365, 0.38 , 0.405, 0.39 , 0.38 ,
0.37 , 0.365, 0.415, 0.385, 0.425, 0.39 , 0.44 , 0.39 , 0.41 ,
0.46 , 0.37 , 0.38 , 0.39 , 0.405, 0.42 , 0.395, 0.39 , 0.48 ,
0.36 , 0.39 , 0.365, 0.36 , 0.385, 0.395, 0.35 , 0.315, 0.38 ,
0.385, 0.395, 0.345, 0.41 , 0.42 , 0.425, 0.42 , 0.4  , 0.375,
0.375, 0.415, 0.315, 0.4  , 0.455, 0.415, 0.4  , 0.365, 0.425,
0.425, 0.41 , 0.455, 0.395, 0.405, 0.405, 0.44 , 0.385, 0.45 ,
0.43 , 0.365, 0.36 , 0.415, 0.41 , 0.375, 0.42 , 0.425, 0.38 ,
0.45 , 0.37 , 0.405, 0.42 , 0.405, 0.355, 0.42 , 0.47 , 0.395,
0.39 , 0.34 , 0.395, 0.37 , 0.39 , 0.395, 0.4  , 0.39 , 0.44 ,
```

```
                0.445, 0.41 , 0.37 , 0.39 , 0.365, 0.435, 0.36 , 0.415, 0.35 ,
                0.405, 0.425, 0.41 , 0.365, 0.39 , 0.37 , 0.38 , 0.385, 0.415,
                0.37 , 0.365, 0.44 , 0.4  , 0.355, 0.4  , 0.375, 0.37 , 0.41 ,
                0.415, 0.36 , 0.365, 0.41 , 0.375, 0.375, 0.44 , 0.395, 0.36 ,
                0.41 , 0.39 , 0.395, 0.405, 0.39 ])
```

```
In [208]: #: grade 2.4
          _ = ok.grade('q2_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 5. Now, compute the average of `apples_empirical_props`. You claim that this average is a good estimate of the proportion of apples in the farm. Store your proportion in `apples_claim_prop`.

```
In [209]: apples_claim_prop = np.mean(apples_empirical_props)
          apples_claim_prop
```

```
Out[209]: 0.39821000000000006
```

```
In [210]: #: grade 2.5
          _ = ok.grade('q2_5')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 6. How far away is your claim from the true proportion of apples. Compute the absolute difference between the two and store it in the variable `error`. Remember that you calculated the true proportion of apples in Question 2

```
In [211]: error = abs(apples_true_prop - apples_claim_prop)
          error
```

```
Out[211]: 0.0017899999999999583

In [212]: #: grade 2.6
          _ = ok.grade('q2_6')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

1.4 3. Broken Phones

A phone manufacturing company claims that it produces phones that are 99% non-faulty. In other words, only 1% of the phones that they manufacture have some fault in them. They open a retail shop in the friendly neighbourhood of La Jolla. Because the phones are cheap and nice, 100 UCSD students have bought phones at this shop. However, it is soon discovered that four of the students had faulty phones. You're angry and argue that the company's claim is wrong. But the company is adament that they are right. You decide to investigate.

Question 1. Assign `null_probabilities` to a two-item *array* such that the first element contains the chance of a phone being non-faulty and the second element contains the chance that the phone is faulty under the **null hypothesis**.

```
In [213]: null_probabilities = make_array(0.99, 0.01)
          null_probabilities

Out[213]: array([0.99, 0.01])

In [214]: #: grade 3.1
          _ = ok.grade('q3_1')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

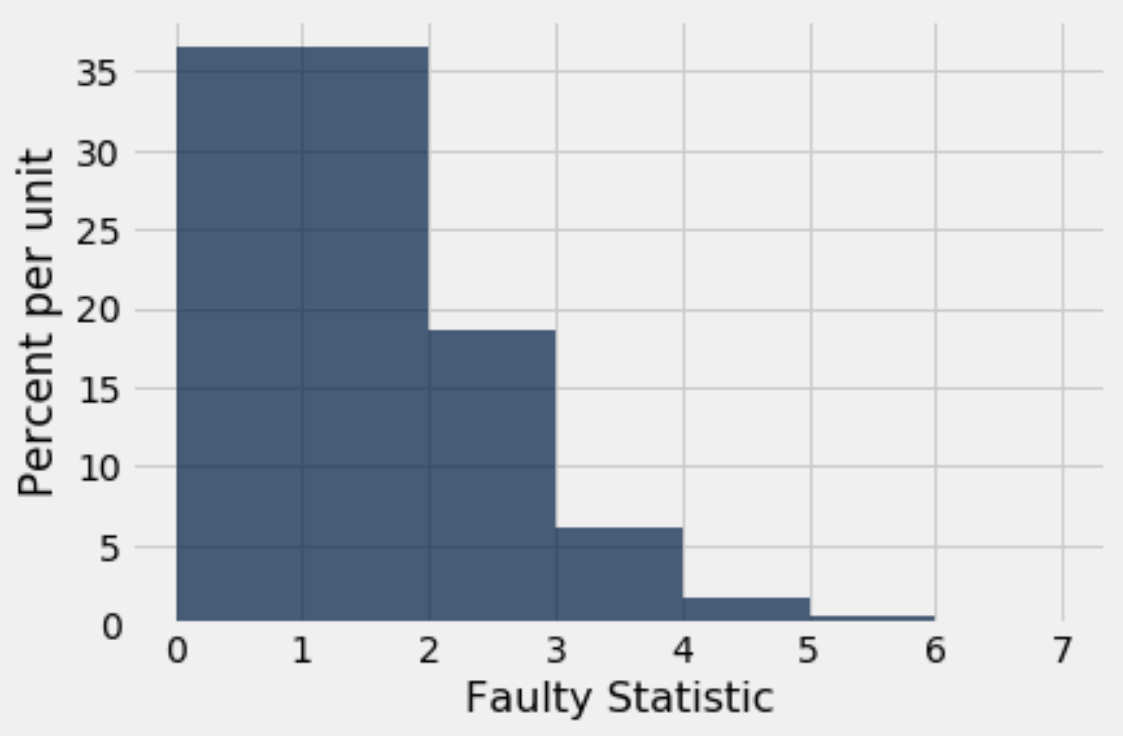
Question 2. Using the function you wrote above, simulate the buying of 100 phones 5,000 times, using the proportions that you assigned to `null_probabilities`. Create an *array* `simulations` with the number of faulty phones in each simulation.

Note that the number of faulty phones in a simulation of sample size x is the proportion of faulty phones in the simulation multiplied by x.

In [215]:
```
# Place your answer here. It may contain several lines of code.

simulations = make_array()

for i in np.arange(5000):
    ind_case = make_array()
    ind_case = sample_proportions(100, null_probabilities)*100
    simulations = np.append(simulations, ind_case.item(1))

simulations
```

Out[215]:
```
array([1., 1., 4., ..., 3., 1., 1.])
```

In [216]:
```
#: Consider the resulting histogram of the fault_statistics array
Table().with_column("Faulty Statistic", simulations).hist(bins=np.arange(8))
```



In [217]:
```
#: grade 3.2
_ = ok.grade('q3_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 3. Using the results of your simulation, calculate an estimate of the p-value, i.e., the probability of observing four or more faulty phones under the null hypothesis. Assign your answer to `p_value_3_3`

```
In [218]: p_value_3_3 = np.count_nonzero(simulations >= 4) / 5000
          p_value_3_3

Out[218]: 0.0222

In [219]: #: grade 3.3
          _ = ok.grade('q3_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 4. Given the results of your above experiment, do you reject the null hypothesis? Explain why.

Our p-value is 0.0222. Since our p-value is less than the significance level of 0.05, we are rejecting the null hypothesis.

1.5 4. Bias towards customers

The insurance company LivLife10 classifies its customers into 3 categories - low-income, mid-income and high-income. The company claims that it treats all of its customers equally and makes no compromises on the quality of the products that it provides. You know that the company has 10,000 customers, 40% of which are low-income customers, 30% mid-income and 30% high-income customers. However, over the past year, 60% of the complaints that the company has received are from low-income customers, 30% from mid-income customers and 10% from high-income customers.

```
In [220]: #: Don't change the below three lines
          type_of_customers = ["low-income", "mid-income", "high-income"]
```

```
proportion_of_customers = np.array([0.4, 0.3, 0.3])
proportion_of_complaints = np.array([0.6, 0.3, 0.1])

insurance_customers = Table().with_columns(
                          "Type of Customers", type_of_customers,
                          "Proportion of Customers", proportion_of_customers,
                          "Proportion of Complaints", proportion_of_complaints)
insurance_customers
```

Out[220]:

Type of Customers	Proportion of Customers	Proportion of Complaints
low-income	0.4	0.6
mid-income	0.3	0.3
high-income	0.3	0.1

You have a suspicion that the insurance company is biased towards its high-income customers. That is, the insurance company is providing a better product to the high-income customers than to others. A better product is one that generates lesser complaints. You decide to test your idea. Your null hypothesis is:

Null hypothesis: The complaints are drawn from the population according to the proportion of customers which are low-, mid-, and high-income.

Question 1. What is the expected proportion of complaints that should be heard from the high-income customers under the null hypothesis? Assign your answer to `complaints_proportion_null`.

```
In [221]: complaints_proportion_null = 0.3
          complaints_proportion_null
```

```
Out[221]: 0.3
```

```
In [222]: #: grade 4.1
          _ = ok.grade('q4_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 2. You wish to check the bias in the insurance company towards different categories of customers. However, there are three categories of customers: high-, mid-, and low-income. Which among the following do you think is **not** a reasonable choice of test statistic for your hypothesis. You may include more than one answer. Append all your choices in a *list* called `unreasonable_test_statistics`. For example, if you think statistics 1, 2, and 3 are unreasonable, you should have `unreasonable_test_statistics = [1,2,3]`

1. Average of the absolute difference between proportion of customers and proportion of corresponding complaints
2. Sum of the absolute difference between proportion of customers and proportion of corresponding complaints
3. The total number of complaints that the company has received in the past year
4. The total variation distance between the probability distribution of customers and the distribution of complaints
5. The absolute difference between the sum of proportion of customers and the sum of proportion of corresponding complaints
6. Average of the sum of the proportion of customers and the proportion of corresponding complaints

```
In [223]: unreasonable_test_statistics = [2,3,5,6]
          unreasonable_test_statistics

Out[223]: [2, 3, 5, 6]

In [224]: #: grade 4.2
          _ = ok.grade('q4_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 3. Say you went ahead with the total variation distance as your test statistic

Write a function called `total_variation_distance` that takes in two probability distributions as arrays and calculates the total variation distance between them.

```
In [225]: # Place your answer here. It may contain several lines of code.
          def total_variation_distance(array_one, array_two):
              return sum(np.abs(array_one - array_two)) / 2

In [226]: #: Use the below code to test your function
          total_variation_distance(np.array([1,0,0]), np.array([0,0,1]))  # Output should be 1.0

Out[226]: 1.0

In [227]: #: grade 4.3
          _ = ok.grade('q4_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 4. Write a simulation which computes the TVD statistic 5000 times on data generated under the null hypothesis. Save the simulated statistics in an *array* called `empirical_tvds`.

Hint: Use `sample_proportions`.

```
In [228]: # Place your answer here. It may contain several lines of code.

          proportions = insurance_customers.column('Proportion of Customers')
          empirical_tvds = make_array()

          for i in np.arange(5000):
              sample_distribution = sample_proportions(10000, proportions)
              new_tvd = total_variation_distance(sample_distribution, proportions)
              empirical_tvds = np.append(empirical_tvds, new_tvd)

          empirical_tvds

Out[228]: array([0.008 , 0.0069, 0.0012, ..., 0.0063, 0.0073, 0.0039])

In [229]: #: grade 4.4
          _ = ok.grade('q4_4')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 5. Calculate the total variation distance in the actual scenario, that is, the observed scenario. Save the result in `observed_tvd`.

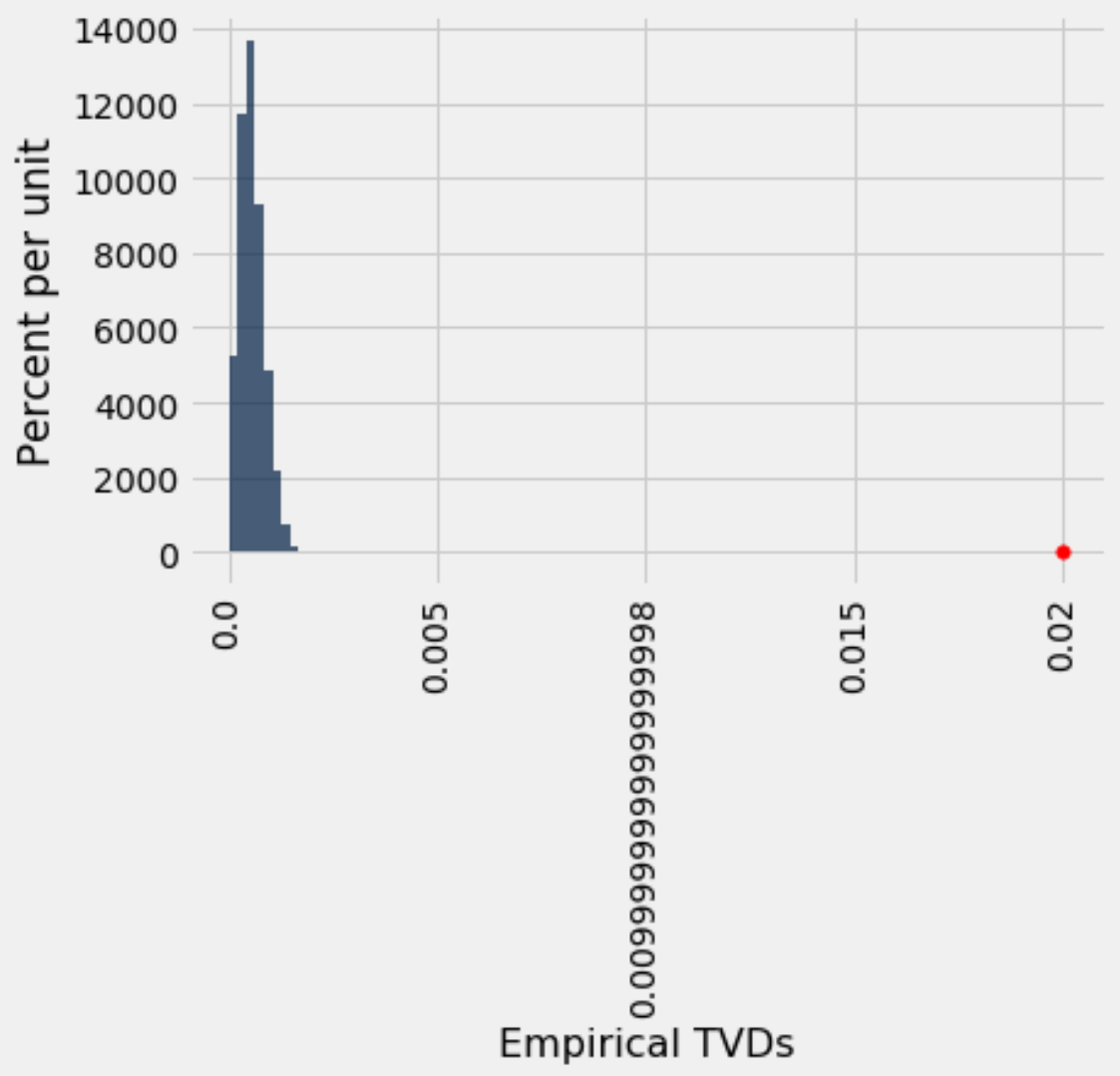
```
In [230]: # proportions = insurance_customers.column('Proportion of Customers')
          proportions_2 = insurance_customers.column('Proportion of Complaints')
          # sample_distribution = sample_proportions(10000, proportions)


          observed_tvd = total_variation_distance(proportions_2, proportions)
          observed_tvd
```

```
Out[230]: 0.19999999999999996
```

Let us plot a histogram of `empirical_tvds` and compare that to our `observed_tvd`

```
In [231]: #: Visualize
          Table().with_column("Empirical TVDs", empirical_tvds).hist()
          plt.scatter(observed_tvd, 0, color='red', s=30)
```

```
Out[231]: <matplotlib.collections.PathCollection at 0x7f9c4eeea6a0>
```




```
In [232]: #: grade 4.5
          _ = ok.grade('q4_5')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 6. Recall that the null hypothesis was that the complaints are drawn from the population according to the proportion of customers which where low-, mid-, and high-income. Looking at the histogram above, do you think it is likely that the null hypothesis is true? Write your answer in the variable `insurance_claim_true`. The value of the boolean variable should be `True` if you agree that the null hypothesis is true, and `False` otherwise.

```
In [233]: insurance_claim_true = False
          insurance_claim_true

Out[233]: False

In [234]: #: grade 4.6
          _ = ok.grade('q4_6')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooo] 100.0% passed
```

Question 7. Does rejecting the null hypothesis in this case prove (or otherwise highly suggest) that the company is biased in its treatment of customers? Why or why not?

This study highly suggests that the company is biased in its treatments. Because our observation's and the sample distribution's total variation distance are very different.

1.6 5. Loaded Die

... And we are back to rolling dice! A loaded die is one that is unfair, i.e., does not have equal probability for each of the outcomes 1–6 (inclusive).

Question 1. Your friend Aby has a model that says that the die is loaded in a way such that the probability of "1" coming up is 0.5 and all the other values have the same probabilities.

Write down Aby's model's distribution as an *array*. It should contain 6 elements, each describing the probability of seeing the corresponding face of the die, and it should sum to 1.

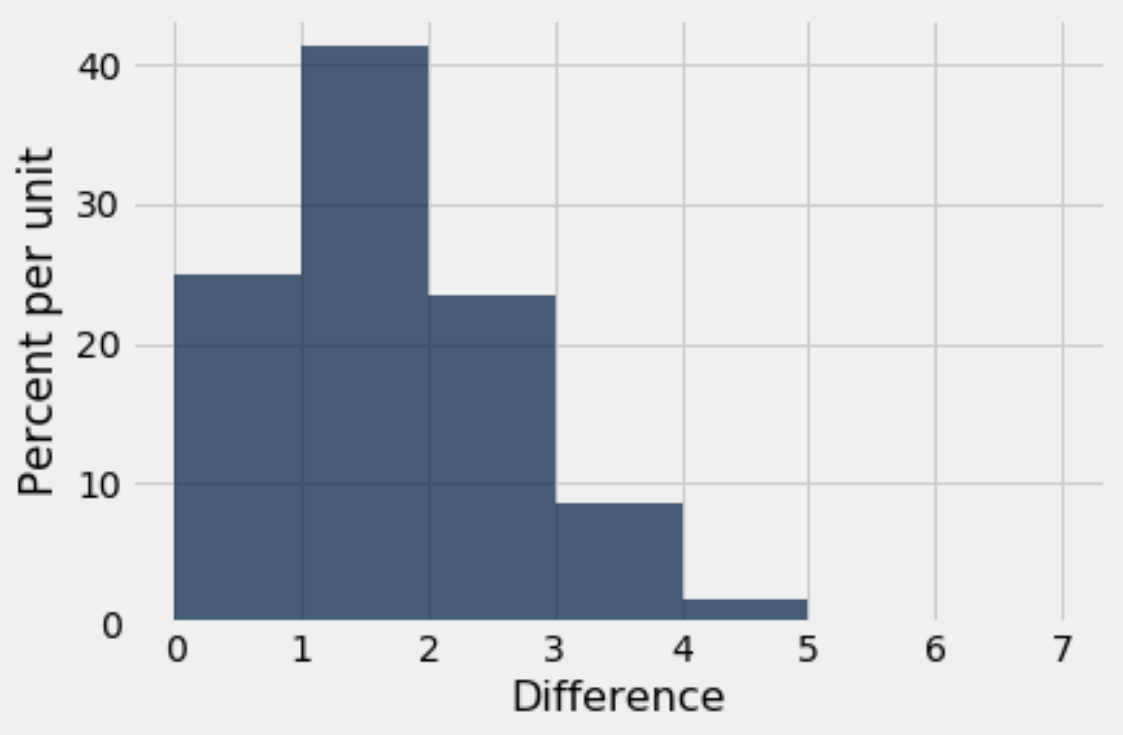
```
In [235]: aby_hypothesis_model_distribution = make_array(0.5, 0.1, 0.1, 0.1, 0.1, 0.1)
          aby_hypothesis_model_distribution
```

```
Out[235]: array([0.5, 0.1, 0.1, 0.1, 0.1, 0.1])

In [236]: #: grade 5.1
          _ = ok.grade('q5_1')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 2. Say we want to test Aby's model. In particular, we wish to test if the probability of "1" coming up is 0.5. We roll the die 10 times and we got "1" a whopping 8 times. We claim that Aby's model is wrong. In order to substantiate our claim, we run a simulation of the die-roll.

Write a simulation and run it 5000 times, maintaining an array `differences` which keeps track of the absolute difference between number of '1's that were seen and the expected number (5) in each simulation.

```
In [237]: # Place your answer here. It may contain several lines of code.

          differences = make_array()
          for i in np.arange(5000):
              case_ind = sample_proportions(10, aby_hypothesis_model_distribution) *10
              differences = np.append(differences, abs(5 - case_ind.item(0)))

          differences

Out[237]: array([1., 0., 1., ..., 1., 0., 2.])

In [238]: #: Visualize with a histogram
          Table().with_column("Difference", differences).hist(bins=np.arange(8))
```




```
In [239]: #: grade 5.2
          _ = ok.grade('q5_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 3. Recall that we saw the die come up "1" eight times. Set the variable `null_hypothesis_boolean` below to `True` if you think Aby's model is plausible or `False` if it should be rejected.

```
In [240]: null_hypothesis_boolean
          null_hypothesis_boolean

Out[240]: True

In [241]: #: grade 5.3
          _ = ok.grade('q5_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

Question 4. Now, we check the p-value of our claim. That is, compute the proportion of times in our simulation that we saw a difference of 3 or more between the number of '1's and the expected number of '1's. Assign your result to `p_value_5_4`

```
In [242]: p_value_5_4 = np.count_nonzero(differences >= 3) / 5000
          p_value_5_4

Out[242]: 0.1038

In [243]: #: grade 5.4
          _ = ok.grade('q5_4')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

To submit:

1. Select `Run All` from the `Cell` menu to ensure that you have executed all cells, including the test cells.
2. Read through the notebook to make sure everything is fine.
3. Submit using the cell below.
4. Save PDF and submit to gradescope

```
In [244]: #: For your convenience, you can run this cell to run all the tests at once!
          import os
          _ = [ok.grade(q[:-3]) for q in os.listdir('tests') if q.startswith('q')]

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
```

```
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
```

```
    Failed: 0
[oooooooooook] 100.0% passed

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[oooooooooook] 100.0% passed
```

1.7 Before submitting, select "Kernel" -> "Restart & Run All" from the menu!

Then make sure that all of your cells ran without error.

```
In [245]: #: submit your notebook
          _ = ok.submit()
```

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


<IPython.core.display.Javascript object>


Saving notebook... Saved 'hw06.ipynb'.
Submit... 100% complete
Submission successful for user: wec149@ucsd.edu
URL: https://okpy.org/ucsd/dsc10/fa18/hw06/submissions/KZgvmR
```

1.8 Don't forget to submit to both OK and Gradescope!