

HW 06 Autograder

● Graded

Student

Sangwon Ji

Total Points

76 / 80 pts

Autograder Score

76.0 / 80.0

Passed Tests

Public Tests

Question 2

Early Submission Bonus

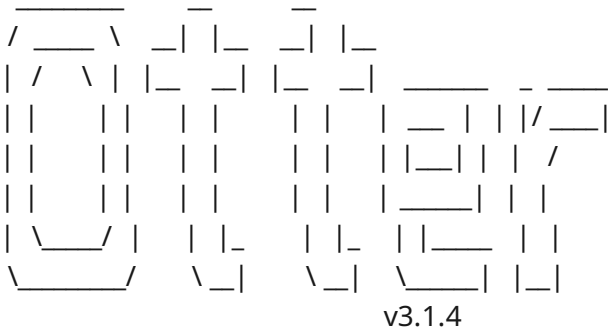
0 / 0 pts

+ 5 pts Early Submission Bonus

✓ + 0 pts No bonus

Autograder Results

Matplotlib is building the font cache; this may take a moment.



Sample Size: 318
Vaccine 1 Percent: 66.35220125786164
<IPython.core.display.HTML object>
<IPython.core.display.HTML object>

----- GRADING SUMMARY -----

Error encountered while trying to verify scores with log:
'TestCaseResult' object has no attribute 'hidden'

Successfully uploaded submissions for: sangwon@berkeley.edu

Total Score: 76.000 / 80.000 (95.000%)

	name	score	max_score
0	Public Tests	NaN	NaN
1	q1_1	4.0	4.0
2	q1_4	4.0	4.0
3	q1_5	4.0	4.0
4	q1_6	4.0	4.0
5	q1_8	4.0	4.0
6	q1_9	4.0	4.0
7	q2_2	4.0	4.0
8	q2_3	4.0	4.0
9	q2_4	4.0	4.0
10	q3_1	4.0	4.0
11	q3_2	4.0	4.0
12	q3_3	4.0	4.0
13	q3_4	0.0	4.0
14	q3_5	4.0	4.0
15	q3_6	4.0	4.0
16	q3_7	4.0	4.0
17	q3_8	4.0	4.0
18	q3_9	4.0	4.0

19	q3_10	4.0	4.0
20	q3_11	4.0	4.0

Public Tests

q1_1 results: All test cases passed!

q1_4 results: All test cases passed!

q1_5 results: All test cases passed!

q1_6 results: All test cases passed!

q1_8 results: All test cases passed!

q1_9 results: All test cases passed!

q2_2 results: All test cases passed!

q2_3 results: All test cases passed!

q2_4 results: All test cases passed!

q3_1 results: All test cases passed!

q3_2 results: All test cases passed!

q3_3 results: All test cases passed!

q3_4 results: All test cases passed!

q3_5 results: All test cases passed!

q3_6 results: All test cases passed!

q3_7 results: All test cases passed!

q3_8 results: All test cases passed!

q3_9 results: All test cases passed!

q3_10 results: All test cases passed!

q3_11 results: All test cases passed!

Submitted Files

In [1]:

```
# Initialize Otter
import otter
grader = otter.Notebook("hw06.ipynb")
```

Homework 6: Testing Hypotheses

Please complete this notebook by filling in the cells provided. Before you begin, execute the previous cell to load the provided tests.

Helpful Resource:

- [Python Reference](#): Cheat sheet of helpful array & table methods used in Data 8!

Recommended Readings:

- [Sampling Methods Guide](#)
- [Testing Hypotheses](#)
- [A/B Testing](#)

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to setup the notebook by importing some helpful libraries. Each time you start your server, you will need to execute this cell again.

For all problems that you must write 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. Otherwise, you will fail tests that you thought you were passing previously!

Deadline:

This assignment is due **Tuesday, 7/19 at 11:59pm PT**. Turn it in by Monday, 7/18 at 11:59pm PT for 5 extra credit points. Late work will not be accepted as per the [policies](#) page.

Note: This homework has hidden tests on it. That means even though tests may say 100% passed, it doesn't mean your final grade will be 100%. We will be running more tests for correctness once everyone turns in the homework.

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.

You should start early so that you have time to get help if you're stuck. Office hours are held Tuesday through Friday. The schedule appears on <http://data8.org/su22/office-hours.html>.

In [2]:

```
# Run this cell to set up the notebook, but please don't change it.

# These lines import the Numpy and Datascience modules.
import numpy as np
from datascience import *
import d8error

# 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. Vaccinations Across The Nation

A vaccination clinic has two types of vaccines against a disease. Each person who comes in to be vaccinated gets either Vaccine 1 or Vaccine 2. One week, everyone who came in on Monday, Wednesday, and Friday was given Vaccine 1. Everyone who came in on Tuesday and Thursday was given Vaccine 2. The clinic is closed on weekends.

Doctor DeNero at the clinic said, "Oh wow, it's just like tossing a coin that lands heads with chance $\frac{3}{5}$. Heads you get Vaccine 1 and Tails you get Vaccine 2."

But Doctor Sahai said, "No, it's not. We're not doing anything like tossing a coin."

That week, the clinic gave Vaccine 1 to 211 people and Vaccine 2 to 107 people. Conduct a test of hypotheses to see which doctor's position is better supported by the data.

Question 1.1. Given the information above, what was the sample size for the data, and what was the percentage of people who got **Vaccine 1**? (4

points)

Note: Your percent should be a number between 0 and 100.

In [3]:

```
sample_size = 211 + 107
percent_V1 = 211/sample_size *100

print(f"Sample Size: {sample_size}")
print(f"Vaccine 1 Percent: {percent_V1}")
```

```
Sample Size: 318
Vaccine 1 Percent: 66.35220125786164
```

In [4]:

```
grader.check("q1_1")
```

Out [4]: q1_1 results: All test cases passed!

Question 1.2. State the null hypothesis. It should reflect the position of either Dr. DeNero or Dr. Sahai. **(4 points)**

Note: Check out [11.3](#) for a refresher on hypotheses.

The probability of getting vaccine 1 is $3/5$. $P = 3/5$. Probability of getting vaccine 2 is $2/5$.

Question 1.3. State the alternative hypothesis. It should reflect the position of the doctor you did not choose to represent in Question 1.2. **(4 points)**

Note: Check out [11.3](#) for a refresher on hypotheses.

The probability of getting vaccine 1 is not $3/5$.

Question 1.4. One of the test statistics below is appropriate for testing these hypotheses. Assign the variable `valid_test_stat` to the number corresponding to the correct test statistic. **(4 points)**

1. percent of heads - 60
2. |percent of heads - 60|
3. percent of heads - 50
4. |percent of heads - 50|

In [5]:

```
valid_test_stat = 2
valid_test_stat
```

Out [5]: 2

In [6]:

```
grader.check("q1_4")
```

Out [6]: q1_4 results: All test cases passed!

Question 1.5. Using your answer from Questions 1.1 and 1.4, find the observed value of the test statistic and assign it to the variable

`observed_statistic`. (4 points)

In [7]:

```
observed_statistic = abs(percent_V1 - 60)
observed_statistic
```

Out [7]: 6.352201257861637

In [8]:

```
grader.check("q1_5")
```

Out [8]: q1_5 results: All test cases passed!

Question 1.6. In order to perform this hypothesis test, you must simulate the test statistic. From the four options below, pick the assumption that is needed for this simulation. Assign `assumption_needed` to an integer corresponding to the assumption. (4 points)

1. The statistic must be simulated under the null hypothesis.
2. The statistic must be simulated under the alternative hypothesis.
3. The statistic must be simulated under both hypotheses.
4. No assumptions are needed. We can just simulate the statistic.

In [9]:

```
assumption_needed = 1
assumption_needed
```

Out [9]: 1

In [10]:

```
grader.check("q1_6")
```

Out [10]: q1_6 results: All test cases passed!

Question 1.7. Simulate 20,000 values of the test statistic under the assumption you picked in Question 1.6. **(4 points)**

As usual, start by defining a function that simulates one value of the statistic. Your function should use `sample_proportions`. (You may find a variable defined in Question 1.1 useful here!) Then, write a `for` loop to simulate multiple values and collect them in the array `simulated_statistics`.

Use as many lines of code as you need. We have included the code that visualizes the distribution of the simulated values. The red dot represents the observed statistic you found in Question 1.5.

In [11]:

```
def one_simulated_statistic():
    sample_distribution = abs(sample_proportions(sample_size, [3/5,2/5]).item(0))
    return 100 * abs(sample_distribution - 0.6)

num_simulations = 20000
simulated_statistics = make_array()
for i in np.arange(num_simulations):
    simulated_statistics= np.append(simulated_statistics,
    one_simulated_statistic())

# Run the this cell a few times to see how the simulated statistic changes
one_simulated_statistic()
```

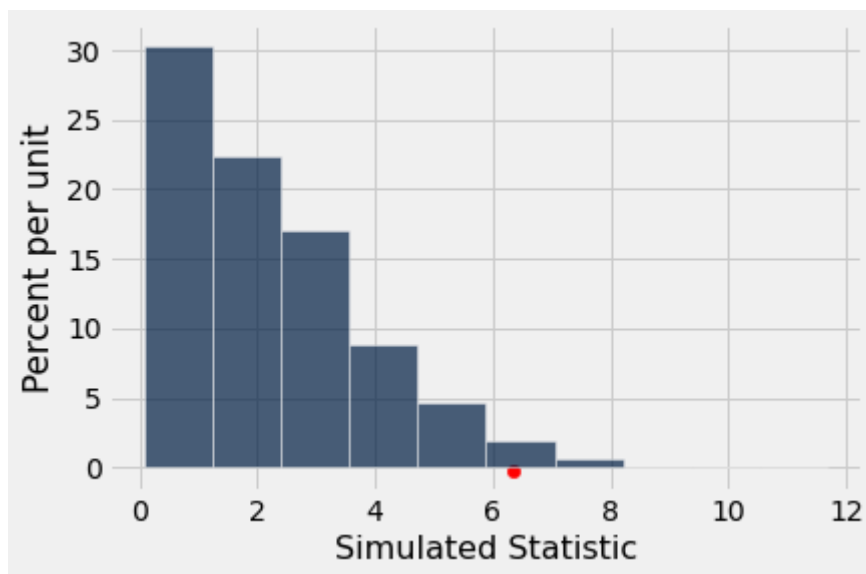
Out [11]:

3.836477987421383

In [12]:

```
# Run this cell to produce a histogram of the simulated statistics

Table().with_columns('Simulated Statistic', simulated_statistics).hist()
plt.scatter(observed_statistic, -0.002, color='red', s=40);
```



Question 1.8. Using `simulated_statistics`, `observed_statistic`, and `num_simulations`, find the empirical p-value based on the simulation. **(4 points)**

In [13]:

```
p_value = np.count_nonzero(simulated_statistics >=
observed_statistic)/num_simulations
p_value
```

Out [13]: 0.01725

In [14]:

```
grader.check("q1_8")
```

Out [14]: q1_8 results: All test cases passed!

Question 1.9. Assign `correct_doctor` to the number corresponding to the correct statement below. Use the 5% cutoff for the p-value. **(4 points)**

1. The data support Dr. DeNero's position more than they support Dr. Sahai's.
2. The data support Dr. Sahai's position more than they support Dr. DeNero's.

As a reminder, here are the two claims made by Dr. DeNero and Dr. Sahai:

Doctor DeNero: "Oh wow, it's just like tossing a coin that lands heads with chance $\frac{3}{5}$. Heads you get Vaccine 1 and Tails you get Vaccine 2."

Doctor Sahai: "No, it's not. We're not doing anything like tossing a coin."

In [15]:

```
correct_doctor = 2
correct_doctor
```

Out [15]: 2

```
In [16]: grader.check("q1_9")
```

Out [16]: q1_9 results: All test cases passed!

2. Using TVD as a Test Statistic

Before beginning this section, please read [this section](#) of the textbook on TVD!

Total variation distance (TVD) is a special type of test statistic that we use when we want to compare two distributions of *categorical data*. It is often used when we observe that a set of observed proportions/probabilities is different than what we expect under the null model.

Consider a six-sided die that we roll 6,000 times. If the die is fair, we would expect that each face comes up $\frac{1}{6}$ of the time. By random chance, a fair die won't always result in equal proportions (that is, we won't get exactly 1,000 of each face). However, if we suspect that the die might be unfair based on the data, we can conduct a hypothesis test using TVD to compare the expected $\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}$ distribution to what is actually observed.

In this part of the homework, we'll look at how we can use TVD to determine the effect that different factors have on happiness.

We will be working with data from the [Gallup World Poll](#) that is presented in the World Happiness Report, a survey of the state of global happiness. The survey ranked 155 countries by overall happiness and estimated the influence that economic production, social support, life expectancy, freedom, absence of corruption, and generosity had on population happiness. The study has been repeated for several years, but we'll be looking at data from the 2016 survey.

Run the cell below to load in the `happiness_scores` table.

```
In [17]: happiness_scores = Table.read_table("happiness_scores.csv")
happiness_scores.show(5)
```

<IPython.core.display.HTML object>

Participants in the study were asked to evaluate their life satisfaction from a scale of 0 (worst possible life) to 10 (best possible life). The responses for each country were averaged to create the `Happiness Score`.

The columns `Economy (GDP per Capita)`, `Family`, `Health (Life Expectancy)`, `Freedom`, `Trust (Government Corruption)`, and `Generosity` estimate the extent to which each factor influences happiness, both for better or for worse. The happiness score is the sum of these factors; the larger a factor is, the more it contributes to overall happiness. [In other words, if you add up all the factors \(in addition to a "Difference from Dystopia" value we excluded in the dataset\), you get the happiness score.](#)

Let's look at the different factors that affect happiness in the United States. Run the cell below to view the row in `us_happiness` that contains data for the United States.

```
In [18]: us_happiness = happiness_scores.where("Country", "United States")
us_happiness
```

```
Out [18]: Country      | Region      | Happiness Rank | Happiness Score | Lower Confidence Int
United States | North America | 13             | 7.104           | 7.02            | 7.188
```

To compare the different factors, we'll look at the proportion of the happiness score that is attributed to each variable. You can find these proportions in the table `us_happiness_factors` after running the cell below.

Note: The factors shown in `us_happiness` don't add up exactly to the happiness score, so we adjusted the proportions to only account for the data we have access to. The proportions were found by dividing each Happiness Factor value by the sum of all Happiness Factor values in `us_happiness`.

```
In [19]: us_happiness_factors = Table().read_table("us_happiness_factors.csv")
us_happiness_factors
```

```
Out [19]: Happiness Factor      | Proportion of Happiness Score
Economy (GDP per Capita)      | 0.344609
Family                        | 0.239455
Health (Life Expectancy)      | 0.178022
Freedom                      | 0.110065
Trust (Government Corruption) | 0.0339773
Generosity                    | 0.0938718
```

Question 2.1. Suppose we want to test whether or not each factor

contributes the same amount to the overall Happiness Score. Define the null hypothesis, alternative hypothesis, and test statistic in the cell below. Feel free to check your work with another student or course staff. **(4 points)**

Note: Please format your answer as follows:

- Null Hypothesis: ...
- Alternative Hypothesis: ...
- Test Statistic: ...

Null hypothesis: Each factor accounts for 1/6 of the total happiness score

Alternative Hypothesis: Each factor does not account for 1/6 of the total

happiness score Test Statistic: Using the TVD, total variation difference, the difference in distribution, $\text{sum}(\text{abs}(\text{observed_distribution} - \text{null_distribution}))/2$

Question 2.2. Write a function `calculate_tvd` that takes in the observed distribution (`obs_dist`) and expected distribution under the null hypothesis (`null_dist`) and calculates the total variation distance. Use this function to set `observed_tvd` to be equal to the observed test statistic. **(4 points)**

In [20]:

```
null_distribution = np.ones(6) * (1/6)

def calculate_tvd(obs_dist, null_dist):
    test = sum(abs(obs_dist-null_dist))/2
    return test

observed_tvd = calculate_tvd(us_happiness_factors.column(1), null_distribution)
observed_tvd
```

Out [20]: 0.26208562431156396

In [21]:

```
grader.check("q2_2")
```

Out [21]: q2_2 results: All test cases passed!

Question 2.3. Create an array called `simulated_tvds` that contains 10,000 simulated values under the null hypothesis. Assume that the original sample consisted of 1,000 individuals. **(4 points)**

Hint: The `sample_proportions` function may be helpful to you. Refer to the [Python Reference Sheet](#) to read up on it!

In [22]:

```
simulated_tvds = make_array()

for i in np.arange(10000):
    one = sample_proportions(1000, null_distribution)
    two = us_happiness_factors.column(1)
    three = np.mean(abs(one-two))

    simulated_tvds= np.append(simulated_tvds, three)
simulated_tvds
```

Out [22]:

```
array([0.09868783, 0.08702854, 0.08836187, ..., 0.09436187, 0.07535449,
       0.08268783])
```

In [23]:

```
grader.check("q2_3")
```

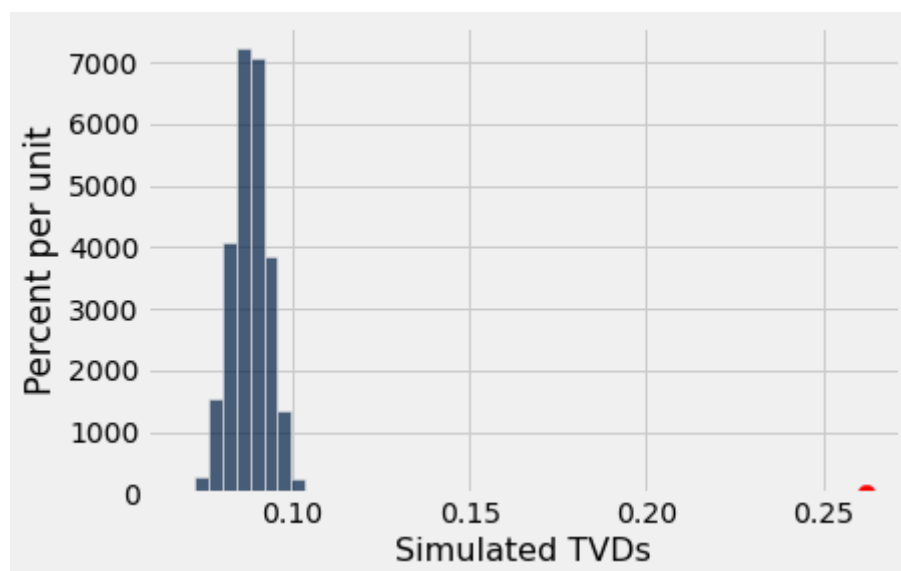
Out [23]:

q2_3 results: All test cases passed!

Run the cell below to plot a histogram of your simulated test statistics, as well as a red dot representing the observed value of the test statistic.

In [24]:

```
Table().with_column("Simulated TVDs", simulated_tvds).hist()
plt.scatter(observed_tvd, 0, color='red', s=70, zorder=2);
plt.show();
```



Question 2.4. Use your simulated statistics to calculate the p-value of your test. Make sure that this number is consistent with what you observed in the histogram above. **(4 points)**

```
In [25]: p_value_tvd = np.count_nonzero(simulated_tvds >= observed_tvd)/10000  
p_value_tvd
```

Out [25]: 0.0

```
In [26]: grader.check("q2_4")
```

Out [26]: q2_4 results: All test cases passed!

Question 2.5. What can you conclude about how each factor contributes to the overall happiness score in the US? Explain your answer using the results of your hypothesis test. Assume a p-value cutoff of 5%. **(4 points)**

The result, test statistic is lower than the cutoff of the p-value which is 5%. So, we reject the null hypothesis. We can conclude and indicate that factors do not account for the overall happiness score at the same proportion, which is proportion of 1/6.

3. Who is Older?

Data scientists have drawn a simple random sample of size 500 from a large population of adults. Each member of the population happened to identify as either "male" or "female". (Though many people identify outside of the gender binary, in this particular population of interest, each member happened to identify as either male or female.) Data was collected on several attributes of the sampled people, including age. The table `sampled_ages` contains one row for each person in the sample, with columns containing the individual's gender identity.

```
In [27]: sampled_ages = Table.read_table('age.csv')  
sampled_ages.show(5)
```

<IPython.core.display.HTML object>

Question 3.1. How many females were there in our sample? Please use the provided skeleton code. **(4 points)**

Hint: Keep in mind that `.group` sorts categories in alphabetical order!

```
In [28]: num_females = sampled_ages.group('Gender').column(1).item(0)
```

```
num_females
```

Out [28]: 260

```
In [29]: grader.check("q3_1")
```

Out [29]: q3_1 results: All test cases passed!

Question 3.2. Complete the cell below so that `avg_male_vs_female` evaluates to `True` if the sampled males are older than the sampled females on average, and `False` otherwise. Use Python code to achieve this. **(4 points)**

```
In [30]: group_mean_tbl = sampled_ages.group('Gender', np.mean)
group_means = group_mean_tbl.column(1) # array of mean ages
avg_male_vs_female = group_means.item(1) > group_means.item(0)
avg_male_vs_female
```

Out [30]: True

```
In [31]: grader.check("q3_2")
```

Out [31]: q3_2 results: All test cases passed!

Question 3.3. The data scientists want to use the data to test whether males are older than females—or, in other words, whether the ages of the two groups have the same distribution. One of the following statements is their null hypothesis and another is their alternative hypothesis. Assign `null_statement_number` and `alternative_statement_number` to the numbers of the correct statements in the code cell below. **(4 points)**

1. In the sample, the males and females have the same distribution of ages; the sample averages of the two groups are different due to chance.
2. In the population, the males and females have the same distribution of ages; the sample averages of the two groups are different due to chance.
3. The age distributions of males and females in the population are different due to chance.
4. The males in the sample are older than the females, on average.
5. The males in the population are older than the females, on average.
6. The average ages of the males and females in the population are different.

```
In [32]: null_statement_number = 2
alternative_statement_number = 5
```



```
In [33]: grader.check("q3_3")
```

Out [33]: q3_3 results: All test cases passed!

Question 3.4. The data scientists have decided to use a permutation test. Assign `permutation_test_reason` to the number corresponding to the reason they made this choice. **(4 points)**

1. Since a person's age shouldn't be related to their gender, it doesn't matter who is labeled "male" and who is labeled "female", so you can use permutations.
2. Under the null hypothesis, permuting the labels in the `sampled_ages` table is equivalent to drawing a new random sample with the same number of males and females as in the original sample.
3. Under the null hypothesis, permuting the rows of `sampled_ages` table is equivalent to drawing a new random sample with the same number of males and females as in the original sample.

```
In [34]: permutation_test_reason = 1  
permutation_test_reason
```

Out [34]: 1

```
In [35]: grader.check("q3_4")
```

Out [35]: q3_4 results: All test cases passed!

Question 3.5. To test their hypotheses, the data scientists have followed our textbook's advice and chosen a test statistic where the following statement is true: Large values of the test statistic favor the alternative hypothesis.

The data scientists' test statistic is one of the two options below. Which one is it? Assign the appropriate number to the variable `correct_test_stat`. **(4 points)**

1. "male age average - female age average" in a sample created by randomly shuffling the male/female labels
2. " $|\text{male age average} - \text{female age average}|$ " in a sample created by randomly shuffling the male/female labels

```
In [36]: correct_test_stat = 1
```

```
correct_test_stat
```

Out [36]: 1

```
In [37]: grader.check("q3_5")
```

Out [37]: q3_5 results: All test cases passed!

Question 3.6. Complete the cell below so that `observed_statistic_ab` evaluates to the observed value of the data scientists' test statistic. Use as many lines of code as you need, and remember that you can use any quantity, table, or array that you created earlier. **(4 points)**

```
In [38]: observed_statistic_ab = group_means.item(1) - group_means.item(0)
observed_statistic_ab
```

Out [38]: 1.314102564102562

```
In [39]: grader.check("q3_6")
```

Out [39]: q3_6 results: All test cases passed!

Question 3.7. Assign `shuffled_labels` to an array of shuffled male/female labels. The rest of the code puts the array in a table along with the data in `sampled_ages`. **(4 points)**

Note: Check out [12.1](#) for a refresher on random permutations.

```
In [40]: shuffled_labels = sampled_ages.sample(with_replacement = False).column(0)
original_with_shuffled_labels = sampled_ages.with_columns('Shuffled Label',
shuffled_labels)
original_with_shuffled_labels
```

Out [40]:

Gender	Age	Shuffled Label
male	23	male
male	29	female
male	29	male
female	49	male
female	33	male
male	31	female
male	60	female
male	38	female

```
female | 60 | male
female | 27 | male
... (490 rows omitted)
```

In [41]: `grader.check("q3_7")`

Out [41]: q3_7 results: All test cases passed!

Question 3.8. The comparison below uses the array `shuffled_labels` from Question 3.7 and the count `num_females` from Question 3.1.

For this comparison, assign the correct number from one of the following options to the variable `correct_q8`. **Pretend this is a midterm problem and solve it without doing the calculation in a code cell. (4 points)**

```
comp = np.count_nonzero(shuffled_labels == 'female') == num_females
```

1. `comp` is set to `True`.
2. `comp` is set to `False`.
3. `comp` is set to `True` or `False`, depending on how the shuffle came out.

In [42]: `correct_q8 = 1`
`correct_q8`

Out [42]: 1

In [43]: `grader.check("q3_8")`

Out [43]: q3_8 results: All test cases passed!

Question 3.9. Define a function `simulate_one_statistic` that takes no arguments and returns one simulated value of the test statistic. We've given you a skeleton, but feel free to approach this question in a way that makes sense to you. Use as many lines of code as you need. Refer to the code you have previously written in this problem, as you might be able to re-use some of it. **(4 points)**

In [44]: `def simulate_one_statistic():`
 `"Returns one value of our simulated test statistic"`
 `shuffled_labels = sampled_ages.sample(with_replacement = False).column(0)`
 `shuffled_tbl = sampled_ages.with_columns('Shuffled Label',`

```
shuffled_labels).drop('Gender')
group_meanss = shuffled_tbl.group(1, np.mean).column(1)
group_means = group_meanss.item(1) - group_meanss.item(0)
return group_means
```

In [45]: `grader.check("q3_9")`

Out [45]: q3_9 results: All test cases passed!

After you have defined your function, run the following cell a few times to see how the statistic varies.

In [46]: `simulate_one_statistic()`

Out [46]: 0.0

Question 3.10. Complete the cell to simulate 4,000 values of the statistic. We have included the code that draws the empirical distribution of the statistic and shows the value of `observed_statistic_ab` from Question 3.6. Feel free to use as many lines of code as you need. **(4 points)**

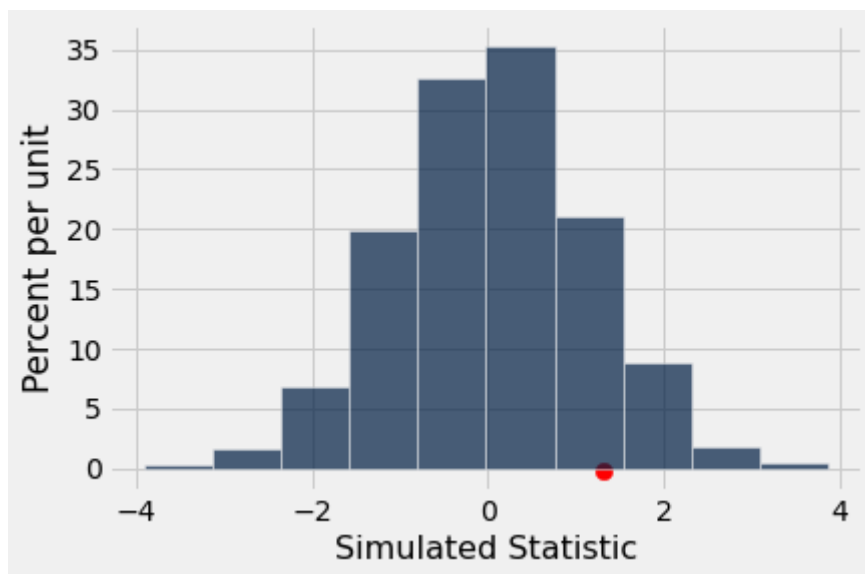
Note: This cell will take around a minute to run.

In [47]:

```
repetitions = 4000

simulated_statistics_ab = make_array()
for i in np.arange(repetitions):
    simulated_statistics_ab = np.append(simulated_statistics_ab,
    simulate_one_statistic())

# Do not change these lines
Table().with_columns('Simulated Statistic', simulated_statistics_ab).hist()
plt.scatter(observed_statistic_ab, -0.002, color='red', s=70);
```



In [48]: `grader.check("q3_10")`

Out [48]: q3_10 results: All test cases passed!

Question 3.11. Use the simulation to find an empirical approximation to the p-value. Assign `p_val` to the appropriate p-value from this simulation. Then, assign `conclusion` to either `null_hyp` or `alt_hyp`. (4 points)

Note: Assume that we use the 5% cutoff for the p-value.

```
In [49]: # These are variables provided for you to use.
null_hyp = 'The data are consistent with the null hypothesis.'
alt_hyp = 'The data support the alternative more than the null.'

p_val = np.count_nonzero(simulated_statistics >= observed_statistic)/repetitions
conclusion = null_hyp

p_val, conclusion # Do not change this line
```

Out [49]: (0.08625, 'The data are consistent with the null hypothesis.')

In [50]: `grader.check("q3_11")`

Out [50]: q3_11 results: All test cases passed!

You're done with Homework 6!

Important submission steps: 1. Run the tests and verify that they all pass. 2. Choose **Save Notebook** from the **File** menu, then **run the final cell**. 3.

Click the link to download the zip file. 4. Go to [Gradescope](#) and submit the zip file to the corresponding assignment. The name of this assignment is "HW 06 Autograder".

It is your responsibility to make sure your work is saved before running the last cell.

To double-check your work, the cell below will rerun all of the autograder tests.

In [51]:

```
grader.check_all()
```

Out [51]:

q1_1 results: All test cases passed!

q1_4 results: All test cases passed!

q1_5 results: All test cases passed!

q1_6 results: All test cases passed!

q1_8 results: All test cases passed!

q1_9 results: All test cases passed!

q2_2 results: All test cases passed!

q2_3 results: All test cases passed!

q2_4 results: All test cases passed!

q3_1 results: All test cases passed!

q3_10 results: All test cases passed!

q3_11 results: All test cases passed!

q3_2 results: All test cases passed!

q3_3 results: All test cases passed!

q3_4 results: All test cases passed!

q3_5 results: All test cases passed!

q3_6 results: All test cases passed!

q3_7 results: All test cases passed!

q3_8 results: All test cases passed!

q3_9 results: All test cases passed!

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

In [150]:

```
# Save your notebook first, then run this cell to export your submission.  
grader.export(pdf=False)
```


<IPython.core.display.HTML object>

▼ .OTTER_LOG

 Download

1	Large file hidden. You can download it using the button above.
---	--

▼ __zip_filename__

 Download

1	hw06_2022_07_19T23_51_17_757713.zip
---	---