# Homework 7: Testing Hypotheses

**Reading**:

- [Testing Hypotheses (https://www.inferentialthinking.com/chapters/11/testing-hypotheses.html)](https://www.inferentialthinking.com/chapters/11/testing-hypotheses.html)

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

Homework 7 is due **Thursday, 3/12 at 11:59pm**. You will receive an early submission bonus point if you turn in your final submission by **Wednesday, 3/11 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 (http://data8.org/sp20/policies.html)](http://data8.org/sp20/policies.html) 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.

In [1]:
```
# 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('hw07.ok')
_ = ok.auth(inline=True)
```

...

## 1. Spam Calls

## Part 1: 781 Fun

Yanay gets a lot of spam calls. An area code is defined to be a three digit number from 200-999 inclusive. In reality, many of these area codes are not in use, but for this question we'll simplify things and assume they all are. **Throughout these questions, you should assume that Yanay's area code is 781.**

**Question 1.** Assuming each area code is just as likely as any other, what's the probability that the area code of two back to back spam calls are 781?

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

In [2]:
```
prob_781 = (1/800)**2 #SOLUTION
prob_781
```

Out[2]: 1.5625e-06

In [3]:
```
# TEST
0 <= prob_781 <= 1
```

Out[3]: True

In [4]:
```
# HIDDEN TEST
np.round(prob_781, 9) == 1.562e-6
```

Out[4]: True

**Question 2.** Rohan already knows that Yanay's area code is 781. Rohan randomly guesses the last 7 digits (0-9 inclusive) of his phone number. What's the probability that Rohan correctly guesses Yanay's number, assuming he's equally likely to choose any digit?

*Note: A phone number contains an area code and 7 additional digits, i.e. xxx-xxx-xxxx*

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

In [5]:
```
prob_yanay_num = 1/(10**7) #SOLUTION
prob_yanay_num
```

Out[5]: 1e-07

In [6]:
```
# TEST
0 <= prob_yanay_num <= 1
```

Out[6]: True

In [7]:
```
# HIDDEN TEST
prob_yanay_num == 1e-07
```

Out[7]: True

Yanay suspects that there's a higher chance that the spammers are using his area code (781) to trick him into thinking it's someone from his area calling him. Ashley thinks that this is not the case, and that spammers are just choosing area codes of the spam calls at random from all possible area codes (*Remember, for this question we're assuming the possible area codes are 200-999, inclusive*). Yanay wants to test his claim using the 50 spam calls he received in the past month.

Here's a dataset of the area codes of the 50 spam calls he received in the past month.

In [8]:
```
# Just run this cell
spam = Table().read_table('spam.csv')
spam
```

Out[8]:

| Area Code |
|---|
| 891 |
| 924 |
| 516 |
| 512 |
| 328 |
| 613 |
| 214 |
| 781 |
| 591 |
| 950 |

... (40 rows omitted)

**Question 3.** Define the null hypothesis and alternative hypothesis for this investigation.

*Hint: Don't forget that your null hypothesis should fully describe a probability model that we can use for simulation later.*

```
BEGIN QUESTION
name: q1_3
manual: true
```

**SOLUTION: Null hypothesis:** Area codes for Yanay's spam calls are chosen at random, and each area code (200-999) is equally likely to be chosen. **Alternative hypothesis:** There's a higher chance of getting a spam call with an area code of 781.

**Question 4.** Which of the following test statistics would be a reasonable choice to help differentiate between the two hypotheses?

*Hint*: For a refresher on choosing test statistics, check out the textbook section on [Test Statistics (https://www.inferentialthinking.com/chapters/11/3/decisions-and-uncertainty.html#Step-2:-The-Test-Statistic)](https://www.inferentialthinking.com/chapters/11/3/decisions-and-uncertainty.html#Step-2:-The-Test-Statistic).

1. The proportion of area codes that are 781 in 50 random calls
2. The total variation distance (TVD) between probability distribution of randomly chosen area codes, and the observed distribution of area codes. (*Remember the possible area codes are 200-999 inclusive*)
3. The probability of getting an area code of 781 out of all the possible area codes.
4. The proportion of area codes that are 781 in 50 random calls divided by 2
5. The number of times you see the area code 781 in 50 spam calls

Assign `reasonable_test_statistics` to an array of numbers corresponding to these test statistics.

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

In [9]:
```
reasonable_test_statistics = make_array(1, 4, 5) # SOLUTION
```

In [10]:
```
# TEST
type(reasonable_test_statistics) == np.ndarray
```

Out[10]: True

In [11]:
```
# TEST
len(reasonable_test_statistics) > 0
```

Out[11]: True

In [12]:
```
# HIDDEN TEST
set(reasonable_test_statistics) == set([1, 4, 5])
```

Out[12]: True

**For the rest of this question, suppose you decide to use the number of times you see the area code 781 in 50 spam calls as your test statistic.**

**Question 5.** Write a function called `simulate` that generates exactly one simulated value of your test statistic under the null hypothesis. It should take no arguments and simulate 50 area codes under the assumption that the result of each area is sampled from the range 200-999 inclusive with equal probability. Your function should return the number of times you saw the 781 area code in those 50 random spam calls.

```
BEGIN QUESTION
name: q1_5
manual: false
```

In [13]:
```
possible_area_codes = np.arange(200,1000) # SOLUTION
def simulate():
    # BEGIN SOLUTION
    random_area_codes = np.random.choice(possible_area_codes, 50)
    return np.count_nonzero(random_area_codes == 781)
    # END SOLUTION

# Call your function to make sure it works
simulate()
```

Out[13]:
```
0
```

In [14]:
```
# TEST
# It looks like your simulation isn't random.
np.std([simulate() for _ in range(1000)]) > 0
```

Out[14]:
```
True
```

In [15]:
```
# HIDDEN TEST
all(possible_area_codes == np.arange(200, 1000))
```

Out[15]:
```
True
```

In [16]:
```
# HIDDEN TEST
np.random.seed(10)
abs(np.mean([simulate() for _ in range(1000)]) - (1/16)) <= 0.02
```

Out[16]:
```
True
```

**Question 6.** Generate 20,000 simulated values of the number of times you see the area code 781 in 50 random spam calls. Assign `test_statistics_under_null` to an array that stores the result of each of these trials.

*Hint*: Use the function you defined in Question 5.

```
BEGIN QUESTION
name: q1_6
manual: false
```

In [17]:
```
test_statistics_under_null = make_array() # SOLUTION
repetitions = 20000 # SOLUTION

# BEGIN SOLUTION
for i in np.arange(repetitions):
    one_statistic = simulate()
    test_statistics_under_null = np.append(test_statistics_under_null, one_
# END SOLUTION

test_statistics_under_null
```

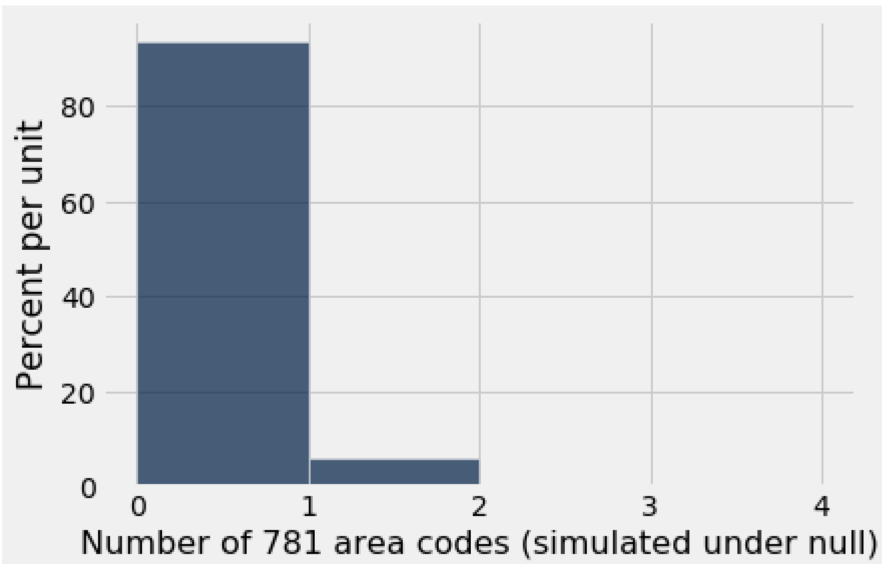
Out[17]:
```
array([0., 0., 0., ..., 0., 0., 0.])
```

In [18]:
```
# TEST
len(test_statistics_under_null) == 20000
```
Out[18]:
```
True
```

**Question 7.** Using the results from Question 6, generate a histogram of the empirical distribution of the number of times you saw the area code 781 in your simulation. **NOTE: Use the provided bins when making the histogram**

```
BEGIN QUESTION
name: q1_7
manual: true
```

In [19]:
```
bins = np.arange(0,5,1) # Use these provided bins
# BEGIN SOLUTION
Table().with_column("Number of 781 area codes (simulated under null)", test
# END SOLUTION
```




**Question 8.** Compute an empirical P-value for this test.

```
BEGIN QUESTION
name: q1_8
manual: false
```

In [20]:
```
# First calculate the observed value of the test statistic from the `spam`
observed_val = spam.where("Area Code", 781).num_rows # SOLUTION
p_value = np.count_nonzero(test_statistics_under_null >= observed_val) / 20
p_value
```
Out[20]:
```
0.0016
```

In [21]:
```
# TEST
0 <= p_value < 1
```

Out[21]: True

In [22]:
```
# HIDDEN TEST
observed_val == 2
```

Out[22]: True

**Question 9.** Suppose you use a P-value cutoff of 1%. What do you conclude from the hypothesis test? Why?

```
BEGIN QUESTION
name: q1_9
manual: true
```

**SOLUTION:** The p-value we observed is below the 1% cutoff, so we conclude that the data support the alternative hypothesis.

# Part 2: Multiple Spammers

Instead of checking if the area code is equal to his own, Yanay decides to check if the area code matches the area code of one of the 8 places he's been to recently, and wants to test if it's more likely to receive a spam call with an area code from any of those 8 places. These are the area codes of the places he's been to recently: 781, 617, 509, 510, 212, 858, 339, 626.

**Question 10.** Define the null hypothesis and alternative hypothesis for this investigation.

*Reminder: Don't forget that your null hypothesis should fully describe a probability model that we can use for simulation later.*

```
BEGIN QUESTION
name: q1_10
manual: true
```

**SOLUTION: Null hypothesis:** Area codes for Yanay's spam calls are chosen at random, and each area code (200-999) is equally likely to be chosen. **Alternative hypothesis:** There's a higher chance of getting a spam call with an area code of one of the 8 places he's been to recently (781, 617, 509, 510, 212, 858, 339, 626).

**Suppose you decide to use the number of times you see any of the area codes of the places Yanay has been to in 50 spam calls as your test statistic.**

**Question 11.** Write a function called `simulate_visited_area_codes` that generates exactly one simulated value of your test statistic under the null hypothesis. It should take no arguments and simulate 50 area codes under the assumption that the result of each area is sampled from the range 200-999 inclusive with equal probability. Your function should return the number of times you saw any of the area codes of the places Yanay has been to in those 50 spam calls.

*Hint*: You may find the textbook [section (https://www.inferentialthinking.com/chapters/11/1/Assessing_Models#Predicting-the-Statistic-Under-the-Model)](https://www.inferentialthinking.com/chapters/11/1/Assessing_Models#Predicting-the-Statistic-Under-the-Model) on the `sample_proportions` function to be useful.

```
BEGIN QUESTION
name: q1_11
manual: false
```

In [23]:
```
model_proportions = make_array(8/800, 792/800) # SOLUTION
def simulate_visited_area_codes():
    # BEGIN SOLUTION
    sampled_props = sample_proportions(50, model_proportions)
    prop_visited = sampled_props.item(0)
    return prop_visited * 50
    # END SOLUTION

# Call your function to make sure it works
simulate_visited_area_codes()
```

Out[23]: 1.0

In [24]:
```
# TEST
# It looks like your simulation isn't random.
np.std([simulate_visited_area_codes() for _ in range(1000)]) > 0
```

Out[24]: True

In [25]:
```
# TEST
# The sum of the items in model proportions should be 1
model_proportions.item(0) + model_proportions.item(1)
```

Out[25]: 1.0

In [26]:
```
# HIDDEN TEST
np.random.seed(10)
abs(np.mean([simulate_visited_area_codes() for _ in range(1000)]) - (1/2))
```

Out[26]: True

**Question 12.** Generate 20,000 simulated values of the number of times you see any of the area codes of the places Yanay has been to in 50 random spam calls. Assign `test_statistics_under_null` to an array that stores the result of each of these trials.

*Hint*: Use the function you defined in Question 11.

```
BEGIN QUESTION
name: q1_12
manual: false
```

In [27]:
```
visited_test_statistics_under_null = make_array() # SOLUTION

repetitions = 20000 # SOLUTION
# BEGIN SOLUTION
for i in np.arange(repetitions):
    visited_statistic = simulate_visited_area_codes()
    visited_test_statistics_under_null = np.append(visited_test_statistics_
# END SOLUTION
visited_test_statistics_under_null
```

Out[27]:
```
array([0., 1., 0., ..., 0., 0., 0.])
```

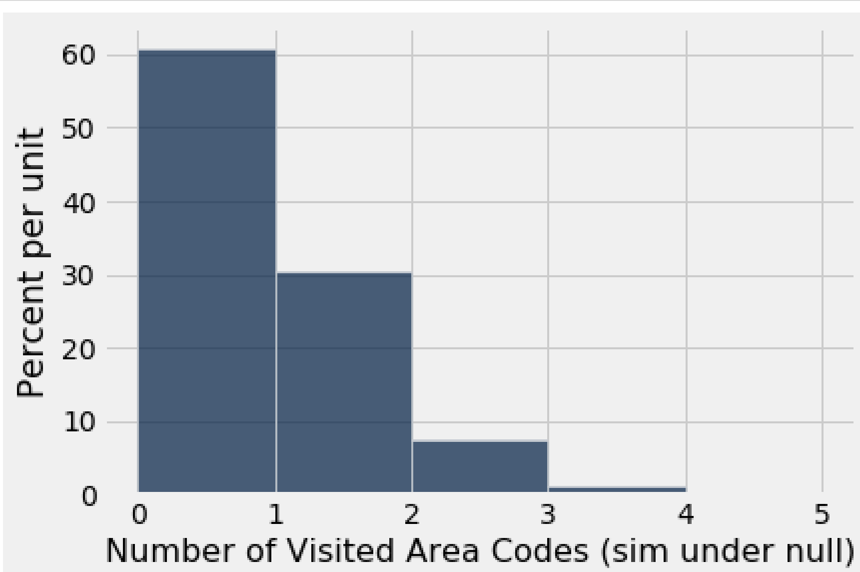
In [28]:
```
# TEST
len(visited_test_statistics_under_null) == 20000
```

Out[28]:
```
True
```

**Question 13.** Using the results from Question 12, generate a histogram of the empirical distribution of the number of times you saw any of the area codes of the places Yanay has been to in your simulation. **NOTE: Use the provided bins when making the histogram**

```
BEGIN QUESTION
name: q1_13
manual: true
```

In [29]:
```
bins_visited = np.arange(0,6,1) # Use these provided bins
# BEGIN SOLUTION
Table().with_column("Number of Visited Area Codes (sim under null)", visite
# END SOLUTION
```



**Question 14.** Compute an empirical P-value for this test.

```
BEGIN QUESTION
name: q1_14
manual: false
```

In [30]:
```
visited_area_codes = make_array(781, 617, 509, 510, 212, 858, 339, 626)
# First calculate the observed value of the test statistic from the `spam`
visited_observed_value = spam.where("Area Code", are.contained_in(visited_a
p_value = np.count_nonzero(visited_test_statistics_under_null >= visited_ob
p_value
```

Out[30]: 0.00165

In [31]:
```
# TEST
0 <= p_value < 1
```

Out[31]: True

In [32]:
```
# HIDDEN TEST
visited_observed_value == 4
```

Out[32]: True

**Question 15.** Suppose you use a P-value cutoff of 0.05% (**Note: that's 0.05%, not our usual cutoff of 5%**). What do you conclude from the hypothesis test? Why?

```
BEGIN QUESTION
name: q1_15
manual: true
```

**SOLUTION:** The p-value we observed is above the 0.05% cutoff, so we conclude that the data support the null hypothesis.

**Question 16.** Is `p_value` :

- (a) the probability that the spam calls favored the visited area codes,
- (b) the probability that they didn't favor, or
- (c) neither

If you chose (c), explain what it is instead.

```
BEGIN QUESTION
name: q1_16
manual: true
```

**SOLUTION:** (c) Neither. The p-value is just the probability that the test statistic (the number of spam calls Yanay got from the eight area codes he's visited) is equal to the value that was observed in the data or is even further in the direction of the alternative if the null hypothesis were true.

**Question 17.** Is 0.05% (the P-value cutoff):

- (a) the probability that the spam calls favored the visited area codes,
- (b) the probability that they didn't favor, or
- (c) neither

If you chose (c), explain what it is instead.

```
BEGIN QUESTION
name: q1_17
manual: true
```

**SOLUTION:** (c) Neither. It's just the cutoff we used to decide whether to reject the null hypothesis. It can be interpreted as the probability of the test rejecting the null hypothesis if the null hypothesis were true.

**Question 18.** Suppose you run this test for 4000 different people after observing each person's last 50 spam calls. When you reject the null hypothesis for a person, you accuse the spam callers of favoring the area codes that person has visited. If the spam callers were not actually favoring area codes that people have visited, can we compute how many times we will incorrectly accuse the spam callers of favoring area codes that people have visited? If so, what is the number? Explain your answer. Assume a 0.05% P-value cutoff.

```
BEGIN QUESTION
name: q1_18
manual: true
```

**SOLUTION:** We will incorrectly accuse the spam callers 2 times, or 0.05% of 4000. Since we're using 0.05% as our P-value cutoff, we have a 0.05% chance of rejecting the null hypothesis when it's actually true. We're running 4000 separate tests, and 0.05% of 4000 is 2.

# Part 3: Practice with A/B Tests

Yanay collects information about this month's spam calls. The table `with_labels` is a sampled table, where the `Area Code Visited` column contains either `"Yes"` or `"No"` which represents whether or not Yanay has visited the location of the area code. The `Picked Up` column is `1` if Yanay picked up and `0` if he did not pick up.

In [33]:
```
# Just run this cell
with_labels = Table().read_table("spam_picked_up.csv")
with_labels
```

Out[33]:

| Area Code Visited | Picked Up |
|---|---|
| No | 0 |
| No | 1 |
| No | 1 |
| Yes | 0 |
| No | 0 |
| No | 0 |
| Yes | 0 |
| No | 1 |
| No | 1 |
| No | 1 |

... (40 rows omitted)

Yanay is going to perform an A/B Test to see whether or not he is more likely to pick up a call from an area code he has visited. Specifically, his null hypothesis is that there is no difference in the distribution of calls he picked up between visited and not visited area codes, with any difference due to chance. His alternative hypothesis is that there is a difference between the two categories, specifically that he thinks that he is more likely to pick up if he has visited the area code. We are going to perform a [permutation test (https://www.inferentialthinking.com/chapters/12/1/AB_Testing.html#Permutation-Test)](https://www.inferentialthinking.com/chapters/12/1/AB_Testing.html#Permutation-Test) to test this. Our test statistic will be the difference in proportion of calls picked up between the area codes Yanay visited and the area codes he did not visit.

**Question 19.** Complete the `difference_in_proportion` function to have it calculate this test statistic, and use it to find the observed value. The function takes in a sampled table which can be any table that has the same columns as `with_labels`. We'll call `difference_in_proportion` with the sampled table `with_labels` in order to find the observed difference in proportion.

```
BEGIN QUESTION
name: q1_19
manual: false
```

In [34]:
```
def difference_in_proportion(sample):
    # Take a look at the code for `proportion_visited` and use that as a
    # hint of what `proportions` should be assigned to
    proportions = sample.group("Area Code Visited", np.mean) # SOLUTION
    proportion_visited = proportions.where("Area Code Visited", "Yes").colu
    proportion_not_visited = proportions.where("Area Code Visited", "No").c
    return proportion_visited - proportion_not_visited # SOLUTION

observed_diff_proportion = difference_in_proportion(with_labels)
observed_diff_proportion
```

Out[34]: 0.21904761904761905

In [35]:
```
# TEST
-1 <= observed_diff_proportion <= 1
```

Out[35]: True

In [36]:
```
# TEST
# The observed difference in proportion should be about 0.219
np.round(observed_diff_proportion, 3) == 0.219
```

Out[36]: True

**Question 20.** To perform a permutation test we shuffle the labels, because our null hypothesis is that the labels don't matter because the distribution of calls he picked up between visited and not visited area codes come from same underlying distribution. The labels in this case is the `"Area Code Visited"` column containing `"Yes"` and `"No"`.

Write a function to shuffle the table and return a test statistic using the function you defined in question 19.

*Hint: To shuffle labels, we sample without replacement and then replace the appropriate column with the new shuffled column.*

```
BEGIN QUESTION
name: q1_20
manual: false
```

In [37]:
```
def simulate_one_stat():
    shuffled = with_labels.sample(with_replacement=False) # SOLUTION
    original_with_shuffled_labels = with_labels.with_column("Area Code Visi
    return difference_in_proportion(original_with_shuffled_labels)

one_simulated_test_stat = simulate_one_stat()
one_simulated_test_stat
```

Out[37]: 0.21904761904761905

In [38]:
```
# TEST
-0.75 <= one_simulated_test_stat <= 0.75
```

Out[38]:
```
True
```

**Question 21.** Generate 1,000 simulated test statistic values. Assign `test_stats` to an array that stores the result of each of these trials.

*Hint*: Use the function you defined in Question 20.

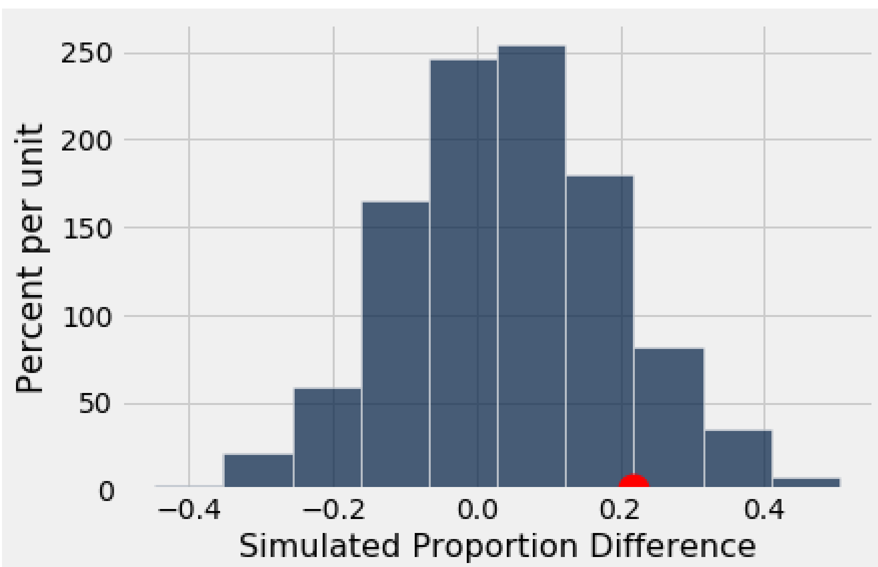
We also provided code that'll generate a histogram for you after generating a 1000 simulated test statistic values.

```
BEGIN QUESTION
name: q1_21
manual: true
```

In [39]:
```
trials = 1000 # SOLUTION
test_stats = make_array() # SOLUTION

# BEGIN SOLUTION
for i in np.arange(trials):
    one_stat = simulate_one_stat()
    test_stats = np.append(test_stats, one_stat)
# END SOLUTION

# here's code to generate a histogram of values and the red dot is the obse
Table().with_column("Simulated Proportion Difference", test_stats).hist("Si
plt.plot(observed_diff_proportion, 0, 'ro', markersize=15);
```




**Question 22.** Compute the empirical p-value for this test, and assign it to `p_value_ab` .

```
BEGIN QUESTION
name: q1_22
manual: false
```

In [40]:
```
p_value_ab = np.count_nonzero(test_stats >= observed_diff_proportion) / ler
p_value_ab
```

Out[40]:
```
0.118
```

In [41]:
```
# TEST
p_value_ab > 0.05
```

Out[41]:
```
True
```

For `p_value_ab` , you should be getting a value around 10-15%. If our p-value cutoff is 5%, the data is more consistent with the null hypothesis - that there is no difference in the distribution of calls Yanay picked up between visited and not visited area codes.

## 2. Mid-Semester Survey

Once you have submitted, please also take the time to complete the Mid-Semester Survey! We really appreciate your honest feedback and it helps us improve the course!

The Mid-Semester survey is here: https://docs.google.com/forms/d/e/1FAIpQLSdyq7HSgY-pRDSOyIcHKPT8Ojfb4veVjUKG10AQReH6UBG_PQ/viewform?usp=sf_link (https://docs.google.com/forms/d/e/1FAIpQLSdyq7HSgY-pRDSOyIcHKPT8Ojfb4veVjUKG10AQReH6UBG_PQ/viewform?usp=sf_link)

**Question 1.** Fill out the mid-semester survey linked above. Right before submitting, a special string will be displayed. Set `special_string` to the special string at the end of the form.

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

In [42]:
```
special_string = "happy math.pi day on 3/14!" # SOLUTION
```

In [43]:
```
# TEST
special_string
```

Out[43]:
```
'happy math.pi day on 3/14!'
```

## 3. 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 [(https://okpy.org/)](https://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!

In [44]:
```
_ = ok.submit()
```

...

In [45]:
```
# 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
print("Finished running all tests.")
```

...