hw07_master

November 1, 2019

1 Homework 7: Testing Hypotheses

Reading: * 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 7 is due **Friday, 10/18 at 11:59pm**. You will receive an early submission bonus point if you turn in your final submission by **Thursday, 10/17 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('hw07.ok')
_ = ok.auth(inline=True)
```

1.1 1. Landing a Spacecraft

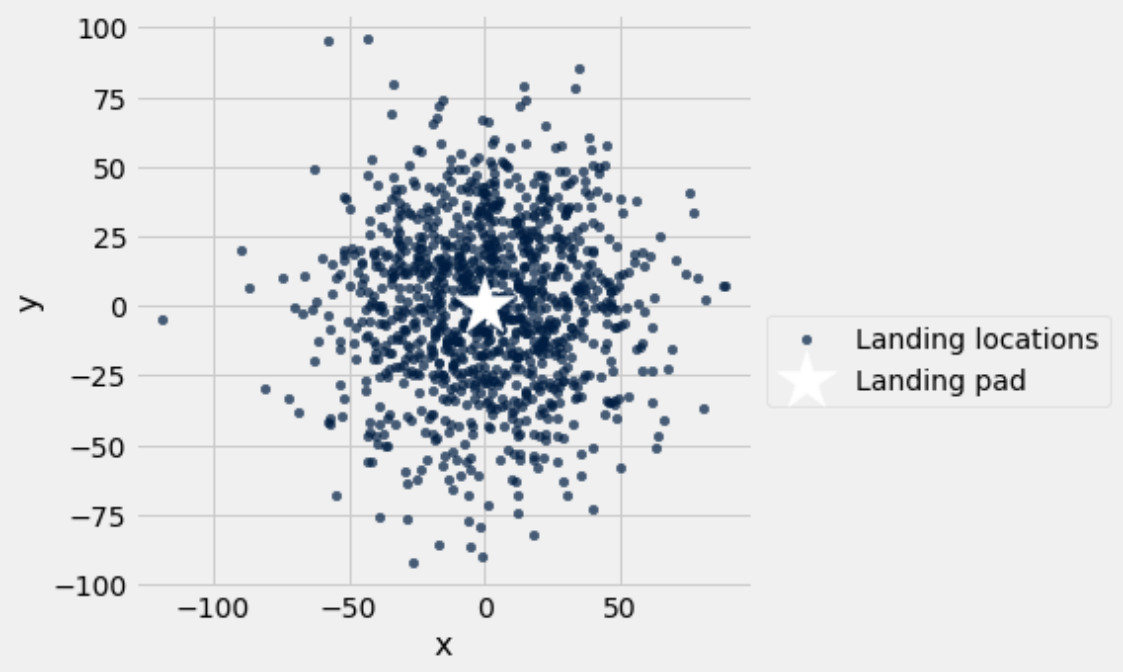
(Note: This problem describes something that's close to a real story with a very exciting video, but the details have been changed somewhat.)

SpaceY, a company that builds and tests spacecraft, is testing a new reusable launch system. Most spacecraft use a "first stage" rocket that propels a smaller payload craft away from Earth, then falls back to the ground and crashes. SpaceY's new system is designed to land safely at a landing pad at a certain location, ready for later reuse. If it doesn't land in the right location, it crashes, and the very, very expensive vehicle is destroyed.

SpaceY has tested this system over 1000 times. Ordinarily, the vehicle doesn't land exactly on the landing pad. For example, a gust of wind might move it by a few meters just before it lands. It's reasonable to think of these small errors as random. That is, the landing locations are drawn from some distribution over locations on the surface of Earth, centered around the landing pad.

Run the next cell to see a plot of those locations.

[2]:
```
ordinary_landing_spots = Table.read_table("ordinary_landing_spots.csv")
ordinary_landing_spots.scatter("x", label="Landing locations")
plt.scatter(0, 0, c="w", s=1000, marker="*", label="Landing pad")
plt.legend(scatterpoints=1, bbox_to_anchor=(1.6, .5));
```



During one test, the vehicle lands far away from the landing pad and crashes. The coordinates of the crash site are contained in the array `crash_site`, where the first item contains the x-coordinate and the second item contains the y-coordinate.

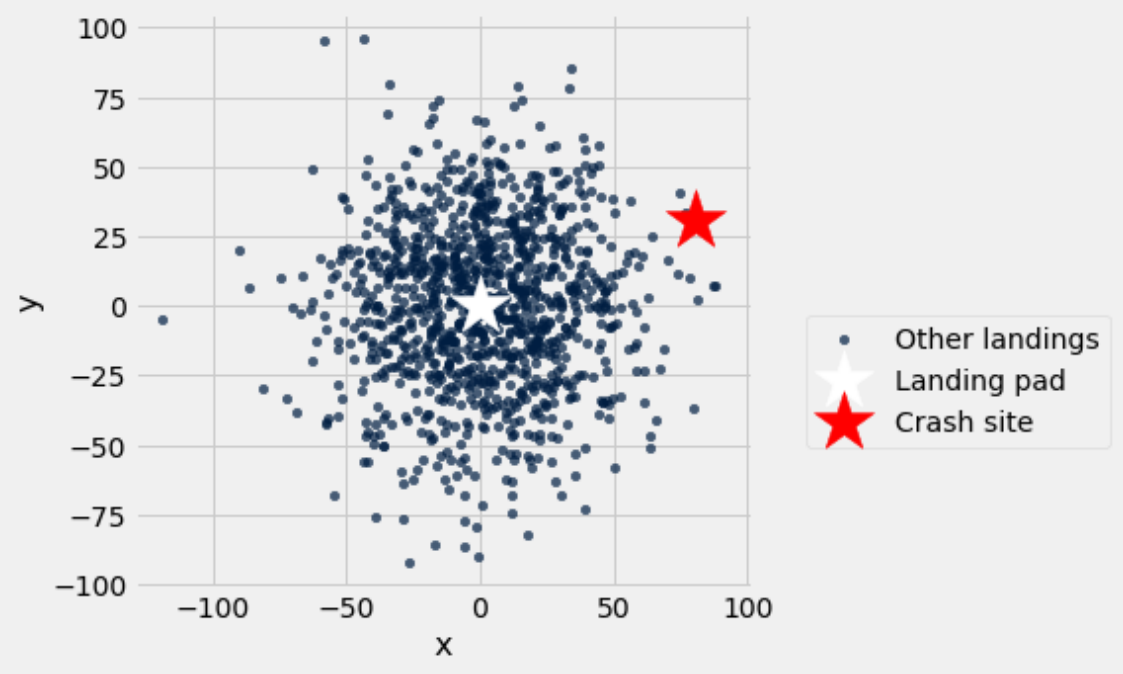
[5]:
```
crash_site = make_array(80.59, 30.91)
crash_site
```

[5]:
```
array([80.59, 30.91])
```

SpaceY investigators suspect there was a problem unique to this landing, a problem that wasn't part of the ordinary pattern of variation in landing locations. They think a software error in the guidance system caused the craft to incorrectly attempt to land at a spot other than the landing pad. The guidance system engineers think there was nothing out of the ordinary in this landing, and that there was no special problem with the guidance system.

Run the cell below to see a plot of the 1100 ordinary landings and the crash.

[6]:
```
ordinary_landing_spots.scatter("x", label="Other landings")
plt.scatter(0, 0, c="w", s=1000, marker="*", label="Landing pad")
plt.scatter(crash_site.item(0), crash_site.item(1), marker="*", c="r", s=1000,␣
 ↪label="Crash site")
plt.legend(scatterpoints=1, bbox_to_anchor=(1.6, .5));
```



Question 1. Let's use distance from the landing pad as our test statistic, to compare the suspicious landing to the distribution of landings we believe to be normal. Write a function called `landing_distance`. It should take two arguments: an "x" location and a "y" location (both numbers), and should return the euclidean distance from the landing spot to the origin!

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

[7]:
```
def landing_distance(x_coordinate, y_coordinate):
    return (x_coordinate**2 + y_coordinate**2)**0.5 #SOLUTION
```

Question 2. Use the `landing_distance` function you wrote above to compute the landing distance for the crash site. Assign this distance to `observed_test_stat`.

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

[11]:
```
observed_test_stat = landing_distance(crash_site.item(0), crash_site.item(1))␣
↪#SOLUTION
observed_test_stat
```

[11]:
```
86.31440320131978
```

Question 3. Next, assign `distances` to an array containing the computed landing distances for each landing in `ordinary_landing_spots`. Use the `landing_distance` function you wrote above to compute these distances!

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

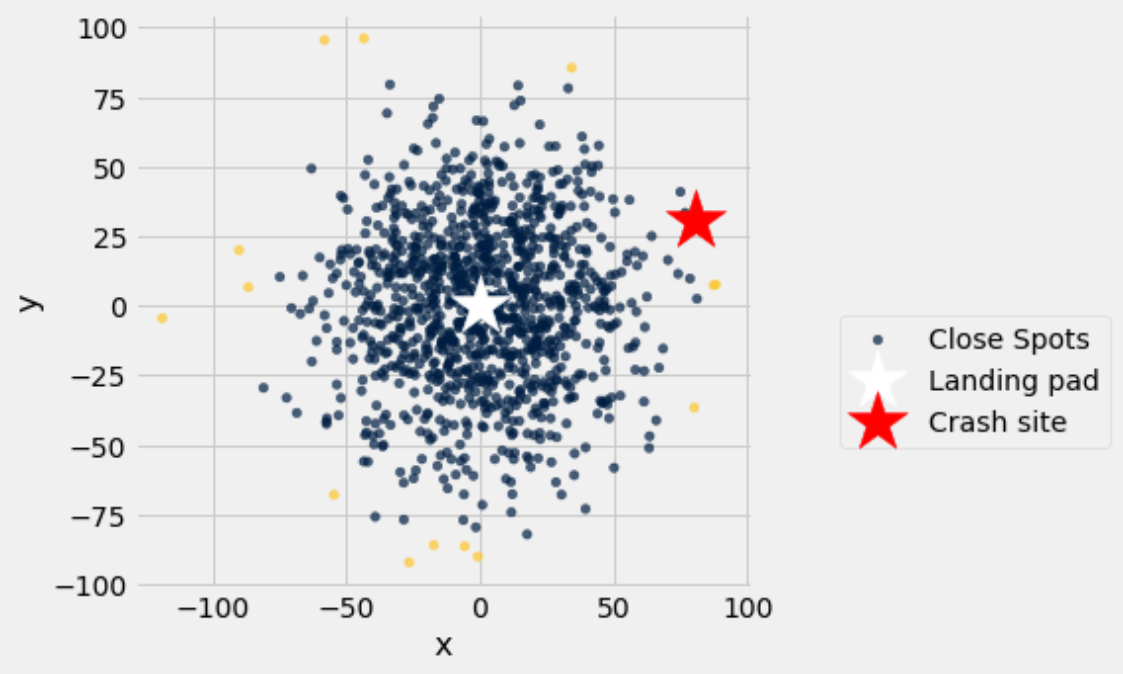
[14]:
```
# BEGIN SOLUTION
distances = make_array()
repetitions = ordinary_landing_spots.num_rows

for i in np.arange(repetitions):
    distance = landing_distance(
        ordinary_landing_spots.column('x').item(i),
        ordinary_landing_spots.column('y').item(i))
    distances = np.append(distances, distance)

# here is another possible solution

distances = ordinary_landing_spots.apply(landing_distance, 'x', 'y')
# END SOLUTION

distances
```

[14]:
```
array([ 7.0373003 , 37.04323884, 28.24132651, …, 20.84023432,
       32.48926398, 23.88691421])
```

Run the next cell below to see the points color-coded by distance! Gold points correspond to landing locations that are as far or farther away than the Crash Site!

[16]:
```
spots_with_distances = ordinary_landing_spots.with_column("Distances",␣
↪distances >= observed_test_stat)
```

```
spots_with_distances.scatter('x', 'y', colors = 'Distances', label='Close␣
↪Spots')
plt.scatter(0, 0, c="w", s=1000, marker="*", label="Landing pad")
plt.scatter(crash_site.item(0), crash_site.item(1), marker="*", c="r", s=1000,␣
↪label="Crash site")
plt.legend(scatterpoints=1, bbox_to_anchor=(1.6, .5));
```



Question 4. What proportion of points were as far or farther away than the specific landing? Assign this value to `proportion_of_points` below.

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

[17]:
```
proportion_of_points = np.count_nonzero(distances >= observed_test_stat) /␣
↪len(distances) # SOLUTION
proportion_of_points
```

[17]:
```
0.012727272727272728
```

Question 5. How can we interpret the value of `proportion_of_points`? Assign `interpretations` to an array with the number(s) of all valid interpretations. 1. There's a `proportion_of_points` chance that this was an abnormal landing. 2. There's a 1 - `proportion_of_points` chance that this was a normal landing. 3. If `proportion_of_points` is less than or equal to a predetermined cutoff, the data is more consistent with the hypothesis that the crash site **did not** come from the ordinary pattern of variation in landing locations.

BEGIN QUESTION
name: q1_5
manual: false

[20]:
```
interpretations = make_array(3) # SOLUTION
```

1.2 2. Catching Cheaters

Suppose you are a casino owner, and your casino runs a very simple game of chance. The dealer flips a coin. The customer wins $$$9 from the casino if it comes up heads and loses $$$10 if it comes up tails.

Question 1. Assuming no one is cheating and the coin is fair, if a customer plays twice, what is the chance they make money?

BEGIN QUESTION
name: q2_1
manual: false

[6]:
```
p_winning_after_two_flips = 0.5 ** 2 #SOLUTION
```

A certain customer plays the game 20 times and wins 13 of the bets. You suspect that the customer is cheating! That is, you think that their chance of winning is higher than the normal chance of winning.

You decide to test your hunch using the outcomes of the 20 games you observed.

Question 2. Define the null hypothesis and alternative hypothesis for this investigation.

BEGIN QUESTION
name: q2_2
manual: true

SOLUTION: Null hypothesis: This customer wasn't cheating. That is, they had a 50% chance of winning each bet during the 20 games. **Alternative hypothesis:** This customer had a chance greater than 50% of winning each bet during the 20 games.

Question 3. Which of the following test statistics would be a reasonable choice to help differentiate between the two hypotheses? You will see the outcome of 20 games.

Hint: For a refresher on choosing test statistics, check out this section on Test Statistics.

1. Whether there is at least one win.
2. Whether there is at least one loss.
3. The number of wins.
4. The number of wins minus 10.
5. The total variation distance between the probability distribution of a fair coin and the observed distribution of heads and tails.
6. The total amount of money that the customer won.

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

BEGIN QUESTION
name: q2_3
manual: false

[9]:
```
reasonable_test_statistics = make_array(3, 4, 6) # SOLUTION
```

 Suppose you decide to use the number of wins as your test statistic.

Question 4. 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 20 games under the assumption that the result of each game is sampled from a fair coin (one that lands heads or lands tails with 50% chance). Your function should return the number of wins in those 20 games.

Hint: You may find the textbook section on the `sample_proportions` function to be useful.

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

[13]:
```
fair_coin = make_array(0.5, 0.5) # SOLUTION
def simulate():
    # BEGIN SOLUTION
    proportion_heads = sample_proportions(20, fair_coin).item(0)
    return 20*proportion_heads
    # END SOLUTION

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

[13]:
```
11.0
```

Question 5. Generate 10,000 simulated values of the number of wins in 20 games. 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 4.

```
BEGIN QUESTION
name: q2_5
manual: false
```

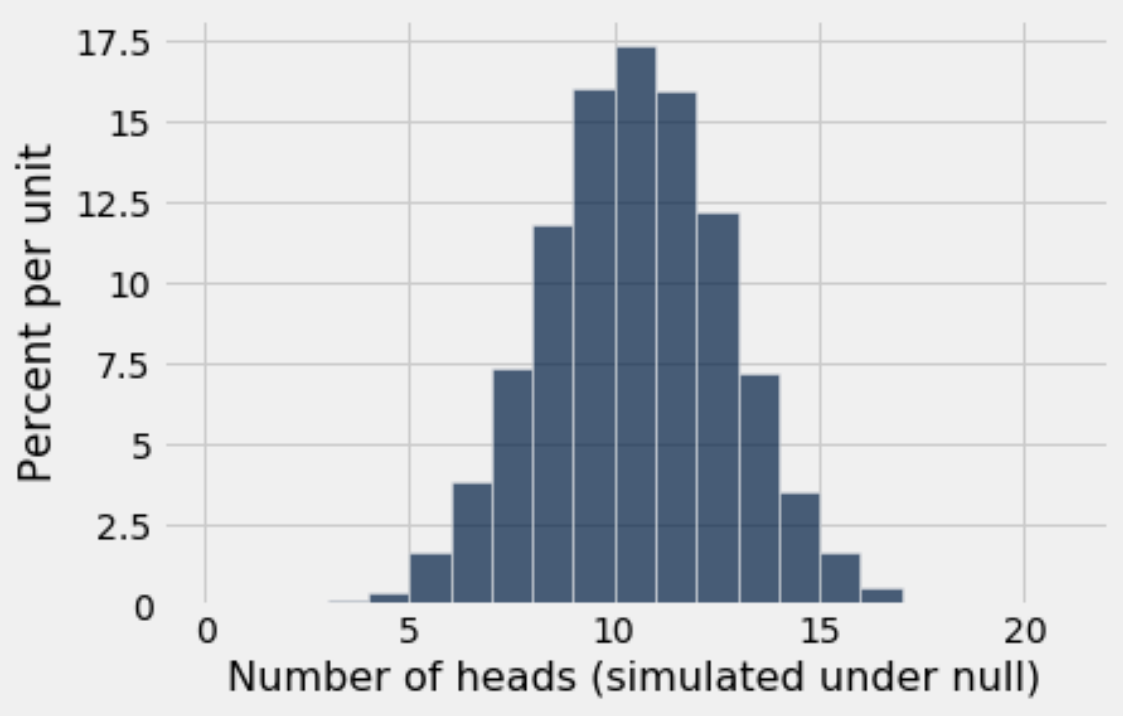
[17]:
```
test_statistics_under_null = make_array() # SOLUTION
repetitions = 10000 # SOLUTION

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

test_statistics_under_null
```

[17]:
```
array([ 8., 12., 12., …,  4., 12., 11.])
```

Question 6. Using the results from Question 5, generate a histogram of the empirical distribution of the number of wins in 20 games.

```
BEGIN QUESTION
name: q2_6
manual: true
```

[14]:
```
# BEGIN SOLUTION
Table().with_column("Number of heads (simulated under null)",␣
 ↪test_statistics_under_null).hist("Number of heads (simulated under null)",␣
 ↪bins=np.arange(22))
# END SOLUTION
```



Question 7. Compute an empirical P-value for this test.

Hint: Would you expect large or small values of the test statistic under the alternative hypothesis?

```
BEGIN QUESTION
name: q2_7
manual: false
```

[19]:
```
p_value = np.count_nonzero(test_statistics_under_null >= 13) / 10000 # SOLUTION
p_value
```

[19]:
```
0.1301
```

Question 8. Suppose you use a P-value cutoff of 1%. What do you conclude from the hypothesis test? Why?

```
BEGIN QUESTION
name: q2_8
manual: true
```

SOLUTION: With our 1% cutoff, we would decide that our result isn't "strange" enough for us to reject the null hypothesis. We do not find the result of the test statistically significant, and we fail to reject the null hypothesis.

Question 9. Is `p_value` the probability that the customer cheated, or the probability that the customer didn't cheat, or neither? If neither, what is it?

```
BEGIN QUESTION
name: q2_9
manual: true
```

SOLUTION: Neither. A P-value is not a probability that any hypothesis about the world is correct. It is just the probability that the test statistic 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 10. Is 1% (the P-value cutoff) the probability that the customer cheated, or the probability that the customer didn't cheat, or neither? If neither, what is it?

```
BEGIN QUESTION
name: q2_10
manual: true
```

SOLUTION: 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 11. Suppose you run this test for 400 different customers after observing each customer play 20 games. When you reject the null hypothesis for a customer, you accuse that customer of cheating. If no customers were actually cheating, can we compute how many we will incorrectly accuse of cheating? If so, what is the number? Explain your answer. Assume a 1% P-value cutoff.

```
BEGIN QUESTION
name: q2_11
```

manual: true

SOLUTION: 4 customers, or 1% of 400. Since we're using 1% as our P-value cutoff, we have a 1% chance of rejecting the null hypothesis when it's actually true. We're running 400 separate tests, and 1% of 400 is 4.

1.3 3. 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://forms.gle/zWHPFcMvTekFEJrT8

Question 1. Fill out the mid-semester survey linked above. Once you have submitted, a secret word will be displayed. Set `secret_word` to the secret string at the end of the form.

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

[5]:
```
secret_word = "boo-lean" # SOLUTION
```

1.4 4. 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.")
```