Name: Allan Gongora

Section: 0131

Homework 7: Testing Hypotheses

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

In [1]:
```
pip install gofer-grader
```

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-packages
(1.1.0)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages (from
gofer-grader) (3.0.3)
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-packages (fr
om gofer-grader) (2.11.2)
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages (fro
m gofer-grader) (6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-packa
ges (from jinja2->gofer-grader) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
# 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)

# These lines load the tests.

from gofer.ok import check
```

Recommended Reading:

- Testing Hypotheses 1) For all problems that you must write explanations and sentences for, you **must** provide your answer in the designated space. This can include: A) Sentence reponses to questions that ask for an explanation B) Numeric responses to multiple choice questions C) Programming code

2) 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!

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing. Then click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

1. 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's heads and loses $10 if it's tails.

Question 1.1
Assuming no one is cheating and the coin is fair, if a customer plays twice, what is the chance they make money?

In [3]:
```
p_winning_after_two_flips = .5**2
```

In [4]:
```
check('tests/q1_1.py')
```

Out[4]: All tests passed!

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 1.2
Define the null hypothesis and alternative hypothesis for this investigation.

Null hypothesis: Chance of getting heads is .5 because this is a fair coin

Alternative hypothesis: Chance of getting heads is not .5 and therefore is unfair

Question 1.3
Given the outcome of 20 games, which of the following test statistics would be a reasonable choice for this hypothesis test?

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 the number of losses.
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 a **list** of numbers corresponding to these test statistics.

In [7]:
```
reasonable_test_statistics = [3, 4, 5]
```

In [8]:
```
check('tests/q1_3.py')
```

Out[8]: All tests passed!

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

Question 1.4

Write a function called `simulate` that generates exactly one simulation of your test statistic under the Null Hypothesis. It should take no arguments. It should return the number of wins in 20 games simulated under the assumption that the result of each game is sampled from a fair coin that lands heads or lands tails with a 50% chance.

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

In [13]:
```
def simulate() -> int:
    wins = 0
    for _ in range(20):
        if np.random.choice(["H", "T"]) == "H":
            wins += 1
    return wins

simulate()
```

Out[13]: 10

In [14]:
```
check('tests/q1_4.py')
```

Out[14]: All tests passed!

Question 1.5

Using 10,000 trials, generate 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: Feel free to use the function you defined in Question 1.4.

In [28]:
```
test_statistics_under_null = []
repetitions = 10000
```

```
for _ in range(repetitions):
    test_statistics_under_null.append(simulate())

test_statistics_under_null = np.array(test_statistics_under_null)
```

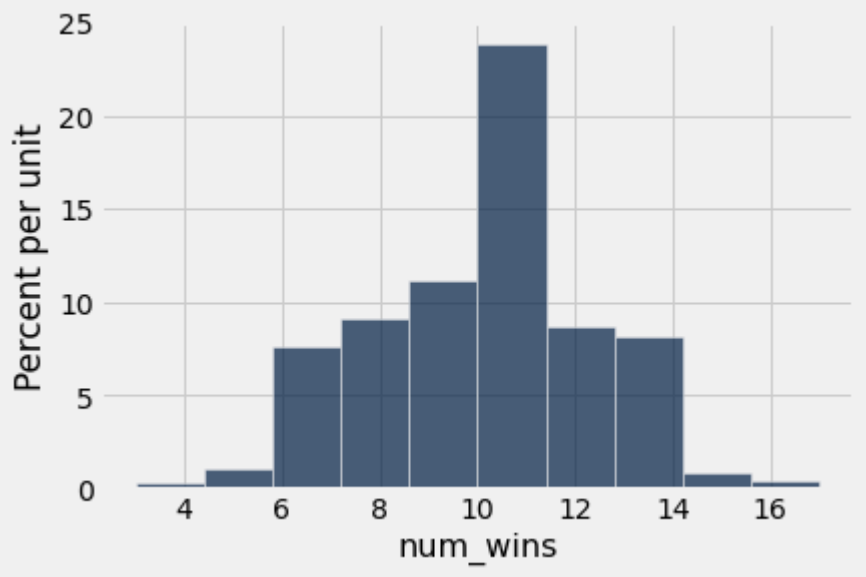
In [29]:
```
check('tests/q1_5.py')
```

Out[29]: All tests passed!

Question 1.6

Using the results from Question 1.5, generate a histogram of the empirical distribution of the number of wins in 20 games.

In [24]:
```
Table().with_column(
    "num_wins", test_statistics_under_null
).hist()
```



Question 1.7

Compute an empirical P-value for this test.

Hint: Which values of our test statistic are in the direction of the alternative hypothesis?

In [30]:
```
p_value = len(test_statistics_under_null[test_statistics_under_null >= 13]) / len(tes
```

In [31]:
```
check('tests/q1_7.py')
```

Out[31]: All tests passed!

In [32]:
```
p_value
```

Out[32]: 0.1375

Question 1.8

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

The customer did not cheat because the the p_value is greater than the p_value cutoff

Question 1.9

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?

Neither. It is the probability we got a result at least as extreme as what the customer got. We can draw conlcusions from these stats but they don't outright claim that the customer cheated.

Question 1.10

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?

Neither. It is saying that we will conclude the coin is fair (and therefore they didn't cheat) as long as the probability we got a result as extreme as what the customer got is greater than 1%, otherwise we conclude they did cheat.

Question 1.11

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 customers we will incorrectly accuse of cheating? If so, what is the number? Explain your answer. Assume a 1% P-value cutoff.

Yes. 4

2. 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.)

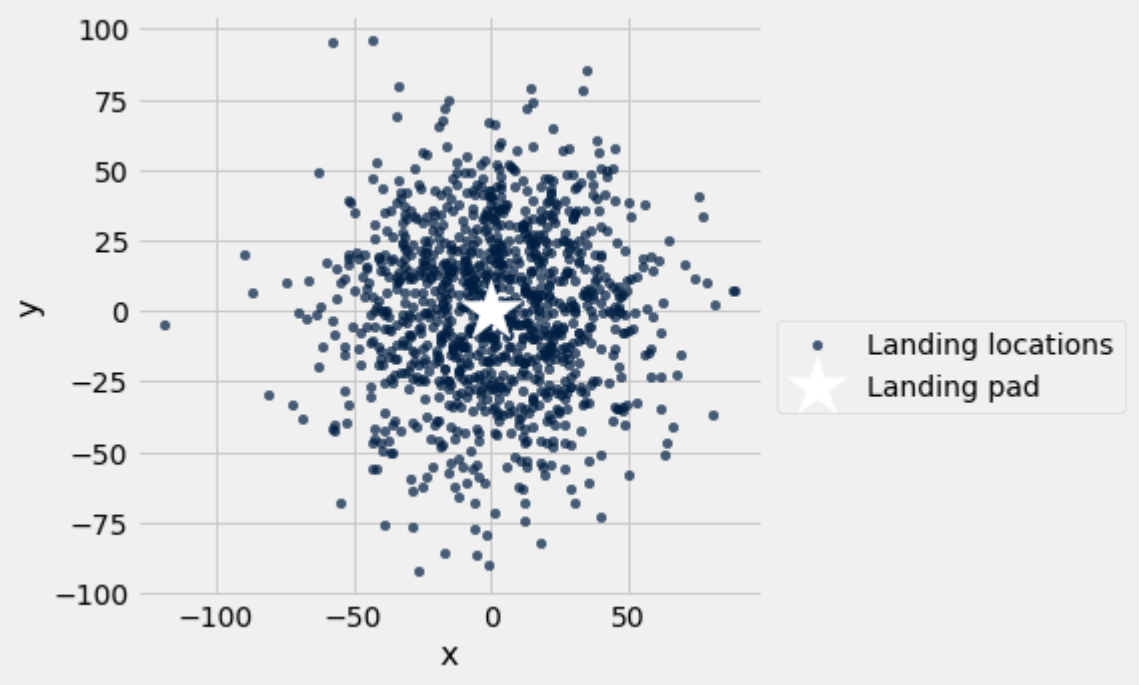
SpaceY, a company that builds and tests spacecraft, is testing a new reusable launch system. Most spacecrafts use a "first stage" rocket that propels a smaller payload craft away from Earth, then falls back to the ground and crashes. SpaceY's new system is designed to land safely at a landing pad at 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.

In [34]:
```
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));
```

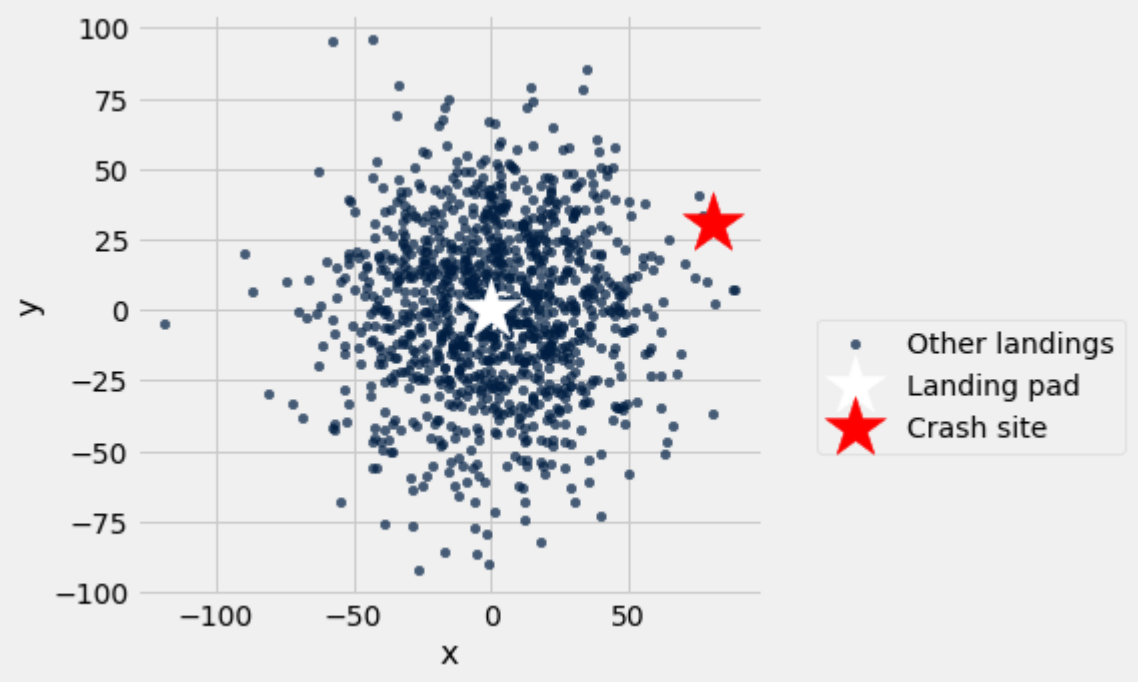


During one test, the vehicle lands far away from the landing pad and crashes. SpaceY investigators suspect there was a problem unique to this landing, a problem that wasn't part of the ordinary pattern of variation in landing locations. They think a software error in the guidance system caused the craft 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.

In [35]:

```
landing_spot = make_array(80.59, 30.91)
ordinary_landing_spots.scatter("x", label="Other landings")
plt.scatter(0, 0, c="w", s=1000, marker="*", label="Landing pad")
plt.scatter(landing_spot.item(0), landing_spot.item(1), marker="*", c="r", s=1000, la
plt.legend(scatterpoints=1, bbox_to_anchor=(1.6, .5));
```



Question 2.1

Suppose we'd like to use hypothesis testing to shed light on this question. We've written down an alternative hypothesis below. What is a reasonable null hypothesis?

Null hypothesis: The landing was not special. It happened by `chance` and not any underlying problems with the ship.

Alternative hypothesis: This landing was special; its location was a draw from some other distribution, not the distribution from which the other 1100 landing locations were drawn.

Question 2.2

What's a good test statistic for this hypothesis test?

Hint: A test statistic can be almost anything, but a *good* test statistic varies informatively depending on whether the null is true. So for this example, we might think about a test statistic that would be small if the null is true, and large otherwise. If we want to compare landings, we might want to see *how far* each landing is from some *reference point*, so we can compare all landings from the same vantage point.

Test statistic: Distance from the landing pad

Question 2.3

Write a function called `landing_test_statistic` . It should take two arguments: an "x" location and a "y" location (both numbers). It should return the value of your test statistic for a landing at those coordinates.

In [36]:
```
def landing_test_statistic(x_coordinate, y_coordinate) -> float:
    # put 0 just to explicitly tell what the origin is
    return np.sqrt((x_coordinate - 0)**2 + (y_coordinate - 0)**2)
```

Question 2.4

The next three cells compute a P-value using your test statistic. Describe the test procedure in

words. Is there a simulation involved? If so, what is being simulated? If not, why not? Where are we getting the data from? What kind of calculations are being performed? How are we calculating our P-value?

Hint: Think about what a simulation actually consists of.

In [37]:
```
observed_test_stat = landing_test_statistic(
    landing_spot.item(0),
    landing_spot.item(1))

observed_test_stat
```

Out[37]:
```
86.31440320131978
```

In [38]:
```
null_stats = make_array()
repetitions = ordinary_landing_spots.num_rows

for i in np.arange(repetitions):
    null_stat = landing_test_statistic(
        ordinary_landing_spots.column('x').item(i),
        ordinary_landing_spots.column('y').item(i))
    null_stats = np.append(null_stats, null_stat)

null_stats
```

Out[38]:
```
array([ 7.0373003 , 37.04323884, 28.24132651, ..., 20.84023432,
       32.48926398, 23.88691421])
```

In [39]:
```
p_value = np.count_nonzero(null_stats >= observed_test_stat) / len(null_stats)
p_value
```

Out[39]:
```
0.012727272727272728
```

It's iterating over all the observed landing spots and calculating the test stat and then checking what % of landing had at least as extreme test stats.

3. Testing Dice

Students in a Data Science class want to figure out whether a six-sided die is fair or not. On a fair die, each face of the die appears with chance 1/6 on each roll, regardless of the results of other rolls. Otherwise, a die is called unfair. We can describe a die by the probability of landing on each face. This table describes an example of a die that is unfairly weighted toward 1:

Face	Probability
1	.5
2	.1
3	.1
4	.1

Face	Probability
5	.1
6	.1

Question 3.1

Define a null hypothesis and an alternative hypothesis to test whether a six-sided die is fair or not.

Hint: Remember that an unfair die is one for which each face does not have an equal chance of appearing.

Null hypothesis: This is a fair die

Alternative hypothesis: This in an unfair die

We decide to test the die by rolling it 5 times. The proportions of the 6 faces in these 5 rolls are stored in a table with 6 rows. For example, here is the table we'd make if the die rolls ended up being 1, 2, 3, 3, and 5:

Face	Proportion
1	.2
2	.2
3	.4
4	.0
5	.2
6	.0

The function `mystery_test_statistic`, defined below, takes a single table like this as its argument and returns a number (which we will use as a test statistic).

In [41]:
```
# Note: We've intentionally used unhelpful function and
# variable names to avoid giving away answers.  It's rarely
# a good idea to use names like "x" in your code.

def mystery_test_statistic(sample):
    x = np.ones(1) * (1/6)
    y = (sample.column('Proportion') - x)
    return np.mean(y**2)
```

Question 3.2

Describe in words what the test statistic is. Is it equivalent to the total variation distance between the observed face distribution and the fair die distribution?

It returns the average squared distance from the expected distribution (a uniform one where each has a 1/6 chance)

The function `simulate_observations_and_test` takes as its argument a table describing the probability distribution of a die. It simulates one set of 5 rolls of that die, then tests the null

hypothesis about that die using our test statistic function above. It returns `False` if it *rejects* the null hypothesis about the die, and `True` otherwise.

In [43]:
```
# The probability distribution table for a fair die:
fair_die = Table().with_columns(
        "Face", np.arange(1, 6+1),
        "Probability", [1/6, 1/6, 1/6, 1/6, 1/6, 1/6])

def simulate_observations_and_test(actual_die):
    """Simulates die rolls from actual_die and tests the hypothesis that the die is f

    Returns False if that hypothesis is rejected, and True otherwise.

    """

    sample_size = 5
    p_value_cutoff = .2
    num_simulations = 250

    # Compute the observed value of the test statistic.
    observation_set = sample_proportions(sample_size, actual_die.column("Probability"
    observation_props_table = Table().with_columns('Face', actual_die.column('Face'),
    observed_statistic = mystery_test_statistic(observation_props_table)

    # Simulate the test statistic repeatedly to get an
    # approximation to the probability distribution of
    # the test statistic, as predicted by the model in
    # the null hypothesis. Store the simulated values
    # of the test statistic in an array.
    simulated_statistics = make_array()
    for _ in np.arange(num_simulations):
        one_observation_set_under_null = sample_proportions(sample_size, fair_die.col
        simulated_props_table = Table().with_columns('Face', fair_die.column('Face'),
        simulated_statistic = mystery_test_statistic(simulated_props_table)
        simulated_statistics = np.append(simulated_statistics, simulated_statistic)

    # Compute the P-value
    p_value = np.count_nonzero(simulated_statistics >= observed_statistic) / num_simu

    # If the P-value is below the cutoff, reject the
    # null hypothesis and return False. Otherwise,
    # return True.
    return p_value >= p_value_cutoff

# Calling the function to simulate a test of a fair die:
simulate_observations_and_test(fair_die)
```

Out[43]:
```
True
```

Question 3.3

Use your knowledge of hypothesis tests and interpretation of the code above to compute the probability that `simulate_observations_and_test` returns `False` when its argument is `fair_die` (which is defined above the function). In other words, what are the odds that we reject the Null Hypothesis if the die is actually fair?

You can call the function a few times to see what it does, but **don't** perform a simulation to compute this probability. Use your knowledge of hypothesis tests. You shouldn't have to write any code to answer this question.

In [46]:
```
probability_of_false = .2
```

In [47]:
```
check('tests/q3_3.py')
```

Out[47]: All tests passed!

Question 3.4

Why is your answer to Question 3.3 the correct probability?

Because .2 is the cutoff, any result that happens less than or equal to 20% of the time is considered unfair. If we assert that the die is fair, we should still see that 20% of the time we get such extreme results and therefore that's the false positive rate

Question 3.5

Simulate the process of running `simulation_observations_and_test` 300 times. Assign `test_results` to an array that stores the result of each of these trials.

Note: This will be a little slow. 300 repetitions of the simulation should require a minute or so of computation, and it should suffice to get an answer that's roughly correct.

In [50]:
```
num_test_simulations = 300
test_results = []

for _ in range(num_test_simulations):
    test_results.append(simulate_observations_and_test(fair_die))

test_results = np.array(test_results)
# Don't change the following line.
test_results.astype(bool)
```

Out[50]:
```
array([ True,  True,  True, False,  True,  True,  True, False,  True,
        True,  True,  True,  True, False,  True,  True,  True, False,
        True, False,  True,  True,  True,  True,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True, False, False,
        True,  True,  True,  True, False,  True,  True,  True,  True,
        True, False,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True, False,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True, False,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False, False,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True, False,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True, False,  True,  True,
        True,  True, False, False, False,  True,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True, False, False, False,  True,  True,  True,  True,
        True,  True, False, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True, False,  True, False,  True,  True,  True, False,
        True, False,  True,  True,  True, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True, False,
        True,  True, False])
```

In [51]:
```
check('tests/q3_5.py')
```

Out[51]: All tests passed!

Question 3.6

Verify your answer to Question 3.3 by computing an approximate probability that `simulation_observations_and_test` returns `False`.

In [52]:
```
approximate_probability_of_false = 1 - (np.count_nonzero(test_results) / len(test_res
approximate_probability_of_false
```

Out[52]:
```
0.16666666666666663
```

In [53]:
```
check('tests/q3_6.py')
```

Out[53]: All tests passed!

Question 3.7

From the perspective of someone who wants to know the truth about the die, is it good or bad

for the function to return `False` when its argument is `fair_die` ? Why is it good or bad?

Good. It returns false if it isn't a fair die and true if it is. Maybe extra information like the p-value would be nice but a bool is good enough.

4. A Potpourri of Tests

The rest of this homework is optional. Do it for your own practice, but it will not be incorporated into the final grading!

Question 4.1 (Optional)
Many scientific disciplines use 5% as a standard cutoff for rejecting the null hypothesis when conducting hypothesis tests. Suppose for the sake of argument that every scientific paper hinges on exactly one hypothesis test with a 5% cutoff. After learning about hypothesis testing, Thomas despairs about the state of scientific research, wondering:

> "Doesn't this mean that 5% of all scientific papers are wrong?"

Under what conditions would Thomas's worry be realistic, and why is it not entirely accurate?

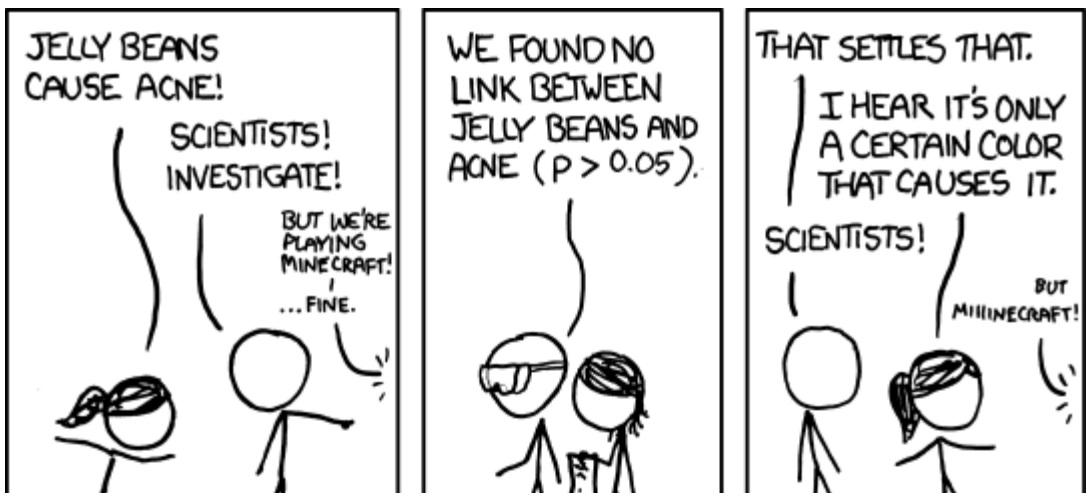
Write your answer here, replacing this text.

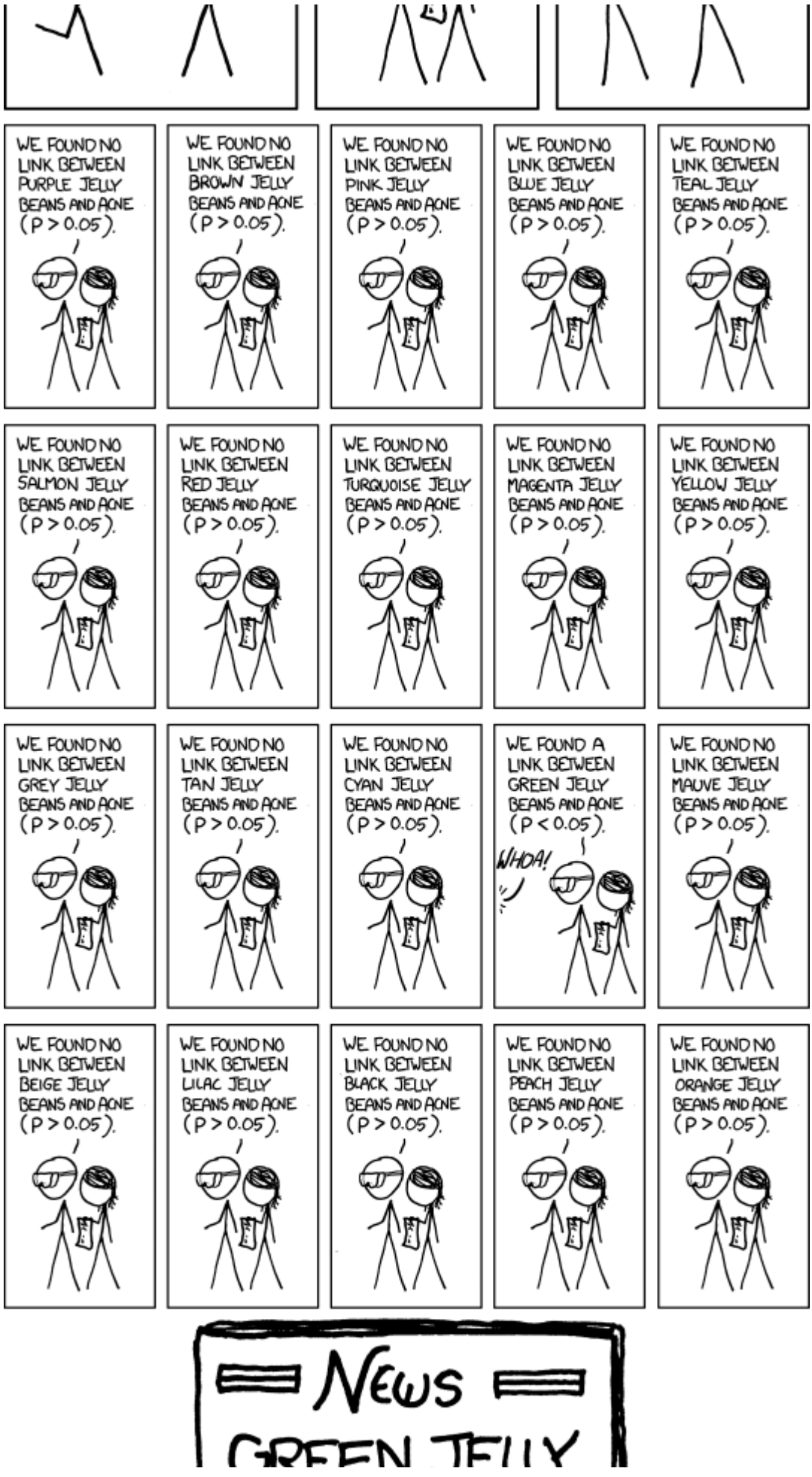
Question 4.2 (Optional)
Many scientists hope to make exciting and unintuitive discoveries. Often, the null hypothesis in an hypothesis test is something boring ("the sky is blue"), while the alternative is surprising ("the sky is not blue!").

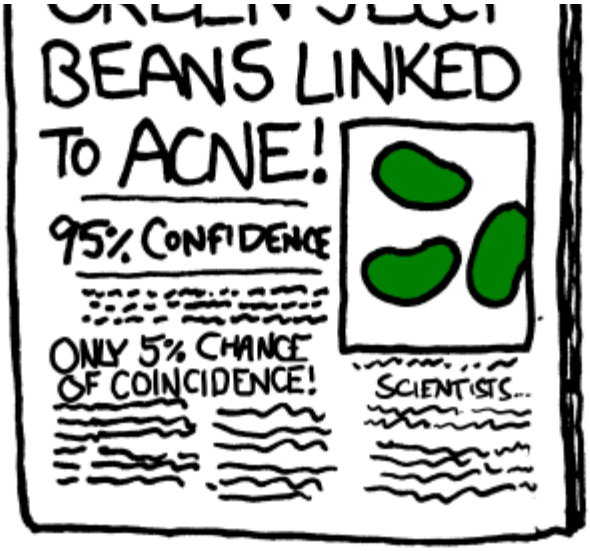
Suppose a scientist has an exciting but incorrect idea, so that their null hypothesis is *truly correct*. When an hypothesis test is run on a sample of data, it fails to reject the null when using a 5% cutoff. Disappointed but determined, the scientist gathers 10 more samples and runs the same test on each sample (running 10 separate hypothesis tests, each with a 5% cutoff). If any of the tests rejects the null, the scientist publishes that one. What is the chance that any of those tests rejects the null hypothesis?

Write your answer here, replacing this text.

Note: The scientist in this scenario is acting very unethically, but (probably milder) forms of this *publication bias* seem to be a real problem in science today. See a relevant xkcd below.




WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN SALMON JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN RED JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TURQUOISE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN MAGENTA JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN YELLOW JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN GREY JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE (P < 0.05).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BEIGE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN LILAC JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLACK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PEACH JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN ORANGE JELLY BEANS AND ACNE (P > 0.05).
NEWS
GREEN JELLY



5. Submission

Once you're finished, select "Save and Checkpoint" in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the autograder does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this homework.

When ready, click "Print Preview" in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose "save as pdf" from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

In [54]:
```
# For your convenience, you can run this cell to run all the tests at once!
import glob
from gofer.ok import grade_notebook
if not globals().get('__GOFER_GRADER__', False):
    display(grade_notebook('hw07.ipynb', sorted(glob.glob('tests/q*.py'))))
```

```
['tests/q1_1.py', 'tests/q1_3.py', 'tests/q1_4.py', 'tests/q1_5.py', 'tests/q1_7.py',
'tests/q3_3.py', 'tests/q3_5.py', 'tests/q3_6.py']
Question 1:
```

All tests passed!

```
Question 2:
```

All tests passed!

```
Question 3:
```

All tests passed!

```
Question 4:
```

All tests passed!

```
Question 5:
```

All tests passed!

```
Question 6:
```

All tests passed!

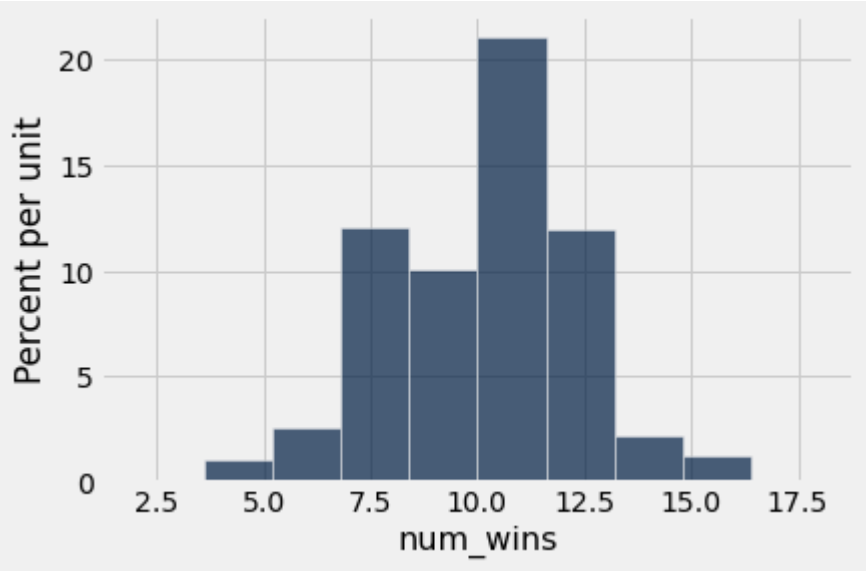
```
Question 7:
```

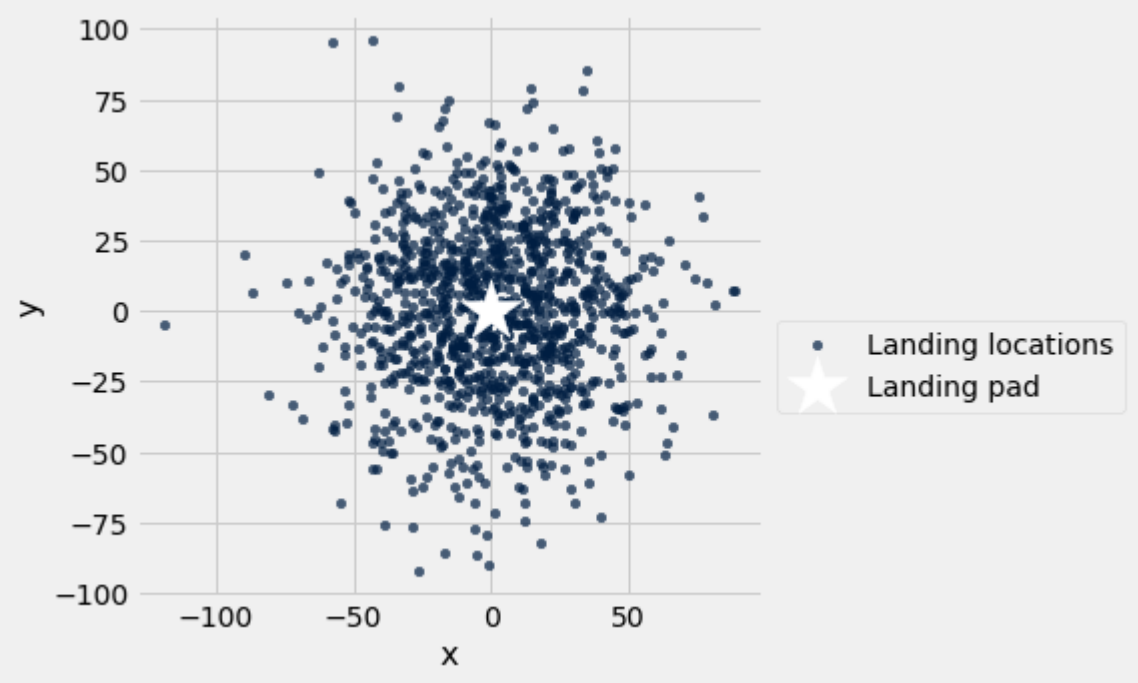
All tests passed!

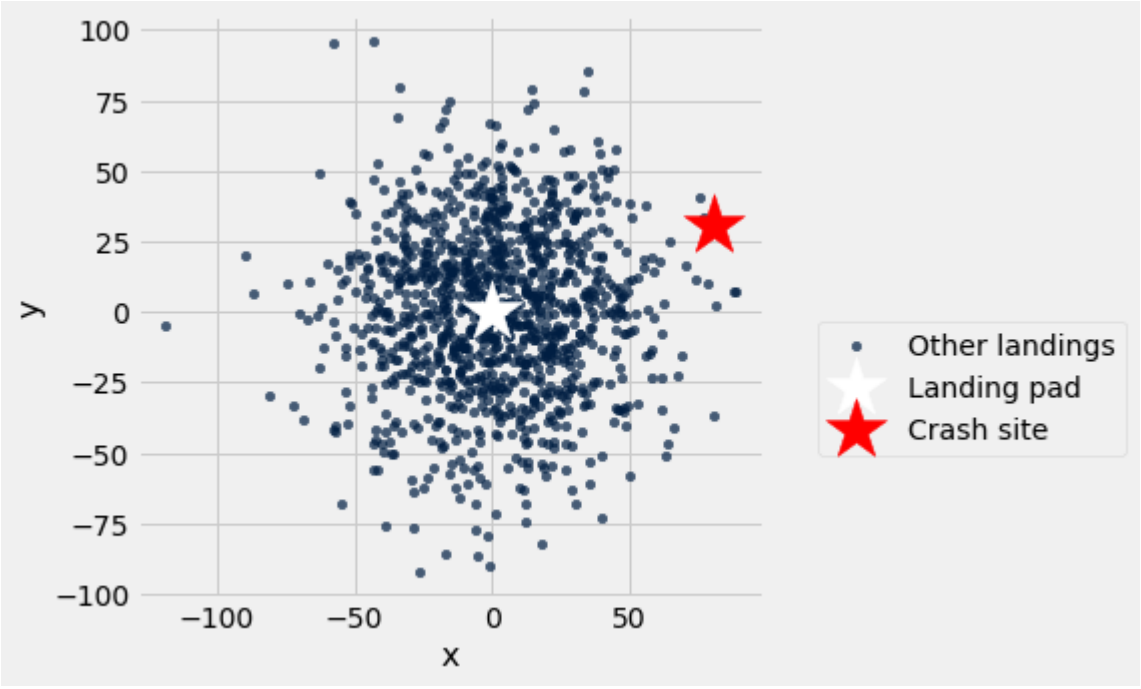
```
Question 8:
```

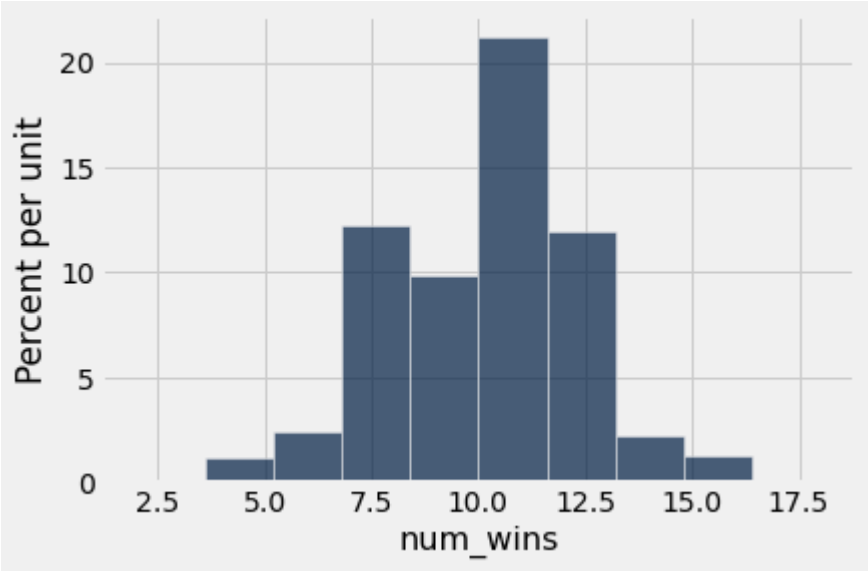
All tests passed!

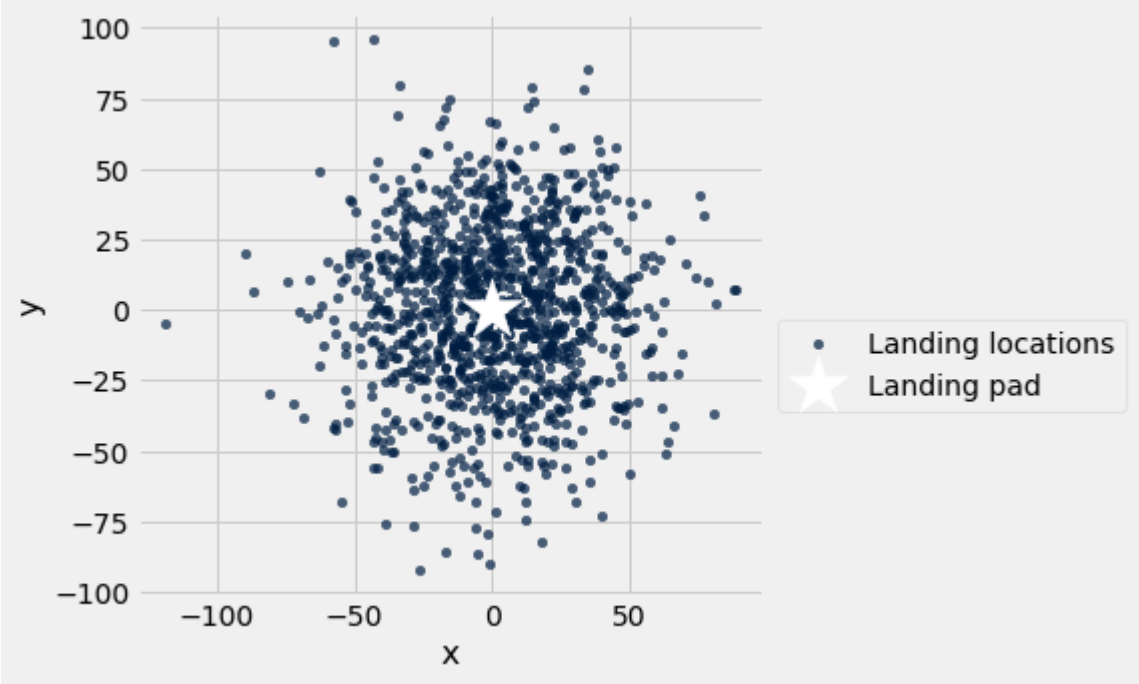
```
1.0
```

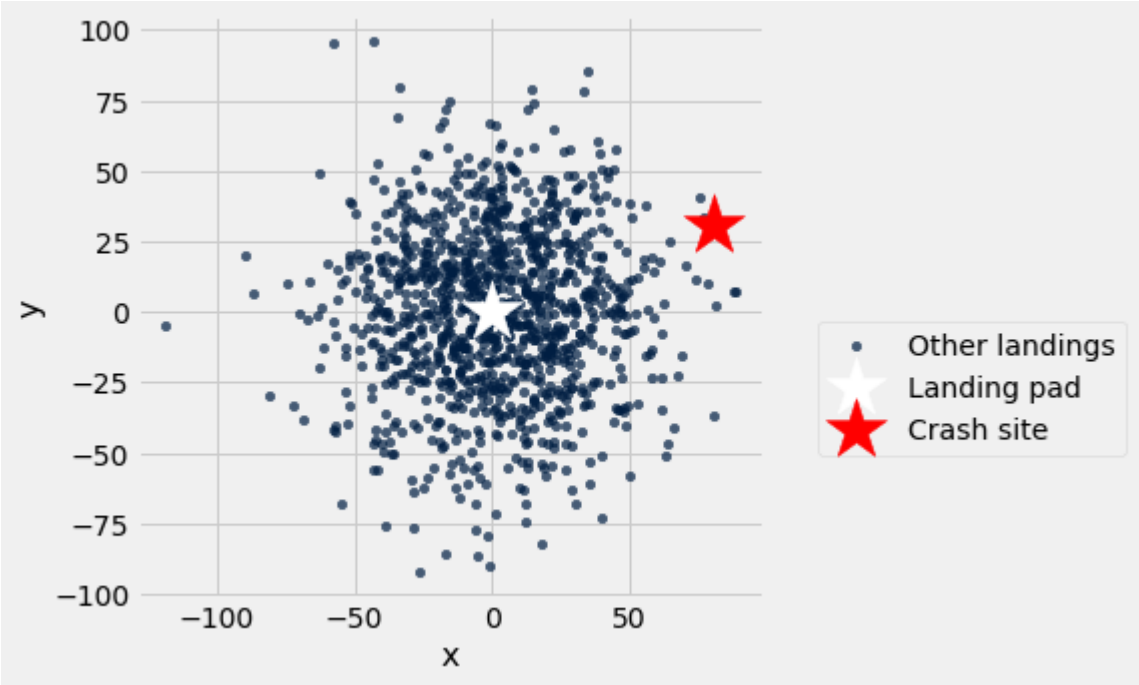





100
75
50
25
0
−25
−50
−75
−100
y
−100
−50
0
50
x
Other landings
Landing pad
Crash site


20
15
10
5
0
Percent per unit
2.5
5.0
7.5
10.0
12.5
15.0
17.5
num_wins





Name: Allan Gongora

Section: 0131