HW 10 Autograder

● Graded

Student

Sangwon Ji

Total Points

65 / 60 pts

Autograder Score

60.0 / 60.0

Passed Tests

Public Tests

Question 2

Early Submission Bonus **5** / 0 pts

✓ **+ 5 pts** Early Submission Bonus

+ 0 pts Click here to replace this description.

Autograder Results

Autograder Output

```
  _________        __        __
 /   _____ \    __|  |__  __|  |__
|  /     \  |  |__    __||__    __|   _______    _  _____
| |       | |     |  |       |  |    |   __  |  | |/ ____|
| |       | |     |  |       |  |    |  |__| |  |   /
| |       | |     |  |       |  |    |  _____|  |  |
|  \_____/  |     |  |_      |  |_   |  |____   |  |
 \_________/       \___|      \___|   \______|  |__|
                                          v3.1.4


95% confidence interval for slope: [0.598031, 0.81229]
95% Confidence interval for predictions for x=8: ( 5.62614784137 , 5.76819126819 )




------------------------------ 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: 60.000 / 60.000 (100.000%)

          name  score  max_score
0   Public Tests    NaN        NaN
1          q1_2    8.0        8.0
2          q1_3    8.0        8.0
3          q1_4    1.0        1.0
4          q1_5    8.0        8.0
5          q2_1    8.0        8.0
6          q2_2    8.0        8.0
7          q2_3    8.0        8.0
8          q2_4    1.0        1.0
9          q2_5    6.0        6.0
10         q2_6    6.0        6.0
```

Public Tests

```
q1_2 results: All test cases passed!

q1_3 results: All test cases passed!

q1_4 results: All test cases passed!

q1_5 results: All test cases passed!

q2_1 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!

q2_5 results: All test cases passed!

q2_6 results: All test cases passed!
```

Submitted Files

▾ hw10.ipynb Download

In [1]:

```
# Initialize Otter
import otter
grader = otter.Notebook("hw10.ipynb")
```

Homework 10: Regression Inference

Helpful Resource:

- Python Reference: Cheat sheet of helpful array & table methods used in Data 8!

Recommended Reading:

- Using Confidence Intervals
- The Regression Line
- Inference for Regression

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, 8/2 at 11:59pm PT**. Turn it in by Monday, 8/1 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. The OH schedule appears on http://data8.org/su22/office-hours.html.

In [2]:
```
# Don't change this cell; just run it.

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')
from datetime import datetime
```

An Introduction to Regression Inference

Previously in this class, we've used confidence intervals to quantify uncertainty about estimates. We can also run hypothesis tests using a confidence interval under the following procedure:

1. Define a null and alternative hypothesis (they must be of the form "The parameter is X" and "The parameter is not X").
2. Choose a p-value cutoff, and call it q.
3. Construct a $(100-q)\%$ interval using bootstrap sampling (for example, if your p-value cutoff is 0.01, or 1%, then construct a 99% confidence interval).
4. Using the confidence interval, determine if your data are more consistent with your null or alternative hypothesis:
    - If the null hypothesis parameter X is in your confidence interval, the data are more consistent with the null hypothesis.
    - If the null hypothesis parameter X is *not* in your confidence interval, the data are more consistent with the alternative hypothesis.

More recently, we've discussed the use of linear regression to make predictions based on correlated variables. For example, we can predict the height of children based on the heights of their parents.

We can combine these two topics to make powerful statements about our population by using the following techniques:

- Bootstrapped interval for the true slope
- Bootstrapped prediction interval for y (given a particular value of x)

This homework explores these two methods.

The Data

The Snowy Plover is a tiny bird that lives on the coast in parts of California and elsewhere. It is so small that it is vulnerable to many predators, including people and dogs that don't look where they are stepping when they go to the beach. It is considered endangered in many parts of the U.S.

The data are about the eggs and newly-hatched chicks of the Snowy Plover. Here's a picture of a parent bird incubating its eggs.

The data were collected at the Point Reyes National Seashore by a former student at Berkeley. The goal was to see how the size of an egg could be used to predict the weight of the resulting chick. The bigger the newly-hatched chick, the more likely it is to survive.

Each row of the table below corresponds to one Snowy Plover egg and the resulting chick. Note how tiny the bird is:

- `Egg Length` and `Egg Breadth` (widest diameter) are measured in millimeters
- `Egg Weight` and `Bird Weight` are measured in grams; for comparison, a standard paper clip weighs about one gram

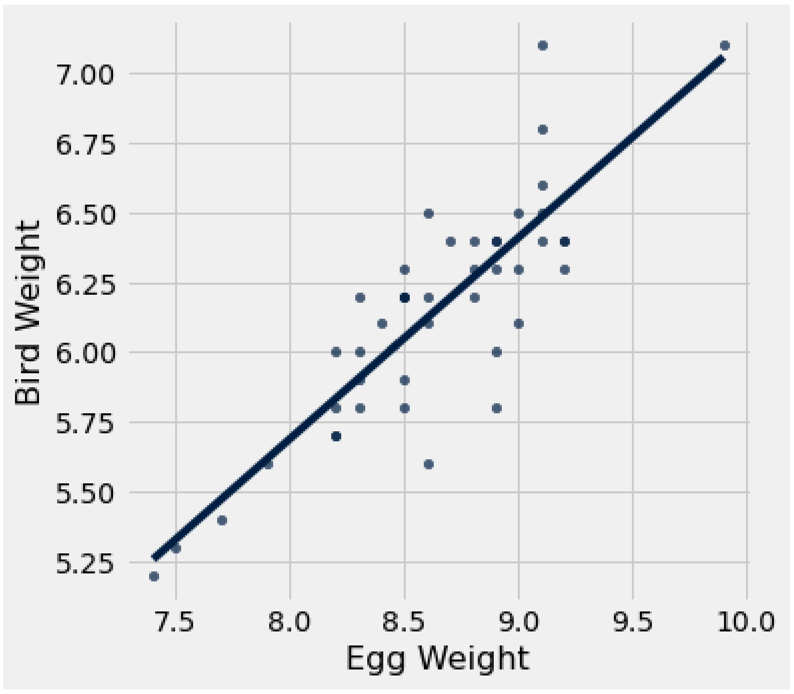
In [3]:
```
birds = Table.read_table('snowy_plover.csv')
birds
```

Out [3]:
```
Egg Length | Egg Breadth | Egg Weight | Bird Weight
28.8       | 21.84       | 7.4        | 5.2
29.04      | 22.45       | 7.7        | 5.4
29.36      | 22.48       | 7.9        | 5.6
30.1       | 21.71       | 7.5        | 5.3
30.17      | 22.75       | 8.3        | 5.9
30.34      | 22.84       | 8.5        | 5.8
30.36      | 22.5        | 8.2        | 5.8
30.46      | 22.72       | 8.3        | 6
30.54      | 23.31       | 9          | 6.1
30.62      | 22.94       | 8.5        | 6.2
... (34 rows omitted)
```

In this investigation, we will be using the egg weight to predict bird weight. Run the cell below to create a scatter plot of the egg weights and bird weights, along with their line of best fit.

In [4]:

```
# Just run this cell and examine the scatter plot.
birds.scatter('Egg Weight', "Bird Weight", fit_line=True)
```



1. Finding the Bootstrap Confidence Interval for the True Slope

Looking at the scatter plot of our sample, we observe a linear relationship between egg weight and bird weight. However, relationships that appear in a sample might not exist in the population from which the sample was taken.

We want to know if there truly exists a linear relationship between egg weight and bird weight for Snowy Plovers. If there is no linear relationship between the two variables, then we'd expect a correlation of 0. Consequently, the slope of the regression line would also be 0.

Question 1.1. Let's run a hypothesis test using confidence intervals to see if there is a linear relationship between egg weight and bird weight. Define the null and alternative hypotheses that will allow you to conduct this test. **(8 points)**

Note: Please write your answer **in the cell below** in the following format:

- **Null Hypothesis:**
- **Alternative Hypothesis:**

Null Hypothesis: There is no association between egg weight and bird weight. The true correlation is 0. Alternative Hypothesis: There is relationship between egg weight and bird weight. And the association between them are not 0.

Question 1.2. Define the following two functions:

1. `standard_units`: This function takes in an array of numbers and returns an array containing those numbers converted to standard units.
2. `correlation`: This function takes in a table and two column names (one for *x* and one for *y*) and returns the correlation between these columns.

(8 points)

In [5]:
```
def standard_units(arr):
    return (arr - np.mean(arr))/np.std(arr)

def correlation(tbl, x_col, y_col):
    return np.mean(standard_units(tbl.column(x_col)) *
standard_units(tbl.column(y_col)))
```

In [6]:
```
grader.check("q1_2")
```

Out [6]:
```
q1_2 results: All test cases passed!
```

Question 1.3. Using the functions you just implemented, create a function called `fit_line`. It should take a table like `birds` and the column names associated to *x* and *y* as its arguments and return an *array* containing the slope and intercept of the regression line (in that order) that predicts the *y* column in the table using the *x* column. **(8 points)**

In [7]:
```
def fit_line(tbl, x_col, y_col):
    slope = correlation(tbl, x_col, y_col) * np.std(tbl.column(y_col))/
np.std(tbl.column(x_col))
    intercept = np.mean(tbl.column(y_col))- slope * np.mean(tbl.column(x_col))
    return make_array(slope, intercept)

fit_line(birds, "Egg Weight", "Bird Weight")
```

Out [7]:
```
array([ 0.71851534, -0.05827226])
```

In [8]:
```
grader.check("q1_3")
```

Out [8]:

```
q1_3 results: All test cases passed!
```

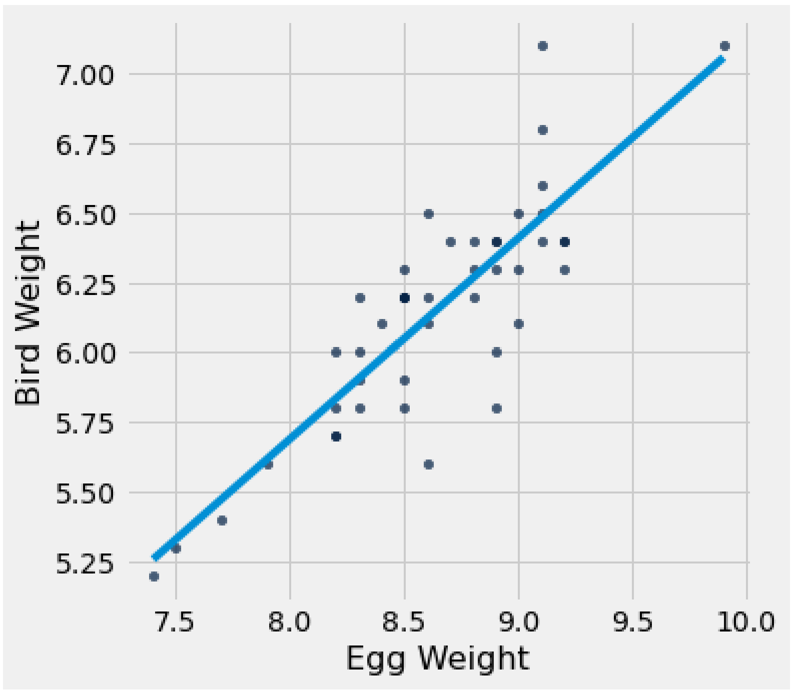
Run this cell to plot the line produced by calling `fit_line` on the `birds` table.

Note: You are not responsible for the code in the cell below, but make sure that your `fit_line` function generated a reasonable line for the data.

In [9]:

```
# Ensure your fit_line function fits a reasonable line
# to the data in birds, using the plot below.

# Just run this cell
slope, intercept = fit_line(birds, "Egg Weight", "Bird Weight")
birds.scatter("Egg Weight", "Bird Weight")
plt.plot([min(birds.column("Egg Weight")), max(birds.column("Egg Weight"))],
        [slope*min(birds.column("Egg Weight"))+intercept,
slope*max(birds.column("Egg Weight"))+intercept])
plt.show()
```



Now we have all the tools we need to create a confidence interval that quantifies our uncertainty about the true relationship between egg weight and bird weight.

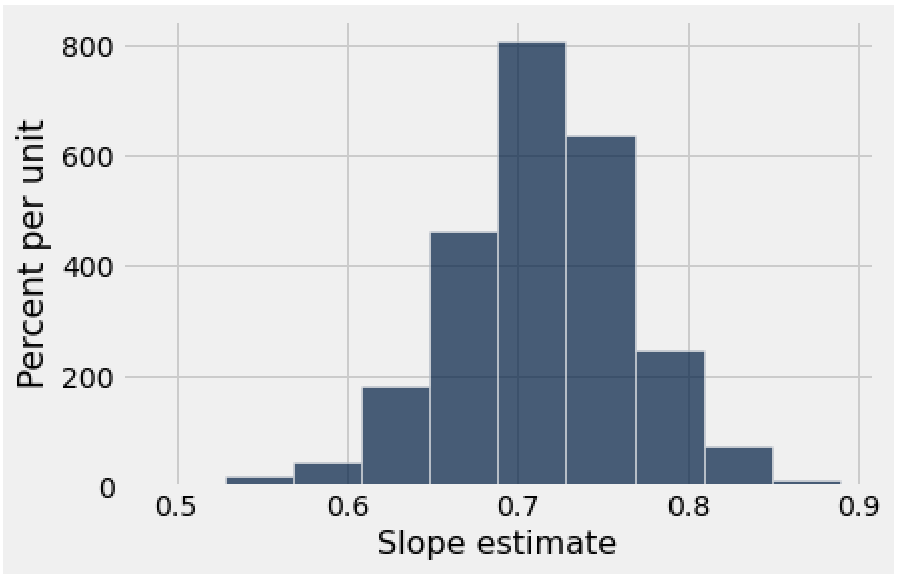
Question 1.4. Create an array called `resampled_slopes` that contains the slope of the best fit line for 1000 bootstrap resamples of `birds`. Plot the distribution of these slopes. **(8 points)**

In [10]:

```
resampled_slopes = make_array()
```

```
for i in np.arange(1000):
    birds_bootstrap = birds.sample()
    bootstrap_line = fit_line(birds_bootstrap, "Egg Weight", "Bird Weight")
    bootstrap_slope = bootstrap_line.item(0)
    resampled_slopes = np.append(resampled_slopes, bootstrap_slope)

# DO NOT CHANGE THIS LINE
Table().with_column("Slope estimate", resampled_slopes).hist()
```



In [11]:

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

Out [11]:

```
q1_4 results: All test cases passed!
```

Question 1.5. Use your resampled slopes to construct an 95% confidence interval for the true value of the slope. **(8 points)**

In [12]:

```
lower_end = percentile(2.5, resampled_slopes)
upper_end = percentile(97.5, resampled_slopes)
print("95% confidence interval for slope: [{:g}, {:g}]".format(lower_end,
upper_end))
```

```
95% confidence interval for slope: [0.603565, 0.817153]
```

In [13]:

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

Out [13]:

```
q1_5 results: All test cases passed!
```

Question 1.6. Based on your confidence interval, would you accept or reject the null hypothesis that the true slope is 0? Why? What p-value cutoff are you using? **(8 points)**

Hint: Read the introduction of this homework!

Based on my confidence interval, 95%, we would reject the null that the true slope is 0. Using the p-value cutoff of 0.05, and it's because 0 is not in our range for the 95% confidence interval.

Question 1.7. What do you think the true slope is? You do not need an exact number. How confident are you of this estimate? **(8 points)**

Hint: Can you provide an interval that you think the true slope falls in?

I think the true slope is about 0.71. We can also find that mean and median close to this value, and this is from slope estimation from 1,000 bootstraped samples. Also, I can say I'm 95% confident of this answer since I was using 95 confidence interval to get the range of it, 0.603565, 0.817153

2. Finding the Bootstrap Prediction Interval

Suppose we're visiting Point Reyes and stumble upon some Snowy Plover eggs; we'd like to know how heavy they'll be once they hatch. In other words, we want to use our regression line to make predictions about a bird's weight based on the weight of the corresponding egg.

However, just as we're uncertain about the slope of the true regression line, we're also uncertain about the predictions made based on the true regression line.

Question 2.1. Define the function `fitted_value`. It should take in four arguments:

1. `table`: a table like `birds`. We'll be predicting the values in the second column using the first.
2. `x_col`: the name of our x-column within the input `table`
3. `y_col`: the name of our y-column within the input `table`
4. `given_x`: a number, the value of the predictor variable for which we'd like to make a prediction.

The function should return the line’s prediction for the given x. **(8 points)**

Hint: Make sure to use the `fit_line` function you defined in Question 1.3.

In [14]:
```
def fitted_value(table, x_col, y_col, given_x):
    line = fit_line(table, x_col, y_col)
    slope = line.item(0)
    intercept = line.item(1)
    return slope * given_x + intercept

# Here's an example of how fitted_value is used. The code below
# computes the prediction for the bird weight, in grams, based on
# an egg weight of 8 grams.
egg_weight_eight = fitted_value(birds, "Egg Weight", "Bird Weight", 8)
egg_weight_eight
```

Out [14]:
```
5.689850497215146
```

In [15]:
```
grader.check("q2_1")
```

Out [15]:
```
q2_1 results: All test cases passed!
```

Question 2.2. Raymond, the resident Snowy Plover expert at Point Reyes, tells us that the egg he has been carefully observing has a weight of 9 grams. Using `fitted_value` above, assign the variable `experts_egg` to the predicted bird weight for Raymond's egg.

In [16]:
```
experts_egg = fitted_value(birds, "Egg Weight", "Bird Weight", 9)
experts_egg
```

Out [16]:
```
6.408365842108825
```

In [17]:
```
grader.check("q2_2")
```

Out [17]:
```
q2_2 results: All test cases passed!
```

In [18]:
```
# Let's look at the number of rows in the birds table.
birds.num_rows
```

Out [18]:
```
44
```

A fellow parkgoer raises the following objection to your prediction:

> "Your prediction depends on your sample of 44 birds. Wouldn't your prediction change if you had a different sample of 44 birds?"

Having read section 16.3 of the textbook, you know just the response! Had the sample been different, the regression line would have been different too. This would ultimately result in a different prediction. To see how good our prediction is, we must get a sense of how variable the prediction can be.

Question 2.3. Define a function `compute_resampled_line` that takes in a table `tbl` and two column names, `x_col` and `y_col`, and returns an array containing the parameters of the best fit line (slope and intercept) for one bootstrapped resample of the table.

In [19]:

```
def compute_resampled_line(tbl, x_col, y_col):
    resample = tbl.sample()
    resampled_line = fit_line(resample, x_col, y_col)
    return resampled_line
```

In [20]:

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

Out [20]:

```
q2_3 results: All test cases passed!
```

Run the following cell below in order to define the function `bootstrap_lines`. It takes in four arguments: 1. `tbl`: a table like `birds` 2. `x_col`: the name of our x-column within the input `tbl` 3. `y_col`: the name of our y-column within the input `tbl` 4. `num_bootstraps`: an integer, a number of bootstraps to run.

It returns a *table* with one row for each bootstrap resample and the following two columns: 1. `Slope`: the bootstrapped slopes 2. `Intercept`: the corresponding bootstrapped intercepts

In [21]:

```
# Just run this cell
def bootstrap_lines(tbl, x_col, y_col, num_bootstraps):
    resampled_slopes = make_array()
    resampled_intercepts = make_array()
    for i in np.arange(num_bootstraps):
        resampled_line = compute_resampled_line(tbl, x_col, y_col)
        resampled_slope = resampled_line.item(0)
        resampled_intercept = resampled_line.item(1)
        resampled_slopes = np.append(resampled_slopes,resampled_slope)
        resampled_intercepts =
np.append(resampled_intercepts,resampled_intercept)
```

```
    tbl_lines = Table().with_columns('Slope', resampled_slopes, 'Intercept',
resampled_intercepts)
    return tbl_lines

regression_lines = bootstrap_lines(birds, "Egg Weight", "Bird Weight", 1000)
regression_lines
```

Out [21]:

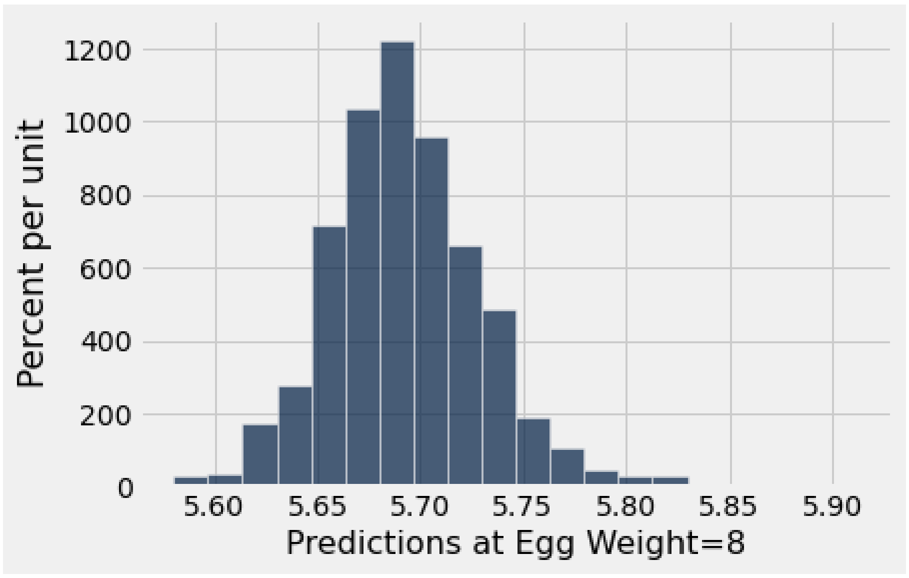
```
Slope    | Intercept
0.775998 | -0.532435
0.664716 | 0.415541
0.758781 | -0.390646
0.69801  | 0.127025
0.660995 | 0.40759
0.741814 | -0.256655
0.669197 | 0.38463
0.726797 | -0.22521
0.692813 | 0.130816
0.806576 | -0.777825
... (990 rows omitted)
```

Question 2.4. Create an array called `predictions_for_eight` that contains the predicted bird weights based on an egg of weight 8 grams for each regression line in `regression_lines`. **(8 points)**

In [22]:

```
predictions_for_eight = regression_lines.column(1) + regression_lines.column(0)
* 8

# This will make a histogram of your predictions:
table_of_predictions = Table().with_column('Predictions at Egg Weight=8',
predictions_for_eight)
table_of_predictions.hist('Predictions at Egg Weight=8', bins=20)
```



In [23]:
```
grader.check("q2_4")
```
Out [23]:
```
q2_4 results: All test cases passed!
```

Question 2.5. Create an approximate 95% confidence interval for these predictions. **(6 points)**

In [24]:
```
lower_bound = percentile(2.5, predictions_for_eight)
upper_bound = percentile(97.5, predictions_for_eight)

print('95% Confidence interval for predictions for x=8: (', lower_bound,",",
upper_bound, ')')
```
```
95% Confidence interval for predictions for x=8: ( 5.623992197659299 , 5.77267410651
```

In [25]:
```
grader.check("q2_5")
```
Out [25]:
```
q2_5 results: All test cases passed!
```

Question 2.6. Set `plover_statements` to an array of integer(s) that correspond to statement(s) that are true. **(6 points)**

1. The 95% confidence interval covers 95% of the bird weights for eggs that had a weight of eight grams in `birds`.
2. The 95% confidence interval quantifies the uncertainty in our estimate of what the true line would predict.
3. The 95% confidence interval gives a sense of how much actual weights differ from your prediction.

In [26]:
```
plover_statements = make_array(2)
plover_statements
```
Out [26]:
```
array([2])
```

In [27]:
```
grader.check("q2_6")
```
Out [27]:
```
q2_6 results: All test cases passed!
```

You're done with Homework 10!

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 10 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 [28]:
```
grader.check_all()
```

Out [28]:
```
q1_2 results: All test cases passed!

q1_3 results: All test cases passed!

q1_4 results: All test cases passed!

q1_5 results: All test cases passed!

q2_1 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!

q2_5 results: All test cases passed!

q2_6 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 [ ]:
```
# Save your notebook first, then run this cell to export your submission.
grader.export(pdf=False)
```

.OTTER_LOG

Download

```
Binary file hidden. You can download it using the button above.
```

__zip_filename__

Download

```
hw10_2022_08_01T22_22_01_979373.zip
```