lab11_student_solutions

December 6, 2019

1 Lab 11: Regression Inference

Welcome to Lab 11!

Today we will get some hands-on practice with regression inference. You can find more information about this topic in section 16.

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

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

# These lines do some fancy plotting magic.
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('lab11.ok')
_ = ok.submit()
```

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 q is .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 mean X is in your confidence interval, the data are more consistent with the null hypothesis.
- If the null hypothesis mean 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 lab explores these two advanced methods.

Recall the Old Faithful dataset from our correlation lab (Lab 10). The table contains two pieces of information for each eruption of the Old Faithful geyser in Yellowstone National Park: 1. `duration`: the duration of the eruption, in minutes. 2. `wait`: the time between this eruption and the next eruption (the "waiting time"), in minutes.

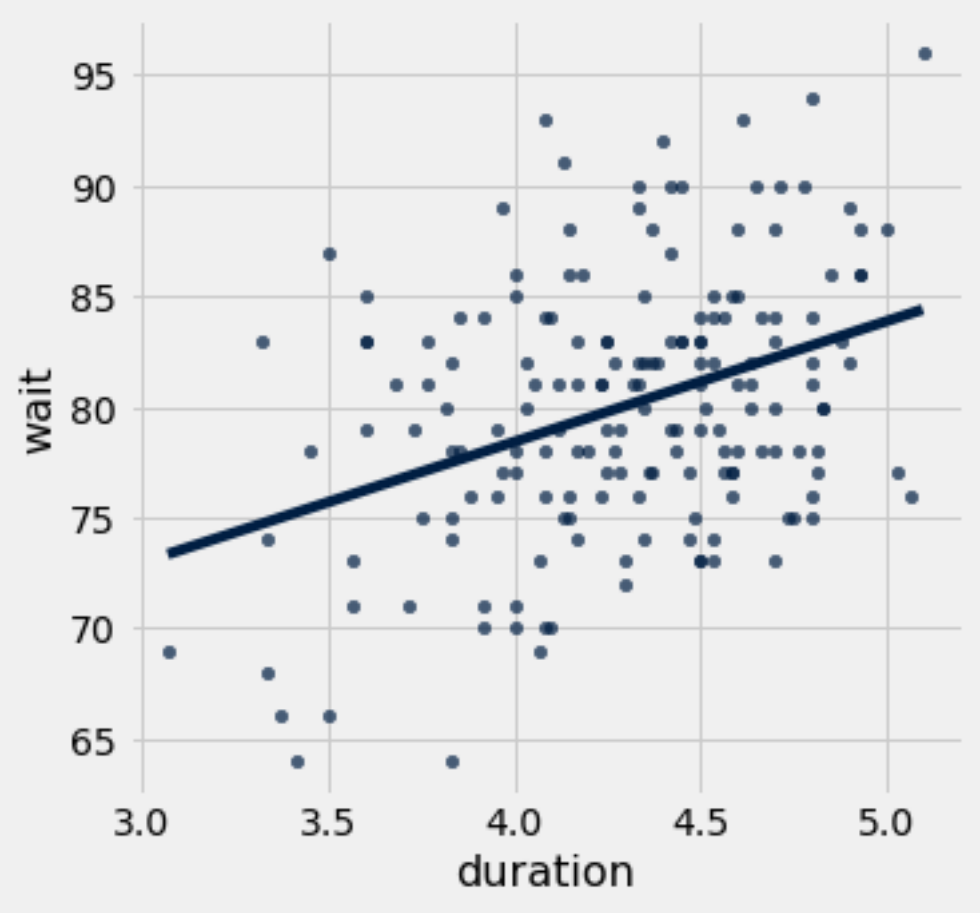
For the purposes of this lab, we'll only look at eruptions that have a duration that is greater than or equal to three minutes.

[65]:
```
faithful = Table.read_table('faithful_inference.csv').where("duration", are.
 ↪above_or_equal_to(3))
faithful
```

[65]:
```
duration | wait
3.6      | 79
3.333    | 74
4.533    | 85
4.7      | 88
3.6      | 85
4.35     | 85
3.917    | 84
4.2      | 78
4.7      | 83
4.8      | 84
… (165 rows omitted)
```

Run the cell below to plot the duration and waiting time for eruptions, along with their line of best fit.

[66]:
```
faithful.scatter('duration', fit_line=True)
```



1.1 1. Finding the Bootstrap Confidence Interval for the True Slope

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

We want to know whether there truly exists a linear relationship between duration and wait time. 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 Let's run a hypothesis test using confidence intervals to see if there is a linear relationship between duration and wait time. Define the null and alternative hypotheses that will allow you to conduct this test.

```
BEGIN QUESTION
name: q1_1
manual: true
```

SOLUTION:

Null Hypothesis: The true slope of the regression line that predicts wait from duration, computed using the population of all eruptions that have ever happened, is 0. If the slope of the regression line computed from our sample isn't 0, that is just a result of the particular eruptions we have in our sample.

Alternate Hypothesis: The true slope of the regression line is not 0.

Question 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 with 2 columns and returns the correlation between these columns.

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

[67]:
```
def standard_units(arr):
    return (arr - np.mean(arr))/np.std(arr) #SOLUTION

def correlation(tbl):
    return np.mean(standard_units(tbl.column(0)) * standard_units(tbl.
 ↪column(1))) #SOLUTION
```

Question 3 Using the functions you just implemented, create a function called `fit_line`. It should take a table as its argument. It should return an array containing the slope and intercept of the regression line (in that order) that predicts the second column in the table using the first.

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

[71]:
```
def fit_line(tbl):
    # BEGIN SOLUTION
    x = tbl.column(0)
    y = tbl.column(1)
    r = correlation(tbl)
    slope = correlation(tbl) * (np.std(y)/np.std(x))
    intercept = np.mean(y) - slope*np.mean(x)
    return make_array(slope, intercept)
    # END SOLUTION

fit_line(faithful)
```

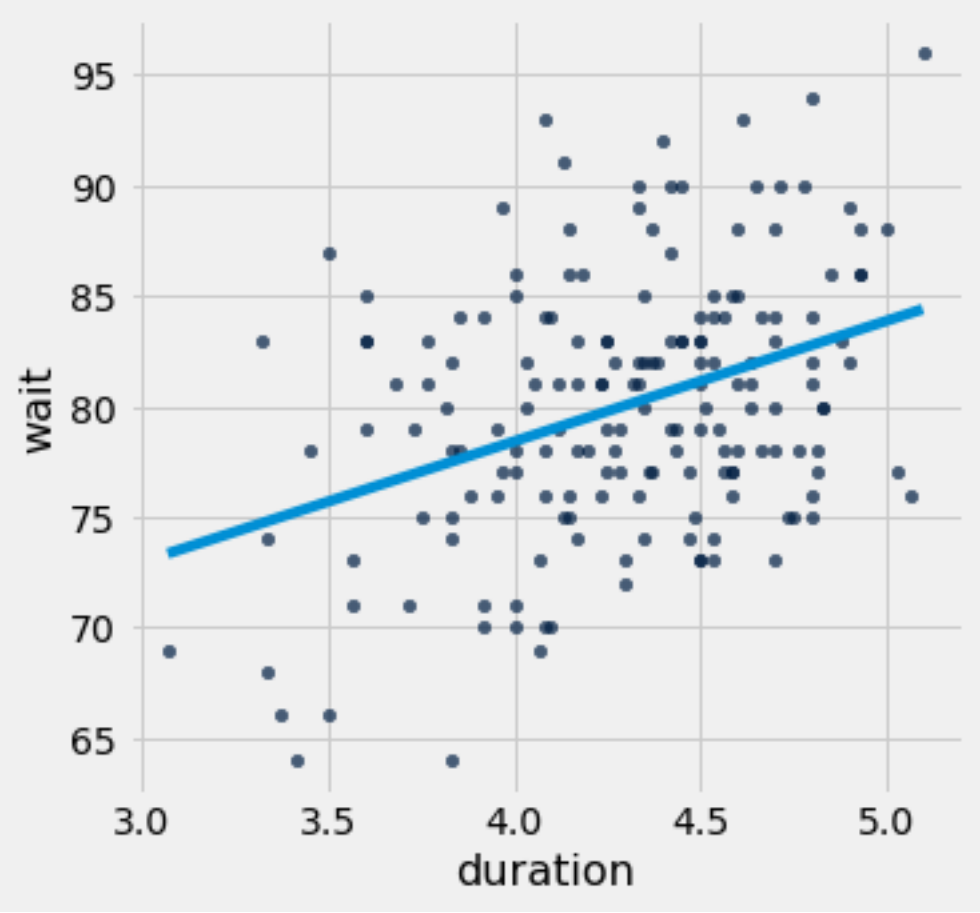
[71]:
```
array([ 5.43881507, 56.64896878])
```

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

Note: you are not responsible for this code, but make sure that your `fit_line` function generated a reasonable line for the data.

[74]:
```
# Ensure your fit_line function fits a reasonable line
# to the data in faithful, using the plot below

slope, intercept = fit_line(faithful)
faithful.scatter("duration")
plt.plot([min(faithful.column("duration")), max(faithful.column("duration"))],
         [slope*min(faithful.column("duration"))+intercept, slope*max(faithful.
 ↪column("duration"))+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 duration and wait time.

Question 4 Create an array called `resampled_slopes` that contains the slope of the best fit line for 1000 bootstrap resamples of `faithful`. Plot the distribution of these slopes.

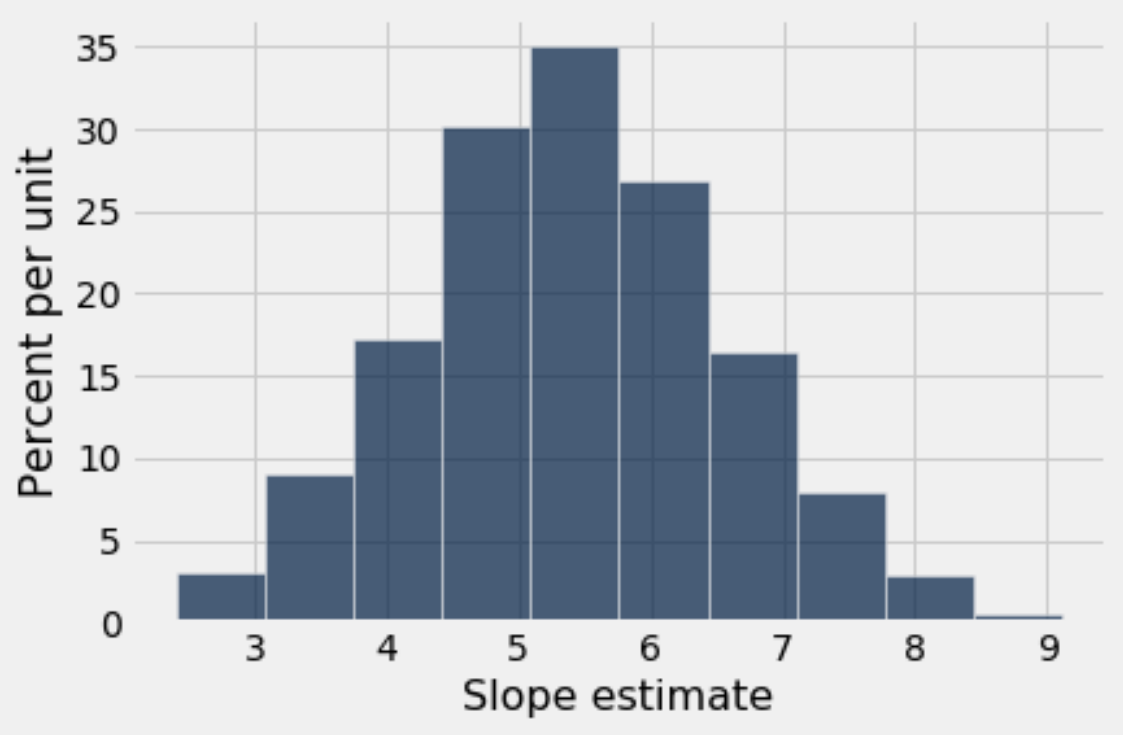
`BEGIN QUESTION`

```
name: q1_4
manual: true
image: true
```

[75]:
```
resample_slopes = make_array() # SOLUTION

for i in np.arange(1000):
    faithful_resample = faithful.sample(with_replacement=True) #SOLUTION
    resample_line = fit_line(faithful_resample) #SOLUTION
    resample_slope = resample_line.item(0) #SOLUTION
    resample_slopes = np.append(resample_slopes, resample_slope) #SOLUTION

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



Question 5 Use your resampled slopes to construct an 95% confidence interval for the true value of the slope.

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

[77]:
```
lower_end = percentile(2.5, resample_slopes) #SOLUTION
upper_end = percentile(97.5, resample_slopes) #SOLUTION
```

```
print("95% confidence interval for slope: [{:g}, {:g}]".format(lower_end,␣
↪upper_end))
```

```
95% confidence interval for slope: [3.13751, 7.72107]
```

Question 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?

```
BEGIN QUESTION
name: q1_6
manual: true
```

SOLUTION: We would reject the null, since 0 is not within the approximate 95% confidence interval. If we use an approximate 95% confidence interval, we're using a 0.05 cutoff.

Question 7 If you think the true slope is not 0, what do you think it is? You do not need an exact number.

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

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

SOLUTION: With 95% confidence, we can say that the true slope is somewhere between

$$[3.12054, 7.59651]$$

The confidence interval generated may be different for the student. The approximate 95% confidence interval that we generate in the experiment not only allows us to make a conclusion about whether the true slope is 0, but also allows us to define a range that the true slope falls in!

1.2 2. Finding the Bootstrap Prediction Interval

Suppose we're tourists at Yellowstone, and we'd like to know how long we'll have to wait for the next eruption. We decide to use our regression line to make some predictions for the waiting times.

But 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 1 Define the function `fitted_value`. It should take 2 arguments:

1. `table`: a table with 2 columns. We'll be predicting the values in the second column using the first.
2. `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.

Make sure to use your `fit_line` function.

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

[22]:
```
def fitted_value(table, given_x):
    line = fit_line(table) # SOLUTION
    slope = line.item(0) # SOLUTION
    intercept = line.item(1) # SOLUTION
    return slope * given_x  + intercept #SOLUTION

# Here's an example of how fitted_value is used.  This should
# compute the prediction for the wait time of an eruption that lasts
# two minutes .
two_minutes_wait = fitted_value(faithful, 2)
two_minutes_wait
```

[22]:
```
67.52659892156797
```

Question 2 The park ranger tells us that the most recent eruption lasted 6 minutes. Using `fitted_value` above, assign the variable `most_recent_wait` to the predicted wait time for the next eruption.

```
BEGIN QUESTION
name: q2_2
manual: false
```

[26]:
```
most_recent_wait = fitted_value(faithful, 6) #SOLUTION
most_recent_wait
```

[26]:
```
89.28185919744949
```

A fellow tourist raises the following objection to your prediction:

> "Your prediction depends on your sample of 272 eruptions. Wouldn't your prediction change if you had a different sample of eruptions?"

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 3 Define a function `compute_resampled_line` that takes in a table `tbl` with two columns (where we predict the second column using the first) and returns the parameters of the best fit line for one resample of the table.

```
BEGIN QUESTION
name: q2_3
manual: false
```

[29]:
```
def compute_resampled_line(tbl):
    resample = tbl.sample() # SOLUTION
    resampled_line = fit_line(resample) # SOLUTION
    return resampled_line # SOLUTION
```

Run the following cell below in order to define the function `bootstrap_lines`. It takes in two arguments: 1. `tbl`: a table with two columns. As usual, we'll be predicting the second column using the first. 2. `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

[53]:
```
def bootstrap_lines(tbl, num_bootstraps):
    resampled_slopes = make_array()
    resampled_intercepts = make_array()
    for i in np.arange(num_bootstraps):
        resampled_line = compute_resampled_line(tbl)
        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(faithful, 1000)
regression_lines
```

[53]:
```
Slope   | Intercept
6.21444 | 53.0906
5.56935 | 56.3169
5.46153 | 56.0237
3.58488 | 64.9247
4.93445 | 60.0921
4.70836 | 59.2091
5.66927 | 55.2682
5.46112 | 56.4293
6.15556 | 54.1042
4.39267 | 61.0405
… (990 rows omitted)
```

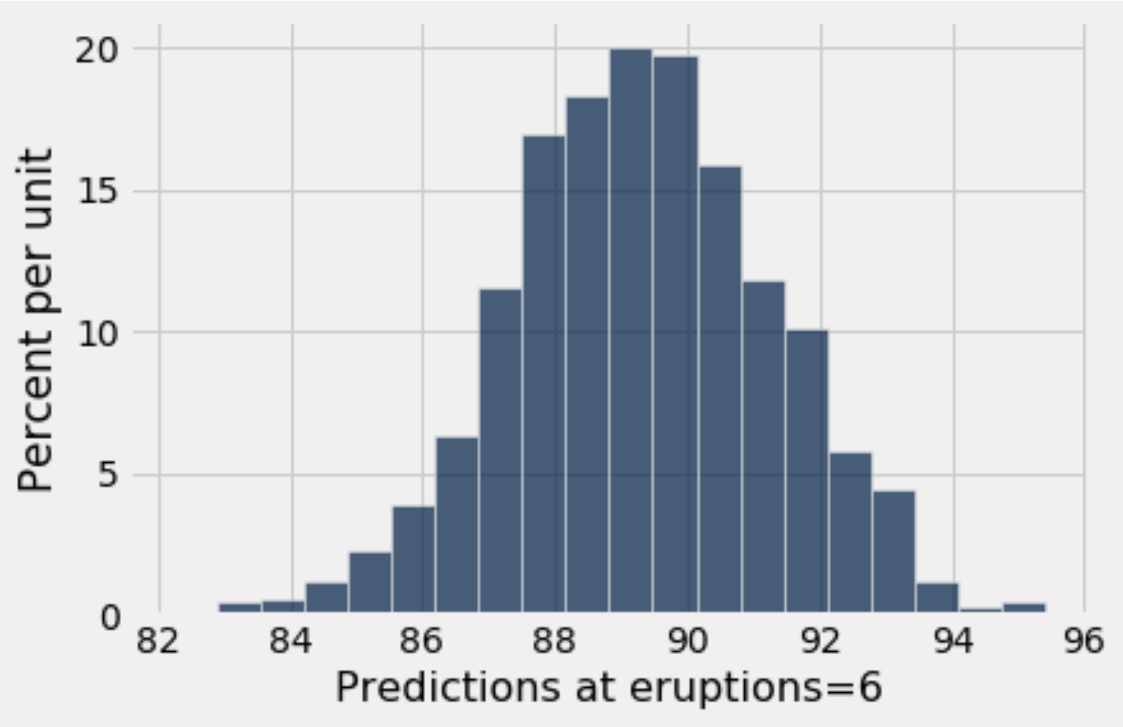
Question 4 Create an array called `predictions_for_six` that contains the predicted waiting times after an eruption with **a duration of 6 minutes** for each regression line in

regression_lines.

BEGIN QUESTION
name: q2_4
manual: true
image: true

[55]:
```
predictions_for_six = regression_lines.column('Slope') * 6 + regression_lines.
 ↪column('Intercept') # SOLUTION

# This will make a histogram of your predictions:
table_of_predictions = Table().with_column('Predictions at eruptions=6',␣
 ↪predictions_for_six)
table_of_predictions.hist('Predictions at eruptions=6', bins=20)
```



Question 5 Create an approximate 95 percent confidence interval for these predictions.

BEGIN QUESTION
name: q2_5
manual: false

[58]:
```
lower_bound = percentile(2.5,predictions_for_six) #SOLUTION
upper_bound = percentile(97.5, predictions_for_six) #SOLUTION
```

```
print('95% Confidence interval for predictions for x=6: (', lower_bound,",",␣
 ↪upper_bound, ')')
```

```
95% Confidence interval for predictions for x=6: ( 85.31911273735038 ,
93.08001136486781 )
```

Question 6 Set `faithful_statements` to an array of integer(s) that correspond to statement(s) that are true.

1. The 95% confidence interval covers 95% of the waiting times for eruptions that had a duration of 6 minutes in `faithful`.
2. The 95% confidence interval gives a sense of how much actual wait times differ from your prediction.
3. The 95% confidence interval quantifies the uncertainty in our estimate of what the true line would predict.

```
BEGIN QUESTION
name: q2_6
manual: false
```

[62]:
```
faithful_statements = make_array(3) #SOLUTION
```

Congratulations you've just completed the last lab of the semester!

Be sure to: - **run all the tests** (the next cell has a shortcut for that), - **Save and Checkpoint** from the `File` menu, - **run the last cell to submit your work**, - and ask one of the staff members to check you off.

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

[ ]:
```
# Run this cell to submit your work *after* you have passed all of the test␣
 ↪cells.
# It's ok to run this cell multiple times. Only your final submission will be␣
 ↪scored.

_ = ok.submit()
```