lab10_student_solutions

December 6, 2019

1 Lab 10: Correlation

Welcome to Lab 10!

In today's lab, we will learn about ways to understand and quantify the association between two variables.

[ ]:
```
# Run this cell, 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 plots
plots.style.use('fivethirtyeight')
import warnings
warnings.simplefilter('ignore', FutureWarning)

# These lines load the tests.
from client.api.notebook import Notebook
ok = Notebook('lab10.ok')
_ = ok.submit()
```

2 1. How Faithful is Old Faithful?

Old Faithful is a geyser in Yellowstone National Park that is famous for eruption on a fairly regular schedule. Run the cell below to see Old Faithful in action!

[2]:
```
# For the curious: this is how to display a YouTube video in a
# Jupyter notebook.  The argument to YouTubeVideo is the part
# of the URL (called a "query parameter") that identifies the
# video.  For example, the full URL for this video is:
#   https://www.youtube.com/watch?v=wE8NDuzt8eg
from IPython.display import YouTubeVideo
```

```
YouTubeVideo("wE8NDuzt8eg")
```

[2]:

Some of Old Faithful's eruptions last longer than others. Whenever there is a long eruption, it usually followed by an even longer wait before the next eruption.

If you visit Yellowstone, you might want to predict when the next eruption will happen, so you can see the rest of the park in the meantime instead of waiting by the geyser. Today, we will use a dataset on eruption durations and waiting times to see how closely these variables are related - if there is a strong relationship, we should be able to predict one from the other.

The dataset has one row for each observed eruption. It includes the following columns: - duration: Eruption duration, in minutes. - wait: Time between this eruption and the next, also in minutes.

Run the next cell to load the dataset.

[3]:
```
faithful = Table.read_table("faithful.csv")
faithful
```

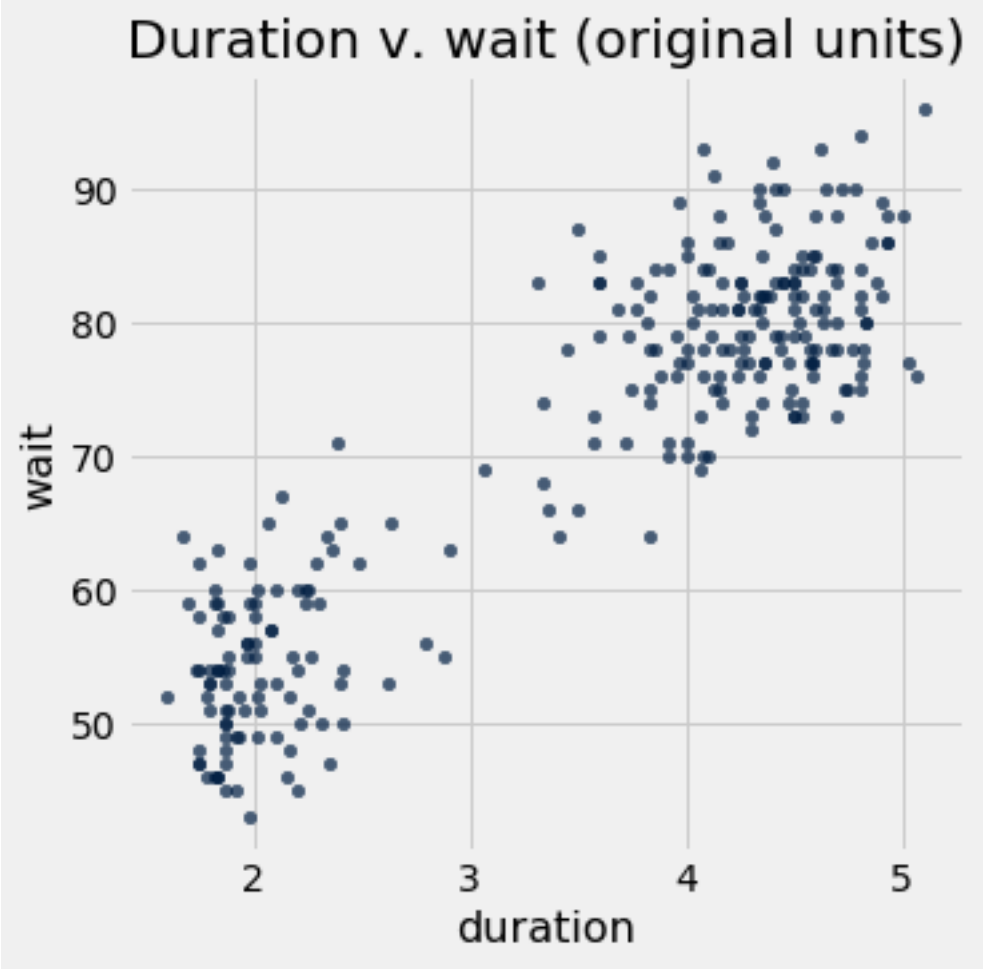
[3]:
```
duration | wait
3.6      | 79
1.8      | 54
3.333    | 74
2.283    | 62
4.533    | 85
2.883    | 55
4.7      | 88
3.6      | 85
1.95     | 51
4.35     | 85
… (262 rows omitted)
```

Let's first look at our data to see whether we can visually identify a linear relationship, which is what the correlation coefficient measures.

Question 1.1. Make a scatter plot of the data. It's convention to put the column we want to predict on the vertical axis and the other column on the horizontal axis.

BEGIN QUESTION
name: q1_1

[4]:
```
faithful.scatter("duration") #SOLUTION
plots.title('Duration v. wait (original units)');
```



Question 1.2. Are eruption duration and waiting time roughly linearly related based on the scatter plot above? Is this relationship positive?

BEGIN QUESTION
name: q1_2

SOLUTION: Yes, they are roughly linearly related. The eruption durations seem to cluster; there are a bunch of short eruptions and a bunch of longer ones. But the data in both clusters fall roughly on a line, and that's what's important when it comes to predicting waiting times with linear regression. The relationship is positive, meaning that longer eruptions have longer waiting times, as we claimed.

We're going to continue with the assumption that they are linearly related, and quantify the strength of this linear relationship.

We'd next like to plot the data in standard units. If you don't remember the definition of standard units, textbook section 14.2 might help!

Question 1.3. Compute the mean and standard deviation of the eruption durations and waiting times. **Then** create a table called faithful_standard containing the

eruption durations and waiting times in standard units. The columns should be named duration (standard units) and wait (standard units).

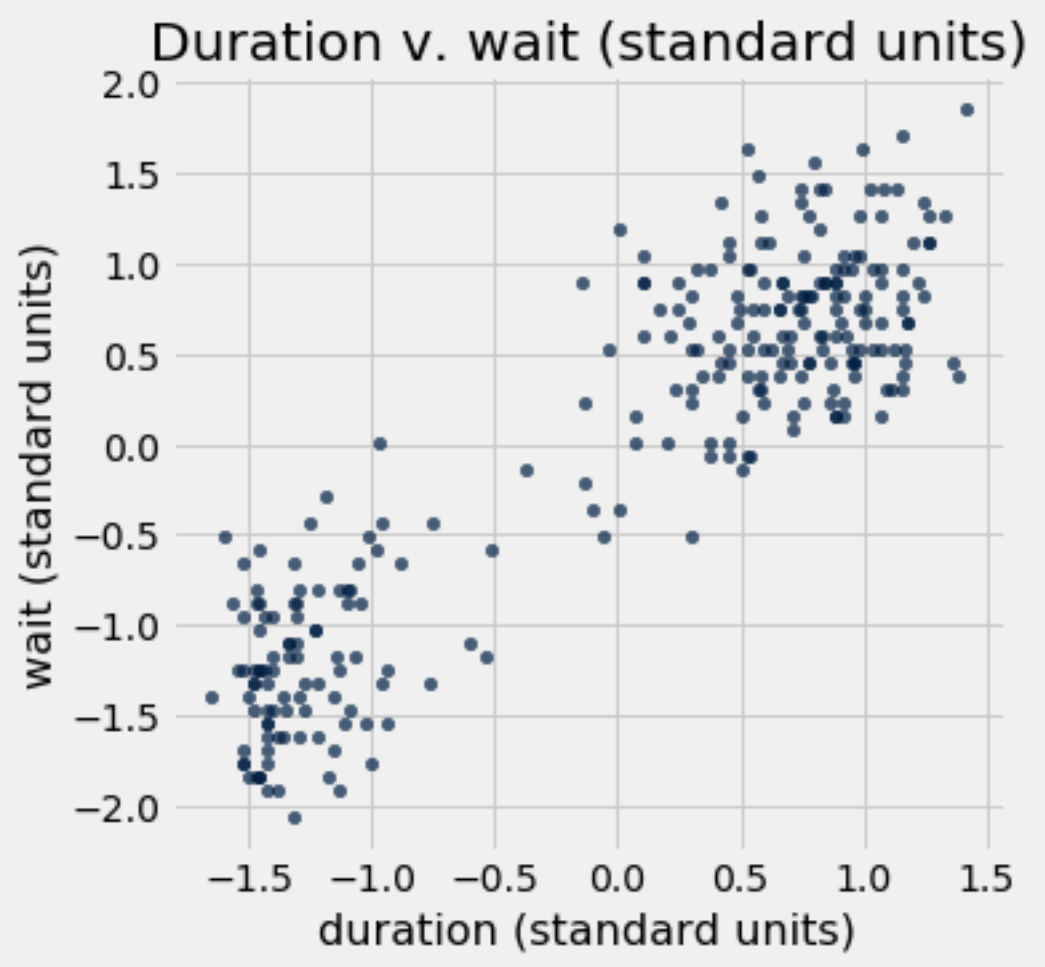
BEGIN QUESTION
name: q1_3

[5]:
```
# BEGIN SOLUTION NO PROMPT
duration_mean = np.mean(faithful.column("duration"))
duration_std = np.std(faithful.column("duration"))
wait_mean = np.mean(faithful.column("wait"))
wait_std = np.std(faithful.column("wait"))

faithful_standard = Table().with_columns(
    "duration (standard units)", (faithful.column("duration") - duration_mean) /
 ↪ duration_std,
    "wait (standard units)", (faithful.column("wait") - wait_mean) / wait_std)
faithful_standard
# END SOLUTION
""" # BEGIN PROMPT
duration_mean = ...
duration_std = ...
wait_mean = ...
wait_std = ...

faithful_standard = Table().with_columns(
    "duration (standard units)", ...,
    "wait (standard units)", ...)
faithful_standard
"""; # END PROMPT
```

Question 1.4. Plot the data again, but this time in standard units.

BEGIN QUESTION
name: q1_4

[9]:
```
faithful_standard.scatter(0) #SOLUTION
plots.title('Duration v. wait (standard units)');
```



You'll notice that this plot looks the same as the last one! However, the data and axes are scaled differently. So it's important to read the ticks on the axes.

Question 1.5. Among the following numbers, which would you guess is closest to the correlation between eruption duration and waiting time in this dataset?

1. -1
2. 0
3. 1

Assign correlation to the option (1, 2, or 3) corresponding to your guess.

BEGIN QUESTION
name: q1_5

[10]:
```
correlation = 3 # SOLUTION
```

Question 1.6. Compute the correlation r.

Hint: Use faithful_standard. Section 15.1 explains how to do this.

BEGIN QUESTION
name: q1_6

[12]:
```
r = np.mean(faithful_standard.column(0) * faithful_standard.column(1)) #SOLUTION
r
```

[12]:
```
0.9008111683218132
```

3 2. Cheese and Doctorates

We'll now investigate the relationship between two unusual variables. For every year between 2000 and 2009 (inclusive), we have data on the per-capita consumption of mozzarella cheese in that year and the number of civil engineering doctorates awarded in that year. These are real data from the U.S. Department of Agriculture and the National Science Foundation.

We can perform the same process that we performed above to investigate whether there is an association between the cheese consumption in a certain year and the number of civil engineering degrees awarded in that year. Just because we can do something, though, doesn't mean it is meaningful. While you carry out the following process, think about whether the analysis you are performing is meaningful or not.

Question 2.1. Run the next cell to load in the dataset.

BEGIN QUESTION
name: q2_1

[14]:
```
cheese_doctors = Table().with_columns(
    "Cheese Consumption", make_array(9.3, 9.7, 9.7, 9.7, 9.9, 10.2, 10.5, 11,␣
↪10.6, 10.6),
    "Civil Engineering Doctorates", make_array(480, 501, 540, 552, 547, 622,␣
↪655,701, 712, 708))
cheese_doctors
```

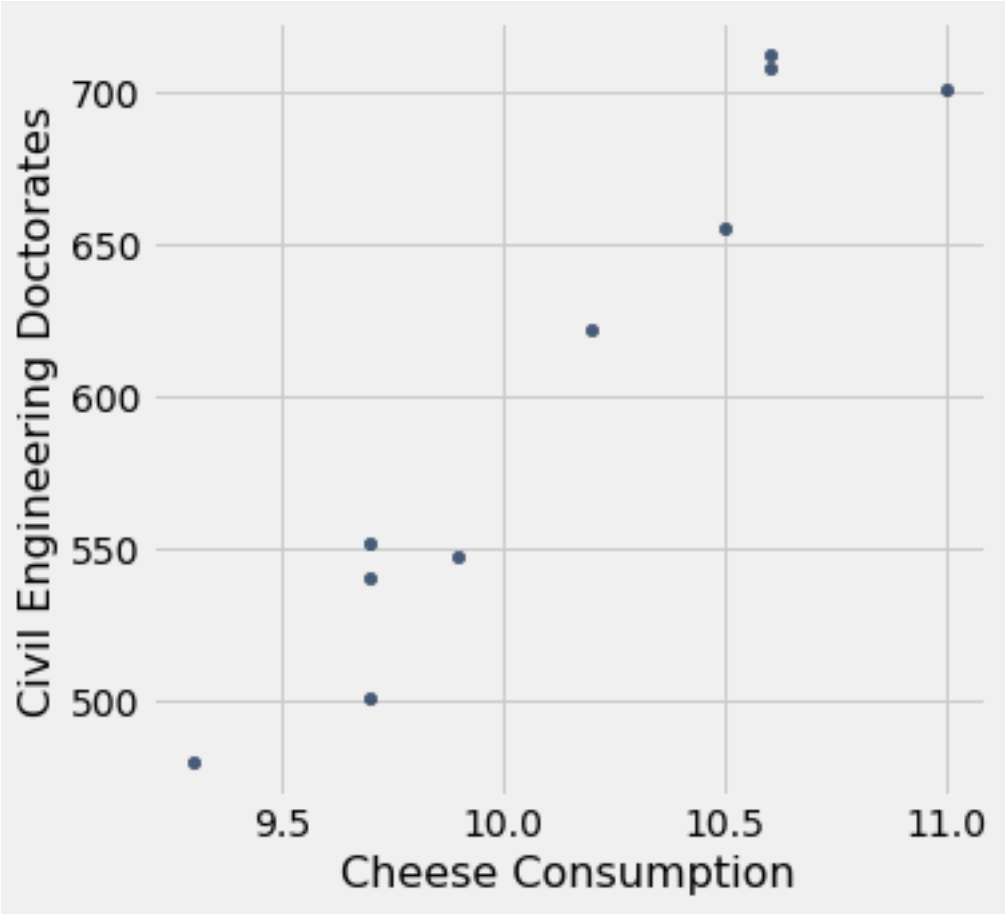
[14]:

Cheese Consumption	Civil Engineering Doctorates
9.3	480
9.7	501
9.7	540
9.7	552
9.9	547
10.2	622
10.5	655
11	701
10.6	712
10.6	708

Question 2.2. Let's visually inspect the relationship in the table. Make a scatter plot which displays mozzarella cheese consumption on the x-axis and number of Civil Engineering degrees on the y-axis.

BEGIN QUESTION
name: q2_2

[15]:
```
cheese_doctors.scatter("Cheese Consumption", "Civil Engineering Doctorates") #␣
↪SOLUTION
```



Question 2.3. Write a function called correlation_from_table that takes as arguments the name of a table and the names of two columns which contain numerical values. The function should return the correlation coefficient (a single number) between the two variables.

BEGIN QUESTION
name: q2_3

[16]:
```
def correlation_from_table(table, col_x, col_y):
    data_x = table.column(col_x) # SOLUTION
```

```
    data_y = table.column(col_y) # SOLUTION
    standard_units_x = (data_x - np.mean(data_x))/np.std(data_x) # SOLUTION
    standard_units_y = (data_y - np.mean(data_y))/np.std(data_y) # SOLUTION
    correlation = np.mean(standard_units_x * standard_units_y) # SOLUTION
    return correlation # SOLUTION
```

Question 2.4. Call your correlation_from_table function on cheese_doctors to find the correlation coefficient between the two variables.

BEGIN QUESTION
name: q2_4

[18]:
```
correlation_from_table(cheese_doctors, "Cheese Consumption", "Civil Engineering␣
↪Doctorates") # SOLUTION
```

[18]: 0.9586477872804794

You should have found a strong correlation (close to 1) between the two variables. But how should we interpret this value?

Question 2.5. Does the high degree of linear association between these two variables tell us anything about a causal relationship between mozzarella cheese consumption and civil engineering doctorates? If we knew the amount of mozzarella cheese eaten per capita in 2010, would it make sense to try to use this to predict the number of civil engineering graduates in that year?

BEGIN QUESTION
name: q2_5

SOLUTION: No, this correlation coefficient only tells us the strength of the association between mozzarella cheese consumption and number of civil engineering doctorates, but it doesn't provide any causal evidence between the two variables. Remember, two variables can have an association, but neither has to cause the other. Since this correlation measures the association only in the time period of 2000-2009, and mozzarella cheese clearly doesn't impact how many people finish civil engineering PhD programs, it wouldn't make sense to try to use this data to predict the number of civil engineering doctorates in 2010 from the per capita cheese consumption in 2010.

What you've just seen is an example of a spurious correlation, in which a relationship between two variables is purely due to chance. There are many such examples at https://www.tylervigen.com/spurious-correlations, where we found these data. This is an amusing website to spend a few minutes on, and it serves as a reminder that we should not use correlation as justification or evidence that two variables are related in a mechanistic or causal way. As you've probably heard before, ``correlation does not imply causation.''

The example above is silly, and makes this statement seem obvious, but it can be dangerous to forget the difference between a correlative link and a causal link.

Now, let's investigate another situation when it's important to recognize the

limitations of correlations. Run the next cell to load a mystery dataset.

[19]:
```
mystery = Table.read_table("mystery.csv")
mystery
```

[19]:
```
x         | y
5.31801   | 220.445
-46.9561  | 2011.91
20.5586   | 60.7178
-25.1235  | 1797.85
16.6883   | -267.489
-83.7817  | 7357.47
-8.96097  | 1077.6
-43.0446  | 1953.85
31.4746   | 2446.56
-3.85866  | 411.589
… (390 rows omitted)
```

Question 2.6. We'll start by committing a sin against data science: we'll calculate correlation before investigating our data. In the following cell, assign mystery_correlation to the value of the correlation coefficient between the two variables in the mystery table.

BEGIN QUESTION
name: q2_6

[20]:
```
mystery_correlation = correlation_from_table(mystery, 'x', 'y') # SOLUTION
mystery_correlation
```

[20]:
```
-0.024711531764254033
```

Question 2.7. Based on the value of the correlation, what can we can say about the relationship between x and y?

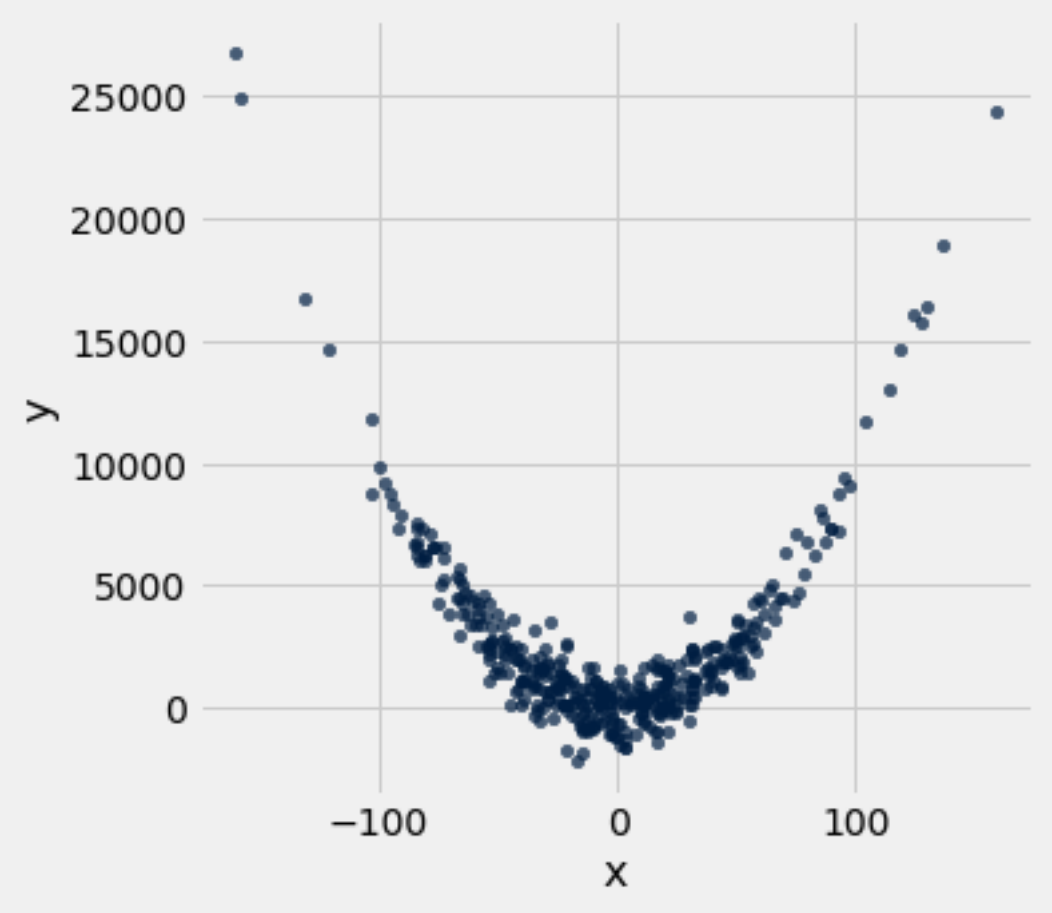
BEGIN QUESTION
name: q2_7

SOLUTION: The correlation coefficient between the two variables is close to 0. There is not a strong linear relationship between x and y.

Question 2.8. Now let's see why it's valuable to investigate our data visually first. Create a scatter plot of the data in the mystery table.

BEGIN QUESTION
name: q2_8

[22]:
```
mystery.scatter('x', 'y') # SOLUTION
```



Question 2.9. What do you see? Is there a linear relationship between x and y? Is there some other kind of relationship? Why did we get the value for the correlation coefficient that we got? Discuss your answers with a neighbor or TA, then summarize below.

BEGIN QUESTION
name: q2_8

SOLUTION: There is a clear relationship between the x and y values, but it is not linear. There is not a general trend in which the y value goes down or up when x goes up. The correlation coefficient cannot capture non-linear relationships, even if they're very strong.

That's it! You've completed Lab 10.

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.

[19]:
```
# 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.")
```

```
Running all tests…
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()
```