KEY_Lesson23_Basic_Stats_III_Correlations

February 18, 2020

1 Basic Statistics III: Correlations

Now that we have learned how to compute basic statistics on single variables, we will look at how to measure the relationship between two variables with *correlations*.

1.1 Background on Correlations

A **correlation** is a measure of the statistical relationship between two variables. Correlation values range from -1 to 1, where the *magnitude* (a.k.a. absolute value) of the correlation indicates the strength of the relationship and the *sign* of the correlation represents the direction of the relationship. The correlation value is often denoted with the variable **r**, so that is what we will use here.

The figure below shows some examples of *perfect*, *strong* and *weak* correlations between two variables in both the positive and negative directions. As you can notice, *perfect* correlation between two variables corresponds to $|\mathbf{r}| = \mathbf{1}$. Stronger correlations have r values with magnitude closer to 1, and weaker correlations have r values with magnitude closer to 0. When r = 0, there is no linear relationship between the two variables.

What do you notice about the difference between *positive* correlations and *negative* correlations?

1.2 Computing Correlations

Let's practice with some test data

[42]:
```
import numpy as np

data_1 = np.array([1,2,3,4,6,7,8,9])
data_2 = np.array([2,4,6,8,10,12,13,15])
data_3 = np.array([-1,-2,-2,-3,-4,-6,-7,-8])
```

Based on how we've constructed our variables, what do you expect the correlation values to be?

Visualizing the relationships may help us understand this better:

Now, let's calculate the actual correlatiton values. We will use the `corrcoef` function from `numpy` to calculate correlation values.

[60]:
```python
r = np.corrcoef([data_1,data_2,data_3])
print(r)
```

```
[[ 1.          0.99535001 -0.9805214 ]
 [ 0.99535001  1.         -0.97172394]
 [-0.9805214  -0.97172394  1.        ]]
```

Does the output of this function make sense to you?

This function returns a *correlation matrix*, which always has 1's along the diagonal and is *symmetric* (i.e. same values above the diagonal as below). This is so you can compute correlations of more than one variable at a time. The correlation values in the matrix above correspond to the following relationships:

Based on these plots, can you figure out why all correlation matrices have: * 1's on the diagonal? **(Talk about how the diagonal is always the correlation of one variable with itself, which will always be perfect correlation)** * symmetric entries? **(Talk about how the corr(data1, data2) == corr(data2, data1))**

So, the output of the `corrcoef` function from above is a correlation matrix follows the following form:

_	data_1	data_2	data_3
data_1	1	0.995	-0.980
data_2	0.995	1	-0.971
data_3	-0.980	-0.971	1

Now, it should be clear why a correlation matrix always has 1's along the diagonal - every variable has perfect positive correlation with itself. Furthermore, it is symmetric because the correlation of `data_1` & `data_2` is the same as the correlation of `data_2` & `data_1`.

Now that we understand our output, let's check the correlations between the variables in the `iris` dataset.

[63]:
```python
# load and preview iris
import pandas as pd
path = 'https://raw.githubusercontent.com/GWC-DCMB/ClubCurriculum/master/'
iris = pd.read_csv(path + 'SampleData/iris.csv')
iris.head()
```

[63]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[62]:
```python
# find correlations between sepal_length, sepal_width, petal_length, petal_width
# HINT: Think back to how we subset certain columns in pandas
```

```python
iris_corrs = np.corrcoef(iris.iloc[:,0:4], rowvar=False)
print(iris_corrs)
```

```
[[ 1.         -0.11756978  0.87175378  0.81794113]
 [-0.11756978  1.         -0.4284401  -0.36612593]
 [ 0.87175378 -0.4284401   1.          0.96286543]
 [ 0.81794113 -0.36612593  0.96286543  1.        ]]
```

You'll notice this time we included the `rowvar` parameter - this is because, by default, the `corrcoef` function expects that each row represents a variable, with observations in the columns. In our case it is the opposite - each column represents a variable, while the rows contain observations. So here we change the value of `rowvar` from the default `True` to `False`.

In this lesson you learned:

- How to measure the relationship between two variables
- The difference between positive/negative correlations and strong/weak correlations
- How to compute and interpret correlations for multiple variables

Now, lets continue to practice!