Cheat Sheet: Pearson Correlation

I399: Research Methods / Prof. Simon DeDeo & AI Alexander Barron

Step One: Covariance

Say you have N measurements of two quantities—in the lecture example, we had "the number of protests" and "the number of violent protests". If we call the first quantity x_i, with i labeling the measurement number, and the second quantity y_i, we can define the covariance of x and y, or "covariance(x, y)" as

$$\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

where xbar is the average value of x in the data, and ybar is the average value of y in the data. In words, for each country, multiply the difference between x and xbar with the difference between y and ybar. In python code, we write two bits of code to do this:

```
def mean(x):
     sum=0.0
     for i in x:
          sum += i
     return sum/len(x)

def covariance(x, y):
     xmean=mean(x)
     ymean=mean(y)
     sum=0
     for i in xrange(len(x)):
          sum += (x[i]-xmean)*(y[i]-ymean)
     return sum/len(x)
```

Here are some examples:

```
>>> covariance([1,2,3], [1,2,3])
0.6666666666666666
```

(when the first list is high [compared to its mean] so is the second; the covariance is positive)

```
>>> covariance([1,2,3], [100,200,300])
66.66666666666667
```

(when the first list is high, so is the second—by a lot; the covariance is positive and larger than before)

```
>>> covariance([1,3,2], [100,300,200])
66.66666666666667
```

(the order doesn't matter: if I'm high when you're high, and I'm low when you're low, we have strong positive covariance)

```
>>> covariance([1,2,3], [5,4,1])
-1.3333333333333333
```

(when the first list is high, the second is low—covariance is negative)

Step Two: Correlation

Notice how correlation worked in a pair of previous examples:

```
>>> covariance([1,2,3], [1, 2, 3])
0.6666666666666666

>>> covariance([1,2,3], [100, 200, 300])
66.66666666666667
```

When we magnify one of the lists, the covariance gets larger. This is strange, since now the covariance will depend upon the units. (Imagine, for example, I am looking at the relationship between salary and age; if I measure salary in thousands of dollars vs. dollars vs. pennies the covariance will change enormously.)

We can correct for effects like these by computing the Pearson correlation. This is defined as the covariance of x and y, divided by the square-root of the covariance of x with itself, and the covariance of y with itself.[1] It's easier to say that in Python:

```
def pearson(x, y):
     return covariance(x, y)/(covariance(x, x)*covariance(y, y))**0.5
```

Pearson correlation is strictly between minus one and one; it is independent of the units you use. It is *exactly one* when the relationship between the two quantities is linear (i.e., when you can draw a straight line through all the data points) and the slope is positive:

```
>>> pearson([1,2,3], [100, 200, 300])
1.0
>>> pearson([1,2,3], [1, 2, 3])
1.0
```

it is exactly minus one when it's linear and the slope is negative:

```
>>> pearson([1,2,3], [-1, -2, -3])
-1.0
```

and it's somewhere in between when the relationship is weaker; here are two examples—the first has a relationship that's mostly positive—high in the first value is usually high in the second (but not always):

```
>>> pearson([1,2,3], [1, 0.5, 7])
0.8293962196513646
```

while the second has a relationship that's mostly negative:

```
>>> pearson([1,2,3], [123.0, 60.0, 78.0])
-0.6933752452815364
```

1. *Why not divide by the means instead of the covariances?* This would make the correlation dependent on the origin (so if you added 100 to all the *x*s, you would get a different answer). *Why the square root?* An easy way to see why is to check the units; say x and y are both in dollars; you want the pearson correlation to be dimensionless, so you have (dollars x dollars) on top, and sqrt(dollars x dollars x dollar x dollars) on the bottom.