Math221_Summary_Statistics

January 30, 2018

Math 221 - Applied Statistics (Data Analysis)
Dr. Kamal Dingle (Spring 2018)
Summary Statistics
Computing summary statistics is easy in Python.
First let's make some data

```
In [1]: import numpy as np

        X = 5*np.random.rand(100) # 100 uniform random numbers between 0 and 5
```

Let's find the mean of X

```
In [2]: np.mean(X)

Out[2]: 2.6291180609569391
```

Which makes sense, because it is about half way beteen 0 and 5.
Now let's find the median

```
In [3]: np.median(X)

Out[3]: 2.8271151434378794
```

It is not worthwhile trying to find the mode of X, because probably the numbers are all different, or with very few repetitions. So, to practice finding the mode we will make some fake data:

```
In [4]: Data = [1,2,3,3,3,4,2,5,6,3,6,7,8,7,5,3,4,3,3]
```

We could use numpy to find the mode, but it is easier to use another library called **scipy**

```
In [5]: from scipy.stats import mode

In [6]: mode(Data)

Out[6]: ModeResult(mode=array([3]), count=array([7]))
```

Which tells you that the mode is 3, and the number 3 occured 7 times.
Now for some measures of spread. We will analyse the data in X again.
We start with the range

```
In [7]: The_Range = np.ptp(X) # The name is funny: ptp comes from Peak-To-Peak (ptp)
        print (The_Range)
```

```
4.91973172095
```

Which is what we expect, because the random numbers were between 0 and 5, so should have a range of about 5

Now for the standard deviation and variance

```
In [8]: sigma = np.std(X) # standard deviation
        print (sigma)
```

```
1.46419290072
```

```
In [9]: sigmasquared = np.var(X) # variance
        print (sigmasquared)
```

```
2.14386085052
```

Recall from the class that std = square root of variance, so we'll quickly verify this.

```
In [10]: np.sqrt(sigmasquared)
```

```
Out[10]: 1.4641929007196142
```

Which is the same as sigma.

Finally we calculate the linear correlation as follows, but we first need to correlate *with something*, so lets make some more random data:

```
In [11]: Y = 5*np.random.rand(100) # 100 uniform random numbers between 0 and 5
         np.corrcoef(X,Y)
```

```
Out[11]: array([[ 1.        ,  0.01873433],
                [ 0.01873433,  1.        ]])
```

The linear correlation is in the top-right (and bottom left). The correlation is very close to zero, which is what we expect for completely random numbers.

But if we make a new variable Z

```
In [12]: Z = X + Y
```

Then we expect Z to correlate with X (and also Y). Let's see if it does.

```
In [13]: np.corrcoef(X,Z)
```

```
Out[13]: array([[ 1.        ,  0.71252614],
                [ 0.71252614,  1.        ]])
```

Yes indeed it does, the correlation is much stronger now, and in fact it is a **positive correlation**