Stats Basic with Python

horizontal line

## Covariance and Correlation

### Covariance

Covariance indicates how two variables are related. A positive covariance means the variables are positively related, while a negative covariance means the variables are inversely related. The formula for calculating covariance of sample data is shown below.



*x* = the independent variable

*y* = the dependent variable

*n* = number of data points in the sample

 = the mean of the independent variable *x*

** = the mean of the dependent variable *y*

#### Python code:

In the following example we will take a speed of webPage and how does it increase the sales of a ecommerce website

%matplotlib inline

**import** numpy **as** np

**from** pylab **import** \*

**def** de\_mean(x):

xmean = mean(x)

**return** [xi - xmean **for** xi **in** x]

**def** covariance(x, y):

n = len(x)

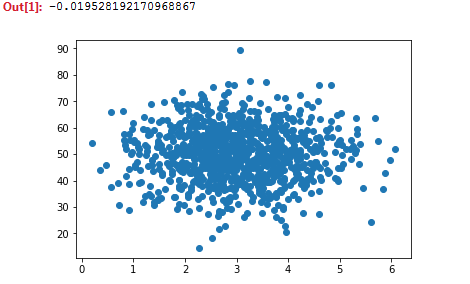
**return** dot(de\_mean(x), de\_mean(y)) / (n-1)

pageSpeeds = np.random.normal(3.0, 1.0, 1000)

purchaseAmount = np.random.normal(50.0, 10.0, 1000)

scatter(pageSpeeds, purchaseAmount)

covariance (pageSpeeds, purchaseAmount)



### Correlation

Correlation is another way to determine how two variables are related. In addition to telling you whether variables are positively or inversely related, correlation also tells you the degree to which the variables tend to move together.

Correlation standardizes the measure of interdependence between two variables and, consequently, tells you how closely the two variables move. The correlation measurement, called a correlation coefficient, will always take on a value between 1 and – 1:  
  
1) If the correlation coefficient is 1, the variables have a perfect positive correlation. This means that if one variable moves a given amount, the second moves proportionally in the same direction. A positive correlation coefficient less than one indicates a less than perfect positive correlation, with the strength of the correlation growing as the number approaches one.  
2) If correlation coefficient is 0, no relationship exists between the variables. If one variable moves, you can make no predictions about the movement of the other variable; they are uncorrelated.  
3) If correlation coefficient is –1, the variables are perfectly negatively correlated (or inversely correlated) and move in opposition to each other. If one variable increases, the other variable decreases proportionally. A negative correlation coefficient greater than –1 indicates a less than perfect negative correlation, with the strength of the correlation growing as the number approaches –1.



*r(x,y)* = correlation of the variables *x* and *y*

*COV*(*x, y*) = covariance of the variables *x* and *y*

*sx* = sample standard deviation of the random variable *x*

*sy* = sample standard deviation of the random variable *y*

#### Python code:

If we make purchase amount and purchase speed related to each other

**def** correlation(x, y):

stddevx = x.std()

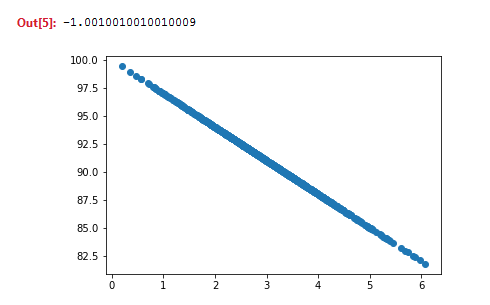
stddevy = y.std()

**return** covariance(x,y) / stddevx / stddevy

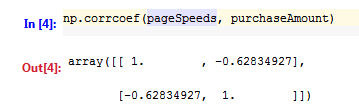
purchaseAmount = 100 - pageSpeeds \* 3

scatter(pageSpeeds, purchaseAmount)

correlation (pageSpeeds, purchaseAmount)



Python has a numpy package. It will correlation between n variables. In the example below, array[0][1] & array[1][0] gives correlation between pageSpeeds and purchaseAmount. array[0][0] and array[1][1] will of course be 1 because they are correlating to themself.



## Mean Median Mode

### Mean

The mean is found by adding up all of the given data and dividing by the number of data entries

Python Code:

We create a fabricated of income 270000, with a normal distribution and standard deviation of 15000, with 10000 data points

**import** numpy **as** np

incomes = np.random.normal(27000, 15000, 10000)

np.mean(incomes)

Output: 27006.630375676905

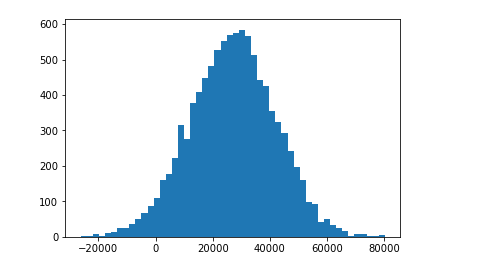
We can segment the income data into 50 buckets, and plot it as a histogram: We can see that max points are around 27000 and there is a sharp dip at 13000 and 42000

%matplotlib inline

**import** matplotlib.pyplot **as** plt

plt.hist(incomes, 50)

plt.show()



### Median

The median is the middle number. First you arrange the numbers in order from lowest to highest, then you find the middle number by crossing off the numbers until you reach the middle.

Python Code:

Looking at the histogram above, we should get it around 27000.

np.median(incomes)

Output: 27195.153719614136

### Mode

It is the number that occurs most often

Python Code:

Lets create an array of ages between 25 and 90, and lets see which comes maximum number of times.

ages = np.random.randint(25, high=90, size=100)

ages

## 

## Conditional Probability Solution

Conditional probability can be thought of as looking at the probability of one event occurring with some relationship to one or more other events. For example:

* Event A is that it is raining outside, and it has a 0.3 (30%) chance of raining today.
* Event B is that you will need to go outside, and that has a probability of 0.5 (50%).

A conditional probability would look at these two events in relationship with one another

#### Python Code:

We will create a scenario to make a purchase if purchase probability is less than 0.4

**from** numpy **import** random

random.seed(0)

totals = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}

purchases = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}

totalPurchases = 0

**for** \_ **in** range(100000):

ageDecade = random.choice([20, 30, 40, 50, 60, 70])

purchaseProbability = 0.4

totals[ageDecade] += 1

**if** (random.random() < purchaseProbability):

totalPurchases += 1

purchases[ageDecade] += 1

PEF = float(purchases[30]) / float(totals[30])

**print**(**"P(purchase | 30s): "** + str(PEF))

PE = float(totalPurchases) / 100000.0

**print**(**"P(Purchase):"** + str(PE))

Output:

P(purchase | 30s): 0.3987604549010169

Probability that a purchase will be made if the person is in 30’s is 0.3998

P(Purchase):0.4003

Probability that a purchased will be made irrespective of the ages is 0.4003, very close to above result.

This implies that purchase and ages are independent of each other