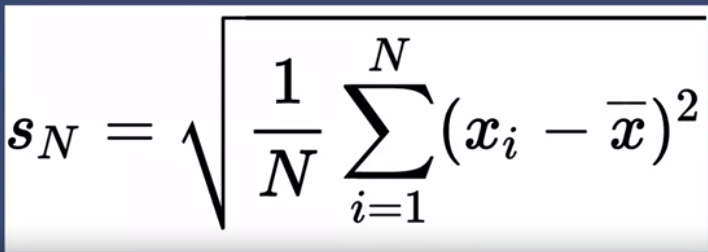
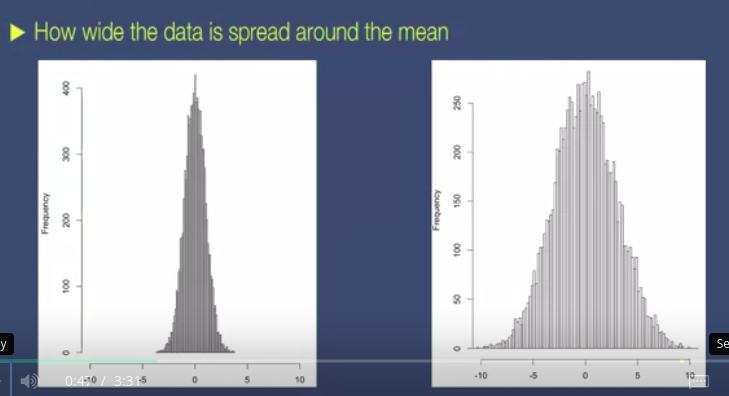
**Mean, Median**

Mean = sum/n ( sum / float(n) )

Median = m1 + m2 / 2

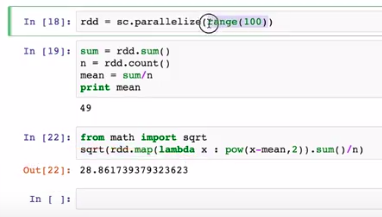
**Standard Deviation**

****

****

**Simple definition: SqRoot ( ( 1/n ) Sum(Each value - mean)^2 )**

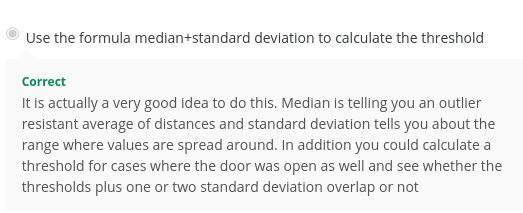
In Spark: (mean & sd)



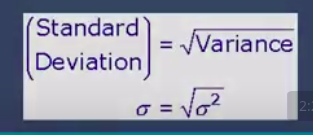
Mean is 49, but all values are spread around 28.86 ( lower/ left of the mean)

Outlier - last/ most significant value : If it s far away from the mean, sd will also change,

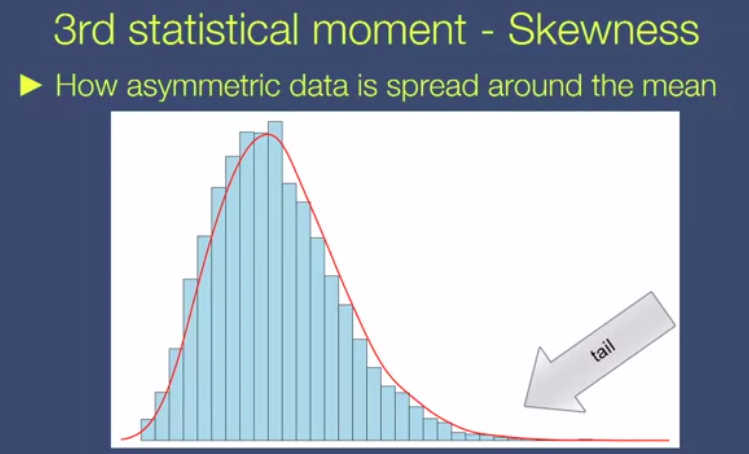


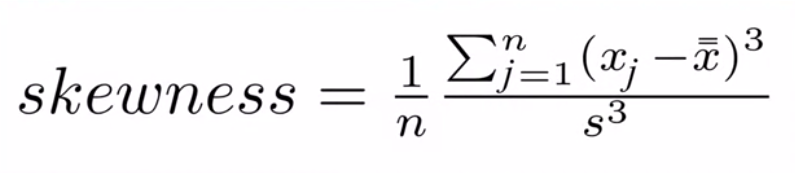


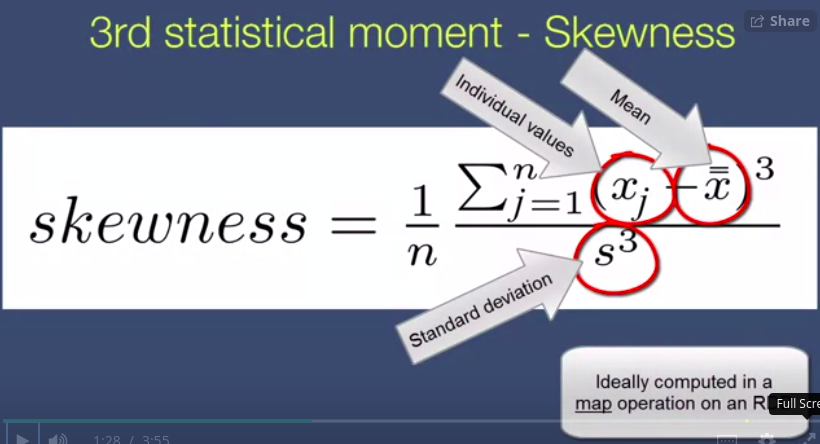
**Variance:**

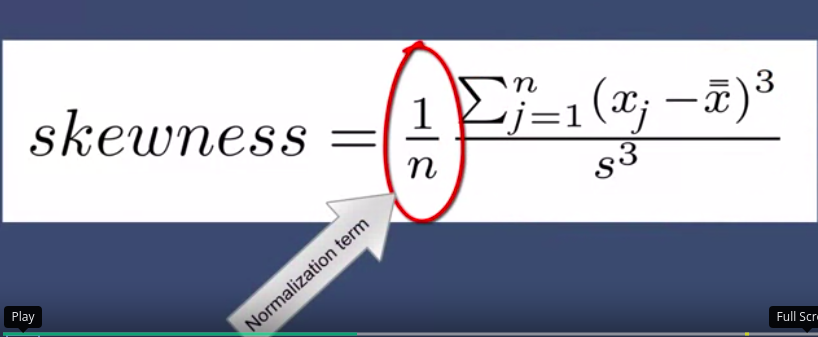


**Skewness**









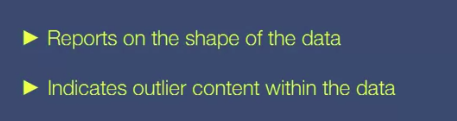
Implementing on spark:



Note: **n/( (n-1)\*(n-2))**  is wrong!!!! Use **1/n** instead

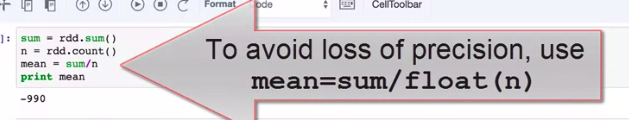
Skewness : in a histogram : **Right Skewed/ Left skewed**

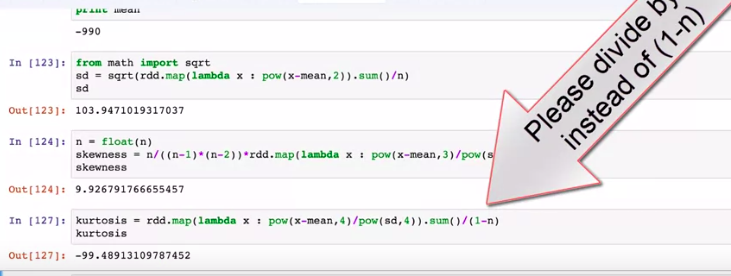
**Kurtosis**





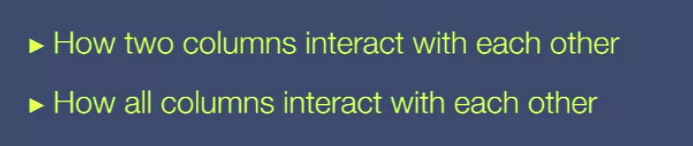
Note : Please use sum/ float(n) to find the mean in spak



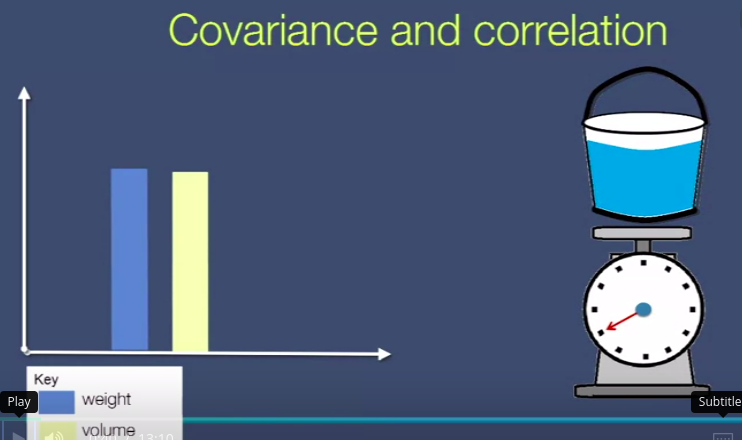


**Note: divide by n, instead of 1/n**

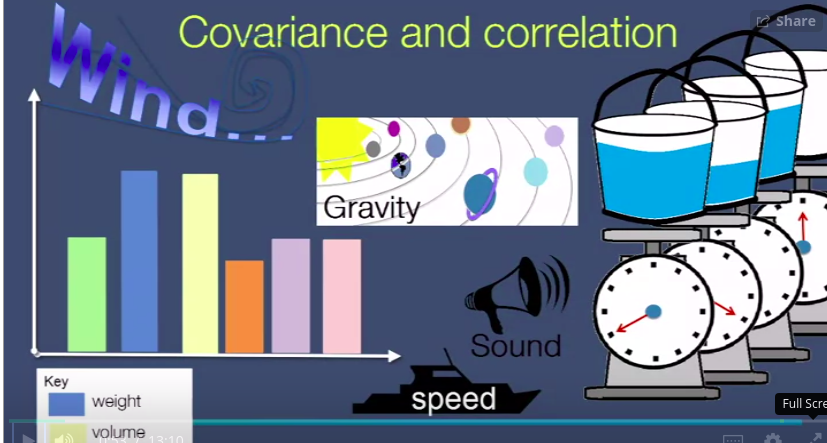
**Covariance and Covariance Matrices**



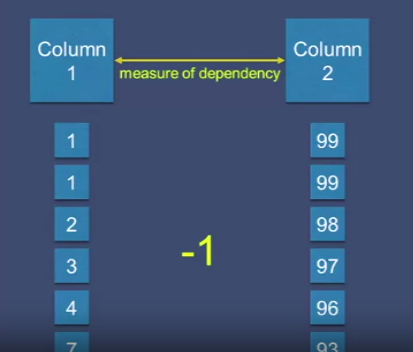
Ex: When volume increases weight increases



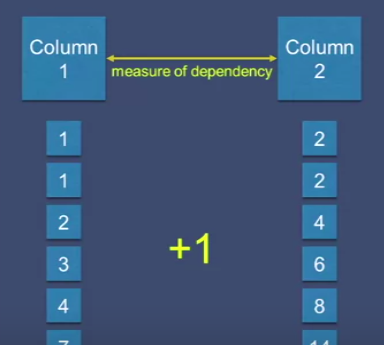
Multiple covariance:



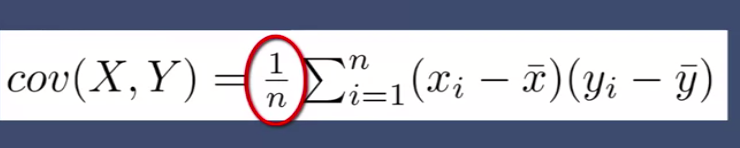
Ex: covariance is -1



Ex: covariance is +1

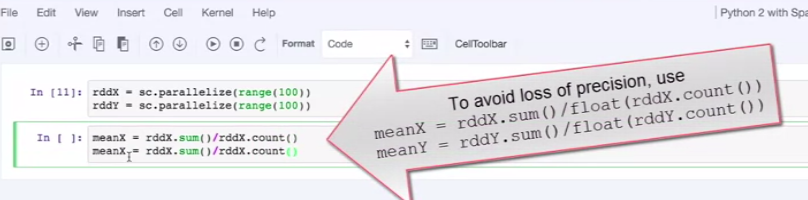


If two columns has no dependent at all, the covariance = 0



Implementation in spark:

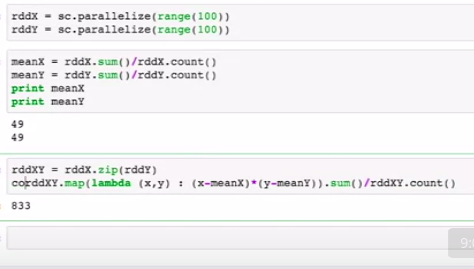
Step 1:



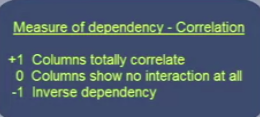
Step 2:

Note: **newRdd = rdd1.zip(rdd2)**

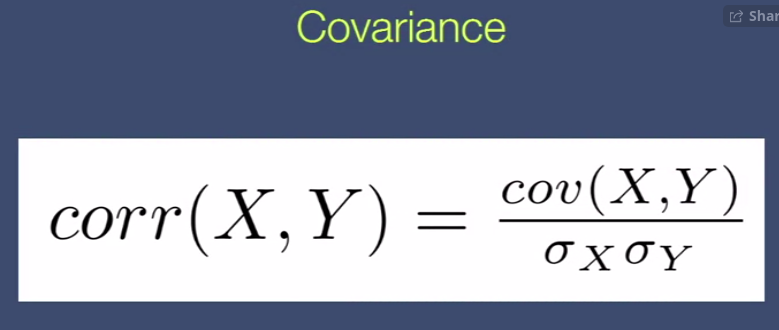
Zipping 2 rdds together as one RDD



Measures of dependency in correlation:



Correlation:

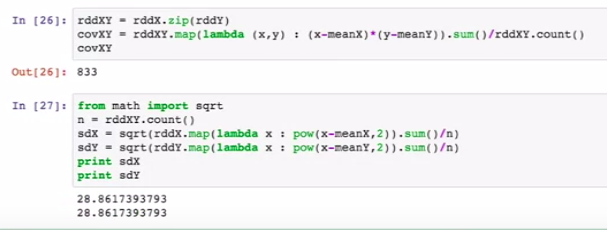


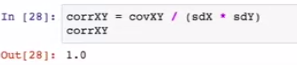
**Covariance of both columns**

**---------------------------------------**

**Sd of colX \* sd of col Y**

Implementation in spark:





Note: **2 RDDs are totally correlating with each other ( Reason: both contain same data)**

Ex 2: Reverse the 2nd RDD

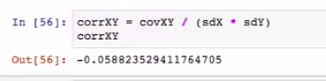


o/p : -1.0

Note: **Total inverse dependency**

Ex 3: Use 2 random lists



O/p : 

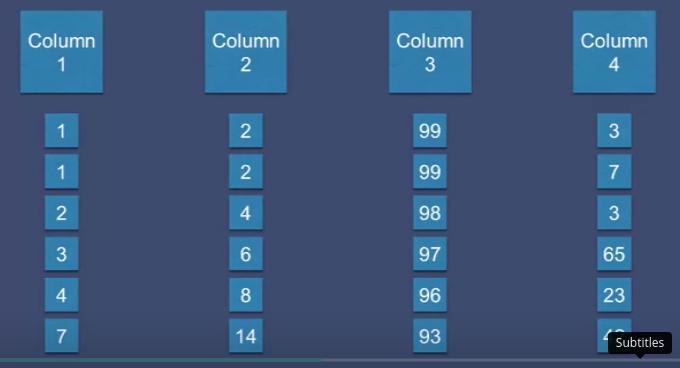
Note: value is close to 0 (no much correlation/ **if 0 - no correlation** at all)

**Correlation Matrix**



To View of all columns correlation at once..

Ex:



Col1 : col 2 (correlation = 1.0)

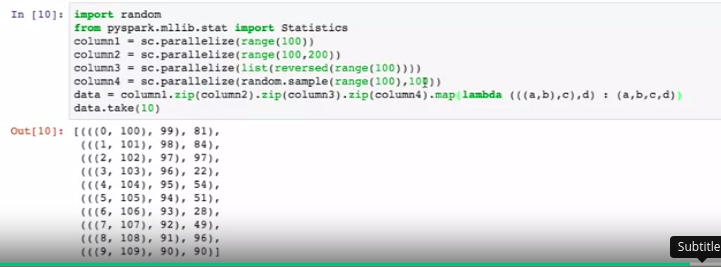
Col1 : col3 (correlation = -1.0)

Col1 : col4 (correlation = close to 0 - random values)

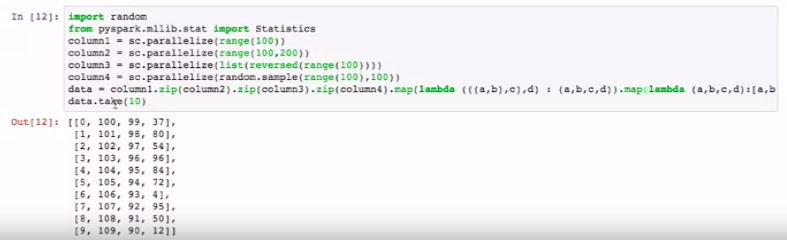
Ex:



Implementation in spark:

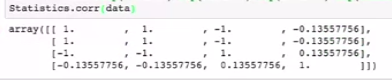


Note: **map** function is used to flatten all data to value 4 tuples



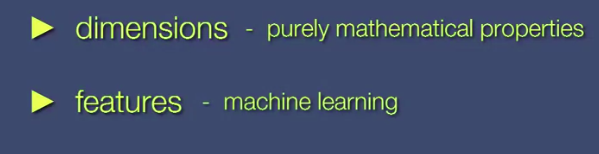
Now it looks like a relational database

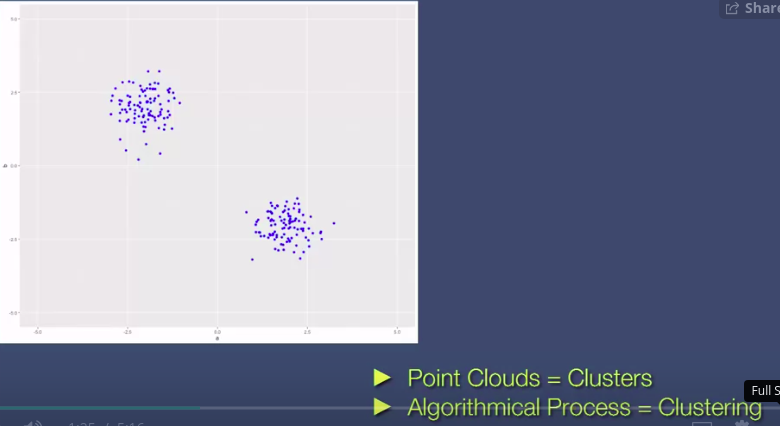
Finally, get correlations:



**Multidimensional vector specs:**

How data scientists call columns?

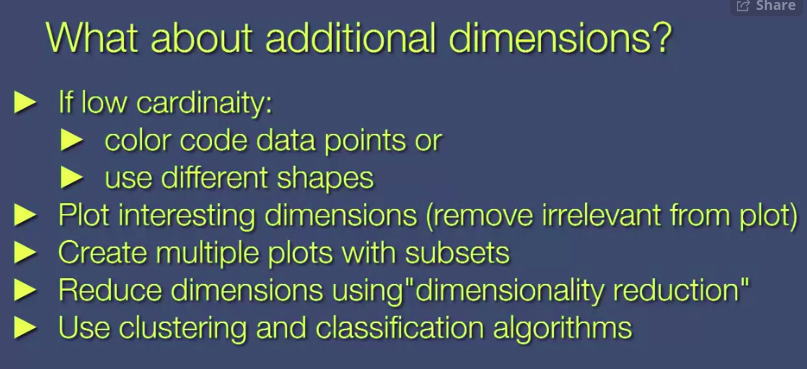




When 3 dimensions: 3-dimensional vector-space:

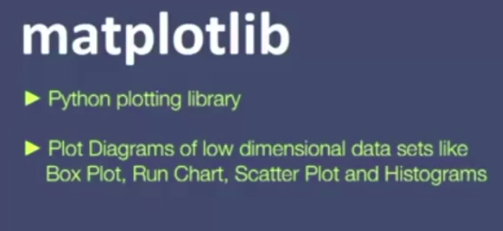


When 3+ dimensions: (hyperplane)



**Data Visualization**

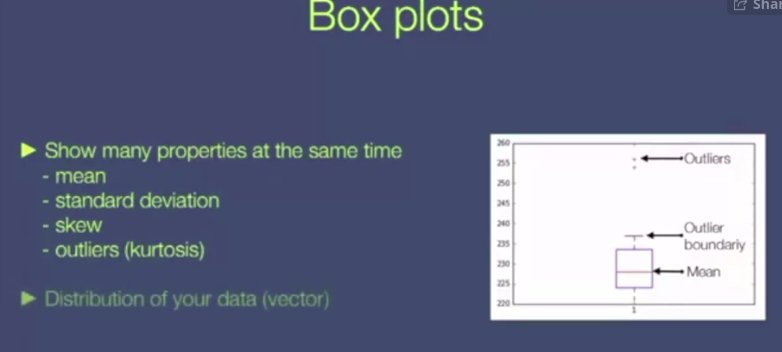
Matplotlib:



We plot only samples:

Reason: data processed in many machines ( not correctly plotted)

1. Box plot - Histogram in a vertical perspective



Showing all created matplotlib diagrams:

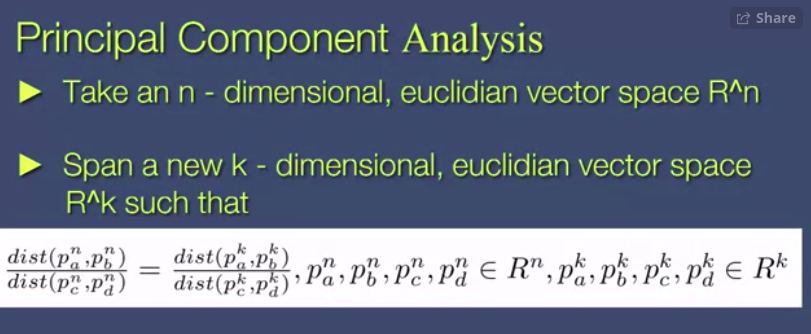
**Matplotlib inline**

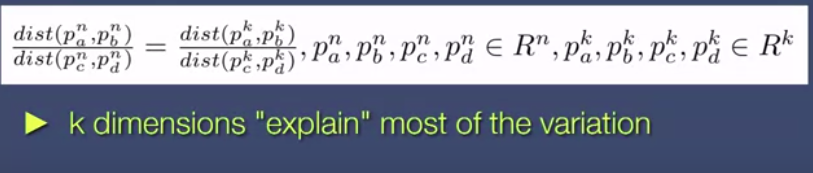
1. Run charts



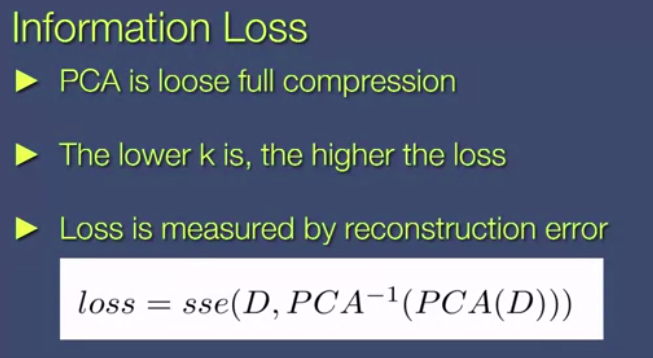
**Dimensionality reduction**

Projection process :





Loss:



D- Original DataSet

PCA - Principal Component Analysis