# Calculating correlation

Correlation is a statistical relationship between two variables such that when one variable changes, it leads to a change in the other variable. Correlation analysis measures the extent to which the two variables are correlated.

If an increase in one variable leads to an increase in another, it is called a **positive correlation**. If an increase in one variable leads to a decrease in the other, it is a **negative correlation**.

Spark supports two correlation algorithms: Pearson and Spearman. Pearson algorithm works with two continuous variables, such as a person's height and weight or house size and house price. Spearman deals with one continuous and one categorical variable, for example, zip code and house price.

## Getting ready

Let's use some real data so that we can calculate correlation more meaningfully. The following are the size and price of houses in the City of Saratoga, California, in early 2014:

| **House size (sq ft)** | **Price** |
| --- | --- |
| 2100 | $1,620,000 |
| 2300 | $1,690,000 |
| 2046 | $1,400,000 |
| 4314 | $2,000,000 |
| 1244 | $1,060,000 |
| 4608 | $3,830,000 |
| 2173 | $1,230,000 |
| 2750 | $2,400,000 |
| 4010 | $3,380,000 |
| 1959 | $1,480,000 |

Start the Spark shell:

**$ spark-shell**

Import the statistics and related classes:

**scala> import org.apache.spark.mllib.linalg.\_**

**scala> import org.apache.spark.mllib.stat.Statistics**

Create an RDD of house sizes:

**scala> val sizes = sc.parallelize(List(2100, 2300, 2046, 4314, 1244, 4608, 2173, 2750, 4010, 1959.0))**

Create an RDD of house prices:

**scala> val prices = sc.parallelize(List(1620000 , 1690000, 1400000, 2000000, 1060000, 3830000, 1230000, 2400000, 3380000, 1480000.00))**

Compute the correlation:

**scala> val correlation = Statistics.corr(sizes,prices)**

**correlation: Double = 0.8577177736252577**

0.85 means a very strong positive correlation.

# Doing hypothesis testing

Hypothesis testing is a way of determining probability that a given hypothesis is true. Let's say a sample data suggests that females tend to vote more for the Democratic Party. This may or may not be true for the larger population. What if this pattern is there in the sample data just by chance?

Another way to look at the goal of hypothesis testing is to answer this question: If a sample has a pattern in it, what are the chances of the pattern being there just by chance?

How do we do it? There is a saying that the best way to prove something is to try to disprove it.

The hypothesis to disprove is called **null hypothesis**. Hypothesis testing works with categorical data. Let's look at the example of a gallop poll of party affiliations.

| **Party** | **Male** | **Female** |
| --- | --- | --- |
| Democratic Party | 32 | 41 |
| Republican Party | 28 | 25 |
| Independent | 34 | 26 |

Start the Spark shell:

**$ spark-shell**

Import the relevant classes:

**scala> import org.apache.spark.mllib.stat.Statistics**

**scala> import org.apache.spark.mllib.linalg.{Vector,Vectors}**

**scala> import org.apache.spark.mllib.linalg.{Matrix, Matrices}**

Create a vector for the Democratic Party:

**scala> val dems = Vectors.dense(32.0,41.0)**

Create a vector for the Republican Party:

**scala> val reps= Vectors.dense(28.0,25.0)**

Create a vector for the Independents:

**scala> val indies = Vectors.dense(34.0,26.0)**

Do the chi-square goodness of fit test of the observed data against uniform distribution:

**scala> val dfit = Statistics.chiSqTest(dems)**

**scala> val rfit = Statistics.chiSqTest(reps)**

**scala> val ifit = Statistics.chiSqTest(indies)**

Print the goodness of fit results:

**scala> print(dfit)**

**scala> print(rfit)**

**scala> print(ifit)**

Create the input matrix:

**scala> val mat = Matrices.dense(2,3,Array(32.0,41.0, 28.0,25.0, 34.0,26.0))**

Do the chi-square independence test:

**scala> val in = Statistics.chiSqTest(mat)**

Print the independence test results:

**scala> print(in)**