

# Principal Component Analysis

## Task #1

Student: Sergi Blanco Cuaresma

December 8, 2010

### Abstract

Solutions to the PCA tasks from the subject Statistics, Monte Carlo Methods and Data Processing - Master in Astrophysics, Particle Physics and Cosmology (Universitat de Barcelona).

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Description . . . . .	2
1.2	Required software . . . . .	2
1.3	Design and implementation . . . . .	2
1.4	Execution . . . . .	3
<b>2</b>	<b>PCA with linear sample</b>	<b>3</b>
2.1	Design and implementation . . . . .	3
2.2	Results . . . . .	3
<b>3</b>	<b>PCA with synthetic sample</b>	<b>5</b>
3.1	Design and implementation . . . . .	5
3.1.1	Non-linear relationship . . . . .	5
3.1.2	Linear relationship . . . . .	6
3.2	Results . . . . .	6
3.2.1	Case 1: Small $\sigma_X$ and $\sigma_{Err} \ll \sigma_X$ . . . . .	6
3.2.2	Case 2: Large $\sigma_X$ and $\sigma_{Err} \ll \sigma_X$ . . . . .	9
3.2.3	Case 3: Small $\sigma_X$ and $\sigma_{Err} \sim \sigma_X$ . . . . .	11
<b>4</b>	<b>PCA with stock market sample</b>	<b>12</b>
4.1	Description of the sample . . . . .	12
4.1.1	Context . . . . .	12
4.1.2	Objects . . . . .	13
4.1.3	Adquisition process . . . . .	13
4.1.4	General characteristics . . . . .	13
4.2	Statistical analysis . . . . .	14
4.2.1	Key indicators . . . . .	14
4.2.2	Histograms . . . . .	14
4.3	Preliminary evaluation . . . . .	15
4.4	Design and implementation . . . . .	15
4.5	Results . . . . .	16
4.5.1	Financial institutions . . . . .	16
4.5.2	Financial institutions and a telecommunication company . . . . .	16

# 1 Introduction

## 1.1 Description

In order to provide a solution for the different tasks, an automatic process has been implemented using the R language, Java and Weka. This document presents the results for each task and describes how to execute the developed program for future results validations.

It is worth to mention that although the second task of the assignment gives the option of working with a synthetic or a real sample, both options has been implemented and documented.

## 1.2 Required software

The automatic process has been implemented using some pre-existing software that it is required in order to execute it:

- Java Development Kit 1.6 or above for compilation (compiled files are also included with the results, so this may not be necessary)
- Java Runtime Kit 1.6 or above
- R language
- GD2 library and development headers

In a Debian-based system (i.e. Ubuntu), the installation of the previous software can be done from a console as a root user:

```
apt-get install openjdk-6-jdk
apt-get install r-base-core
apt-get install libgd2-xpm-dev
```

Once installed, execute the following sentences from a R console (execute 'R' on a console, do not use a QT interface such as 'Rkward' because they are not compatible with the needed Java interface 'rJava'):

```
> install.packages("GDD", dependencies = TRUE)
> install.packages("scatterplot3d", dependencies = TRUE)
> install.packages("rJava", dependencies = TRUE)
```

This will install some extra R packages into your home directory.

## 1.3 Design and implementation

Weka provides the needed algorithm for a Principal Component Analysis and R is the perfect environment for data treatment. The following files use these tools and form part of the automatic process:

- 'dataAnalysis.R': Main R script that contains all the necessary operation to perform each of the Principal Component Analysis presented in this document. It will automatically read and prepare initial data, generate 3D plots, execute PCA, etc.
  - 'input' directory: Initial data.
  - 'output' directory: Results from 'dataAnalysis.R' execution such as data, PCA and 3D plots (PNG image format).
- 'WekaFileFunctions.R': Group of functions for reading and writting files in Weka format (ARFF). They have been extracted from RWeka and adapted for the specifications of the problem.
- 'PCA.java': Java class that performs the Principal Component Analysis using Weka methods. This component has been develop in order to make easier the interaction between R and Weka.

- The compiled class is also included as 'PCA.class', but in case a recompilation is needed:

```
javac -classpath "./weka.jar" PCA.java
```

- 'weka.jar': Unmodified version of weka 3.6.3 used by the previous PCA Java class.

## 1.4 Execution

For the main R script execution, it can be loaded from a R console with 'source("dataAnalysis.R")' or executed directly from a system console:

```
R --slave --vanilla -f dataAnalysis.R
```

## 2 PCA with linear sample

### 2.1 Design and implementation

The automatic process reads the original Linear data, rotate it  $-45^\circ$  around the Z axis and perform a PCA for each of them with variance covered 95% and 70%:

```
1 #-----
2 #-- First task
3 #-----
4 lineal = read.arff("input/00.Lineal.arff")
5 # Just copy to output directory
6 write.arff(lineal, file = "output/00.Lineal.arff")
7 generatePlot("00.Lineal", lineal$X, lineal$Y, lineal$Z)
8
9 # Perform PCA over original linear data
10 varianceCovered = 0.95
11 .jcall(pca, "V", "generateData", "input", "00.Lineal", "output", as.double(varianceCovered))
12
13 # -45° rotation of the linear data around Z axis
14 x = lineal$X * cos(-45 * (pi/180)) - lineal$Y * sin(-45 * (pi/180))
15 y = lineal$X * sin(-45 * (pi/180)) - lineal$Y * cos(-45 * (pi/180))
16 z = lineal$Z
17 type = rep("A", 1000)
18 linealRotated = data.frame(x, y, z, type)
19 write.arff(linealRotated, file = "output/01.Rotated_Lineal.arff")
20
21 generatePlot("01.Rotated_Lineal", linealRotated$x, linealRotated$y, linealRotated$z)
22
23 # Perform PCA over rotated Linear data
24 varianceCovered = 0.95
25 .jcall(pca, "V", "generateData", "output", "01.Rotated_Lineal", "output", as.double(varianceCovered))
```

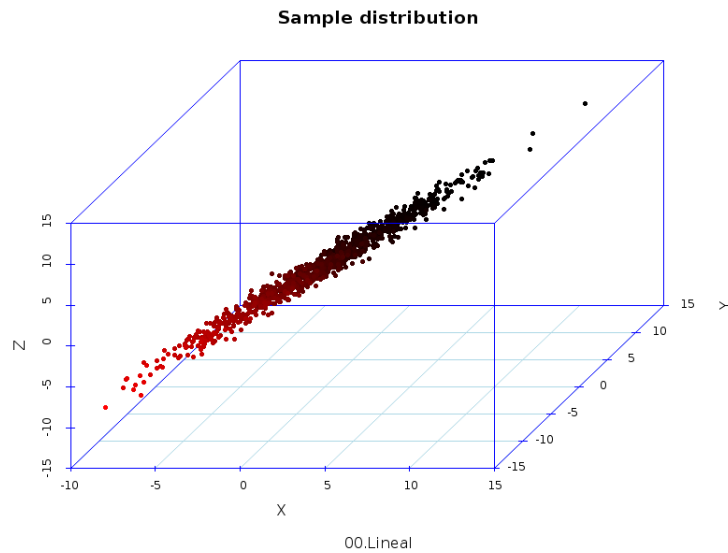
### 2.2 Results

Given the data sample 'Lineal.arff' provided by the task definition (slide 59, class notes):

```
/* Generate xyz */
x= GenGau(X0,SigX);
y= GenGau(Y0,SigY);
z= GenGau(Z0,SigZ);

/* Apply a linear transformation with noise */
x2= 3.*x + 3.*y + 3.*z + GenGau(0.,SigErr);
y2= 3.*x + 3.*y + 3.*z + GenGau(0.,SigErr);
z2= 3.*x + 3.*y + 3.*z + GenGau(0.,SigErr);
```

Which presents the following distribution:



The result of PCA for a covered variance of 95%:

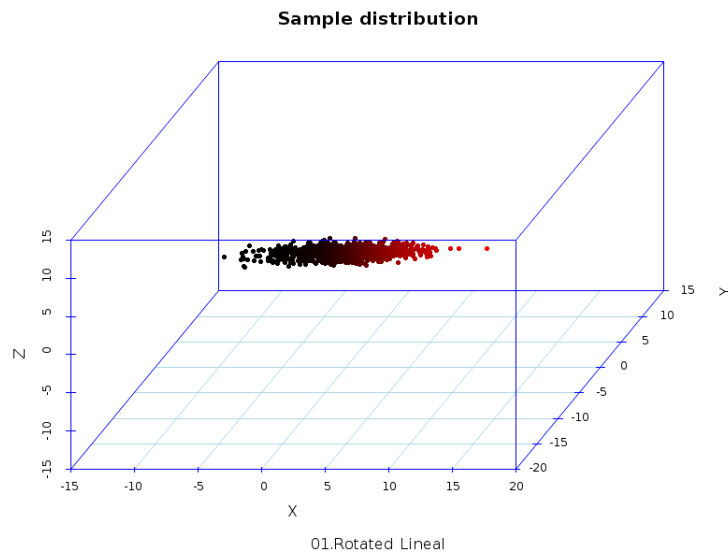
```
Principal Components Attribute Transformer

Correlation matrix
1      0.97  0.97
0.97   1     0.97
0.97  0.97   1

eigenvalue  proportion  cumulative
2.94793     0.98264    0.98264    0.578Z+0.577X+0.577Y

Eigenvectors
V1
0.5774 X
0.5772 Y
0.5776 Z
```

PCA identifies without problem the linear relationship and proposes a single component.  
In case we rotate the sample  $-45^\circ$  around the Z axis:



Then PCA results for a covered variance of 95% still is able to detect the correlation:

```
Principal Components Attribute Transformer

Correlation matrix
1      -1     0.98
-1     1     -0.98
```

```

0.98 -0.98 1

eigenvalue      proportion      cumulative
2.97476         0.99159         0.99159      -0.579x+0.579y-0.575z

Eigenvectors
V1
-0.5786 x
0.5786 y
-0.5749 z

```

This proves that PCA is capable of identifying linear correlation, no matter how the correlation is oriented in the sample space.

## 3 PCA with synthetic sample

### 3.1 Design and implementation

#### 3.1.1 Non-linear relationship

A specific function 'generateSample' has been implemented in order to generate samples for the different cases proposed in the task definition.

The required dimensions:

- $x$  = gaussian distribution  $N(0, \sigma_X)$
- $y = e^x + \text{gaussian noise } N(0, \sigma_{\text{Err}})$
- $z$  = gaussian distribution  $N(0, \sigma_Z)$

Cases proposed by the task definition:

- Small  $\sigma_X$  and  $\sigma_{\text{Err}} \ll \sigma_X$
- Large  $\sigma_X$  and  $\sigma_{\text{Err}} \ll \sigma_X$
- Small  $\sigma_X$  and  $\sigma_{\text{Err}} \sim \sigma_X$

```

1 #####
2 ### Problem functions
3 #####
4
5 #### Generate a 3D sample (3 x N numbers) according to problem specifications
6 generateSample = function(filename, sigmaX = 1, sigmaErr = 1, sigmaZ = 1, n = 1000) {
7
8     x = rnorm(n, mean=0, sd=sigmaX)
9     y = exp(1)**x + rnorm(n, mean=0, sd=sigmaErr)
10    z = rnorm(n, mean=0, sd=sigmaZ)
11    type = rep("A", n)
12
13    df = data.frame(x, y, z, type)
14
15    write.arff(df, file = paste("output/", filename, ".arff", sep=""))
16
17    df # Return data frame
18 }
19
20 #-----
21 #-- Second task
22 #-----
23 d1 = generateSample("02.Small_SigmaX", sigmaX=0.1, sigmaErr=0.1e-5, sigmaZ=1)
24 d2 = generateSample("03.Large_SigmaX", sigmaX=100, sigmaErr=0.1e-5, sigmaZ=1)
25 d3 = generateSample("04.Small_SigmaX_and_similar_SigmaErr", sigmaX=0.1, sigmaErr=0.098, sigmaZ=1)
26
27 generatePlot("02.Small_SigmaX", d1$x, d1$y, d1$z)
28 generatePlot("03.Large_SigmaX", d2$x, d2$y, d2$z)
29 generatePlot("04.Small_SigmaX_and_similar_SigmaErr", d3$x, d3$y, d3$z)
30
31 # Perform PCA
32 varianceCovered = 0.95

```

```

33 .jcall(pca, "V", "generateData", "output", "02.Small_SigmaX", "output", as.double(varianceCovered))
34 .jcall(pca, "V", "generateData", "output", "03.Large_SigmaX", "output", as.double(varianceCovered))
35 .jcall(pca, "V", "generateData", "output", "04.Small_SigmaX_and_similar_SigmaErr", "output", as.double(
    varianceCovered))

```

### 3.1.2 Linear relationship

Aiming to transform the sample so that the correlation among variables is linear instead of exponential, a natural logarithm has been applied to the 'Y' component.

The new dimensions are:

- $x' = x$
- $y' = \ln(y)$
- $z' = z$

Additionally, in order to avoid problems with negative or zero numbers and the logarithm, data has been displaced to positive values (maintaining the distribution) in case of necessity.

```

1  #-----
2  #-- Third task
3  #-----
4
5  # Alternative variables aiming to have more linear relationships
6  type = rep("A", length(d1$y))
7  if (min(d1$y) > 0) {
8    d1_alt = data.frame(x = d1$x, y = log(d1$y), z = d1$z, type)
9  } else {
10   # Natural logarithm should have positive non-zero values to avoid problems
11   # * All data will be displaced by the same quantity, it will not affect PCA
12   d1_alt = data.frame(x = d1$x, y = log(d1$y - min(d1$y) + 0.1e-5), z = d1$z, type)
13 }
14 if (min(d2$y) > 0) {
15   d2_alt = data.frame(x = d2$x, y = log(d2$y), z = d2$z, type)
16 } else {
17   # Natural logarithm should have positive non-zero values to avoid problems
18   # * All data will be displaced by the same quantity, it will not affect PCA
19   d2_alt = data.frame(x = d2$x, y = log(d2$y - min(d2$y) + 0.1e-5), z = d2$z, type)
20 }
21 if (min(d3$y) > 0) {
22   d3_alt = data.frame(x = d3$x, y = log(d3$y), z = d3$z, type)
23 } else {
24   # Natural logarithm should have positive non-zero values to avoid problems
25   # * All data will be displaced by the same quantity, it will not affect PCA
26   d3_alt = data.frame(x = d3$x, y = log(d3$y - min(d3$y) + 0.1e-5), z = d3$z, type)
27 }
28
29 write.arff(d1_alt, file = paste("output/", "02.Small_SigmaX_ALT", ".arff", sep=""))
30 write.arff(d2_alt, file = paste("output/", "03.Large_SigmaX_ALT", ".arff", sep=""))
31 write.arff(d3_alt, file = paste("output/", "04.Small_SigmaX_and_similar_SigmaErr_ALT", ".arff", sep=""))
32
33 generatePlot("02.Small_SigmaX_ALT", d1_alt$x, d1_alt$y, d1_alt$z)
34 generatePlot("03.Large_SigmaX_ALT", d2_alt$x, d2_alt$y, d2_alt$z)
35 generatePlot("04.Small_SigmaX_and_similar_SigmaErr_ALT", d3_alt$x, d3_alt$y, d3_alt$z)
36
37 # Perform PCA
38 varianceCovered = 0.95
39 .jcall(pca, "V", "generateData", "output", "02.Small_SigmaX_ALT", "output", as.double(varianceCovered))
40 .jcall(pca, "V", "generateData", "output", "03.Large_SigmaX_ALT", "output", as.double(varianceCovered))
41 .jcall(pca, "V", "generateData", "output", "04.Small_SigmaX_and_similar_SigmaErr_ALT", "output", as.double(
    varianceCovered))

```

## 3.2 Results

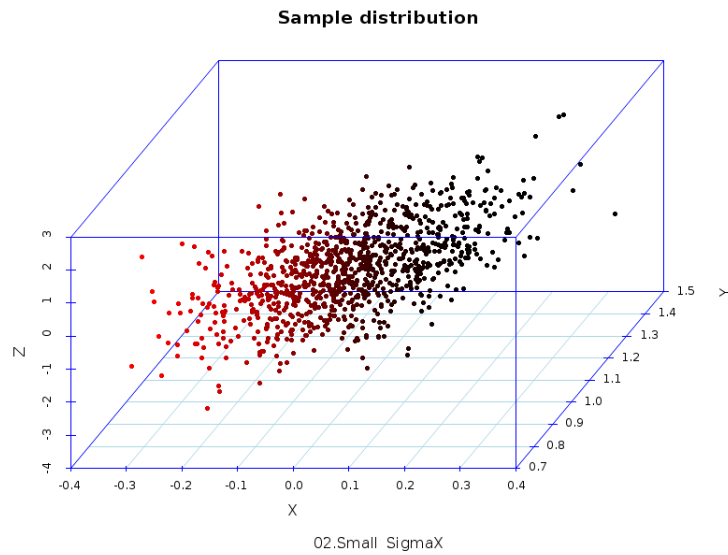
### 3.2.1 Case 1: Small sigmaX and sigmaErr << sigmaX

The proposed values for this case are:

- $x = \text{gaussian distribution } N(0, \text{sigmaX} = 0.1)$

- $y = e^x + \text{gaussian noise } N(0, \text{sigmaErr} = 0.1e-5)$
- $z = \text{gaussian distribution } N(0, \text{sigmaZ} = 1)$

**Non-linear relationship** The distribution of the generated sample:



The PCA results for a covered variance of 95% is able to detect correlation and reduce the number of dimension to two principal components:

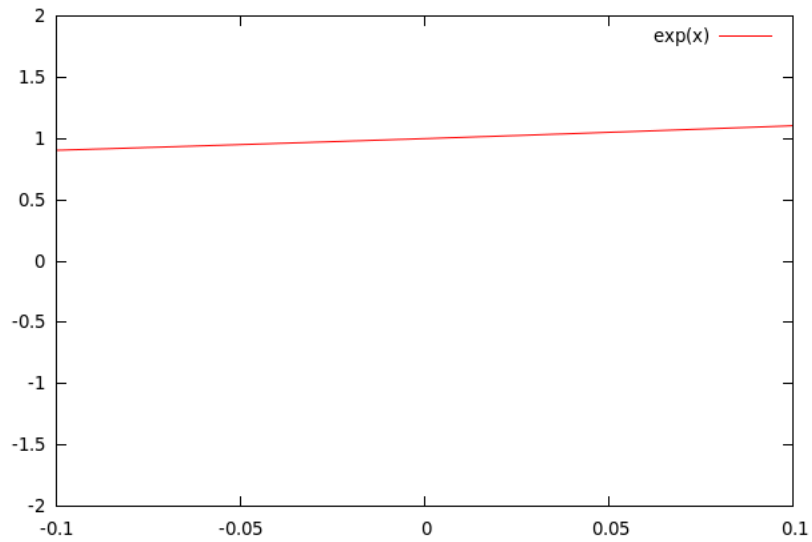
```
Principal Components Attribute Transformer

Correlation matrix
1      1      0.01
1      1      0.01
0.01   0.01   1

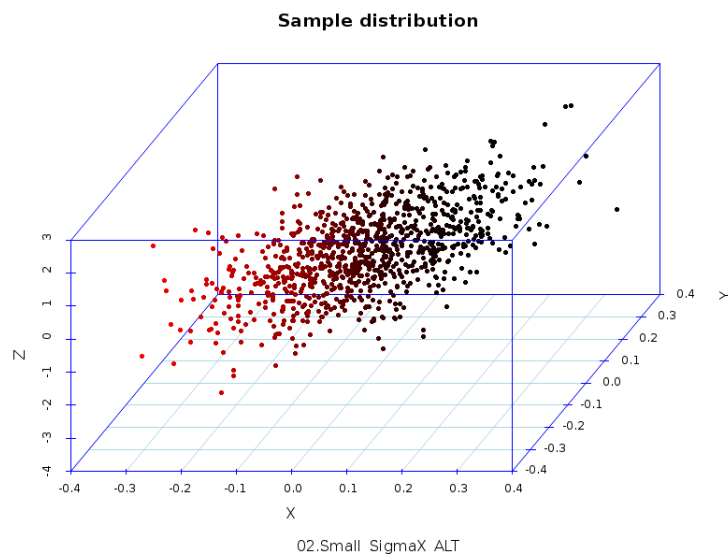
eigenvalue    proportion    cumulative
1.9978        0.66593       0.66593
0.99971       0.33324       0.99917
0.707x+0.707y+0.017z
-1z+0.013y+0.011x

Eigenvectors
V1      V2
0.707   0.0112 x
0.707   0.0129 y
0.017   -0.9999 z
```

Although the correlation between 'x' and 'y' is exponential (PCA only works for linear correlations), PCA can identify it because a small sigmaX (0.1) and sigmaErr (0.000001) has been established for 'x' and the effect of the exponential in the 'y' dimension is relatively small.



**Linear relationship** The same case but with a natural logarithm applied to the Y component presents the following distribution:



The PCA results for a covered variance of 95% works as well as with the previous case, identifying two principal components:

```
Principal Components Attribute Transformer

Correlation matrix
1      1      0.01
1      1      0.01
0.01   0.01   1

eigenvalue   proportion   cumulative
2.00033      0.66678      0.66678      0.707x+0.707y+0.018z
0.99967      0.33322      1            -1z+0.013y+0.013x

Eigenvectors
V1      V2
0.707   0.0128 x
0.707   0.0128 y
0.0182  -0.9998 z
```

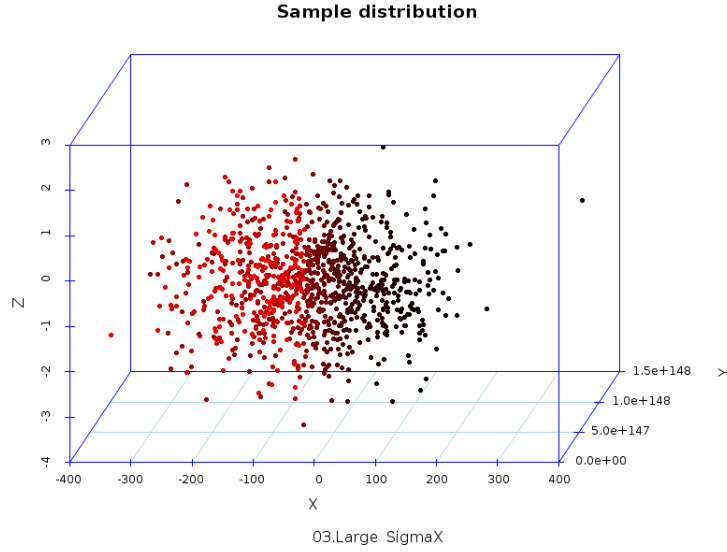


### 3.2.2 Case 2: Large sigmaX and sigmaErr << sigmaX

The proposed values for this case are:

- $x$  = gaussian distribution  $N(0, \text{sigmaX} = 100)$
- $y = e^x + \text{gaussian noise } N(0, \text{sigmaErr} = 0.1e-5)$
- $z$  = gaussian distribution  $N(0, \text{sigmaZ} = 1)$

**Non-linear relationship** The distribution of the generated sample:



The PCA results for a covered variance of 95% presents three principal components, so the algorithm is not able to reduce the number of dimensions:

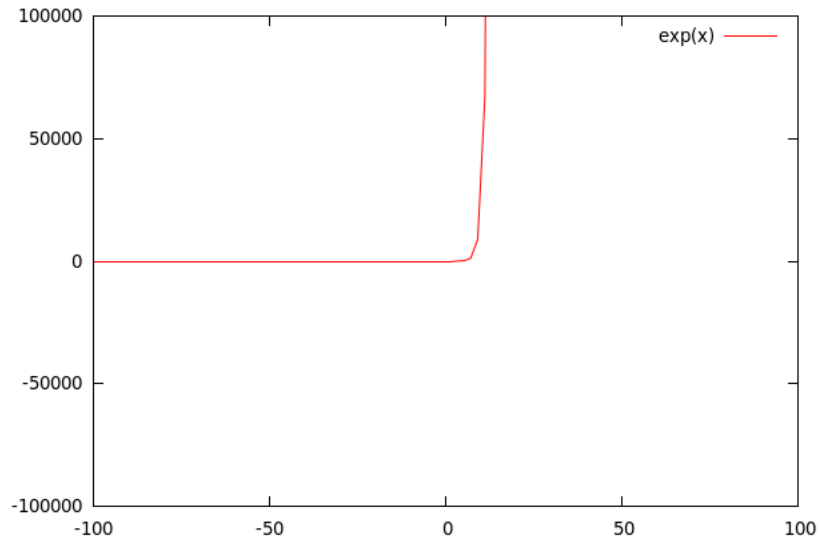
```
Principal Components Attribute Transformer

Correlation matrix
1      0.1      0.02
0.1     1      -0.05
0.02  -0.05     1

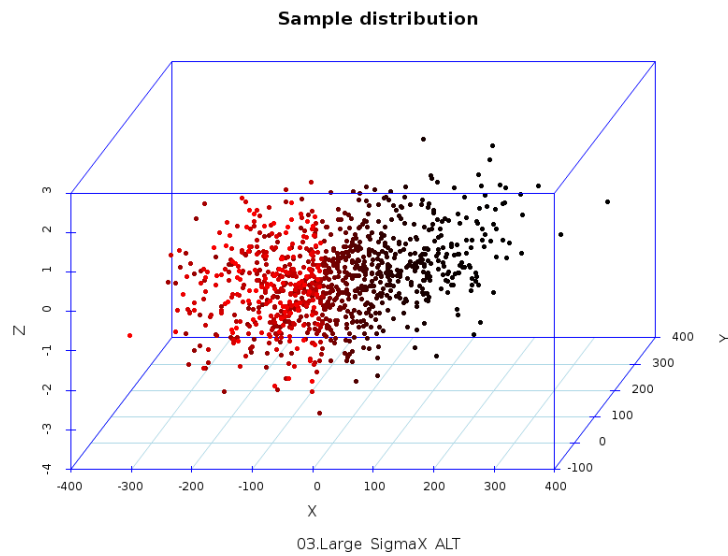
eigenvalue    proportion    cumulative
1.10612       0.36871      0.36871
1.01816       0.33939      0.70809
0.87572       0.29191      1
-0.729y-0.646x+0.227z
0.888z+0.444x-0.117y
0.675y-0.621x+0.399z

Eigenvectors
V1      V2      V3
-0.6463  0.4437 -0.6208 x
-0.7286 -0.1172  0.6748 y
 0.2267  0.8885  0.399  z
```

PCA does not work well if correlations are not linear, therefore the dimensionality of the samples has not been reduced. In this case, a large sigmaX (100) has been used, which magnifies the effect of the exponential correlation between  $y$  and  $x$ .



**Linear relationship** The same case but with a natural logarithm applied to the Y component presents the following distribution:



With the sample conversion applying a natural logarithm to the 'y' dimension, correlation is now linear instead of exponential and PCA is able to detect it and present two principal components:

```
Principal Components Attribute Transformer

Correlation matrix
1      0.86  0.02
0.86   1    0.01
0.02   0.01  1

eigenvalue    proportion    cumulative
1.86178      0.62059      0.62059
0.99967      0.33322      0.95382
0.707x+0.707y+0.024z
-1z+0.026y+0.007x

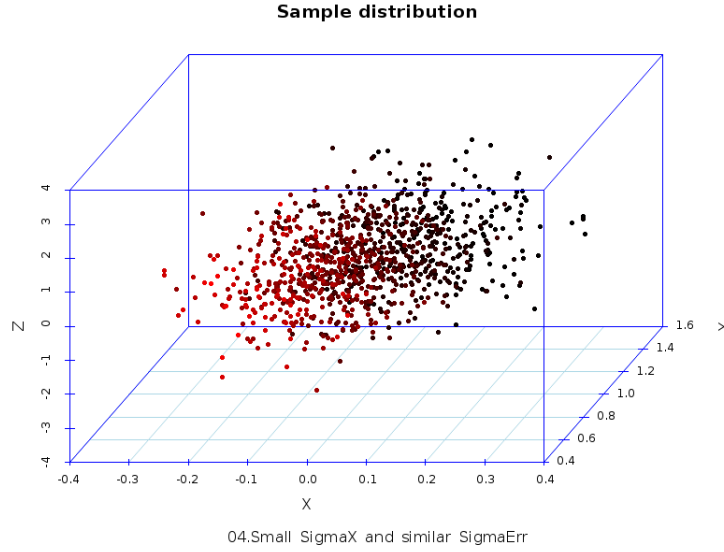
Eigenvectors
V1      V2
0.707   0.0074 x
0.7068  0.026 y
0.0236  -0.9996 z
```

### 3.2.3 Case 3: Small sigmaX and sigmaErr ~ sigmaX

The proposed values for this case are:

- $x$  = gaussian distribution  $N(0, \text{sigmaX} = 0.1)$
- $y = e^x + \text{gaussian noise } N(0, \text{sigmaErr} = 0.098)$
- $z$  = gaussian distribution  $N(0, \text{sigmaZ} = 1)$

**Non-linear relationship** The distribution of the generated sample:



The PCA results for a covered variance of 95% presents three principal components, so the algorithm is not able to reduce the number of dimensions:

```
Principal Components Attribute Transformer

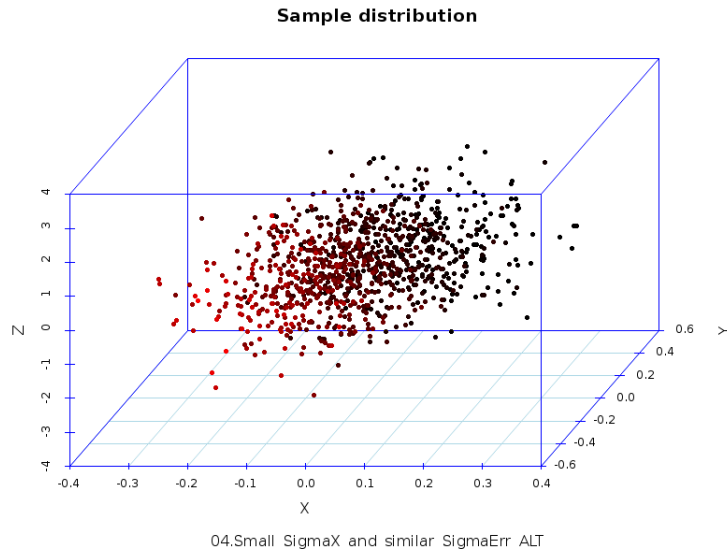
Correlation matrix
1      0.72  0.03
0.72   1    0
0.03   0    1

eigenvalue    proportion    cumulative
1.71904      0.57301      0.57301
0.99986      0.33329      0.9063
0.2811       0.0937      1
0.707x+0.707y+0.027z
-0.999z+0.035y+0.003x
-0.707x+0.707y+0.023z

Eigenvectors
V1      V2      V3
0.7071  0.0028 -0.7071 x
0.7066  0.0348  0.7067 y
0.0266 -0.9994  0.0227 z
```

In this case, a small sigmaX (0.1) has been used like in case 1, therefore the exponential correlation effect is not very strong. What is limiting PCA to detect correlation and reduce dimensionality is that the applied sigmaError (0.098) is very similar to sigmaX (0.1), this produces a lot of noise in the 'y' dimension and breaks the obvious correlation with 'x'.

**Linear relationship** The same case but with a natural logarithm applied to the Y component presents the following distribution:



The PCA results for a covered variance of 95% presents three principal components such as the previous case. The algorithm is not able to reduce the number of dimensions:

```
Principal Components Attribute Transformer
Correlation matrix
1      0.72  0.03
0.72   1     0.01
0.03   0.01  1

eigenvalue    proportion    cumulative
1.71615      0.57205      0.57205      0.707x+0.707y+0.032z
0.99947      0.33316      0.9052      -0.999z+0.035y+0.011x
0.28439      0.0948       1           -0.707x+0.707y+0.017z

Eigenvectors
V1      V2      V3
0.7069  0.0107  -0.7072 x
0.7065  0.035   0.7068 y
0.0323  -0.9993   0.0172 z
```

As explained in the previous case, although the exponential correlation has been transformed into a linear one, there is too much noise in the 'y' dimension as for PCA to detect strong correlations and reduce dimensionality.

## 4 PCA with stock market sample

### 4.1 Description of the sample

#### 4.1.1 Context

In order to test the Principal Component Analysis with real data, the automatic process includes the analysis of the evolution of a group of stock values of the Spanish market over a given period.

The selected stock values can be divided in two groups depending on the company activities:

- Financial institutions
  - BBVA
  - Banco Santander
  - Banesto
  - Bankinter
  - Banco Popular
- Telecommunications
  - Telefonica

#### 4.1.2 Objects

Every object correspond to a given day where a group of stock values has been bought and sold in the Spanish stock market. Each day is determined by the behaviour of the stock values in the following hour range:

- Auction Opening hours: from 8:30 to 9:00
- Open market: from 9:00 to 17:30
- Auction closed: from 17:30 to 17:35

For each stock, the percentatge of change in its value will be considered:

$$Change = \frac{Close\ Value - Open\ Value}{Open\ Value}$$

where *Close Value* is the stock value when the market closes and *Open Value* is the stock value when the market opens.

Days are classified as positive or negative depending on the behaviour of the Spanish index IBEX35, where the top 35 companies of the Spanish market are represented.

$$Positive\ Day \iff \frac{Close\ IBEX\ Value - Open\ IBEX\ Value}{Open\ IBEX\ Value} \geq 0$$

#### 4.1.3 Adquisition process

Data has been adquired from Yahoo Finance services, where anybody can freely download the historical stock values of several markets. The selected period of days start on January, 1st 2003 and ends on November, 5th 2010. Those days for which there were not information for all stock values has been discarded and ignored.

It is worth to mention that effects of events such as dividends or stock splits has not been considered and, for our purpose, they are negligible.

#### 4.1.4 General characteristics

Samples are stored in the 'input' directory, there are individual files in CSV format (comma separated, every line represent a day) for the selected stock values. They contain the following values for each day:

**Date** Year-Month-Day (i.e. 2010-12-03)

**Open** Value of the stock at the market opening hour.

**High** Maximum stock value for that day

**Low** Minimum stock value for that day

**Close** Value of the stock at the market closing hour

**Volume** Number of stock values exchanged (sold/bought).

**Adj Close** Close adjustments (typically equal to the close value because no adjustments are needed)

The units for Open, High, Low, Close and Adj Close are Euros, except for the case of IBEX35 index where it is measured in Points.

Date, Open and Close are the important values for the present analysis, the rest of them can be ignored.

Stock	Number of days
BBVA	2.052
Santander	2.052
Banesto	2.052
Bankinter	2.052
Popular	2.062
Telefónica	2.033
IBEX35	2.009

There are days where not all stocks have values recorded (this is possible due to temporary suspensions ordered by the Spanish market regulator CNMV), those objects are ignored and the analysis will be done over the 2.002 days for which we have values for all stocks.

Those objects has been automatically procesed, selected and recorded into the 'output' directory:

- '05.Stock.Change.only.Banks.arff'
- '05.Stock.Change.Banks.and.Telefonica.arff'

They contain the percentatge of value change over a day and the day classification (positive or not) as explained in previous sections.

## 4.2 Statistical analysis

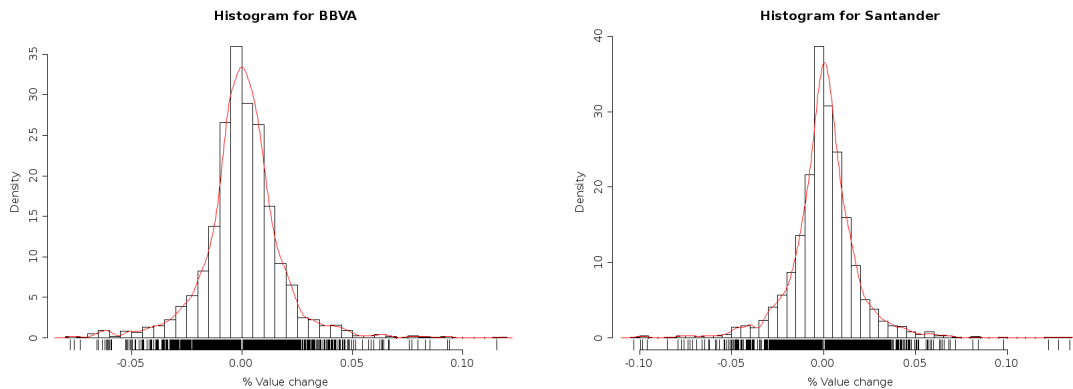
### 4.2.1 Key indicators

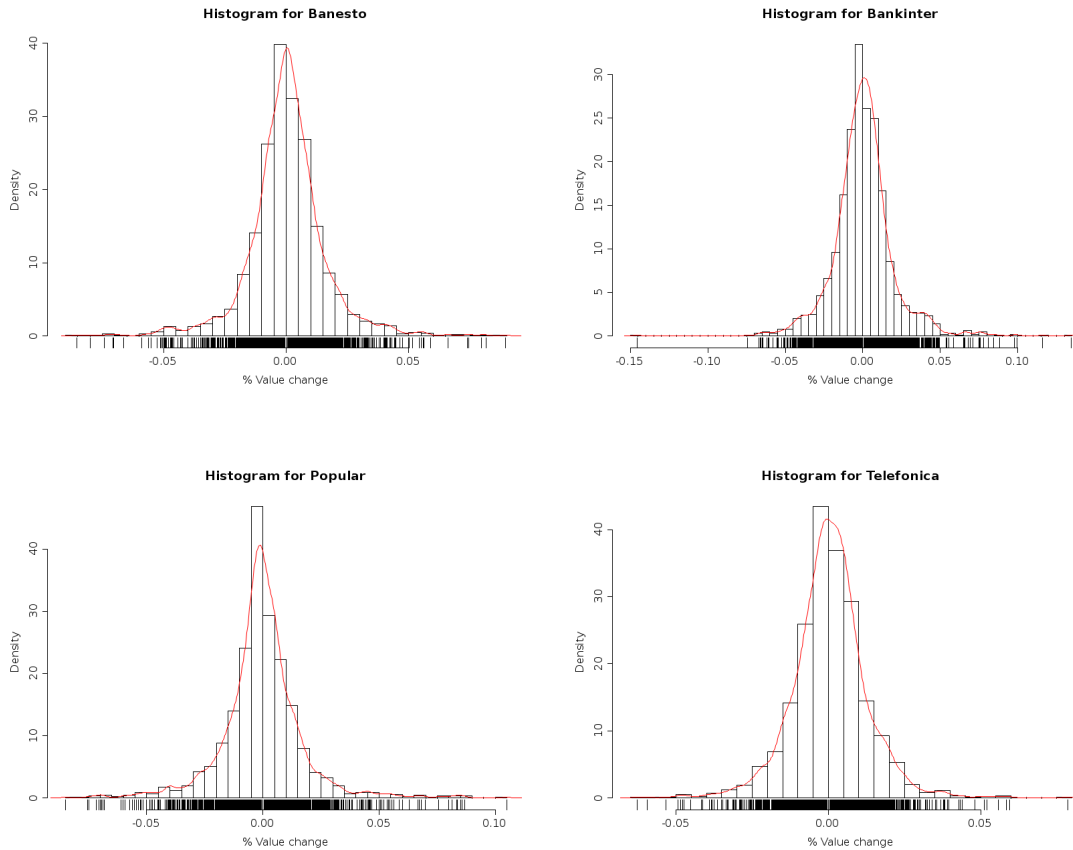
A brief statistical analysis has been perform on the 2.002 days of data (% of stock value change) that will be used in the PCA:

Stock	Min	Max	Mean	Median	Variance	Std Deviation
BBVA	-0.077707	0.1153846	0.0002528539	0	0.0003108627	0.0176313
Santander	-0.1030790	0.1339130	0.0001880890	0	0.0003523623	0.01877132
Banesto	-0.08527919	0.08934708	0.0004157036	0	0.0002505433	0.01582856
Bankinter	-0.1455882	0.1344538	0.0002737273	0	0.0003790459	0.0194691
Popular	-0.08496732	0.1048951	-0.0004328711	-0.0007299367	0.0003017615	0.01737128
Telefónica	-0.06265417	0.07857143	0.0004895762	0.0005037078	0.0001528554	0.01236347

In respect to the day classification, there are 1.063 positive days over a total of 2.002.

### 4.2.2 Histograms





### 4.3 Preliminary evaluation

It is expected that stocks from the financial sector should behave in a correlated way in the long term, while the telecommunication company should not have a clear correlation with the other values.

On the other hand, all the selected stock are part of the computation of the IBEX35 index (which is used to determine if it was a positive day in the market). Therefore, it is expected that positive days most of the values will have a positive change and viceversa.

### 4.4 Design and implementation

Data obtained from Yahoo has been loaded and treated in the R script in order to achive the conditions explained in the previous sections and prepare them for Principal Components Analysis:

```

1  # Read input data
2  ibex = read.table("input/IBEX.csv", header = TRUE, sep = ",", row.names = NULL)
3  bbva = read.table("input/BBVA.csv", header = TRUE, sep = ",", row.names = NULL)
4  santander = read.table("input/Santander.csv", header = TRUE, sep = ",", row.names = NULL)
5  banesto = read.table("input/Banesto.csv", header = TRUE, sep = ",", row.names = NULL)
6  bankinter = read.table("input/Bankinter.csv", header = TRUE, sep = ",", row.names = NULL)
7  popular = read.table("input/Popular.csv", header = TRUE, sep = ",", row.names = NULL)
8  telefonica = read.table("input/Telefonica.csv", header = TRUE, sep = ",", row.names = NULL)
9
10 ## Just interested in 3 columns
11 ibex = data.frame(ibex$Date, ibex$Open, ibex$Close)
12 bbva = data.frame(bbva$Date, bbva$Open, bbva$Close)
13 santander = data.frame(santander$Date, santander$Open, santander$Close)
14 banesto = data.frame(banesto$Date, banesto$Open, banesto$Close)
15 bankinter = data.frame(bankinter$Date, bankinter$Open, bankinter$Close)
16 popular = data.frame(popular$Date, popular$Open, popular$Close)
17 telefonica = data.frame(telefonica$Date, telefonica$Open, telefonica$Close)
18
19 ## Join everything excluding those days for which we do not have values for all stocks

```

```

20 stock = merge(bbva, santander, by.x=c("bbva.Date"), by.y=c("santander.Date"))
21 stock = merge(stock, banesto, by.x=c("bbva.Date"), by.y=c("banesto.Date"))
22 stock = merge(stock, bankinter, by.x=c("bbva.Date"), by.y=c("bankinter.Date"))
23 stock = merge(stock, popular, by.x=c("bbva.Date"), by.y=c("popular.Date"))
24 stock = merge(stock, telefonica, by.x=c("bbva.Date"), by.y=c("telefonica.Date"))
25 stock = merge(stock, ibex, by.x=c("bbva.Date"), by.y=c("ibex.Date"))
26
27
28 #-----
29 #-- Select only % change for banks
30 #-----
31 onlyBanksStockChange = data.frame(bbva.Change = (stock$bbva.Close - stock$bbva.Open)/stock$bbva.Open, santander.
Change = (stock$santander.Close - stock$santander.Open)/stock$santander.Open, banesto.Change = (stock$
banesto.Close - stock$banesto.Open)/stock$banesto.Open, bankinter.Change = (stock$bankinter.Close - stock$
bankinter.Open)/stock$bankinter.Open, popular.Change = (stock$popular.Close - stock$popular.Open)/stock$
popular.Open, positiveDay = ((stock$ibex.Close - stock$ibex.Open) >= 0))
32
33 ## Save data in weka format
34 write.arff(onlyBanksStockChange, file = "output/05.Stock.Change.only.Banks.arff")
35
36 # Perform PCA
37 varianceCovered = 0.70
38 .jcall(pca, "V", "generateData", "output", "05.Stock.Change.only.Banks", "output", as.double(varianceCovered))
39
40
41 #-----
42 #-- Select % change for banks and Telefonica
43 #-----
44 banksAndTelefonicaStockChange = data.frame(bbva.Change = (stock$bbva.Close - stock$bbva.Open)/stock$bbva.Open,
santander.Change = (stock$santander.Close - stock$santander.Open)/stock$santander.Open, banesto.Change = (
stock$banesto.Close - stock$banesto.Open)/stock$banesto.Open, bankinter.Change = (stock$bankinter.Close -
stock$bankinter.Open)/stock$bankinter.Open, popular.Change = (stock$popular.Close - stock$popular.Open)/
stock$popular.Open, telefonica.Change = (stock$telefonica.Close - stock$telefonica.Open)/stock$telefonica.
Open, positiveDay = ((stock$ibex.Close - stock$ibex.Open) >= 0))
45
46 ## Save data in weka format
47 write.arff(banksAndTelefonicaStockChange, file = "output/05.Stock.Change.Banks.and.Telefonica.arff")
48
49 # Perform PCA
50 varianceCovered = 0.70
51 .jcall(pca, "V", "generateData", "output", "05.Stock.Change.Banks.and.Telefonica", "output", as.double(
varianceCovered))

```

## 4.5 Results

### 4.5.1 Financial institutions

The PCA results for a covered variance of 70% presents one principal component, so the algorithm has been able to reduce the number of dimensions from five to one:

```

Principal Components Attribute Transformer

Correlation matrix
1      0.87  0.6  0.56  0.73
0.87   1      0.61  0.54  0.72
0.6    0.61  1      0.55  0.63
0.56   0.54  0.55  1      0.56
0.73   0.72  0.63  0.56  1

eigenvalue      proportion      cumulative      0.478bbva.Change+0.475santander.Change+0.46 popular.Change+0.422banesto.Change+0.395bankinter.Change
3.55921         0.71184         0.71184

Eigenvectors
V1
0.4781 bbva.Change
0.475 santander.Change
0.4224 banesto.Change
0.395 bankinter.Change
0.4597 popular.Change

```

As suspected, financial institutions are correlated in the stock market and PCA has been able to detect it presenting just one Principal Component.

### 4.5.2 Financial institutions and a telecommunication company

The PCA results for a covered variance of 70% presents two principal components, so the algorithm has been able to reduce the number of dimensions from six to two:



```

Principal Components Attribute Transformer

Correlation matrix
1      0.87  0.6   0.56  0.73  0.64
0.87   1      0.61  0.54  0.72  0.65
0.6    0.61  1      0.55  0.63  0.45
0.56   0.54  0.55  1      0.56  0.46
0.73   0.72  0.63  0.56  1      0.53
0.64   0.65  0.45  0.46  0.53  1

eigenvalue  proportion  cumulative  0.449bbva.Change+0.447santander.Change+0.423popular.Change+0.384banesto.Change+0.374telefonica.Change...
4.05382     0.67564     0.67564     -0.567bankinter.Change+0.557telefonica.Change-0.482banesto.Change+0.261santander.Change+0.24 bbva.Change...
0.63533     0.10589     0.78152

Eigenvectors
V1      V2
0.4494  0.2396 bbva.Change
0.4475  0.2614 santander.Change
0.3843 -0.482  banesto.Change
0.3633 -0.5672 bankinter.Change
0.4226 -0.0982 popular.Change
0.3735  0.5573 telefonica.Change

```

When financial institutions are mixed with a telecommunication company, correlations are not so clear and PCA is not able to do such a drastic dimension reduction as seen in the previous case.

## References

- [1] Luri, X. (2010). *Multivariate analysis. PCA*. Universitat de Barcelona.
- [2] Weka API 3.6. Retrieved December 5, 2010, from the World Wide Web: <http://weka.sourceforge.net/doc.stable/>
- [3] *Use Weka in your Java code*. Retrieved December 5, 2010, from the World Wide Web: <http://weka.wikispaces.com/Use+WEKA+in+your+Java+code>
- [4] *Data mining algorithms in R*. Retrieved December 5, 2010, from the World Wide Web: [http://en.wikibooks.org/wiki/Data\\_Mining\\_Algorithms\\_In\\_R](http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R)
- [5] *Weka interface for R. Reference manual*. Retrieved December 5, 2010, from the World Wide Web: <http://cran.r-project.org/web/packages/RWeka/index.html>
- [6] *Java interface for R*. Retrieved December 5, 2010, from the World Wide Web: <http://rosuda.org/rJava/>
- [7] *Bitmap creation for R. Reference manual*. Retrieved December 5, 2010, from the World Wide Web: <http://cran.r-project.org/web/packages/GDD/index.html>
- [8] *3D scatter plot for R. Reference manual*. Retrieved December 5, 2010, from the World Wide Web: <http://cran.r-project.org/web/packages/scatterplot3d/index.html>