# PROJECT IN UNSUPERVISED MACHINE LEARNING

## ASSIGNMENT-2

**AJAY ANAND KUMAR – 013711933**

**PEDRO ALONSOD – 013753179**

**Instructions:**

1. Download the Assignment-2.zip file.
2. Extract the files viz. a) Exercise2.r b) visual.R and put the respective files in the data folder that contains "mixed_images.txt"
3. Platform used is **R-cran** in Windows or Unix.

**Commands:**

- In R command prompt:
  >source ("Exercise2.r")
  >source("visual.R")

**Exercise 1.1**

**Command:** >source("Exercise2.r")
>ex1.1( )

Given matrix $A_1$ and A2, random vector **n** ε $R^2$ which follows Gaussian distribution with mean = 0 and covariance matrix equal to Identity matrix i.e with variance =1. Scatter plot for $y_1 = A_1n$ and $y_2 = A_2n$ for 5000 data points.

The density plot of the transformed data shows that it tends to follow Gaussian distribution having mean = 0.02 for y1 and mean = 0.009 for y2. The covariance matrix for y1 and y2 are:

[1] "****Covariance Matrices*****"
Covaraince matrix for y1
[,1] [,2]
[1,] 2.983993 3.454349
[2,] 3.454349 6.958704

Covariance matrix for y2
[,1] [,2]
[1,] 2.978355 3.452479
[2,] 3.452479 6.967988

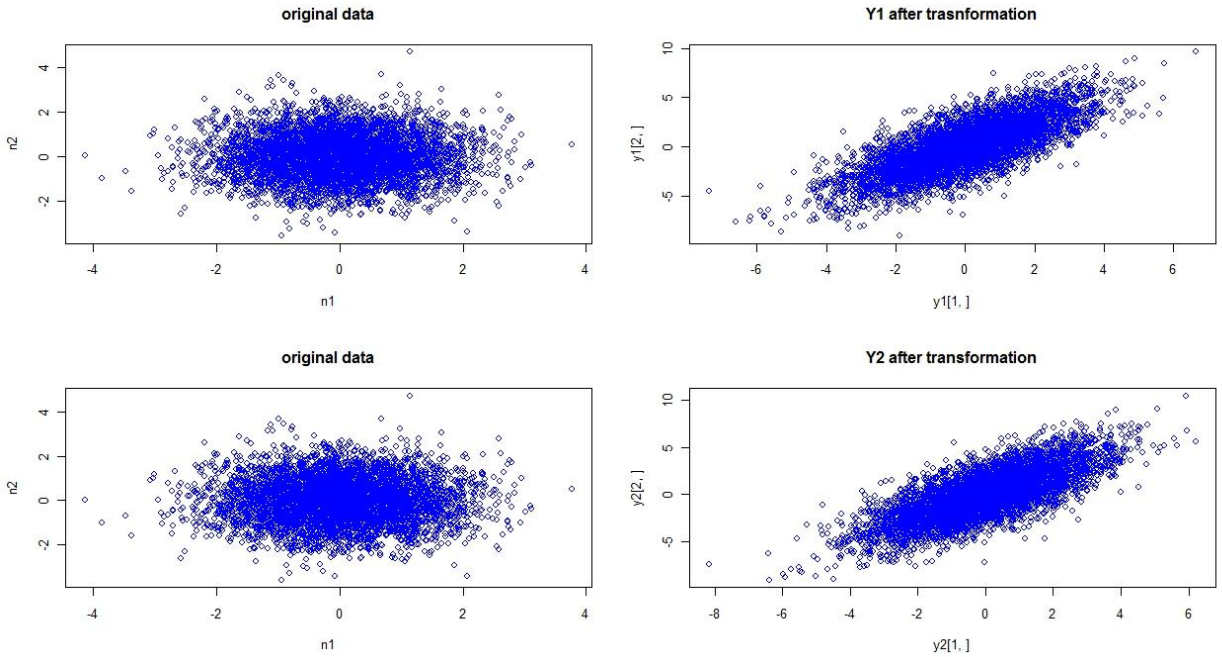


**Figure1.1 :** Scatter plot of Gaussian data n after being transformed to y1 and y2.

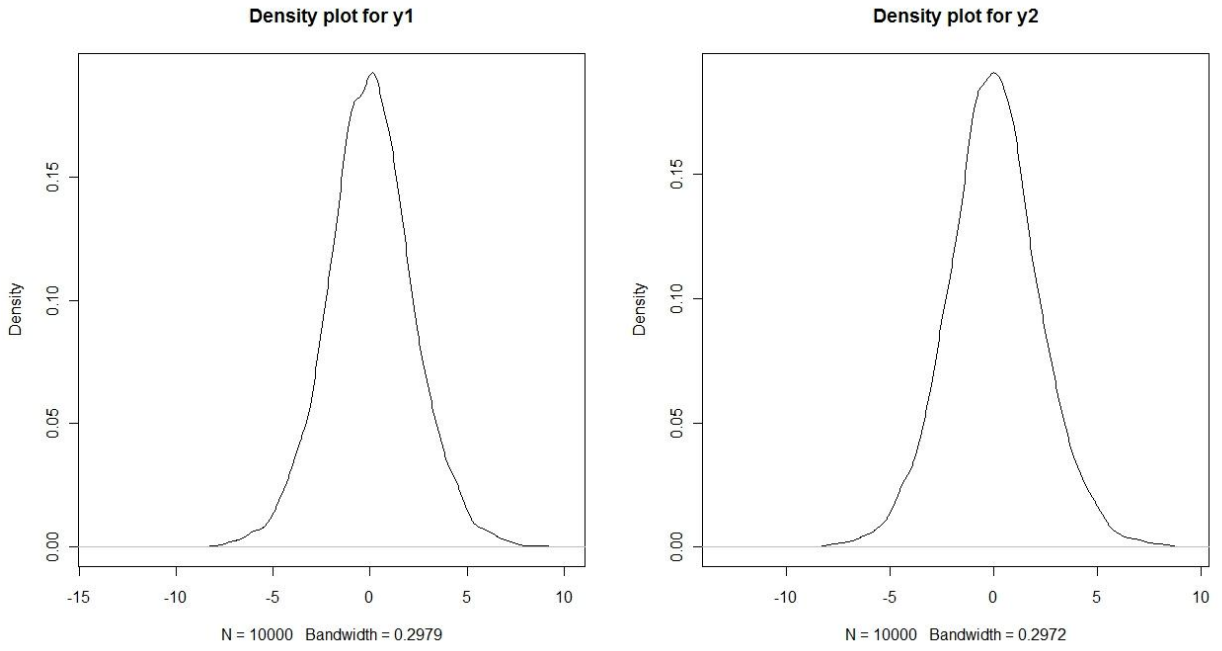


**Figure 1.2:** Density plot of y1 and y2. The transformed data follows more likely Gaussian distribution having mean =0.02 for y1 and mean = 0.009 for y2.

**Exercise1.2**

**Command: >ex1.2 ( )**

Whitening of data: The data matrices **x1, x2, y1 and y2** were whitened using following formula:

$$\tilde{\mathbf{x}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^{T}\mathbf{x}$$

Where E and D are corresponding eigen vectors (orthogonal matrix) and diagonal matrix of eigen values obtained after eigen-value decomposition (EVD) of the covariance matrix $\mathbf{E\{xx^T\}}$. After whitening the data x1, x2, y1 and y2 becomes correlated i.e **cov(t(x1) = I**

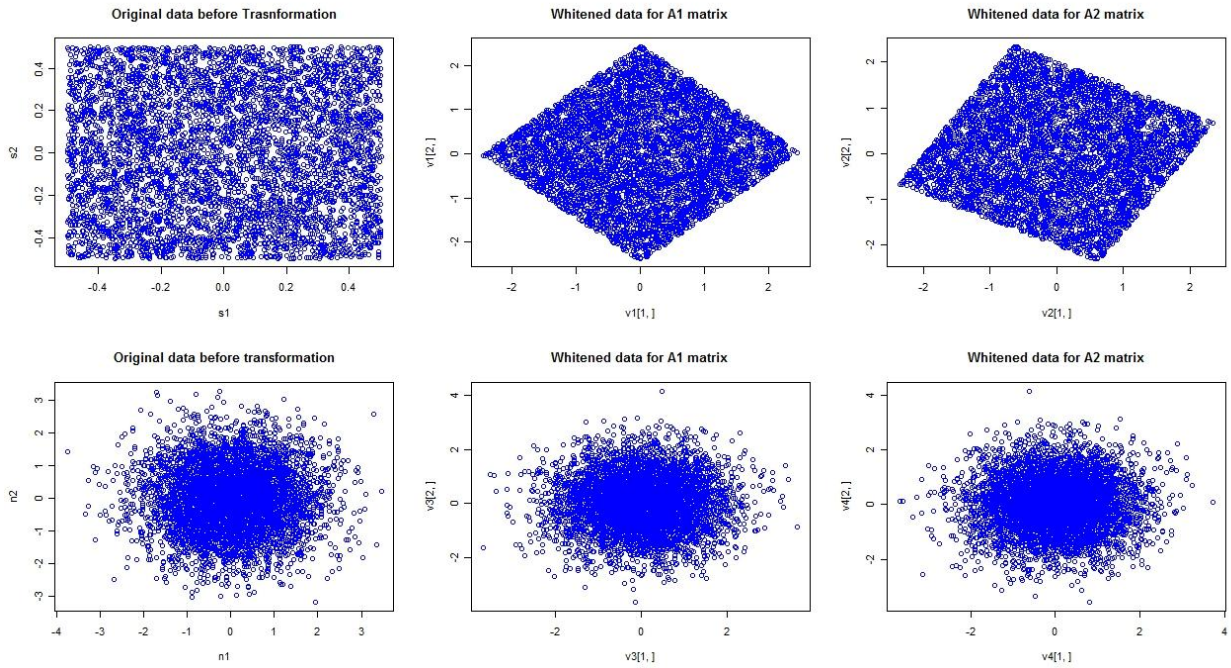


**Figure 1.3:** This represents the original data s and n before transformation and corresponding whitened data. For data **s** that has uniform distribution the whitened data gets rotated but for Gaussian distributed data **'n'** the data is not rotated i.e not transformed.

Whitening transforms the mixing matrix $A_i$ to a new matrix À. The new matrix is obtained as:

$$\tilde{\mathbf{x}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^{T}\mathbf{A}\mathbf{s} = \tilde{\mathbf{A}}\mathbf{s}$$

Thus for the uniformly distributed data **'s'** the data got rotated as seen in Figure1.3 (top). This clearly signifies that the mixing matrix A is altered to new matrix **À.** But for Gaussian data the **'n'** the data remains unchanged and is same as previous. With whitening the number of parameters to be estimated for mixing matrix A got reduced since $\grave{A}\,\grave{A}^T = I$.

**Exercise 1.3**

**Command: >ex1.3 ( )**

For the whitened datasets viz. x1, x2 ,y1 and y2 the data was projected to unit vectors alpha having domain [0 $\pi$]. Then kurtosis was calculated using formula:

$$\text{kurt}(z_i) = E\{z_i^4\} - 3E\{z_i^2\}^2 \quad \text{where } z_i \varepsilon \{x1,x2,x3,x4\}$$

Since the whitened data has been projected to unit vectors thus each of $z_i$ are function of alpha. Now in order to find a vector **w**(alpha) such that the linear combination or projection of $\mathbf{w}^T\mathbf{z}$ is has maximum non-gaussianity. As such plotting the computed kurtosis produces the following curve as shown in Figure 1.3. The plot shows that for uniform data (first row x1, x2) the kurtosis is always negative and this is minimized by value 2.35 and 0.5 radians for the two plots respectively. The second row plot is for Gaussian data (y1 and y2) whose kurtosis is positive as seuch the curve is maximized for values 0.35 and 0.1 radians respectively. The maximum alpha values which maximize the absolute value of kurtosis for this curve in the direction of independent components are:

2.35, 0.5, 0.35, 0.1 respectively (row wise).

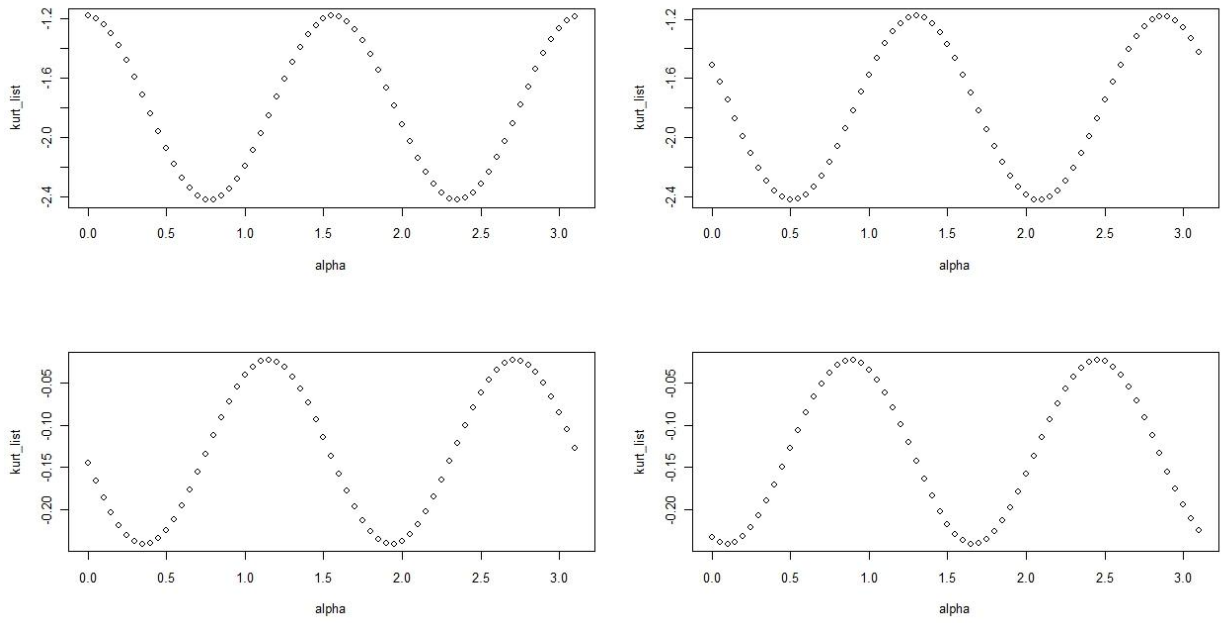


**Figure 1.4:** For alpha in radians in domain [0 $\pi$] kurtosis computed as function of alpha for four data sets (whitened) x1, x2, y1 and y2 respectively (row wise).

**Exercise 1.4**

**Commands: >ex1.3 ( )**

Form previous exercise the corresponding alpha was found for which $\mathbf{w}^T\mathbf{z}$ maximizes the non-gaussianity of the data. For each type of data uniform and Gaussian the maximum value of alpha was obtained. Using this value of alpha **w** vector is calculated and then it is normalized using the formula:

$$\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|$$

The normalized **w** are in fact the rows of inverse of the mixing matrix because of the property of whitened data A is orthogonal as such $\mathbf{A^{-1} = A^T}$. Thus, **w** gives the estimate of the A matrix. In order to get good estimate of these matrices one needs to run several iterations of projections and orthonormalizations. The corresponding estimates of $A_1$ and $A_2$ matrix are:

For Uniform data:

Alpha = 2.35

$A_1$ =

| | [,1] | [,2] |
|---|---|---|
| [1,] | -0.7027131 | -0.7114734 |
| [2,] | 0.7114734 | -0.7027131 |

Alpha = 2.1

$A_2$ =

| | [,1] | [,2] |
|---|---|---|
| [1,] | -0.5048461 | -0.8632094 |
| [2,] | 0.8632094 | -0.5048461 |

For Gaussian data:

Alpha = 0.75

$A_1$ =

| | [,1] | [,2] |
|---|---|---|
| [1,] | 0.7316889 | -0.6816388 |
| [2,] | 0.6816388 | 0.7316889 |

Alpha = 0.5

$A_2$ =

| | [,1] | [,2] |
|---|---|---|
| [1,] | 0.8775826 | -0.4794255 |
| [2,] | 0.4794255 | 0.8775826 |

The corresponding plot of kurtosis as a function of alpha (above values) is given as:

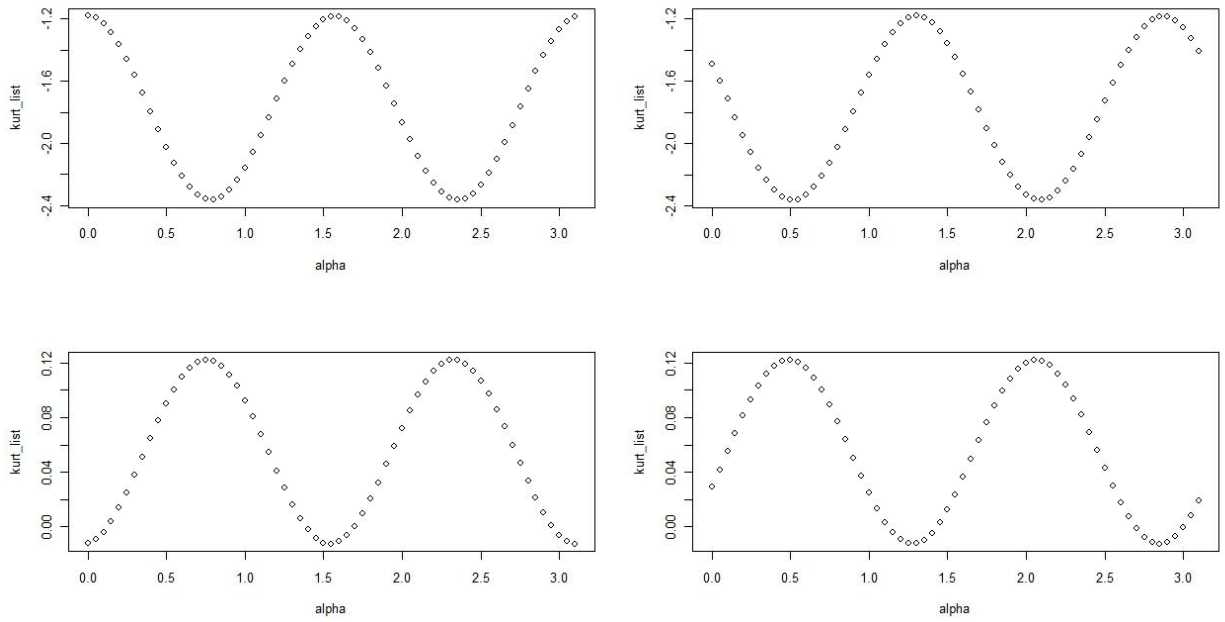


**Figure 1.5:** Plot of kurtosis as function of alpha.

### Exercise 1.5

As seen from the previous exercise in order to get good estimates of mixing matrices one need to run several iterations of projections and ortho-normalizations of **w** matrix. Central Limit theorem is used to evaluate w such that it approximates the rows of inverse of A matrix. It also states that sum of two independent random variable tends to be more Gaussian than original variable. Thus while estimating **w** with uniform distribution data it tends to Gaussian data and hence the estimation of A matrix is possible. Whereas, with Gaussian data it remains same even after several iterations. This can be found from Figure 1.3 that when Gaussian variable is multiplied with A matrices the data remains the same. Hence, logically it can be proven that with non Gaussian data the A matrices can be estimated.

**Exercise 2.1**

**Command: >ex2.1 ( )**

Given **s = (s1, s2…s32)** a random vector at consists of 32 independent random variables, all of which follow Laplacian distribution of mean = 0 and variance one in domain range of [-0.5 0.5).

The $y_m$ was calculated as cumulative sum of m = 1, 2, 4, 8, 16, 32 Laplacian variables and was normalized to unit variance. Similarily, $y_m$ for Gaussian variable was also calculated for these m random variables. Form Central limit theorem, if random variables are statistically independent then cumulative sum of these random variables tends to be more Gaussian. Hence, the final cumulative sum will have Gaussian distribution.

Plot of 6 Log distribution of Gaussian and Laplacian variables can be found in **Figure 2.1 a** and **Figure 2.1 b** respectively. On analyzing the plot one can see that these plots are more or less similar. Thus the $y_m$ satisfies the Central Limit Theorem and it forms the basis for estimating the independent components of input variable **s.**

Histogram plot for Log distribution of Gaussian data

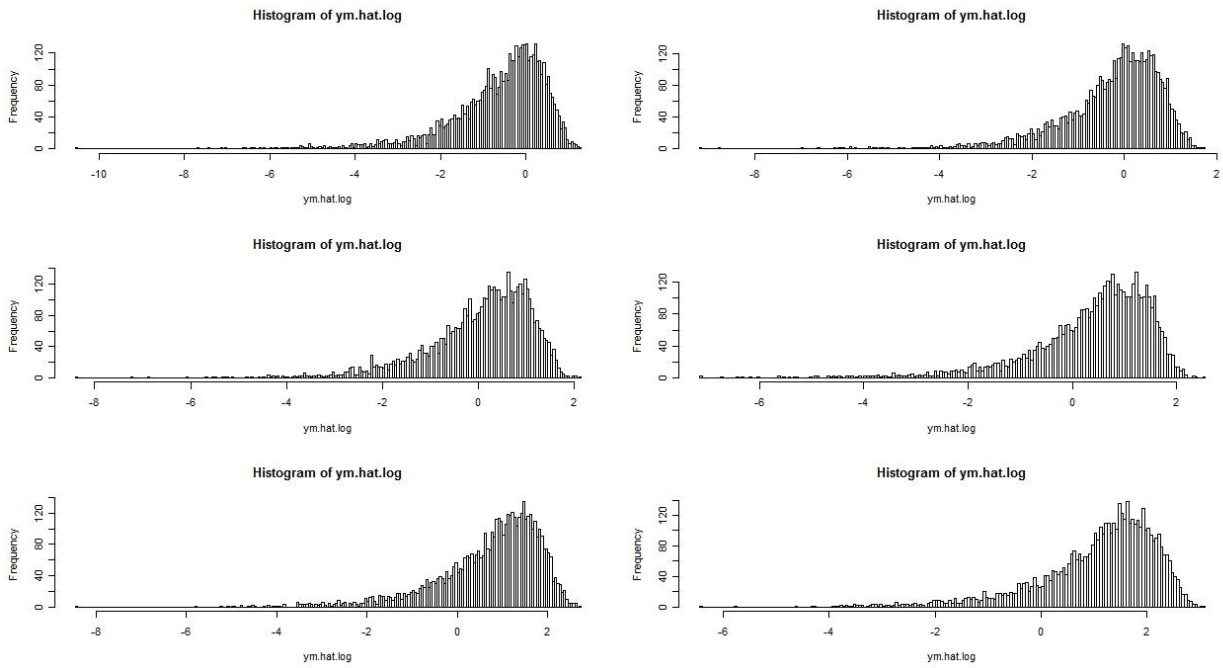


**Figure 2.1 a:** Histogram plot of Log distribution of Gaussian data

Histogram plot for Log distribution of Laplacian data:

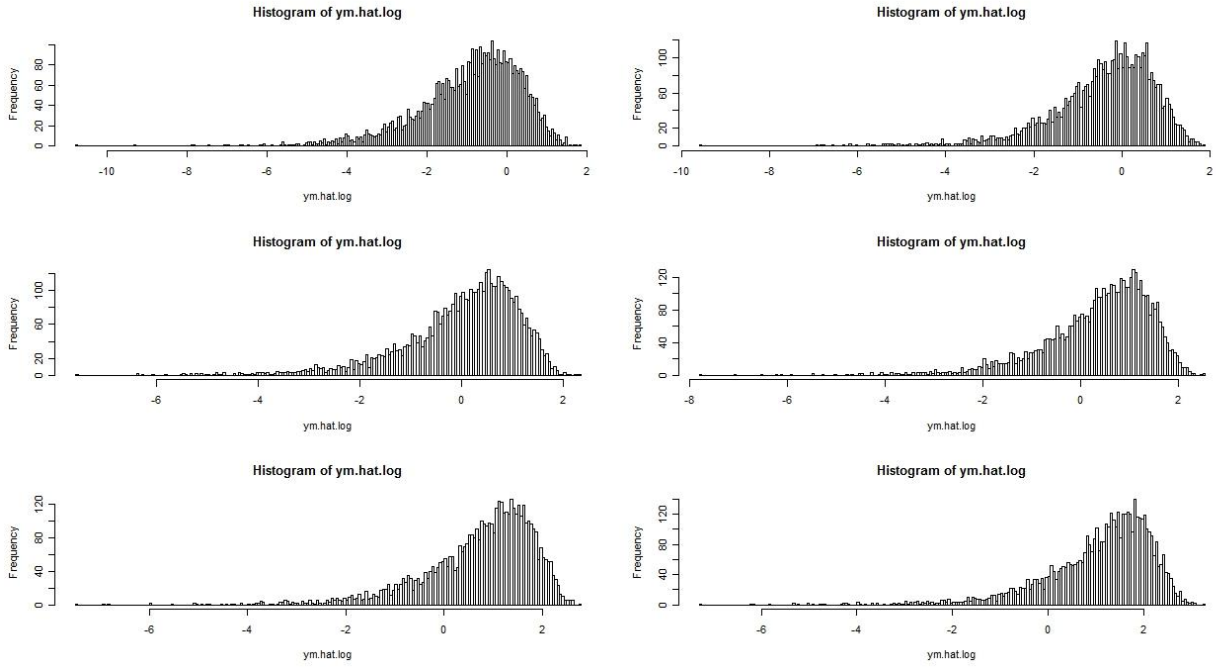


**Figure 2.1 b:** Histogram plot of Log distribution of Laplacian data

**Exercise 2.2**

**Command: >ex2.2 ( )**

For m = 1, 2, 4, 8, 16, 32 the kurtosis of $y_m$ was calculated as:

$$\text{kurt}(z_i) = E\{z_i^4\} - 3E\{z_i^2\}^2 \quad i \varepsilon \{1, 2, 4, 8, 16, 32\}$$

As discussed in previous exercise 2.1 by virtue of Central limit theorem, the sum of non Gaussian random variables tends to become more Gaussian. Since kurtosis of Gaussian random variable is equal to zero, as such kurtosis of cumulative sum of Gaussian random variable should also be zero.

By plotting the curve between Kurtosis and m we found that with increasing m the kurtosis tends to zero in Figure 2.2

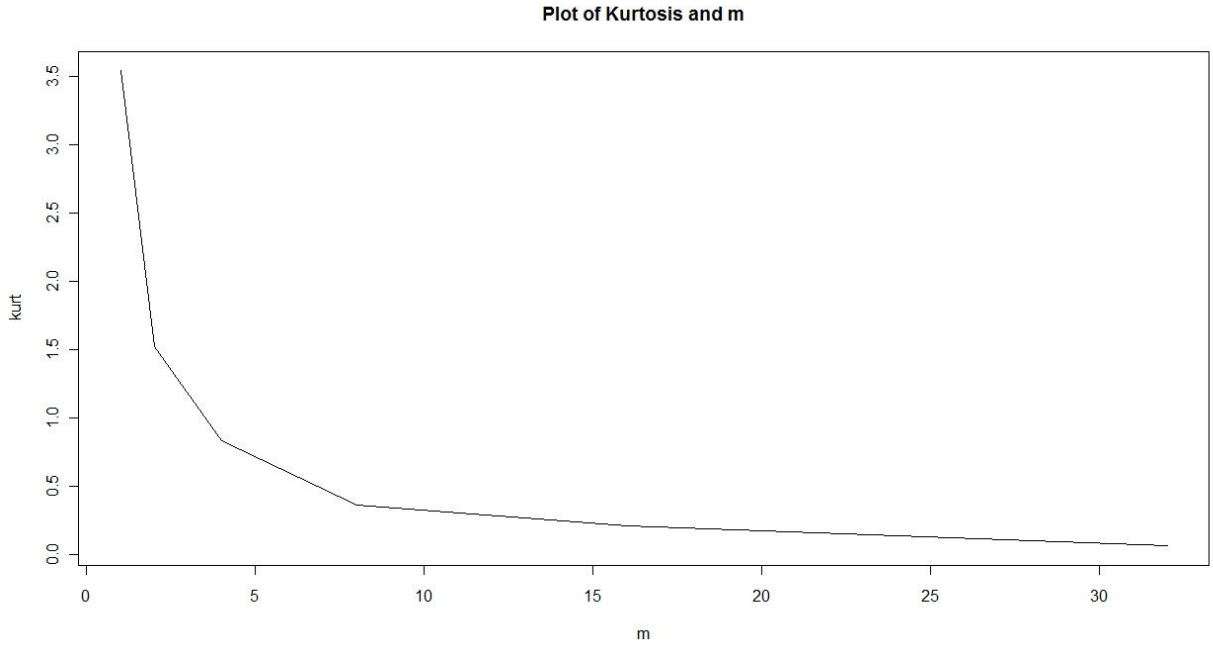


**Figure2.2:** Plot of Kurtosis and m. The kurtosis value decreases with increasing value of m. For larger value of m it tends to zero. This satisfies the Central Limit theorem.

**Exercise2.4**

**Commands: >ex2.3 ( )**

Implementation of ICA algorithm: The alogithm steps are:

1. Choose the number of independent components to estimate. Here m = 2
2. Initialize the $\mathbf{w_i}$ **,** i = 1, 2
3. Do the iteration of a one-unit algorithm on every $\mathbf{w_i}$ parallel.
4. Do a symmetric orthogonalization of the matrix $W = (w_1 , w_2 )^T$
5. Check for convergence. If converge then break else go to step 2 and repeat the process until the objective function in step 3 converges.

Testing: The test data was taken from Exercise 1 with A1 and A2 as mixing matrices. s1 and s2 were chosen randomly having uniform distribution. $W = (w_1 , w_2 )^T$ matrix was initialized having normal distribution with zero mean and unit variance. The algorithm ran for 6 iterations and the final W matrix was obtained as:

W = [,1] [,2]

[1,] -0.7162472 0.6978466

[2,] 0.6978466 0.7162472

The corresponding objective function values for the 6 iterations are:

-1.563408 -2.222108 -2.372243 -2.373285 -2.373287 -2.373287

Thus the ICA algorithm implementation converged at $6^{th}$ iteration and W matrix was obtained.

**Exercise 2.4**

Given y = (y1, y2 ….. y32) and y is cumulative sum of $s_i$ for i in 1 to 32. Thus there is linear relation between **y** and **s**. Mathematically, this can be expressed as:

Y = As

Or [y1 y2 y3 … y32] = [a1 a2 a3 … a32] [s1 s2 s3 …. s32]

Here **s** matrix is 32 × 5000 dimension and **A** matrix is 32 × 32 dimension. The rows of A matrix are just

A = [1 0 0 0 … 32 times
1 1 0 0 0 … 32 times
1 1 1 0 0 …. 32 times
……
1 1 1 1 1 1 1 1 .. 32 times]

**Exercise 2.5**

**Command: >ex2.5 ()**

Using previous implementation of ICA algorithm to estimate **À** matrix, the average squared error obtained:

The algorithm ran for 11 iterations and converged. The resultant error is:

Error: 0.5543 for 10000 samples

and Error: 0.5414 for 20000 samples.

**Exercise 3.1**

**Commands: >ex3.1 ( )**

Visualizing each mixture as 300 × 300 pixel size using visual.R gives the image:

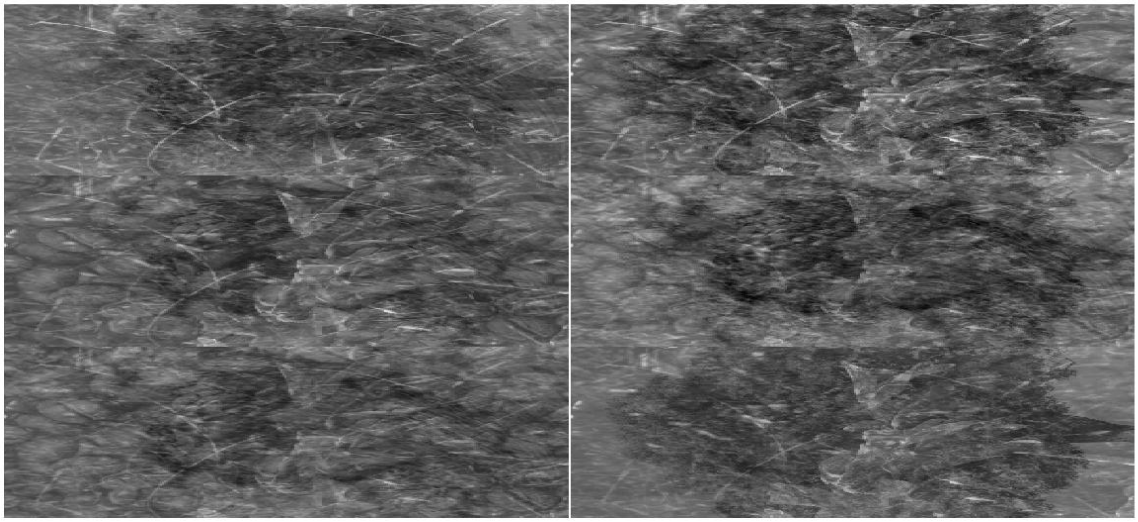

**Figure 3.1:** The image represents mixture of images divided in 300 × 300 pixels in 6 by 6 matrix. Each row has two images.

Originally the input file is read in to a 6 × 90000 dimensional matrix. For ICA model

$$\mathbf{x} = \mathbf{A}\mathbf{s} \text{ where } x \in R^6 \text{ and } s \in R^{6.}$$

For this ICA model it is necessary to have statistical independence assumption for the input matrix **s**. The main reason behind is that when applying maximum likelihood estimation, the original ICA model can be expressed as likelihood model. The density

$$p_x = |\det \mathbf{B}| * p_s(s) = |\det \mathbf{B}| * \mathbf{\Pi_i}\, p_i(s_i)$$

where $\mathbf{B} = \mathbf{A}^{-1}$ and $p_i$ denote the densities of the independent components. With statistical independence condition the density $p_i$ is just the product of individual component densities.

Each row of the input matrix **s** represents mixture of images which was formed by stacking the columns of original un-mixed images with each other to form 90000 row vectors. The separation

between the images is sparsely visible. As such the task of ICA is to separate those images and retrieve the constituent image components in the mixture.

**Exercise 3.2**

**Command: >ex3.2 ( )**

Whitening of input data **x:** The data was initially checked for zero mean and unit variance. The matrix **s** has zero mean and unit variance. Thus whitening can directly be applied to it. The original input data and whitened data looks like:

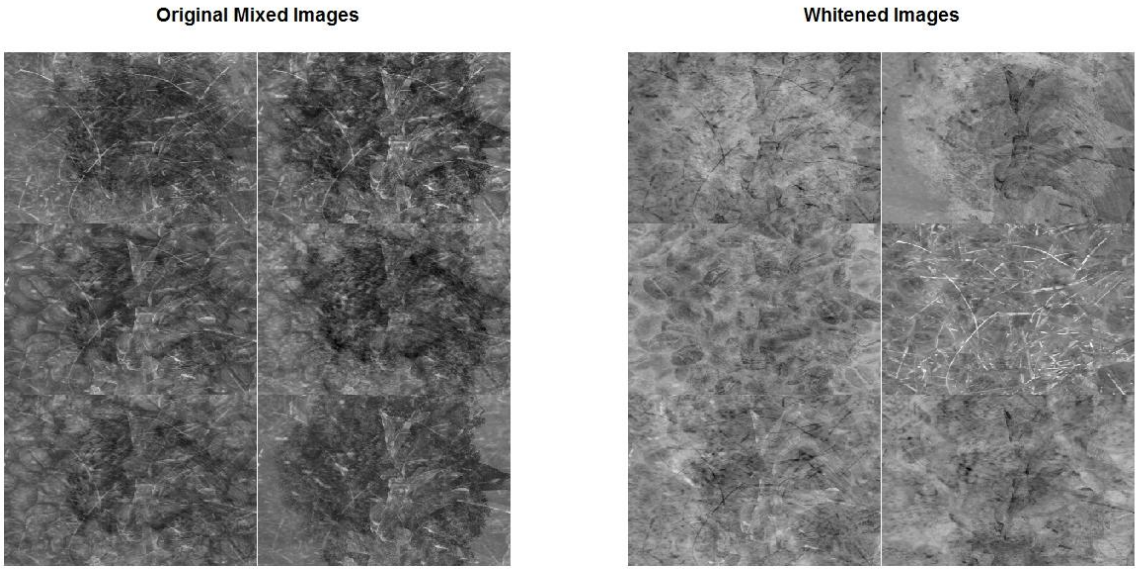


**Figure 3.2:** Represents the Original data (left) and whitened data (right).

From Figure 3.2 we can clearly see that after whitening the mixed images have been separated partially. But still some patches are left and images are not 100% unmixed. Visually, this proves the claim that whitening is just half of ICA. It can be guess from the whitened data that the mixture is composed of 6 original image data.

**Exercise 3.3**

**Commands: >ex3.3 ( )**

The algorithm was implemented. The test data was used from Exercise 1 with A1 and A2 matrices. Input data vector was chosen to be **s = s1 and s2** having uniform distribution. 5000 samples were chosen. The mean was subtracted form the data. Then the data was transformed to **x1** and **x2** vector using A1 and A2 matrices respectively.

1. Each of **x1** and **x2** vector where whitened. The corresponding whitened matrix is represented as **z** of 2×5000 dimension.
2. Matrix **B** of dimension 2×2 was initialized randomly. The step parameter mew_g = 0.8 and mew = 0.2 were chosen.
3. Then **y = B*z** was computed
4. $\gamma_i$ was updated as given in the exercise 3.3 (d)
5. The objective function F was calculated as per rule in exercise 3.3 (e)
6. Then **B** was updated based on the condition given in exercise 3.3 (f)
7. Then **B** was ortho-normalized.
8. Finally, the algorithm was checked for convergence based on the value of objective function F. If it converges then values of F and matrix B from that iteration is returned. If it not converges then step 3 to 8 is repeated until it converges.

With this input data, the algorithm converged after 10 iterations. The final updated **B** matrix is:

Printing the update B matrix

```
          [,1]       [,2]
[1,] -0.4302506  -0.9027095
[2,] -0.9027095   0.4302506
```

The objective function values are:

```
[1] -0.04910144  -0.05892173  -0.06088578  -0.06127860  -0.06135716  -0.06137287
[7] -0.06137601  -0.06137664  -0.06137677  -0.06137679
```

**Exercise 3.4**

**Command: >ex3.4 ( )**

After implementing the modified ICA algorithm in previous exercise, it was tested with input mixture of images. The algorithm returns the de-mixing matrix **B.** This matrix is then multiplied to original data matrix **s** to obtain the independent un-mixed images.

$$y = \mathbf{Bs}$$

The output vector **y** is multiplied with -1 to increase the quality and intensity of the images.

The final un-mixed images look like:

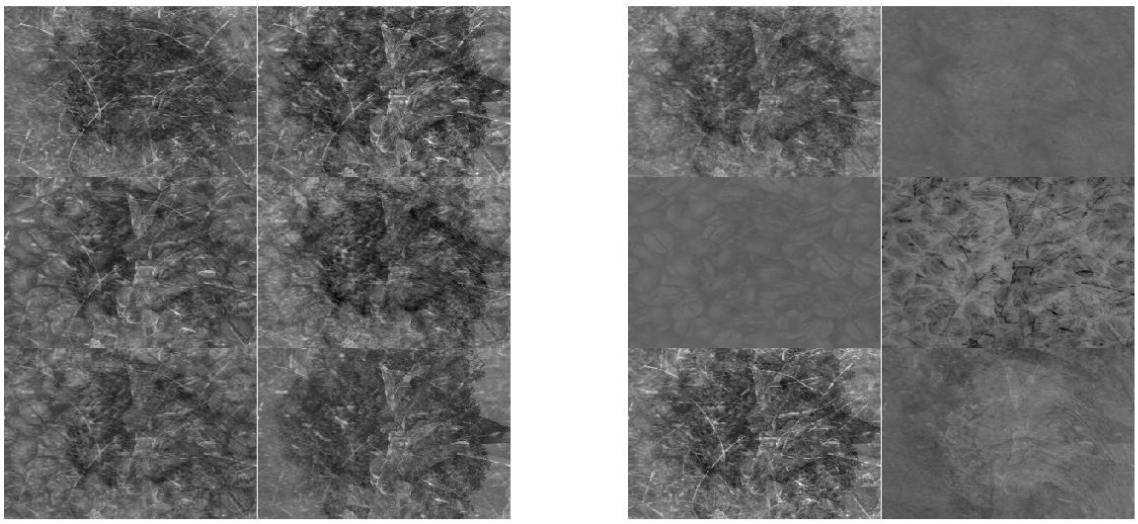

**Figure 3.3:** This represents the images before and after ICA. The left side image is mixture of images. The right side is the un-mixed images.

After running the natural gradient algorithm, it returns the update gamma values and the corresponding estimated **B** matrix. This is the demixing matrix and it is multiplied to the orginal mixture of images matrix **X.** The new **Y** matrix is the un-mixed images obtained as:

**Y = BX**

One can see from Figure 3.3 that left hand side image is mixture of images and separation between the images is sparsely visible. The right side image obtained after applying ICA algorithm is separated in to 6 constituent images. The individual six components of images along with the intensities measure is clearly visible.

The updated gamma values obtained after applying ICA algorithm to the images are:

```
          [,1]
[1,]  0.030056511
[2,] -0.004803479
[3,]  0.031779061
[4,] -0.013116800
[5,]  0.037301494
[6,] -0.012200589
```

From these values of gamma one can infer that the signs changes alternatively. This change in sign corresponds to the images with low intensities. The gamma values indicate the intensity and position of the corresponding images. Thus low intensity images are 2, 4 and 6 (row wise).