hw6_nonames2

```
library(audio)

# install.packages('VGAM')
library(VGAM)

# install.packages('R.matlab')
library(R.matlab)
library(ggplot2)
```

6.1.a Extend your code from the previous problem sheet to get an ICA-learning scheme based on the natural gradient with a learning rate ε that decays slowly to 0 (e.g. εt+1 = λεt with λ ≈ 1, λ < 1). Note that depending on λ you have to iterate over the (shuffled) data more than once for proper convergence.

Initializsation of last Excersise

```
# a)
s1 = read.table("hw5/sound1.dat", header = FALSE)
s2 = read.table("hw5/sound2.dat", header = FALSE)

S = t(as.matrix(data.frame(s1, s2)))

# b)
set.seed(1234)
A = matrix(runif(4, 0, 1), nrow = 2)

X = A %*% S


# e)
X[1, ] = X[1, ] - mean(X[1, ])
X[2, ] = X[2, ] - mean(X[2, ])
```

Extention by lambda decreasing to zero and Plot of S_head (assumption lambda ist not the eigenvalue)

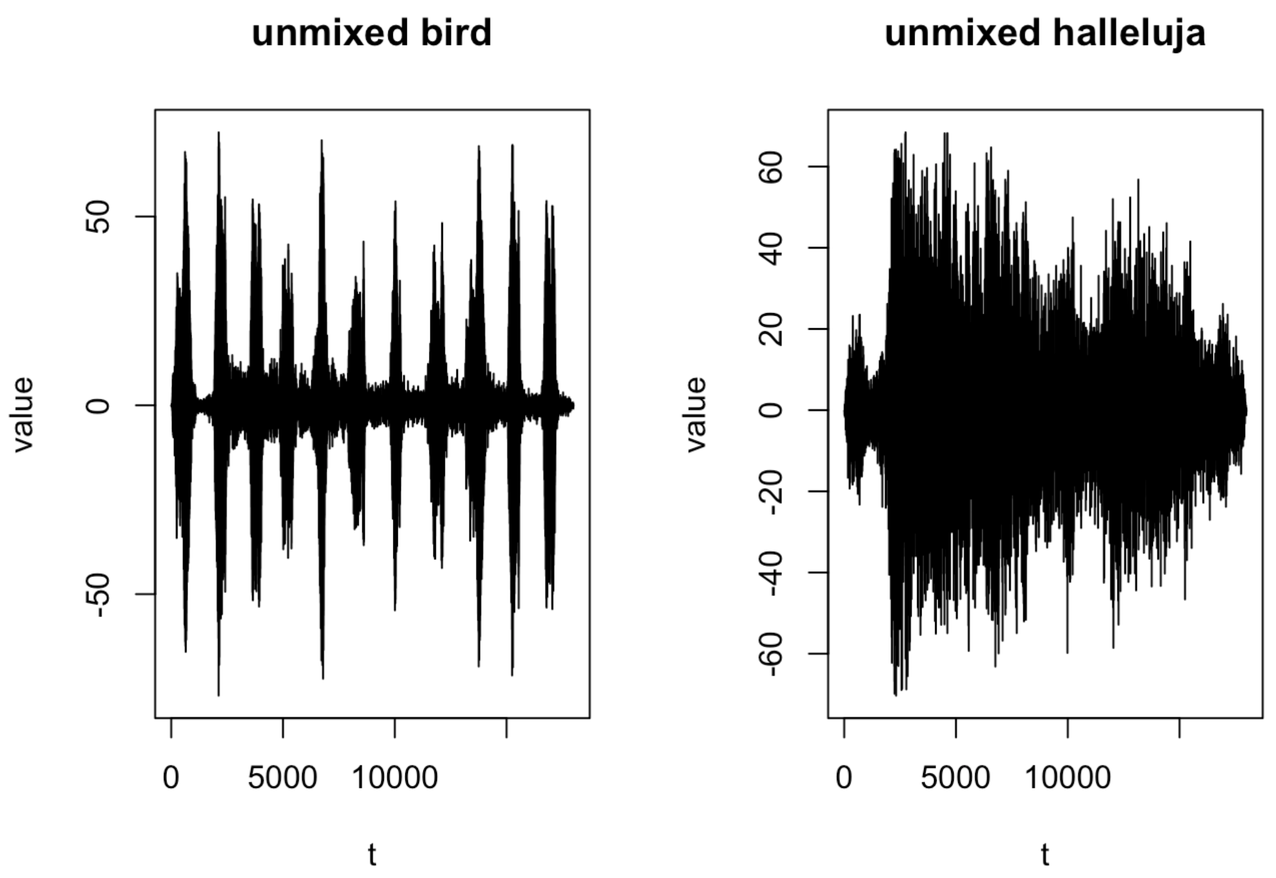
```r
# logistic function
f = function(x) {
    return(1/(1 + exp(-x)))
}


# natural Gradient
t = 1
eta_t = 0.7
alpha = 1
lambda = 0.9102  #0.9112
set.seed(9991)
W2 = matrix(runif(4, 0, 1), ncol = 2)


unmixing_natural = function(W, X, n_steps = 18000) {

    for (t in 1:n_steps) {
        eta_t = eta_t * lambda

        if (alpha > n_steps - 12) {
            print(eta_t)
        }

        x = X[, alpha]

        c <- do.call(cbind, replicate(nrow(W), x, simplify = FALSE))

        f_wx = 1 - 2 * f(W %*% c)

        wx = W %*% c

        k_delta = diag(nrow(W))


        W_delta = eta_t * ((k_delta + f_wx %*% wx) %*% W)

        W = W + W_delta

        alpha = alpha + 1
        if (alpha == n_steps) {
            alpha = 1
        }
    }
    return(W)
}


W_natural = unmixing_natural(W2, X)
```

```
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
```

```
S_hat_natural = W_natural %*% X

cor(t(S_hat_natural), t(S))
```

```
##               V1      V1.1
## [1,]  0.9807428 0.1965396
## [2,] -0.1915089 0.9812487
```

```
par(mfrow = c(1, 2))
plot(1:18000, S_hat_natural[1, ], type = "l", main = "unmixed bird",
    xlab = "t", ylab = "value")
plot(1:18000, S_hat_natural[2, ], type = "l", main = "unmixed halleluja",
    xlab = "t", ylab = "value")
```

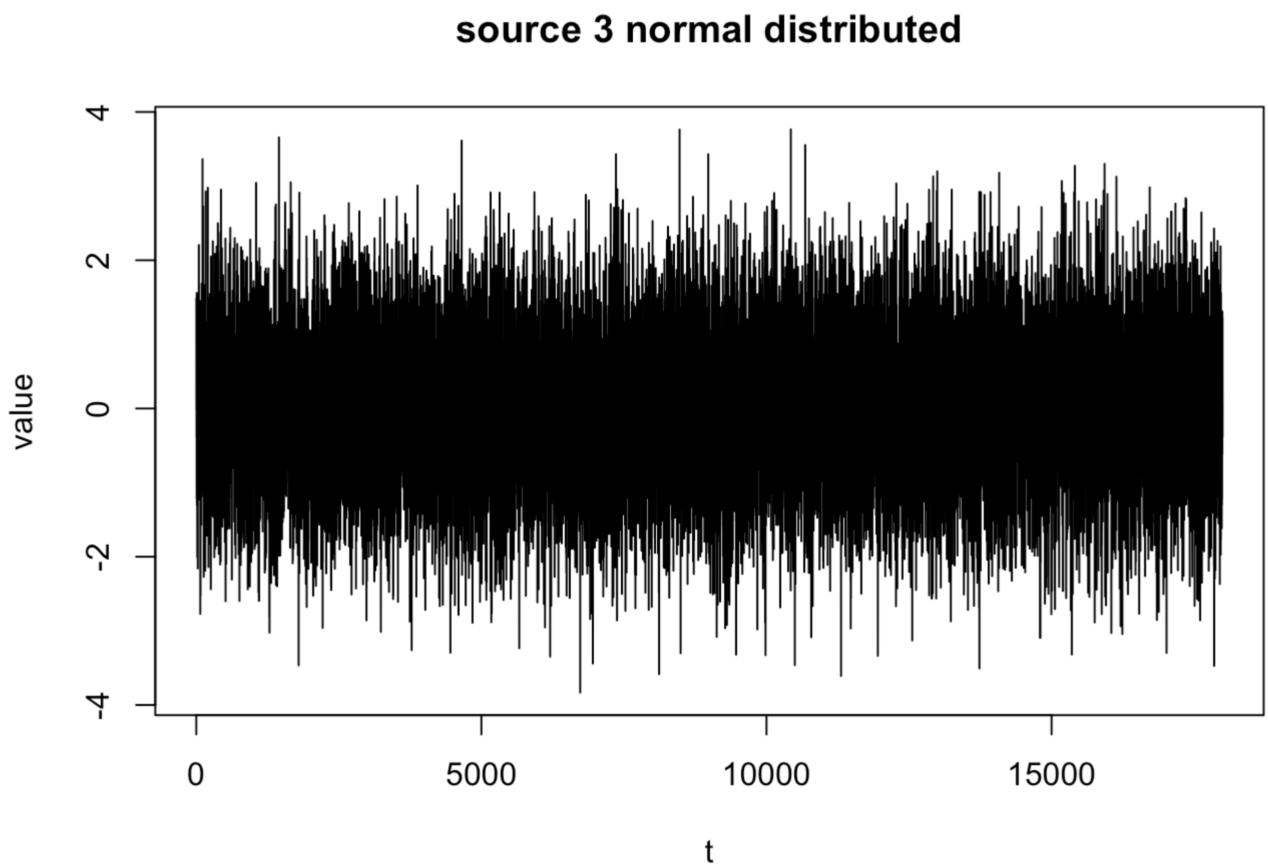


```
# play(audioSample(t(as.matrix(S_hat_natural[1,])), rate =
# 8192)) play(audioSample(t(as.matrix(S_hat_natural[2,])),
# rate = 8192))
```

6.1.b Use the two sound signals from the last problem sheet and add (as third source s3) an ad- ditional "noise" source (normally distributed random numbers with a standard deviation similar to the two signals). Mix the signals using a mixing matrix of your choice and apply your ICA-algorithm. Plot the Mixed Sounds and recovered Sources

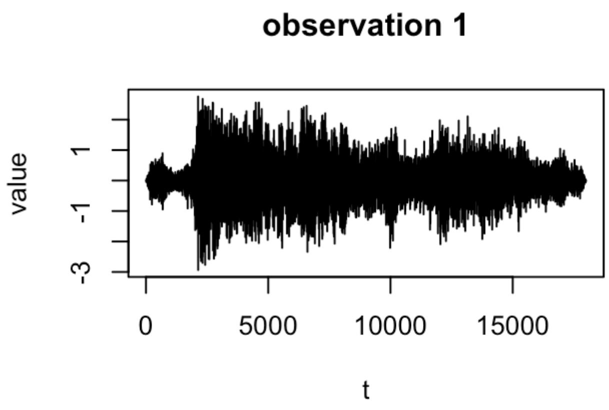
Plot initialised the third source normal distributed

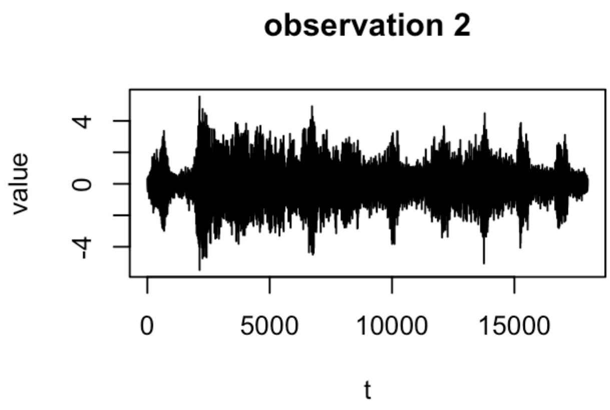
```
set.seed(222)
s3 <- t(matrix(rnorm(nrow(s1), 0, ((sd(as.matrix(s1)) + sd(as.matrix(s2)))/2)),
    nrow = 1))
S3 = t(as.matrix(data.frame(s1, s2, s3)))

par(mfrow = c(1, 1))
plot(1:18000, s3, type = "l", main = "source 3 normal distributed",
    xlab = "t", ylab = "value")
```

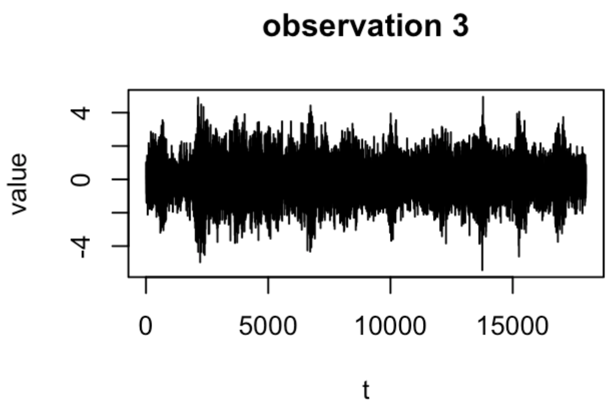


plot the created observations X

```
set.seed(1234)
A3 = matrix(runif(9, 0, 1), nrow = 3)
X3 = A3 %*% S3

# centering
X3[1, ] = X3[1, ] - mean(X3[1, ])
X3[2, ] = X3[2, ] - mean(X3[2, ])
X3[3, ] = X3[3, ] - mean(X3[3, ])

par(mfrow = c(2, 2))
plot(1:18000, X3[1, ], type = "l", main = "observation 1", xlab = "t",
    ylab = "value")
plot(1:18000, X3[2, ], type = "l", main = "observation 2", xlab = "t",
    ylab = "value")
plot(1:18000, X3[3, ], type = "l", main = "observation 3", xlab = "t",
    ylab = "value")
```







Plot the unmixed Xb by modified natural gradient 3x3

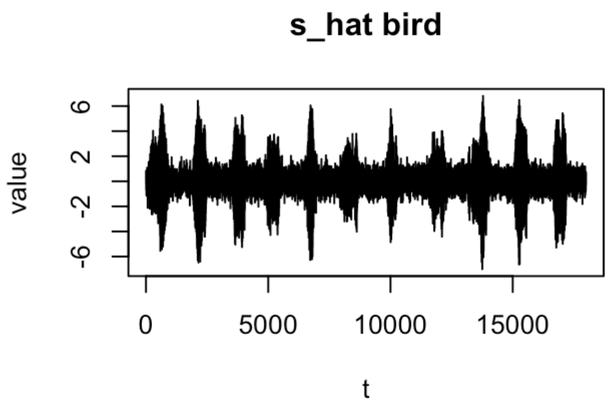
```
# natural Gradient
t = 1
eta_t = 0.3
alpha = 1
lambda = 0.9102  #0.9102
set.seed(9991)
W3 = matrix(runif(9, 0, 1), ncol = 3)

W_natural3 = unmixing_natural(W3, X3)
```

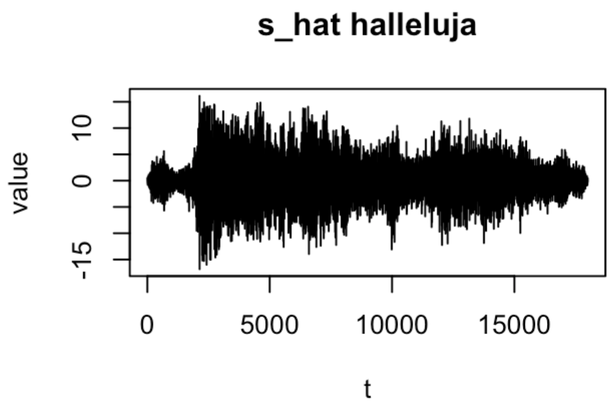
```
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
```

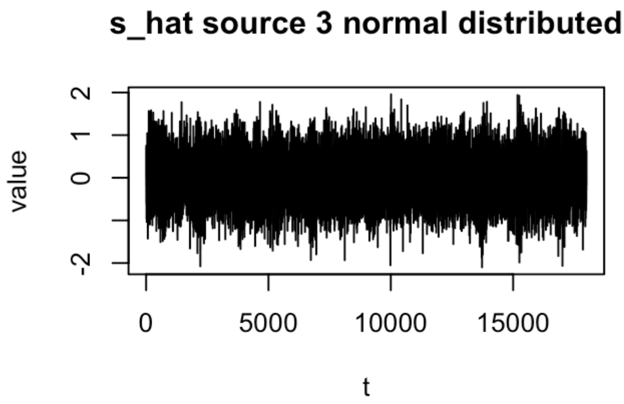
```
S_hat_natural3 = W_natural3 %*% X3

cor(t(S_hat_natural3), t(S3))
```

```
##             V1        V1.1         s3
## [1,] 0.9135854  0.12821027 0.37858387
## [2,] 0.1931624  0.97641253 0.09332271
## [3,] 0.3774333 -0.07263767 0.92045554
```

```
par(mfrow = c(2, 2))
plot(1:18000, S_hat_natural3[1, ], type = "l", main = "s_hat bird",
    xlab = "t", ylab = "value")
plot(1:18000, S_hat_natural3[2, ], type = "l", main = "s_hat halleluja",
    xlab = "t", ylab = "value")
plot(1:18000, S_hat_natural3[3, ], type = "l", main = "s_hat source 3 normal distribu
ted",
    xlab = "t", ylab = "value")
# play(audioSample(t(as.matrix(S_hat_natural3[1,])), rate =
# 8192)) play(audioSample(t(as.matrix(S_hat_natural3[2,])),
# rate = 8192))
# play(audioSample(t(as.matrix(S_hat_natural3[3,])), rate =
# 8192))
```



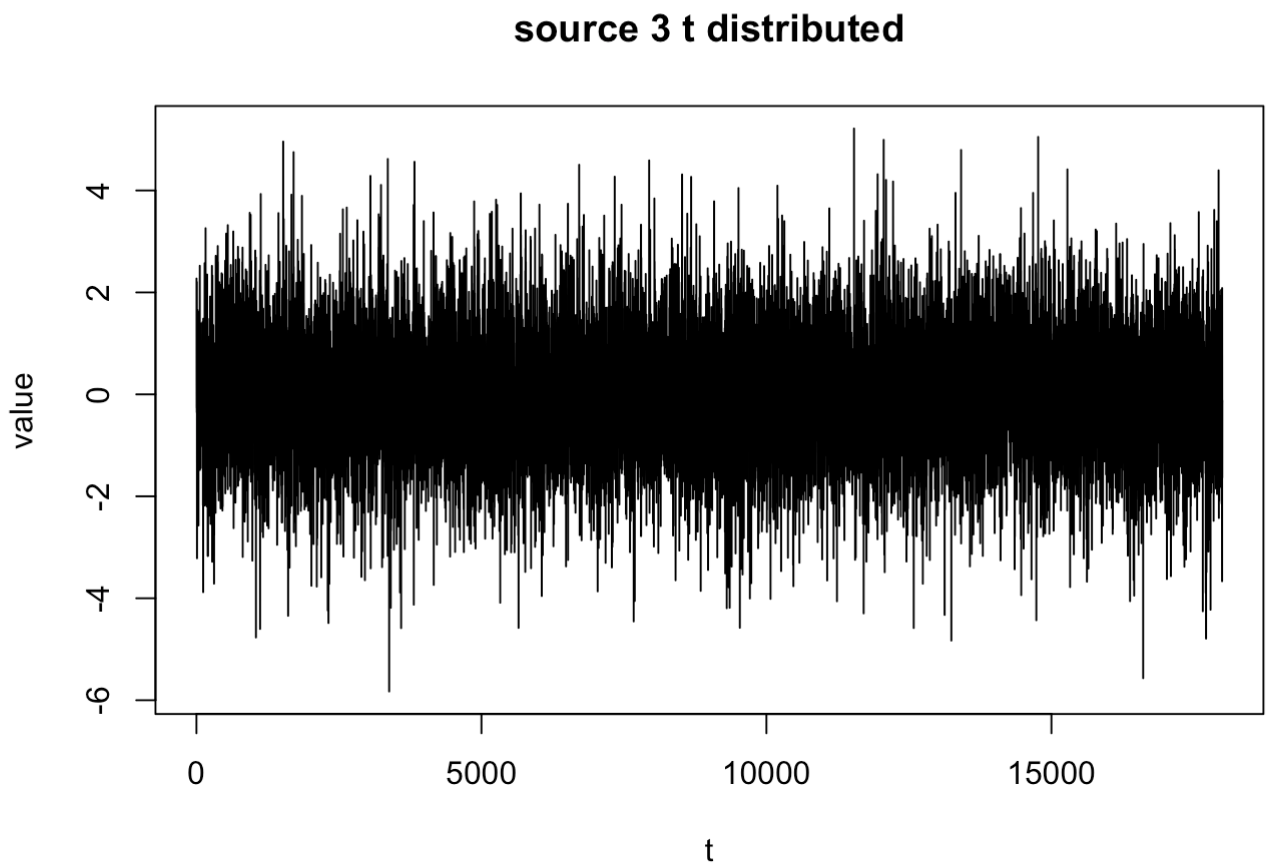




c. Do the same analysis but adding a different “noise”-source (e.g. Laplace distributed) instead of the normal one.

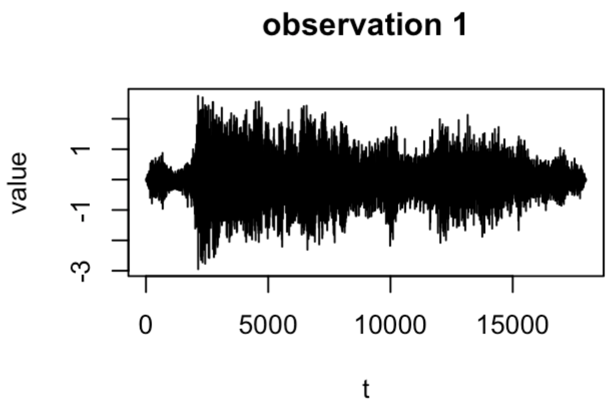
Plot initialised the third source t distributed

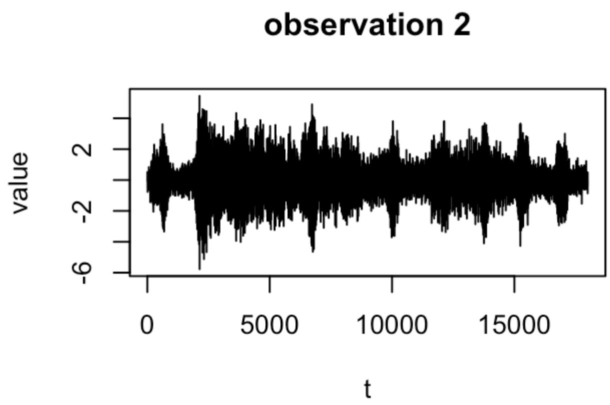
```
set.seed(666666)
st <- t(matrix(rt(nrow(s1), 10), nrow = 1))
St = t(as.matrix(data.frame(s1, s2, st)))

par(mfrow = c(1, 1))
plot(1:18000, st, type = "l", main = "source 3 t distributed",
    xlab = "t", ylab = "value")
```

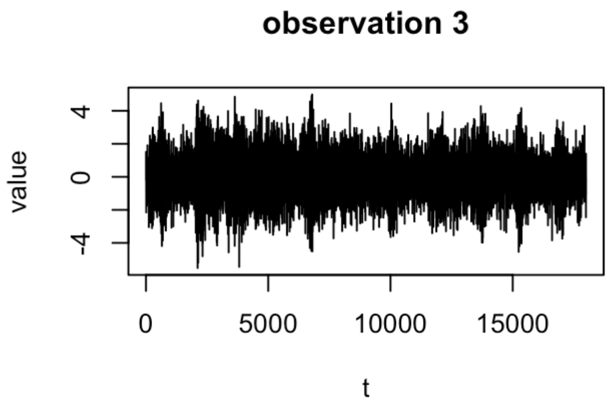


plot the created observations X laplace distributed

```
set.seed(1234)
A3 = matrix(runif(9, 0, 1), nrow = 3)
Xt = A3 %*% St

Xt[1, ] = Xt[1, ] - mean(Xt[1, ])
Xt[2, ] = Xt[2, ] - mean(Xt[2, ])
Xt[3, ] = Xt[3, ] - mean(Xt[3, ])

par(mfrow = c(2, 2))
plot(1:18000, Xt[1, ], type = "l", main = "observation 1", xlab = "t",
    ylab = "value")
plot(1:18000, Xt[2, ], type = "l", main = "observation 2", xlab = "t",
    ylab = "value")
plot(1:18000, Xt[3, ], type = "l", main = "observation 3", xlab = "t",
    ylab = "value")
```







plot S_hat noise source laplace distrubuted

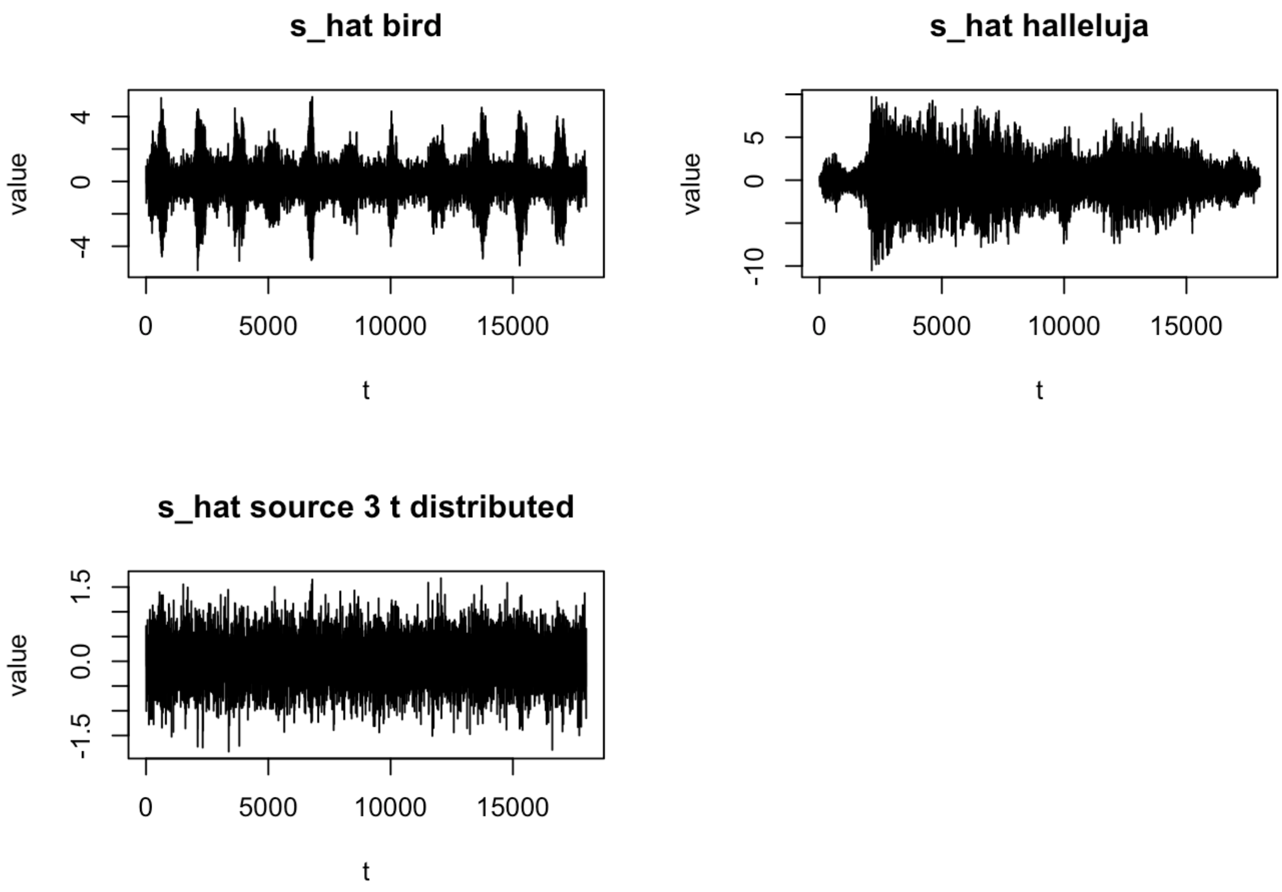
```
# natural Gradient
t = 1
eta_t = 0.25
alpha = 1
lambda = 0.91  #0.9102
set.seed(9991)
Wt = matrix(runif(9, 0, 1), ncol = 3)

W_naturalt = unmixing_natural(Wt, Xt)
```

```
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
## [1] 2.470328e-323
```

```
S_hat_naturalt = W_naturalt %*% Xt

cor(t(S_hat_natural3), t(S3))
```

```
##              V1        V1.1         s3
## [1,] 0.9135854  0.12821027 0.37858387
## [2,] 0.1931624  0.97641253 0.09332271
## [3,] 0.3774333 -0.07263767 0.92045554
```

```
par(mfrow = c(2, 2))
plot(1:18000, S_hat_naturalt[1, ], type = "l", main = "s_hat bird",
    xlab = "t", ylab = "value")
plot(1:18000, S_hat_naturalt[2, ], type = "l", main = "s_hat halleluja",
    xlab = "t", ylab = "value")
plot(1:18000, S_hat_naturalt[3, ], type = "l", main = "s_hat source 3 t distributed",

    xlab = "t", ylab = "value")
# play(audioSample(t(as.matrix(S_hat_natural3[1,])), rate =
# 8192)) play(audioSample(t(as.matrix(S_hat_natural3[2,])),
# rate = 8192))
# play(audioSample(t(as.matrix(S_hat_natural3[3,])), rate =
# 8192))
```

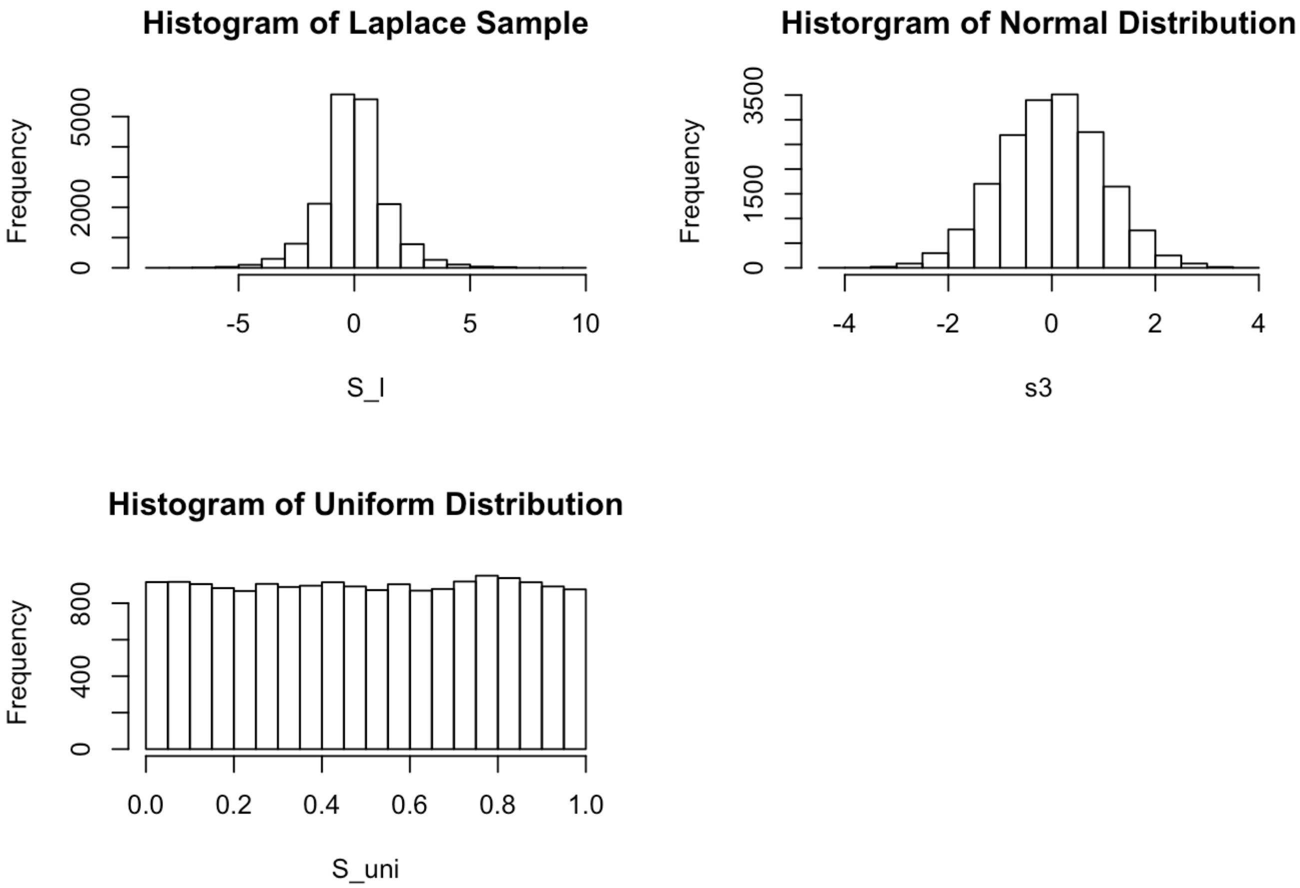


6.2 Moments of univariate distributions

Calculate the first 4 moments of the different random variables depending on the respective pa- rameters. In addition to providing the derivation (e.g. by using the characteristic function) fill the following table:

initalize the sample Laplace, Normal and Uniform Distribution

```
M = matrix(1, ncol = 3, nrow = 4)

set.seed(14321)
# Laplace
b <- 1
mean <- 0
S_l <- rlaplace(nrow(s1), mean, b)

# normal
s3 <- rnorm(nrow(s1), 0, ((sd(as.matrix(s1)) + sd(as.matrix(s2)))/2))

# Uniform
a = 0
b = 1
S_uni <- runif(nrow(s1), a, b)

par(mfrow = c(2, 2))
hist(S_l, main = "Histogram of Laplace Sample")
hist(s3, main = "Historgram of Normal Distribution")
hist(S_uni, main = "Histogram of Uniform Distribution")
```



calculate the first Moment

Moments (derivations)

1) Gaussian: $X \sim \mathcal{N}(\mu, \sigma^2)$

$$CF: \varphi_X(t) = \exp\left(i\mu t - \frac{\sigma^2 t^2}{2}\right)$$

first Moment
~~mean~~
$(E(X))$

$$\frac{\partial}{\partial t}\varphi_X(t) = (i\mu - \sigma^2 t)\cdot \exp\left(-i\mu t - \frac{\sigma^2 t^2}{2}\right)$$

$\rightarrow$ evaluate at $t=0$

$$iE[X] = i\mu \cdot \exp(0) = i\mu$$

$$\Leftrightarrow E[X] = \mu$$

second moment
$(E(X^2))$

$$\frac{\partial^2}{\partial t^2}\varphi_X(t) = -\sigma^2 \exp\left(i\mu t - \frac{\sigma^2 t^2}{2}\right) + (i\mu - \sigma^2 t)\cdot(i\mu - \sigma^2 t)\exp\left(-i\mu t \frac{\sigma^2 t^2}{2}\right)$$

$\rightarrow$ evaluate at $t=0$

$$i^2 E[X^2] = -\sigma^2 \exp(0) + i^2\mu^2 \exp(0)$$
$$= -\sigma^2 - \mu^2$$

$$\Leftrightarrow E[X^2] = \sigma^2 + \mu^2$$

for centered moment set $\mu = E[X] = 0$

$$\Rightarrow E[X^2]_c = \sigma^2$$

third moment
$(E(X^3))$

$$\frac{\partial^3}{\partial t^3}\varphi_X(t) = -(i\mu - \sigma^2 t)\sigma^2 \exp(\dots) + 2(i\mu - \sigma^2 t)(-\sigma^2)\exp(\dots) + (i\mu - \sigma^2 t)^3 \exp(\dots)$$

$\rightarrow$ evaluate at $t=0$: $i^3 E[X^3] = -3i\mu\sigma^2 + i^3\mu^3$

$$\Leftrightarrow E[X^3] = -\frac{1}{i^2}3\mu\sigma^2 + \mu^3 = 3\mu\sigma^2 + \mu^3$$

For standardized moment set $\mu = E[X] = 0$ and $\sigma^2 = E[X^2]_c = 1$ $\Rightarrow E[X_3]_s = 0$

fourth moment:

$$\frac{\partial^4}{\partial t^4} \varphi_X(t) = \ldots$$

$\rightarrow$ evaluate at $t=0$ $\Rightarrow E[X^4] = \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4$

$\Rightarrow$ standardize:

$$E[X^4]_s = 3\underbrace{\sigma^2}_{=1}\underbrace{\sigma^2}_{=1} = 3$$

$Y \sim \text{Laplace}(\mu, b)$

$$CF: \varphi_Y(t) = \frac{e^{it\mu}}{1+b^2t^2}$$

First moment:

$$\frac{\partial}{\partial t}\varphi_Y(t) = \frac{\partial}{\partial t}(1+b^2t^2)^{-1}\exp(it\mu)$$

$$= -(2b^2t)\exp(it\mu) + (1+b^2t^2)^{-1} i\mu \cdot \exp(it\mu)$$

-> evaluate at $t=0$

$$iE[Y] = i\mu$$

$$\Leftrightarrow E[Y] = \mu$$

Second moment:

$$\frac{\partial^2}{\partial t^2}\varphi_Y(t) = -2b^2\exp(it\mu) - 2b^2t^2 i\mu\exp(it\mu) + i^2\mu^2\exp(it\mu) +$$
$$+ 2b^2ti\mu\exp(it\mu) + (bti\mu)^2\exp(it\mu)$$

-> evaluate at $t=0$

$$i^2E[Y^2] = -2b^2 + i^2\mu^2 = -2b^2 - \mu^2$$

$$\Leftrightarrow E[Y^2] = 2b^2 + \mu^2$$

center
i.e. set $\mu = E[Y] = 0$

$$\Rightarrow E[Y^2]_c = 2b^2$$

same proceedure for third and fourth moment

Uniform distr. $z \sim \mathrm{Unif}(a,b)$

first moment:

$\cancel{E[X] = \int_{-\infty}^{\infty} x f_z(x)}$ $f_z(z) = \begin{cases} \frac{1}{b-a} & ; a \leq z \leq b \\ 0 & ; \text{else} \end{cases}$ density

$$E[Z] = \int_{-\infty}^{\infty} z \cdot f_z(z)\, dz$$

$$= \frac{1}{b-a} \int_a^b z\, dz$$

$$= \frac{1}{2} \frac{b^2 - a^2}{b - a} = \frac{a+b}{2}$$

second centered moment:

$$E[Z^2]_{cen} = E[Z^2] - E[Z]^2$$

$$= \frac{1}{b-a} \int_a^b z^2\, dz - \left(\frac{a+b}{2}\right)^2$$

$$= \frac{1}{3} \frac{b^3 - a^3}{b - a} - \left(\frac{a+b}{2}\right)^2$$

$$= \frac{1}{12} (b-a)^2$$

same proceedure for third and fourth moment

```
M = matrix(1, ncol = 3, nrow = 4)

first_moment <- function(x) {
    return(sum(x)/(length(x)))
}

M[1, 1] <- round(first_moment(S_l), 1)
M[1, 2] <- round(first_moment(s3), 1)
M[1, 3] <- round(first_moment(S_uni), 1)

M[1, ]
```

```
## [1] 0.0 0.0 0.5
```

calculate the second Moment

```
second_moment <- function(x) {
    return(sum((x - mean(x))^2)/length(x))
}

M[2, 1] <- round(second_moment(S_l), 1)
M[2, 2] <- round(second_moment(s3), 1)
M[2, 3] <- round(second_moment(S_uni), 1)

M[2, ]
```

```
## [1] 2.0 1.0 0.1
```

calculate the third Moment

```
third_moment <- function(x) {
    return((sum((x - mean(x))^3)/length(x))/((sum((x - mean(x))^2)/length(x))^(3/2)))
}

M[3, 1] <- round(third_moment(S_l), 1)
M[3, 2] <- round(third_moment(s3), 1)
M[3, 3] <- round(third_moment(S_uni), 1)

M[3, ]
```

```
## [1] 0.1 0.0 0.0
```

calculate the fourth Moment

```
vier_moment <- function(x) {
    return((sum((x - mean(x))^4)/length(x))/((sum((x - mean(x))^2)/length(x))^(4/2)))
}

M[4, 1] <- round(vier_moment(S_l), 1)
M[4, 2] <- round(vier_moment(s3), 1)
M[4, 3] <- round(vier_moment(S_uni), 1)

M[4, ]
```

```
## [1] 5.9 3.0 1.8
```

fill the following table: CHECK

the colums represent the distributions (Laplace, Normal, Uniform)

the row represent the Moments, the first, the second (centered), the third (standardized), the fourth(standardized)

```
M
```

```
##      [,1] [,2] [,3]
## [1,]  0.0    0  0.5
## [2,]  2.0    1  0.1
## [3,]  0.1    0  0.0
## [4,]  5.9    3  1.8
```

6.3 The file distrib.mat contains three toy datasets (uniform, normal, laplacian), each 10000 samples of 2 sources. Do the following for each dataset (which can be read for example using Python with loadmat from scipy.io):

```
l = readMat("hw6/distrib.mat")

s1 = l[[1]]
s2 = l[[2]]
s3 = l[[3]]
```

a. Apply the following mixing matrix A to the original data s:

```
A = matrix(4:1, ncol = 2, byrow = TRUE)

x1 = A %*% s1
x2 = A %*% s2
x3 = A %*% s3
```

b. Center the mixed data to zero mean.

```
x1[1, ] = x1[1, ] - mean(x1[1, ])
x1[2, ] = x1[2, ] - mean(x1[2, ])

x2[1, ] = x2[1, ] - mean(x2[1, ])
x2[2, ] = x2[2, ] - mean(x2[2, ])

x3[1, ] = x3[1, ] - mean(x3[1, ])
x3[2, ] = x3[2, ] - mean(x3[2, ])
```

c. Decorrelatethedatabyapplyingprincipalcomponentanalysis(PCA)andprojectthemonto the principal components (PCs).

```
pcs1 = eigen(cov(t(x1)))$vectors
pcs2 = eigen(cov(t(x2)))$vectors
pcs3 = eigen(cov(t(x3)))$vectors

evals1 = eigen(cov(t(x1)))$values
evals2 = eigen(cov(t(x2)))$values
evals3 = eigen(cov(t(x3)))$values

proj1 = t(x1) %*% pcs1
proj2 = t(x2) %*% pcs2
proj3 = t(x3) %*% pcs3

round(cov((proj1)), 5)
```

```
##          [,1]    [,2]
## [1,] 45.46668 0.00000
## [2,]  0.00000 0.20219
```

```
round(cov((proj2)), 5)
```

```
##          [,1]    [,2]
## [1,] 29.86607 0.00000
## [2,]  0.00000 0.13393
```

```
round(cov((proj3)), 5)
```

```
##          [,1]    [,2]
## [1,] 10.05524 0.00000
## [2,]  0.00000 0.04445
```

d. Scale the data to unit variance in each PC direction (now the data is whitened or sphered).

```
proj1_w = t(x1) %*% pcs1 %*% diag(evals1^(-0.5))
proj2_w = t(x2) %*% pcs2 %*% diag(evals2^(-0.5))
proj3_w = t(x3) %*% pcs3 %*% diag(evals3^(-0.5))

round(cov((proj1_w)), 5)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
round(cov((proj2_w)), 5)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
round(cov((proj3_w)), 5)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

e. Rotate the data by different angles θ and calculate the kurtosis1 empirically for each dimension:

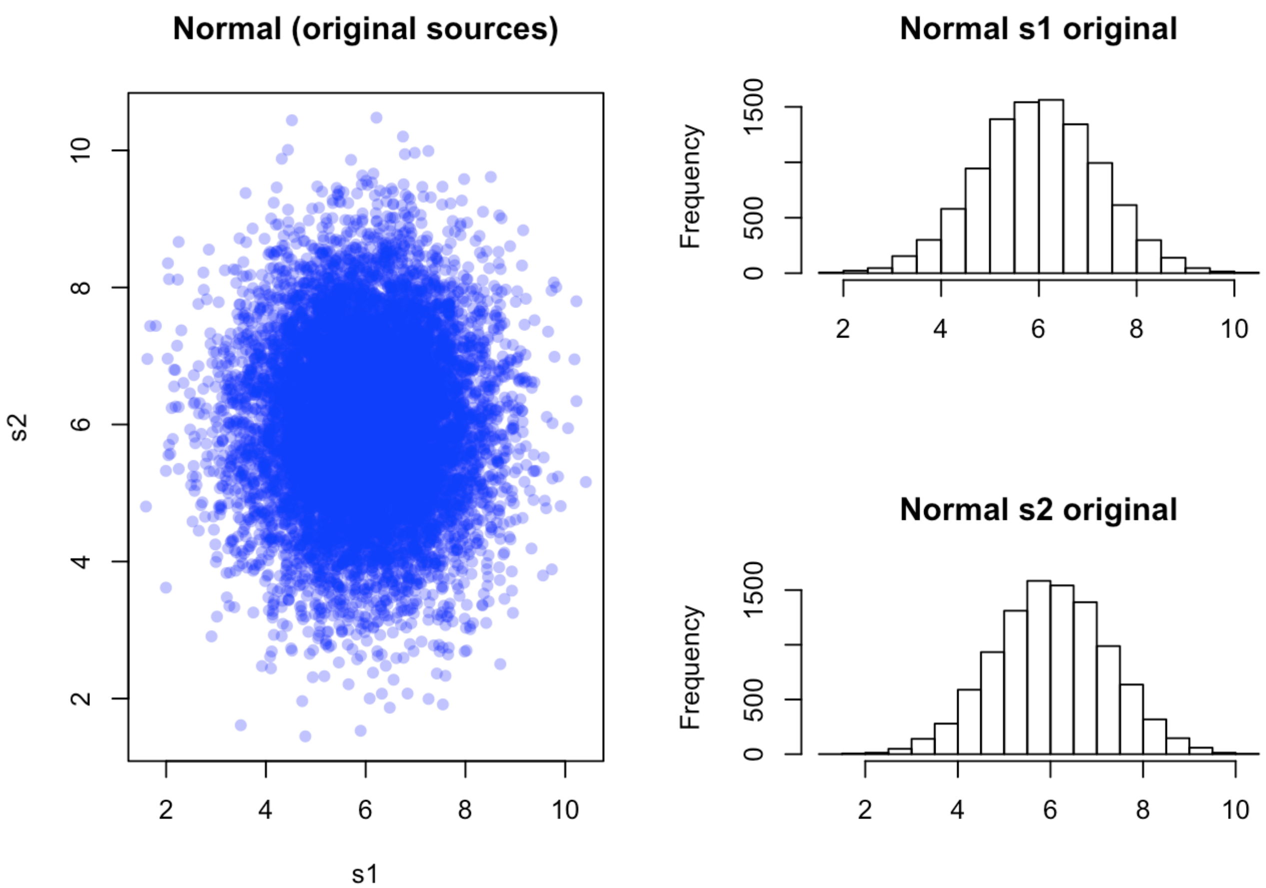
```
rotation_matrix = function(theta) {
    return(matrix(c(cos(theta), -sin(theta), sin(theta), cos(theta)),
        ncol = 2, byrow = TRUE))
}

thetas = seq(0, 2 * pi, pi/50)

kurt = function(x) {
    k = 1/length(x) * sum(x^4) - 3
    return(k)
}

kurt_rotation = function(x, thetas) {

    k_mat = matrix(nrow = length(thetas), ncol = 2)
    i = 1

    if (nrow(x) > ncol(x)) {
        x = t(x)
    }

    for (t in thetas) {
        R = rotation_matrix(t)

        x_theta = t(R %*% x)
        k = apply(x_theta, 2, kurt)

        k_mat[i, ] = k
        i = i + 1

    }
    return(k_mat)
}

k_after_rotation1 = kurt_rotation(proj1_w, thetas)
k_after_rotation2 = kurt_rotation(proj2_w, thetas)
k_after_rotation3 = kurt_rotation(proj3_w, thetas)
```

f. Find the minimum and maximum kurtosis value for the first dimension and rotate the data accordingly.

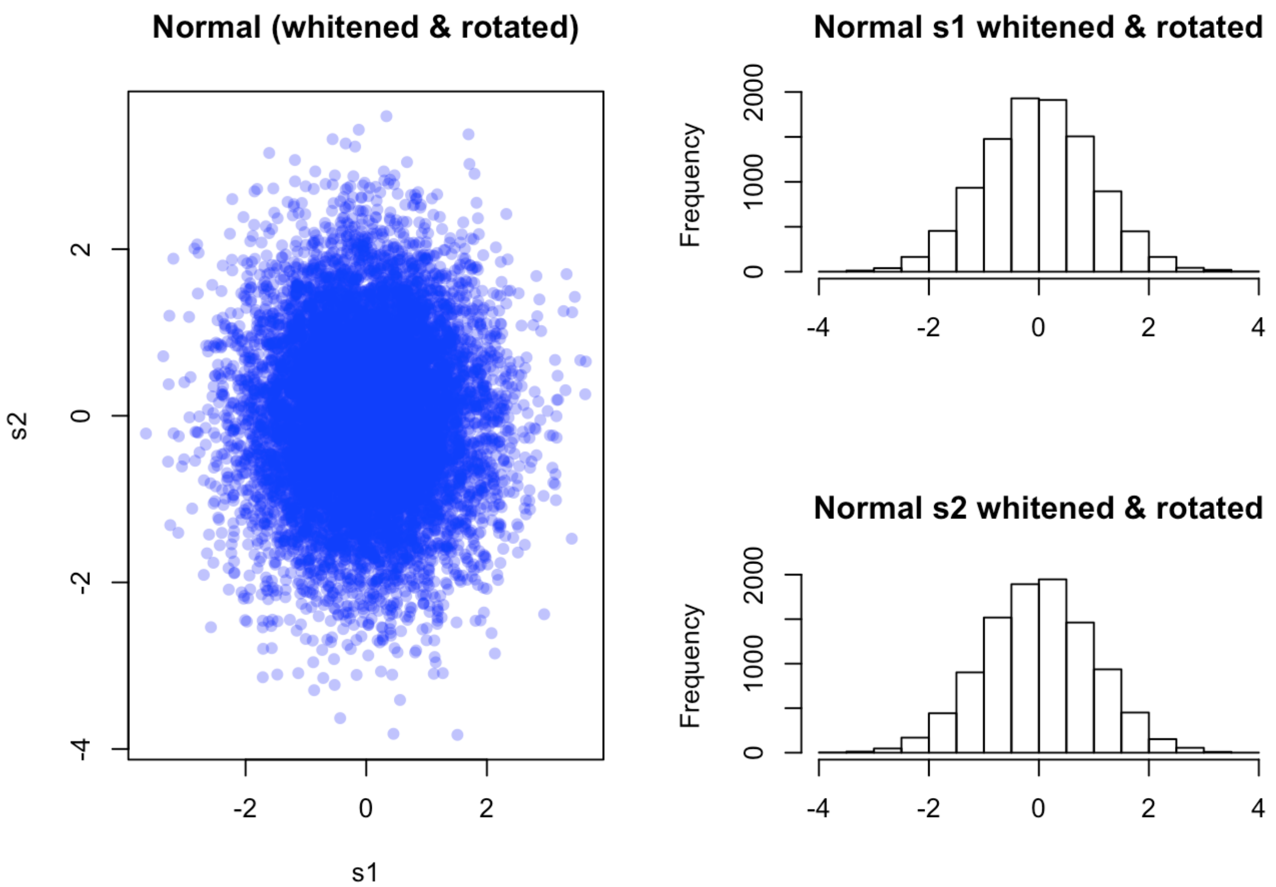
```
max1 = which.max(k_after_rotation1[, 1])
max2 = which.max(k_after_rotation2[, 1])
max3 = which.max(k_after_rotation3[, 1])

R1 = rotation_matrix(thetas[max1])
R2 = rotation_matrix(thetas[max2])
R3 = rotation_matrix(thetas[max3])

proj1_w_rotated = R1 %*% t(proj1_w)
proj2_w_rotated = R2 %*% t(proj2_w)
proj3_w_rotated = R3 %*% t(proj3_w)
```

- Plot the original dataset (sources) and the mixed dataset after the steps (a), (b), (c), (d), and (f) as a scatter plot and display the respective marginal histograms. For step (e) plot the kurtosis value as a function of angle for each dimension.

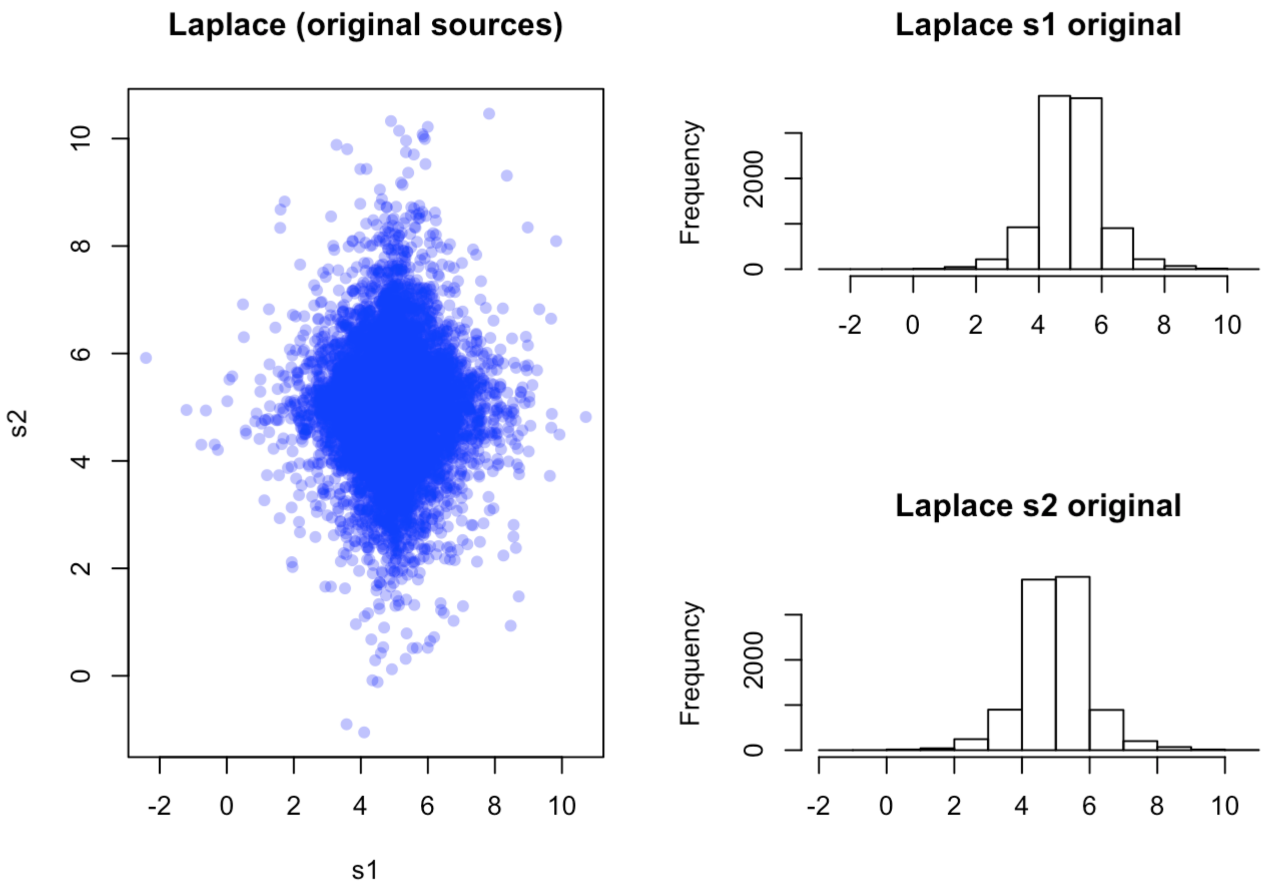
```
m <- rbind(c(1, 2), c(1, 3))
layout(m, widths = 1)

plot(s1[1, ], s1[2, ], col = alpha("blue", 0.3), pch = 16, main = "Normal (original s
ources)",
    xlab = "s1", ylab = "s2")
hist(s1[1, ], main = "Normal s1 original", xlab = "")
hist(s1[2, ], main = "Normal s2 original", xlab = "")
```



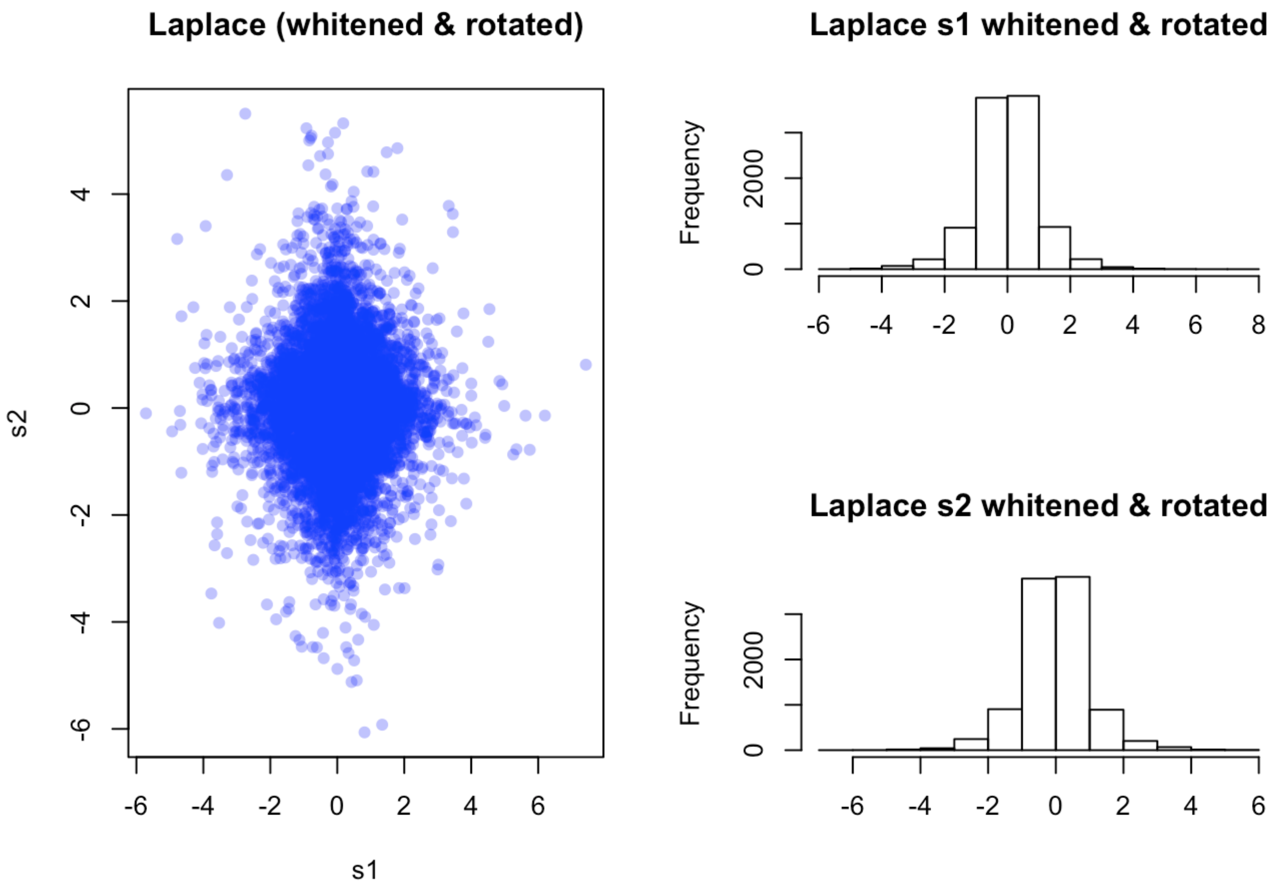
```
plot(proj1_w_rotated[1, ], proj1_w_rotated[2, ], col = alpha("blue",
    0.3), pch = 16, main = "Normal (whitened & rotated)", xlab = "s1",
    ylab = "s2")
hist(proj1_w_rotated[1, ], main = "Normal s1 whitened & rotated",
    xlab = "")
hist(proj1_w_rotated[2, ], main = "Normal s2 whitened & rotated",
    xlab = "")
```



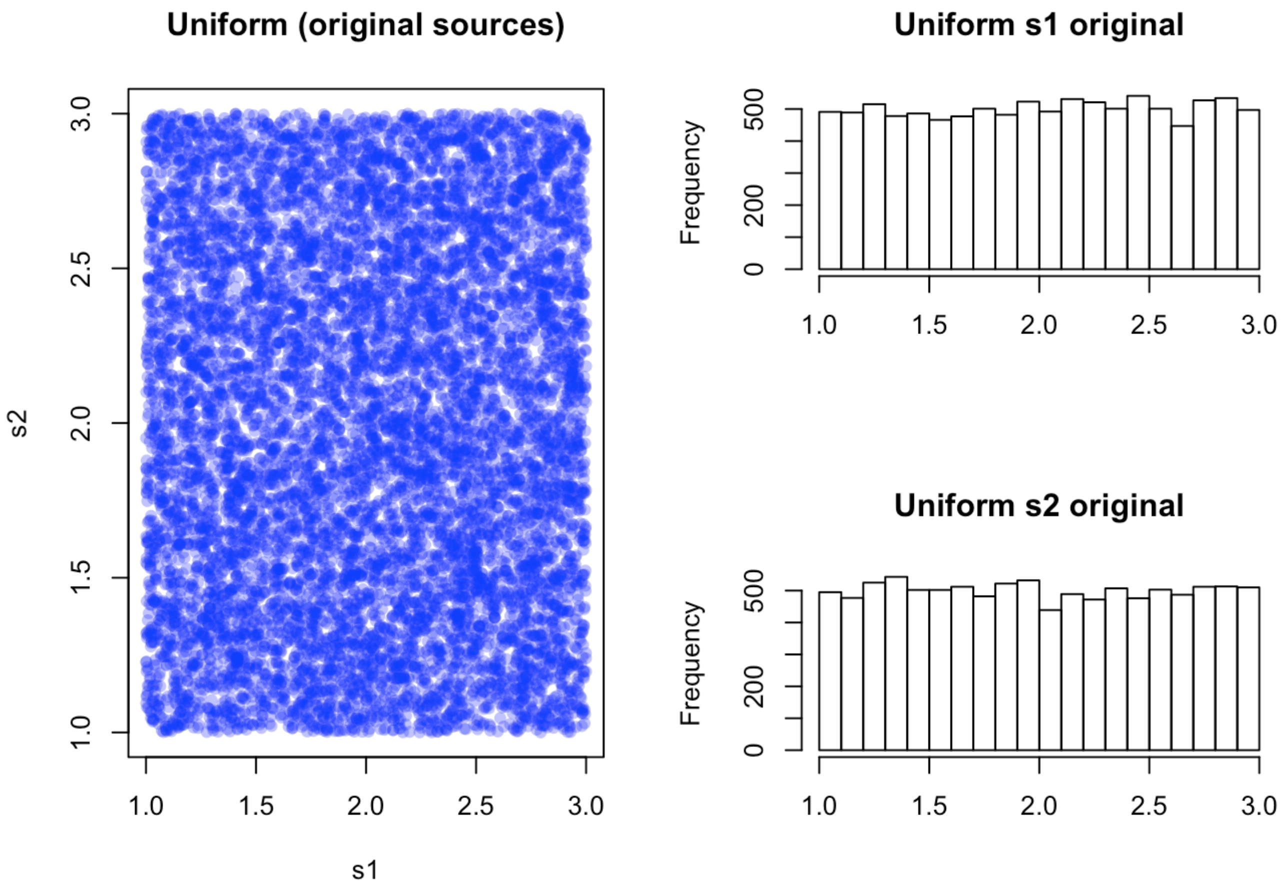

```
plot(s2[1, ], s2[2, ], col = alpha("blue", 0.3), pch = 16, main = "Laplace (original
 sources)",
    xlab = "s1", ylab = "s2")
hist(s2[1, ], main = "Laplace s1 original", xlab = "")
hist(s2[2, ], main = "Laplace s2 original", xlab = "")
```



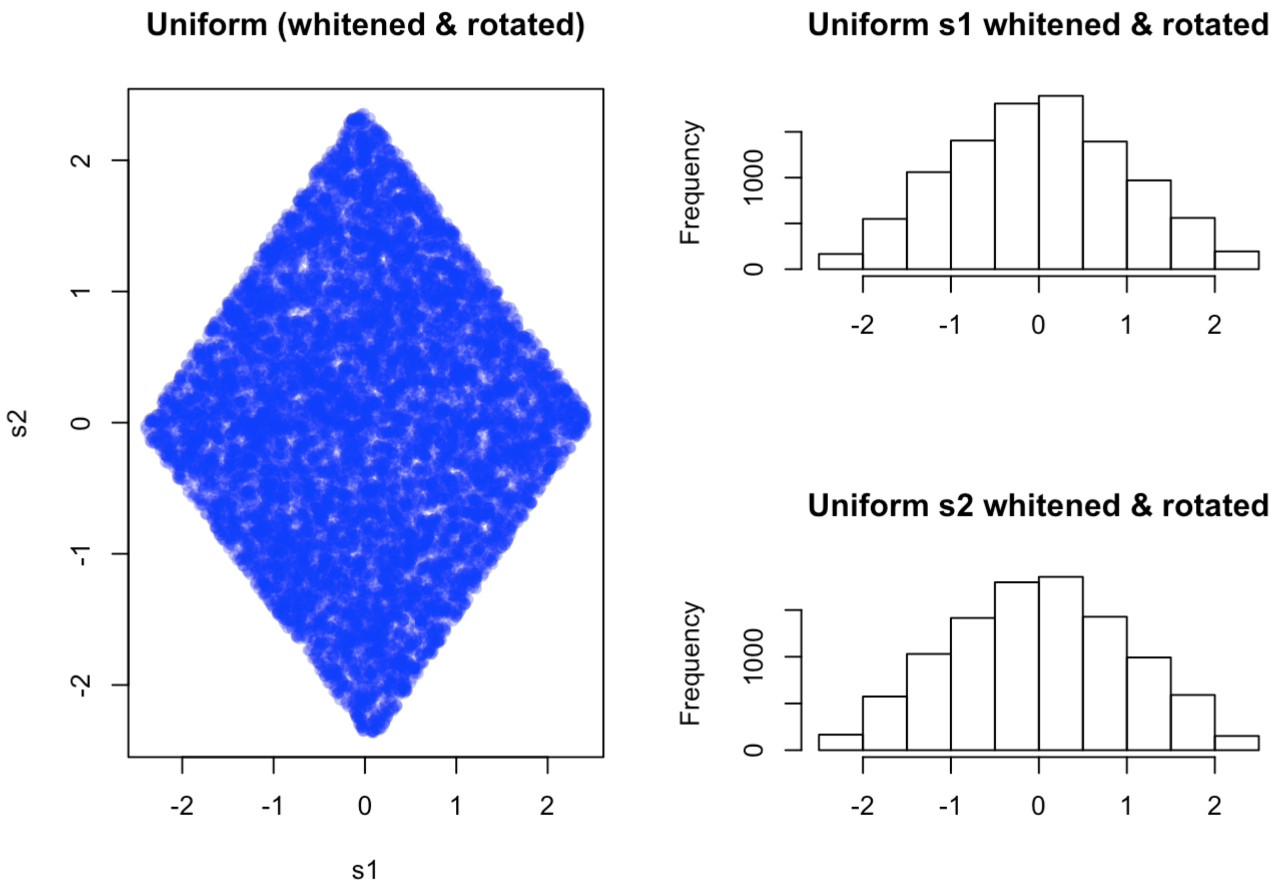

```
plot(proj2_w_rotated[1, ], proj2_w_rotated[2, ], col = alpha("blue",
    0.3), pch = 16, main = "Laplace (whitened & rotated)", xlab = "s1",
    ylab = "s2")
hist(proj2_w_rotated[1, ], main = "Laplace s1 whitened & rotated",
    xlab = "")
hist(proj2_w_rotated[2, ], main = "Laplace s2 whitened & rotated",
    xlab = "")
```




```
plot(s3[1, ], s3[2, ], col = alpha("blue", 0.3), pch = 16, main = "Uniform (original
 sources)",
    xlab = "s1", ylab = "s2")
hist(s3[1, ], main = "Uniform s1 original", xlab = "")
hist(s3[2, ], main = "Uniform s2 original", xlab = "")
```

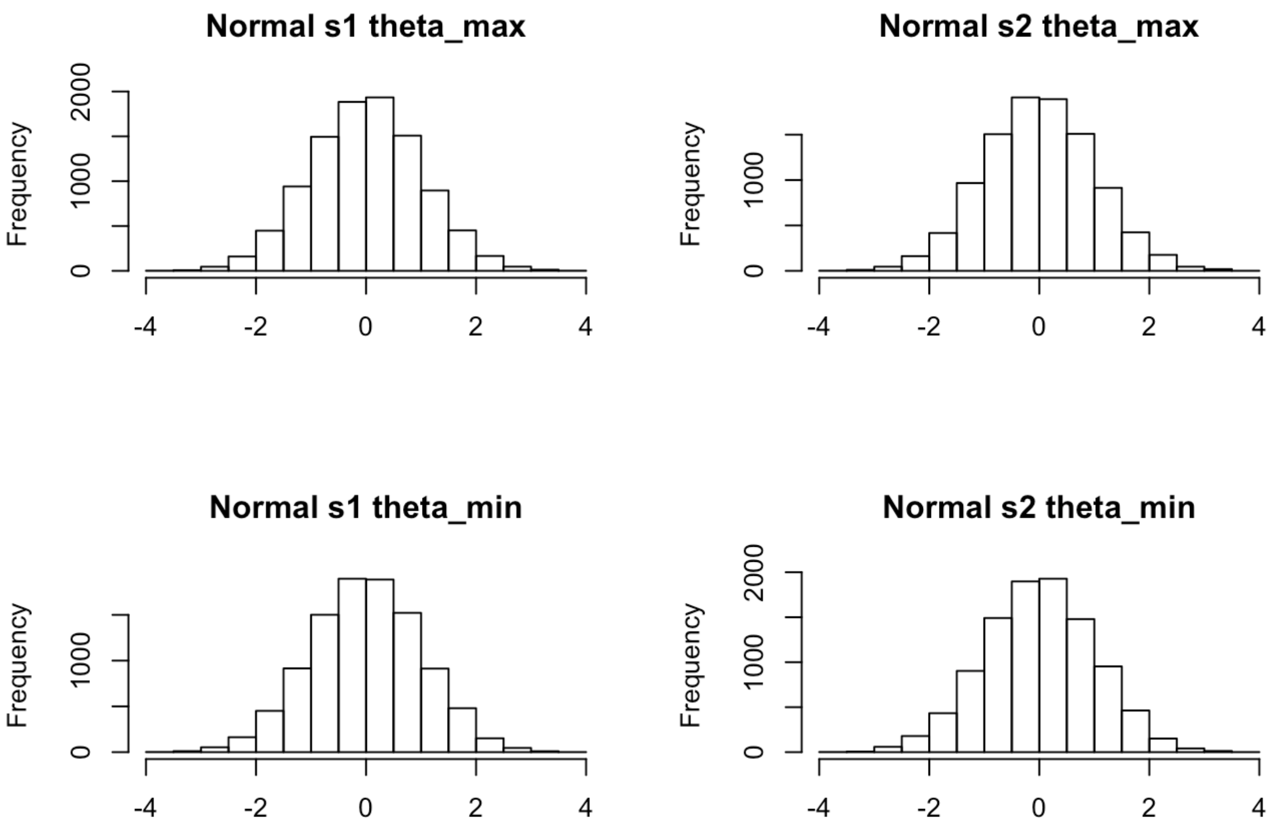



```
plot(proj3_w_rotated[1, ], proj3_w_rotated[2, ], col = alpha("blue",
    0.3), pch = 16, main = "Uniform (whitened & rotated)", xlab = "s1",
    ylab = "s2")
hist(proj3_w_rotated[1, ], main = "Uniform s1 whitened & rotated",
    xlab = "")
hist(proj3_w_rotated[2, ], main = "Uniform s2 whitened & rotated",
    xlab = "")
```

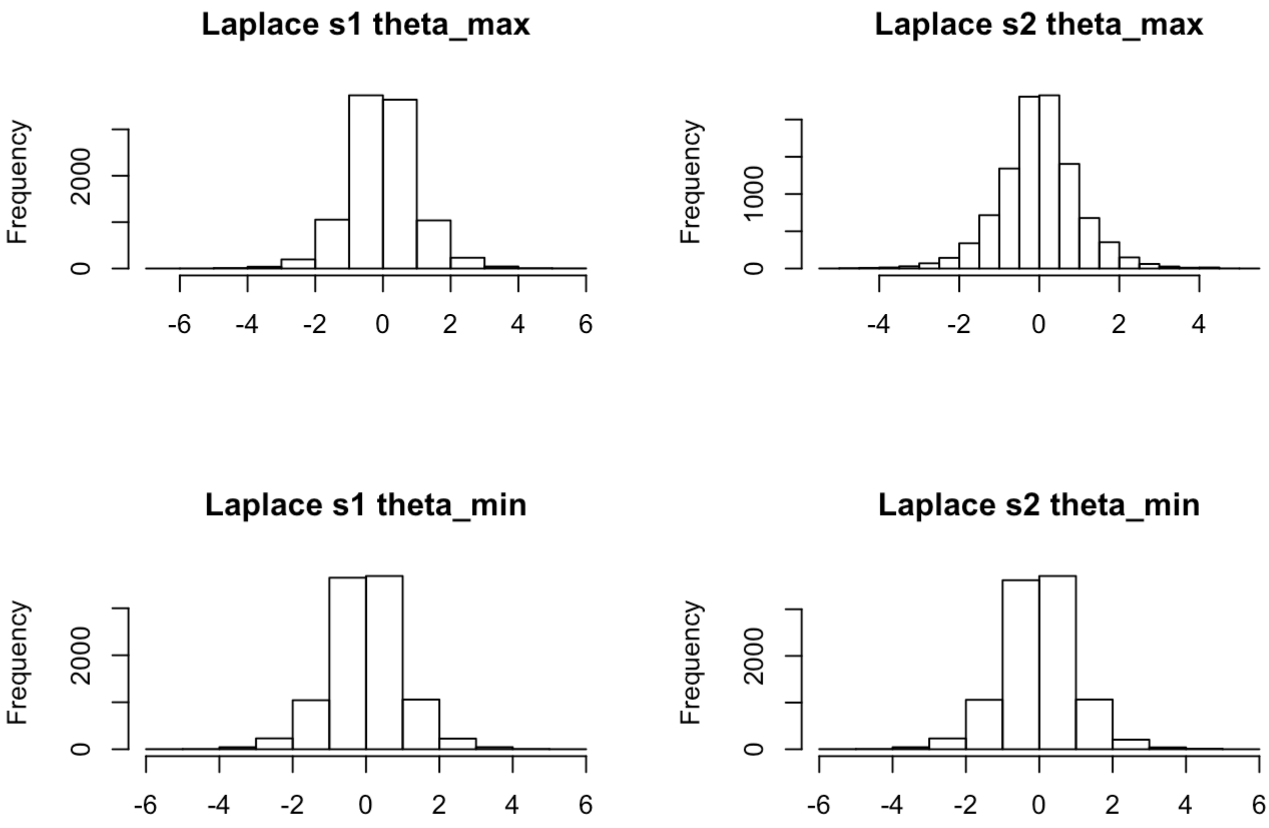


- Compare the histograms after rotation by θmin and θmax for the different distributions.

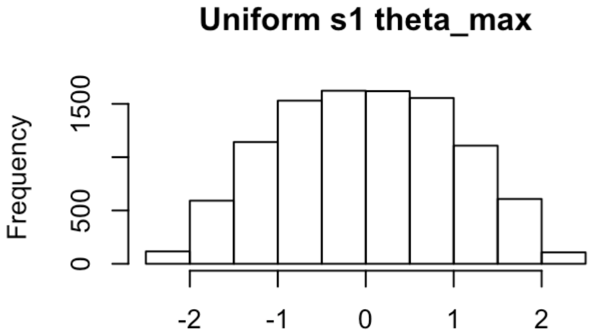
```
par(mfrow = c(2, 2))

min1 = which.min(k_after_rotation1[, 1])
min2 = which.min(k_after_rotation2[, 1])
min3 = which.min(k_after_rotation3[, 1])

R_min1 = rotation_matrix(thetas[min1])
R_max1 = rotation_matrix(max1)
R_min2 = rotation_matrix(thetas[min2])
R_max2 = rotation_matrix(max2)
R_min3 = rotation_matrix(thetas[min3])
R_max3 = rotation_matrix(max3)

proj1_w_rotated_max = R_max1 %*% t(proj1_w)
proj2_w_rotated_max = R_max2 %*% t(proj2_w)
proj3_w_rotated_max = R_max3 %*% t(proj3_w)

proj1_w_rotated_min = R_min1 %*% t(proj1_w)
proj2_w_rotated_min = R_min2 %*% t(proj2_w)
proj3_w_rotated_min = R_min3 %*% t(proj3_w)

hist(proj1_w_rotated_max[1, ], main = "Normal s1 theta_max",
    xlab = "")
hist(proj1_w_rotated_max[2, ], main = "Normal s2 theta_max",
    xlab = "")
hist(proj1_w_rotated_min[1, ], main = "Normal s1 theta_min",
    xlab = "")
hist(proj1_w_rotated_min[2, ], main = "Normal s2 theta_min",
    xlab = "")
```



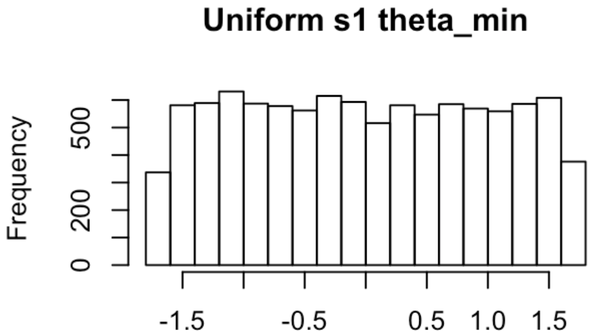
```
hist(proj2_w_rotated_max[1, ], main = "Laplace s1 theta_max",
    xlab = "")
hist(proj2_w_rotated_max[2, ], main = "Laplace s2 theta_max",
    xlab = "")
hist(proj2_w_rotated_min[1, ], main = "Laplace s1 theta_min",
    xlab = "")
hist(proj2_w_rotated_min[2, ], main = "Laplace s2 theta_min",
    xlab = "")
```

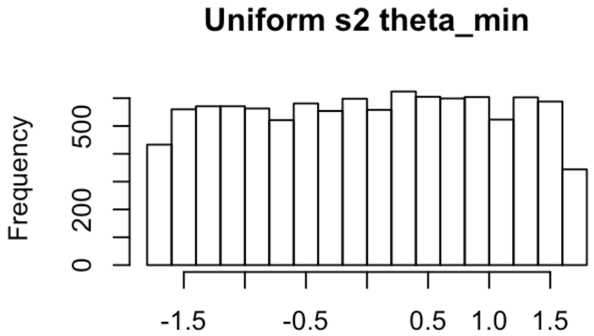



```
hist(proj3_w_rotated_max[1, ], main = "Uniform s1 theta_max",
    xlab = "")
hist(proj3_w_rotated_max[2, ], main = "Uniform s2 theta_max",
    xlab = "")
hist(proj3_w_rotated_min[1, ], main = "Uniform s1 theta_min",
    xlab = "")
hist(proj3_w_rotated_min[2, ], main = "Uniform s2 theta_min",
    xlab = "")
```


Uniform s1 theta_max
Frequency
0
500
1500
-2
-1
0
1
2


Uniform s2 theta_max
Frequency
0
1000
-2
-1
0
1
2


Uniform s1 theta_min
Frequency
0
200
500
-1.5
-0.5
0.5
1.0
1.5


Uniform s2 theta_min
Frequency
0
200
500
-1.5
-0.5
0.5
1.0
1.5