7.1

$y = Ax \quad \rightsquigarrow E\{yy^T\} = A\Sigma A^T \quad (1)$

Σ: Covariance Matrix of n-dimensional Gaussian random Vector x

A: Mixing Matrix

$J(x) = H(x_{gauss}) - H(x) \quad (2)$

$H(\alpha x) = H(x) + \log|\alpha| \quad (3)$

$H(x_{Gauss}) = \frac{1}{2}\log|\det\Sigma| + \frac{n}{2}(1+\log 2\pi) \quad (4)$

(1), (3), (4) in (2):

$$J(Ax) = \frac{1}{2}\log|\det(A\Sigma A^T)| + \frac{n}{2}(1+\log 2\pi) - (H(x) + \log|A|)$$

$$= \frac{1}{2}\log|\det\Sigma| + 2\frac{1}{2}\log|\det A| + \frac{n}{2}(1+\log 2\pi) - H(x) - \log|\det A|$$

$$= \underbrace{\frac{1}{2}\log|\det\Sigma| + \frac{n}{2}(1+\log 2\pi)}_{(4)} - H(x)$$

$$= H(x_{Gauss}) - H(x)$$

$$= J(x)$$

nonames2

```
# install.packages('jpeg')
library(jpeg)

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

# install.packages('reshape2')
library(reshape2)

# install.packages('ggplot2')
library(ggplot2)

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

library(audio)

# install.packages('psych')
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
get_patch <- function(matrix, h, w, return_vector = FALSE) {
    h <- h - 1
    w <- w - 1
    n <- nrow(matrix)
    p <- ncol(matrix)
    h_sample <- sample(1:n, 1, FALSE)
    w_sample <- sample(1:p, 1, FALSE)
    if ((h_sample + h) > n) {
        h_patch <- (h_sample - h):h_sample
    } else {
        h_patch <- h_sample:(h_sample + h)
    }
    if ((w_sample + w) > p) {
        w_patch <- (w_sample - w):w_sample
    } else {
        w_patch <- w_sample:(w_sample + w)
    }
    if (return_vector == FALSE) {
        return(matrix[h_patch, w_patch])
    } else {
        return(as.vector(matrix[h_patch, w_patch]))
    }
}


heatmap_custom <- function(matrix) {
    g1 <- ggplot(data = matrix, aes(x = Var1, y = Var2, fill = value)) +
        geom_tile() + scale_fill_continuous(low = "white", high = "black") +
        guides(fill = FALSE) + theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank(),
        axis.title.y = element_blank(), axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

    return(g1)
}

norm_vec <- function(x) sqrt(sum(x^2))

g1 <- function(x) {
    tanh(x)
}
```

Ex.2

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

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

# whiten S
St <- t(S)
covS <- cov(scale(St, center = TRUE, scale = FALSE))
ev <- eigen(covS)$vectors
eval <- eigen(covS)$values
Zt = scale(St, center = TRUE, scale = FALSE) %*% ev %*% diag((eval)^-0.5)
cov(Zt)
```

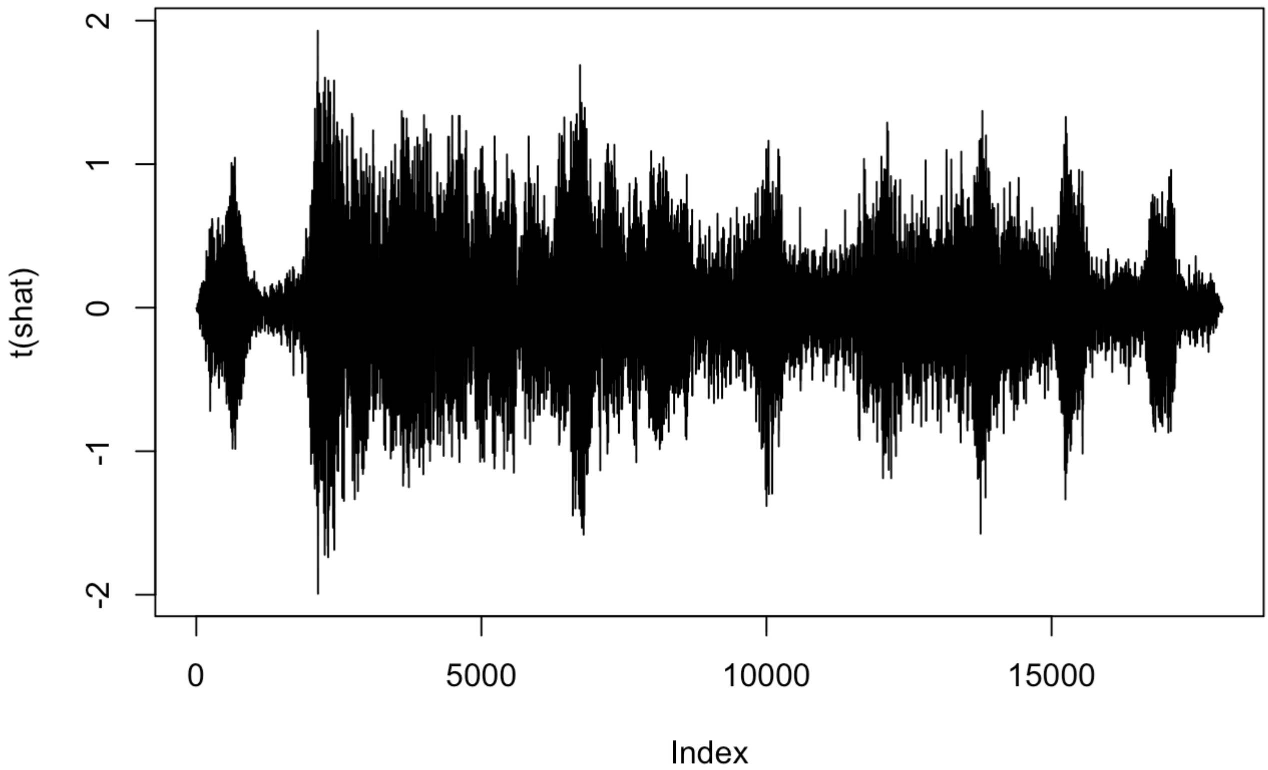
```
##              [,1]         [,2]
## [1,] 1.000000e+00 3.619225e-17
## [2,] 3.619225e-17 1.000000e+00
```

```r
# whitened Data: Z
Z <- t(Zt)

# initialize mixing Matrix A and compute Mixed Signal X
set.seed(1234)
A = matrix(runif(4, 0, 1), nrow = 2)

# mixted signals
X = A %*% Z
```

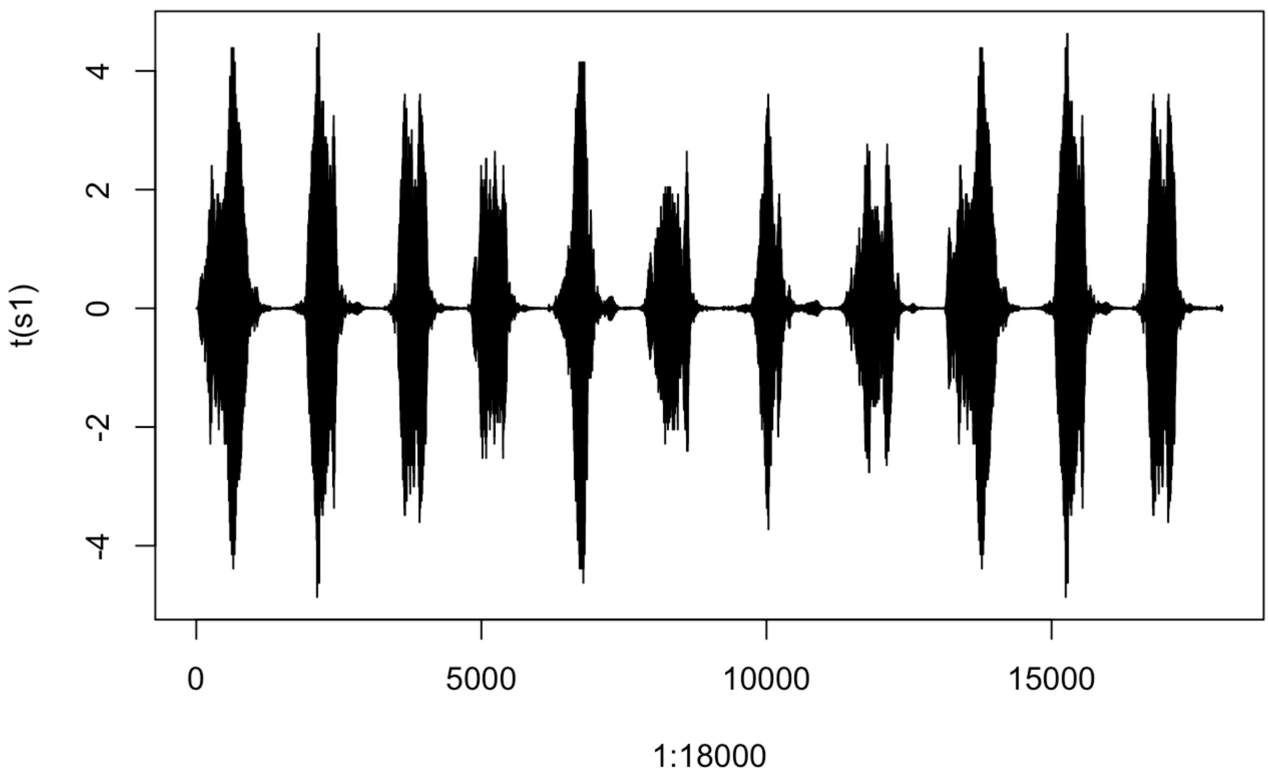
one unit alogorithm (nur eine independent component wird extrahiert)

```r
# init random vector with length 1
vec <- matrix(c(0, 1), nrow = 2, ncol = 1)

# inital learning rate
eta <- 0.01

for (i in 1:18000) {

    eta <- eta * 0.9999

    deltaVec <- eta * (-1) * X[, i] * g1(t(vec) %*% X[, i])

    vec <- vec + deltaVec

    vec <- vec/norm_vec(vec)
}
```

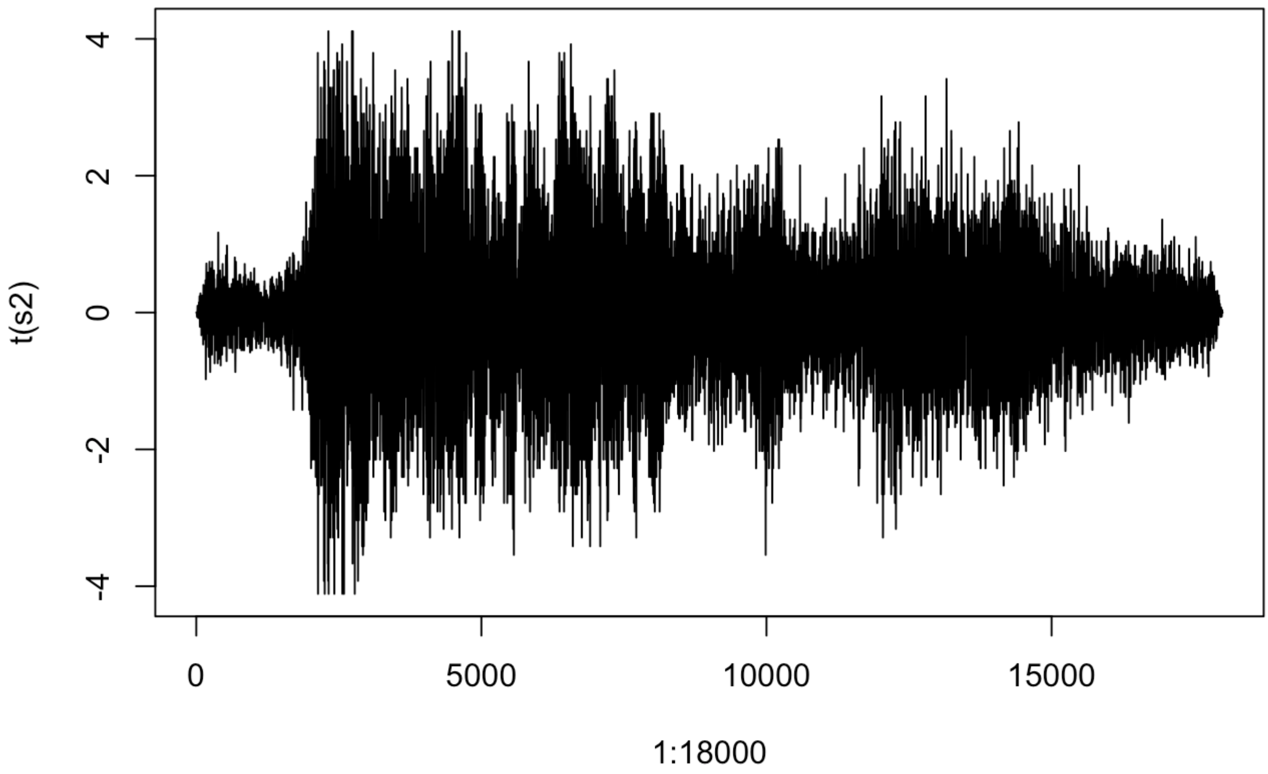
```r
# check
shat <- t(vec) %*% X
plot(t(shat), type = "l")
```




```
plot(1:18000, t(s1), type = "l")
```



```
plot(1:18000, t(s2), type = "l")
```




```
play(audioSample(as.matrix(shat), rate = 8192))
```

multiple unit alogorithm (beide independent components sollen extrahiert werden)

```r
# init random matrix (two orthogonal vectors, each with
# length 1)
set.seed(123)
b1 <- matrix(runif(2, min = 0, max = 1), nrow = 2, ncol = 1)
b1 <- b1/norm_vec(b1)
b2 <- matrix(c(1, (-b1[2]/b1[1])), nrow = 2, ncol = 1)
b2 <- b2/norm_vec(b2)

# initialized unmixing matrix
B <- cbind(b1, b2)

# inital learning rate
eta2 <- 0.01

for (i in 1:18000) {
    # learning rate
    eta2 <- eta2 * 0.9999

    # update b1
    deltaB1 <- eta2 * (-1) * X[, i] * g1(t(b1) %*% X[, i])
    b1 <- b1 + deltaB1

    # update b2
    deltaB2 <- eta2 * (-1) * X[, i] * g1(t(b2) %*% X[, i])
    b2 <- b2 + deltaB2

    # recalculate B
    B <- cbind(b1, b2)

    # normalize B
    maxNorm <- max(norm_vec(b1), norm_vec(b2))
    B <- B/maxNorm

    # decorrelate: FUNKTIONIERT NICHT! B danach linear abh?ngig
    Q <- eigen(cov(B %*% t(B)))$vectors
    EV <- eigen(cov(B %*% t(B)))$values
    Lambda <- diag(EV^-0.5)
    B <- Q %*% Lambda %*% t(Q) %*% B

    # normalize B b1 <- B[,1] b2 <- B[,2] maxNorm <-
    # max(norm_vec(b1), norm_vec(b2)) B <- B/maxNorm b1 <-
    # b1/norm_vec(b1) b2 <- b2/norm_vec(b2)

    # B <- ((B%*%t(B))^-.5) %*% B
}

B
```

```
##      [,1] [,2]
## [1,]  NaN  NaN
## [2,]  NaN  NaN
```

The file imgpca.zip (used also in exercise sheet 2) contains three categories of images: na- ture, buildings, and text (prefixes n,b,t). For each category:

```
path <- "/Users/maxand22/Google Drive/Humboldt/4. Semester/Machine Intelligence II/Ma
chine_Intelligence_II/hw7/imgpca"


fn <- file.path(path, c("n1.jpg", "n2.jpg", "n3.jpg", "n4.jpg",
    "n5.jpg", "n6.jpg", "n7.jpg", "n8.jpg", "n9.jpg", "n10.jpg",
    "n11.jpg", "n12.jpg", "n13.jpg"))
n <- lapply(fn, readJPEG)

fb <- file.path(path, c("b1.jpg", "b2.jpg", "b3.jpg", "b4.jpg",
    "b5.jpg", "b6.jpg", "b7.jpg", "b8.jpg", "b9.jpg", "b9_2.jpg",
    "b9_4.jpg", "b9_8.jpg", "b9_16.jpg", "b10.jpg"))
b <- lapply(fb, readJPEG)

ft <- file.path(path, c("t1.jpg", "t2.jpg", "t3.jpg", "t4.jpg",
    "t5.jpg", "t6.jpg", "t7.jpg", "t8.jpg", "t9.jpg", "t10.jpg",
    "t11.jpg", "t12.jpg", "t13.jpg", "t14.jpg"))
t <- lapply(ft, readJPEG)
```

a. Sample P patches of N × N pixels from all images of this category and rearrange each sample to a column vector. Choose number and size of the patches according to your computing resources. Recommended are $P \geq 20000$ and $N \geq 144$.

```
N <- sqrt(144)
P <- 20000

matrix_n <- matrix(nrow = 0, ncol = N * N)
matrix_b <- matrix(nrow = 0, ncol = N * N)
matrix_t <- matrix(nrow = 0, ncol = N * N)

set.seed(123)
for (i in 1:length(n)) {
    tmp1 <- t(replicate(2000, get_patch(n[[i]], N, N, return_vector = TRUE),
        simplify = "vector"))
    matrix_n <- rbind(matrix_n, tmp1)
}

for (i in 1:length(b)) {
    tmp1 <- t(replicate(2000, get_patch(b[[i]], N, N, return_vector = TRUE),
        simplify = "vector"))
    matrix_b <- rbind(matrix_b, tmp1)
}

for (i in 1:length(t)) {
    tmp1 <- t(replicate(2000, get_patch(t[[i]], N, N, return_vector = TRUE),
        simplify = "vector"))
    matrix_t <- rbind(matrix_t, tmp1)
}



# The required 20000 samples were reached for nature
P < nrow(matrix_n)
```

```
## [1] TRUE
```

```
# for buildungs
P < nrow(matrix_b)
```

```
## [1] TRUE
```

```
# for
P < nrow(matrix_t)
```

```
## [1] TRUE
```

b. Calculate the independent features of the image patches (these are the columns of mixing matrix A). Use a fastICA toolbox to compute this matrix: • Let fastica perform PCA and whitening of the data. • Use the contrast function G(sˆ) = 1 log cosh(asˆ) with a = 1.

```
set.seed(1234)

a_n <- fastICA(matrix_n, N * N, alg.typ = "parallel", fun = "logcosh",
    alpha = 1, method = "R", row.norm = FALSE, maxit = 200, tol = 1e-04,
    verbose = TRUE)
```

```
## Centering
```

```
## Whitening
```

```
## Symmetric FastICA using logcosh approx. to neg-entropy function
```

```
## Iteration 1 tol = 0.08607702
```

```
## Iteration 2 tol = 0.09870396
```

```
## Iteration 3 tol = 0.1101804
```

```
## Iteration 4 tol = 0.1198313
```

```
## Iteration 5 tol = 0.1274759
```

```
## Iteration 6 tol = 0.1336438
```

```
## Iteration 7 tol = 0.1381266
```

```
## Iteration 8 tol = 0.1416144
```

```
## Iteration 9 tol = 0.1474074
```

```
## Iteration 10 tol = 0.1564839
```

```
## Iteration 11 tol = 0.1647405
```

```
## Iteration 12 tol = 0.1706145
```

```
## Iteration 13 tol = 0.1758692
```

```
## Iteration 14 tol = 0.1785981
```

```
## Iteration 15 tol = 0.1804342
```

```
## Iteration 16 tol = 0.1799663
```

```
## Iteration 17 tol = 0.1778074
```

```
## Iteration 18 tol = 0.1733186
```

```
## Iteration 19 tol = 0.1668169
```

```
## Iteration 20 tol = 0.1585373
```

```
## Iteration 21 tol = 0.1486016
```

```
## Iteration 22 tol = 0.1494479
```

```
## Iteration 23 tol = 0.1519989
```

```
## Iteration 24 tol = 0.152655
```

```
## Iteration 25 tol = 0.1520883
```

```
## Iteration 26 tol = 0.1495254
```

```
## Iteration 27 tol = 0.1457614
```

```
## Iteration 28 tol = 0.1403156
```

```
## Iteration 29 tol = 0.1338264
```

```
## Iteration 30 tol = 0.128612
```

```
## Iteration 31 tol = 0.1392278
```

```
## Iteration 32 tol = 0.1511138
```

```
## Iteration 33 tol = 0.1619513
```

```
## Iteration 34 tol = 0.1733361
```

```
## Iteration 35 tol = 0.1827248
```

```
## Iteration 36 tol = 0.191709
```

```
## Iteration 37 tol = 0.1978676
```

```
## Iteration 38 tol = 0.2028241
```

```
## Iteration 39 tol = 0.2042542
```

```
## Iteration 40 tol = 0.2038006
```

```
## Iteration 41 tol = 0.1992296
```

```
## Iteration 42 tol = 0.2023909
```

```
## Iteration 43 tol = 0.213339
```

```
## Iteration 44 tol = 0.2245257
```

```
## Iteration 45 tol = 0.2374667
```

```
## Iteration 46 tol = 0.2514095
```

```
## Iteration 47 tol = 0.2668959
```

```
## Iteration 48 tol = 0.2839582
```

```
## Iteration 49 tol = 0.3015717
```

```
## Iteration 50 tol = 0.3203749
```

```
## Iteration 51 tol = 0.3371173
```

```
## Iteration 52 tol = 0.3646297
```

```
## Iteration 53 tol = 0.4485958
```

```
## Iteration 54 tol = 0.5540273
```

```
## Iteration 55 tol = 0.6950159
```

```
## Iteration 56 tol = 0.8809649
```

```
## Iteration 57 tol = 0.8685676
```

```
## Iteration 58 tol = 0.1794015
```

```
## Iteration 59 tol = 0.004894325
```

```
## Iteration 60 tol = 0.001924101
```

```
## Iteration 61 tol = 0.002000622
```

```
## Iteration 62 tol = 0.002020382
```

```
## Iteration 63 tol = 0.001969766
```

```
## Iteration 64 tol = 0.001844764
```

```
## Iteration 65 tol = 0.00189031
```

```
## Iteration 66 tol = 0.001969358
```

```
## Iteration 67 tol = 0.001999569
```

```
## Iteration 68 tol = 0.002007299
```

```
## Iteration 69 tol = 0.001977582
```

```
## Iteration 70 tol = 0.001888845
```

```
## Iteration 71 tol = 0.001763533
```

```
## Iteration 72 tol = 0.001632492
```

```
## Iteration 73 tol = 0.001646942
```

```
## Iteration 74 tol = 0.001626653
```

```
## Iteration 75 tol = 0.001588775
```

```
## Iteration 76 tol = 0.001547823
```

```
## Iteration 77 tol = 0.001509032
```

```
## Iteration 78 tol = 0.001471211
```

```
## Iteration 79 tol = 0.001430944
```

```
## Iteration 80 tol = 0.001386386
```

```
## Iteration 81 tol = 0.001379614
```

```
## Iteration 82 tol = 0.001522987
```

```
## Iteration 83 tol = 0.001653864
```

```
## Iteration 84 tol = 0.001752844
```

```
## Iteration 85 tol = 0.001806221
```

```
## Iteration 86 tol = 0.001802271
```

```
## Iteration 87 tol = 0.001736971
```

```
## Iteration 88 tol = 0.001620852
```

```
## Iteration 89 tol = 0.001472093
```

```
## Iteration 90 tol = 0.001308815
```

```
## Iteration 91 tol = 0.001145383
```

```
## Iteration 92 tol = 0.0009933767
```

```
## Iteration 93 tol = 0.001041998
```

```
## Iteration 94 tol = 0.001090588
```

```
## Iteration 95 tol = 0.001136374
```

```
## Iteration 96 tol = 0.001174685
```

```
## Iteration 97 tol = 0.001199564
```

```
## Iteration 98 tol = 0.001204172
```

```
## Iteration 99 tol = 0.00118166
```

```
## Iteration 100 tol = 0.00112713
```

```
## Iteration 101 tol = 0.001086272
```

```
## Iteration 102 tol = 0.001022939
```

```
## Iteration 103 tol = 0.0009335636
```

```
## Iteration 104 tol = 0.0008285606
```

```
## Iteration 105 tol = 0.0007188958
```

```
## Iteration 106 tol = 0.0006131558
```

```
## Iteration 107 tol = 0.0005167511
```

```
## Iteration 108 tol = 0.0004322503
```

```
## Iteration 109 tol = 0.0003601802
```

```
## Iteration 110 tol = 0.0003415584
```

```
## Iteration 111 tol = 0.0003644463
```

```
## Iteration 112 tol = 0.0003887234
```

```
## Iteration 113 tol = 0.0004129316
```

```
## Iteration 114 tol = 0.0004351996
```

```
## Iteration 115 tol = 0.0004534113
```

```
## Iteration 116 tol = 0.0004655017
```

```
## Iteration 117 tol = 0.0004698033
```

```
## Iteration 118 tol = 0.0004653361
```

```
## Iteration 119 tol = 0.0004519834
```

```
## Iteration 120 tol = 0.0004305279
```

```
## Iteration 121 tol = 0.0004025111
```

```
## Iteration 122 tol = 0.0003699347
```

```
## Iteration 123 tol = 0.0003349285
```

```
## Iteration 124 tol = 0.0002995009
```

```
## Iteration 125 tol = 0.0002653691
```

```
## Iteration 126 tol = 0.0002338287
```

```
## Iteration 127 tol = 0.0002056773
```

```
## Iteration 128 tol = 0.0001812312
```

```
## Iteration 129 tol = 0.0001604325
```

```
## Iteration 130 tol = 0.0001429895
```

```
## Iteration 131 tol = 0.0001284997
```

```
## Iteration 132 tol = 0.0001165343
```

```
## Iteration 133 tol = 0.000106686
```

```
## Iteration 134 tol = 0.0001007307
```

```
## Iteration 135 tol = 0.0001041893
```

```
## Iteration 136 tol = 0.0001077815
```

```
## Iteration 137 tol = 0.0001113255
```

```
## Iteration 138 tol = 0.0001146447
```

```
## Iteration 139 tol = 0.0001175758
```

```
## Iteration 140 tol = 0.0001199775
```

```
## Iteration 141 tol = 0.0001217393
```

```
## Iteration 142 tol = 0.0001227882
```

```
## Iteration 143 tol = 0.0001230906
```

```
## Iteration 144 tol = 0.0001226503
```

```
## Iteration 145 tol = 0.0001215029
```

```
## Iteration 146 tol = 0.0001197061
```

```
## Iteration 147 tol = 0.000117331
```

```
## Iteration 148 tol = 0.0001144527
```

```
## Iteration 149 tol = 0.0001111443
```

```
## Iteration 150 tol = 0.000107473
```

```
## Iteration 151 tol = 0.0001034985
```

```
## Iteration 152 tol = 9.927379e-05
```

```
a_b <- fastICA(matrix_b, N * N, alg.typ = "parallel", fun = "logcosh",
    alpha = 1, method = "R", row.norm = FALSE, maxit = 200, tol = 1e-04,
    verbose = TRUE)
```

```
## Centering
```

```
## Whitening
```

```
## Symmetric FastICA using logcosh approx. to neg-entropy function
```

```
## Iteration 1 tol = 0.136965
```

```
## Iteration 2 tol = 0.1363683
```

```
## Iteration 3 tol = 0.1292836
```

```
## Iteration 4 tol = 0.1255063
```

```
## Iteration 5 tol = 0.117478
```

```
## Iteration 6 tol = 0.1088217
```

```
## Iteration 7 tol = 0.1153955
```

```
## Iteration 8 tol = 0.1201871
```

```
## Iteration 9 tol = 0.1309147
```

```
## Iteration 10 tol = 0.152878
```

```
## Iteration 11 tol = 0.1850138
```

```
## Iteration 12 tol = 0.2221475
```

```
## Iteration 13 tol = 0.2560113
```

```
## Iteration 14 tol = 0.2819157
```

```
## Iteration 15 tol = 0.2990781
```

```
## Iteration 16 tol = 0.3104952
```

```
## Iteration 17 tol = 0.3133052
```

```
## Iteration 18 tol = 0.3101255
```

```
## Iteration 19 tol = 0.2969038
```

```
## Iteration 20 tol = 0.3164696
```

```
## Iteration 21 tol = 0.3655909
```

```
## Iteration 22 tol = 0.4276045
```

```
## Iteration 23 tol = 0.5185579
```

```
## Iteration 24 tol = 0.6513189
```

```
## Iteration 25 tol = 0.8633186
```

```
## Iteration 26 tol = 0.8077715
```

```
## Iteration 27 tol = 0.3813524
```

```
## Iteration 28 tol = 0.07647103
```

```
## Iteration 29 tol = 0.003174611
```

```
## Iteration 30 tol = 0.002750647
```

```
## Iteration 31 tol = 0.002750885
```

```
## Iteration 32 tol = 0.002649307
```

```
## Iteration 33 tol = 0.002467194
```

```
## Iteration 34 tol = 0.002244824
```

```
## Iteration 35 tol = 0.002020261
```

```
## Iteration 36 tol = 0.001809392
```

```
## Iteration 37 tol = 0.001648076
```

```
## Iteration 38 tol = 0.001625499
```

```
## Iteration 39 tol = 0.001629957
```

```
## Iteration 40 tol = 0.001738715
```

```
## Iteration 41 tol = 0.001830402
```

```
## Iteration 42 tol = 0.001879085
```

```
## Iteration 43 tol = 0.00186296
```

```
## Iteration 44 tol = 0.001772152
```

```
## Iteration 45 tol = 0.001616611
```

```
## Iteration 46 tol = 0.001428364
```

```
## Iteration 47 tol = 0.001243093
```

```
## Iteration 48 tol = 0.001098747
```

```
## Iteration 49 tol = 0.001058642
```

```
## Iteration 50 tol = 0.001079806
```

```
## Iteration 51 tol = 0.001093483
```

```
## Iteration 52 tol = 0.001130807
```

```
## Iteration 53 tol = 0.001199913
```

```
## Iteration 54 tol = 0.001247202
```

```
## Iteration 55 tol = 0.001264866
```

```
## Iteration 56 tol = 0.001250146
```

```
## Iteration 57 tol = 0.001207349
```

```
## Iteration 58 tol = 0.001182586
```

```
## Iteration 59 tol = 0.001187392
```

```
## Iteration 60 tol = 0.001334578
```

```
## Iteration 61 tol = 0.00147156
```

```
## Iteration 62 tol = 0.001581588
```

```
## Iteration 63 tol = 0.001645822
```

```
## Iteration 64 tol = 0.001650846
```

```
## Iteration 65 tol = 0.001596417
```

```
## Iteration 66 tol = 0.001493477
```

```
## Iteration 67 tol = 0.001353278
```

```
## Iteration 68 tol = 0.00118441
```

```
## Iteration 69 tol = 0.001147859
```

```
## Iteration 70 tol = 0.001099787
```

```
## Iteration 71 tol = 0.001025019
```

```
## Iteration 72 tol = 0.0009354995
```

```
## Iteration 73 tol = 0.0008429617
```

```
## Iteration 74 tol = 0.0007538062
```

```
## Iteration 75 tol = 0.0006694548
```

```
## Iteration 76 tol = 0.0006498281
```

```
## Iteration 77 tol = 0.0006538171
```

```
## Iteration 78 tol = 0.0006776679
```

```
## Iteration 79 tol = 0.0006829086
```

```
## Iteration 80 tol = 0.0006667655
```

```
## Iteration 81 tol = 0.0006297134
```

```
## Iteration 82 tol = 0.0006170195
```

```
## Iteration 83 tol = 0.0006360284
```

```
## Iteration 84 tol = 0.0006542787
```

```
## Iteration 85 tol = 0.0006707066
```

```
## Iteration 86 tol = 0.0006839169
```

```
## Iteration 87 tol = 0.0006921463
```

```
## Iteration 88 tol = 0.0006932191
```

```
## Iteration 89 tol = 0.0006847971
```

```
## Iteration 90 tol = 0.0006651788
```

```
## Iteration 91 tol = 0.0006342468
```

```
## Iteration 92 tol = 0.0005937254
```

```
## Iteration 93 tol = 0.000582475
```

```
## Iteration 94 tol = 0.0006478886
```

```
## Iteration 95 tol = 0.0007167721
```

```
## Iteration 96 tol = 0.0007801041
```

```
## Iteration 97 tol = 0.0008264645
```

```
## Iteration 98 tol = 0.0008438634
```

```
## Iteration 99 tol = 0.0008254124
```

```
## Iteration 100 tol = 0.0007745399
```

```
## Iteration 101 tol = 0.0007010468
```

```
## Iteration 102 tol = 0.0006156037
```

```
## Iteration 103 tol = 0.0005274285
```

```
## Iteration 104 tol = 0.0004435061
```

```
## Iteration 105 tol = 0.0003685903
```

```
## Iteration 106 tol = 0.0003050185
```

```
## Iteration 107 tol = 0.0002528999
```

```
## Iteration 108 tol = 0.0002109931
```

```
## Iteration 109 tol = 0.000177571
```

```
## Iteration 110 tol = 0.0001509196
```

```
## Iteration 111 tol = 0.000159681
```

```
## Iteration 112 tol = 0.0001783189
```

```
## Iteration 113 tol = 0.0001997752
```

```
## Iteration 114 tol = 0.0002243042
```

```
## Iteration 115 tol = 0.0002520838
```

```
## Iteration 116 tol = 0.0002831228
```

```
## Iteration 117 tol = 0.0003170926
```

```
## Iteration 118 tol = 0.0003530655
```

```
## Iteration 119 tol = 0.0003892087
```

```
## Iteration 120 tol = 0.0004225952
```

```
## Iteration 121 tol = 0.0004494094
```

```
## Iteration 122 tol = 0.0004656745
```

```
## Iteration 123 tol = 0.0004682317
```

```
## Iteration 124 tol = 0.0004556628
```

```
## Iteration 125 tol = 0.0004289795
```

```
## Iteration 126 tol = 0.0003915613
```

```
## Iteration 127 tol = 0.0003480092
```

```
## Iteration 128 tol = 0.0003281441
```

```
## Iteration 129 tol = 0.00035021
```

```
## Iteration 130 tol = 0.000371355
```

```
## Iteration 131 tol = 0.0003909011
```

```
## Iteration 132 tol = 0.0004078339
```

```
## Iteration 133 tol = 0.0004207886
```

```
## Iteration 134 tol = 0.0004281279
```

```
## Iteration 135 tol = 0.0004281445
```

```
## Iteration 136 tol = 0.0004193966
```

```
## Iteration 137 tol = 0.0004011189
```

```
## Iteration 138 tol = 0.0003842736
```

```
## Iteration 139 tol = 0.0003587408
```

```
## Iteration 140 tol = 0.0003260987
```

```
## Iteration 141 tol = 0.0002898671
```

```
## Iteration 142 tol = 0.000253526
```

```
## Iteration 143 tol = 0.0002199711
```

```
## Iteration 144 tol = 0.0001910282
```

```
## Iteration 145 tol = 0.0001673164
```

```
## Iteration 146 tol = 0.0001485183
```

```
## Iteration 147 tol = 0.0001338118
```

```
## Iteration 148 tol = 0.00012223
```

```
## Iteration 149 tol = 0.0001147205
```

```
## Iteration 150 tol = 0.0001085577
```

```
## Iteration 151 tol = 0.0001022732
```

```
## Iteration 152 tol = 9.584509e-05
```

```
a_t <- fastICA(matrix_t, N * N, alg.typ = "parallel", fun = "logcosh",
    alpha = 1, method = "R", row.norm = FALSE, maxit = 200, tol = 1e-04,
    verbose = TRUE)
```

```
## Centering
```

```
## Whitening
```

```
## Symmetric FastICA using logcosh approx. to neg-entropy function
```

```
## Iteration 1 tol = 0.06549518
```

```
## Iteration 2 tol = 0.07361292
```

```
## Iteration 3 tol = 0.08825801
```

```
## Iteration 4 tol = 0.1013007
```

```
## Iteration 5 tol = 0.112316
```

```
## Iteration 6 tol = 0.1208206
```

```
## Iteration 7 tol = 0.1278176
```

```
## Iteration 8 tol = 0.1315156
```

```
## Iteration 9 tol = 0.1322703
```

```
## Iteration 10 tol = 0.1265494
```

```
## Iteration 11 tol = 0.1144775
```

```
## Iteration 12 tol = 0.102099
```

```
## Iteration 13 tol = 0.1155263
```

```
## Iteration 14 tol = 0.1392235
```

```
## Iteration 15 tol = 0.1558185
```

```
## Iteration 16 tol = 0.1621974
```

```
## Iteration 17 tol = 0.009009335
```

```
## Iteration 18 tol = 0.008520353
```

```
## Iteration 19 tol = 0.008111207
```

```
## Iteration 20 tol = 0.01124011
```

```
## Iteration 21 tol = 0.02051477
```

```
## Iteration 22 tol = 0.04520823
```

```
## Iteration 23 tol = 0.06462653
```

```
## Iteration 24 tol = 0.01665546
```

```
## Iteration 25 tol = 0.004734633
```

```
## Iteration 26 tol = 0.004452541
```

```
## Iteration 27 tol = 0.004148887
```

```
## Iteration 28 tol = 0.003911804
```

```
## Iteration 29 tol = 0.00373016
```

```
## Iteration 30 tol = 0.003667917
```

```
## Iteration 31 tol = 0.003434955
```

```
## Iteration 32 tol = 0.004161929
```

```
## Iteration 33 tol = 0.005719431
```

```
## Iteration 34 tol = 0.007333973
```

```
## Iteration 35 tol = 0.007844071
```

```
## Iteration 36 tol = 0.006210232
```

```
## Iteration 37 tol = 0.004565302
```

```
## Iteration 38 tol = 0.005210513
```

```
## Iteration 39 tol = 0.005654998
```

```
## Iteration 40 tol = 0.005801513
```

```
## Iteration 41 tol = 0.0054958
```

```
## Iteration 42 tol = 0.004682625
```

```
## Iteration 43 tol = 0.003528187
```

```
## Iteration 44 tol = 0.002407848
```

```
## Iteration 45 tol = 0.002364787
```

```
## Iteration 46 tol = 0.002542844
```

```
## Iteration 47 tol = 0.002635421
```

```
## Iteration 48 tol = 0.002747459
```

```
## Iteration 49 tol = 0.003456655
```

```
## Iteration 50 tol = 0.004341014
```

```
## Iteration 51 tol = 0.00513624
```

```
## Iteration 52 tol = 0.005394408
```

```
## Iteration 53 tol = 0.004824918
```

```
## Iteration 54 tol = 0.003566568
```

```
## Iteration 55 tol = 0.002286282
```

```
## Iteration 56 tol = 0.001807385
```

```
## Iteration 57 tol = 0.001764045
```

```
## Iteration 58 tol = 0.001900194
```

```
## Iteration 59 tol = 0.002029516
```

```
## Iteration 60 tol = 0.00212853
```

```
## Iteration 61 tol = 0.00217247
```

```
## Iteration 62 tol = 0.002149114
```

```
## Iteration 63 tol = 0.0020598
```

```
## Iteration 64 tol = 0.001915053
```

```
## Iteration 65 tol = 0.001733318
```

```
## Iteration 66 tol = 0.001534602
```

```
## Iteration 67 tol = 0.001332509
```

```
## Iteration 68 tol = 0.001137258
```

```
## Iteration 69 tol = 0.0009581825
```

```
## Iteration 70 tol = 0.0008254858
```

```
## Iteration 71 tol = 0.0007206119
```

```
## Iteration 72 tol = 0.0006330833
```

```
## Iteration 73 tol = 0.0005637364
```

```
## Iteration 74 tol = 0.000509775
```

```
## Iteration 75 tol = 0.0004669152
```

```
## Iteration 76 tol = 0.0004309167
```

```
## Iteration 77 tol = 0.0003983489
```

```
## Iteration 78 tol = 0.0003668163
```

```
## Iteration 79 tol = 0.0003349356
```

```
## Iteration 80 tol = 0.0003285422
```

```
## Iteration 81 tol = 0.0003274382
```

```
## Iteration 82 tol = 0.0003212323
```

```
## Iteration 83 tol = 0.0003102077
```

```
## Iteration 84 tol = 0.0002951371
```

```
## Iteration 85 tol = 0.0002770947
```

```
## Iteration 86 tol = 0.0002649231
```

```
## Iteration 87 tol = 0.0002539357
```

```
## Iteration 88 tol = 0.0002423235
```

```
## Iteration 89 tol = 0.00023008
```

```
## Iteration 90 tol = 0.0002218832
```

```
## Iteration 91 tol = 0.0002195409
```

```
## Iteration 92 tol = 0.0002176843
```

```
## Iteration 93 tol = 0.0002187746
```

```
## Iteration 94 tol = 0.0002248963
```

```
## Iteration 95 tol = 0.00022904
```

```
## Iteration 96 tol = 0.000230746
```

```
## Iteration 97 tol = 0.000229739
```

```
## Iteration 98 tol = 0.0002259584
```

```
## Iteration 99 tol = 0.0002195638
```

```
## Iteration 100 tol = 0.0002109102
```

```
## Iteration 101 tol = 0.0002004962
```

```
## Iteration 102 tol = 0.000188894
```

```
## Iteration 103 tol = 0.0001766765
```

```
## Iteration 104 tol = 0.0001643562
```

```
## Iteration 105 tol = 0.0001523482
```

```
## Iteration 106 tol = 0.0001409561
```

```
## Iteration 107 tol = 0.0001383968
```

```
## Iteration 108 tol = 0.0001407565
```

```
## Iteration 109 tol = 0.0001424933
```

```
## Iteration 110 tol = 0.0001435204
```

```
## Iteration 111 tol = 0.0001437904
```

```
## Iteration 112 tol = 0.0001432995
```

```
## Iteration 113 tol = 0.0001420838
```

```
## Iteration 114 tol = 0.0001402095
```

```
## Iteration 115 tol = 0.0001377596
```

```
## Iteration 116 tol = 0.0001348209
```

```
## Iteration 117 tol = 0.0001314743
```

```
## Iteration 118 tol = 0.0001277906
```

```
## Iteration 119 tol = 0.0001238295
```

```
## Iteration 120 tol = 0.000119642
```

```
## Iteration 121 tol = 0.0001152731
```

```
## Iteration 122 tol = 0.0001107656
```

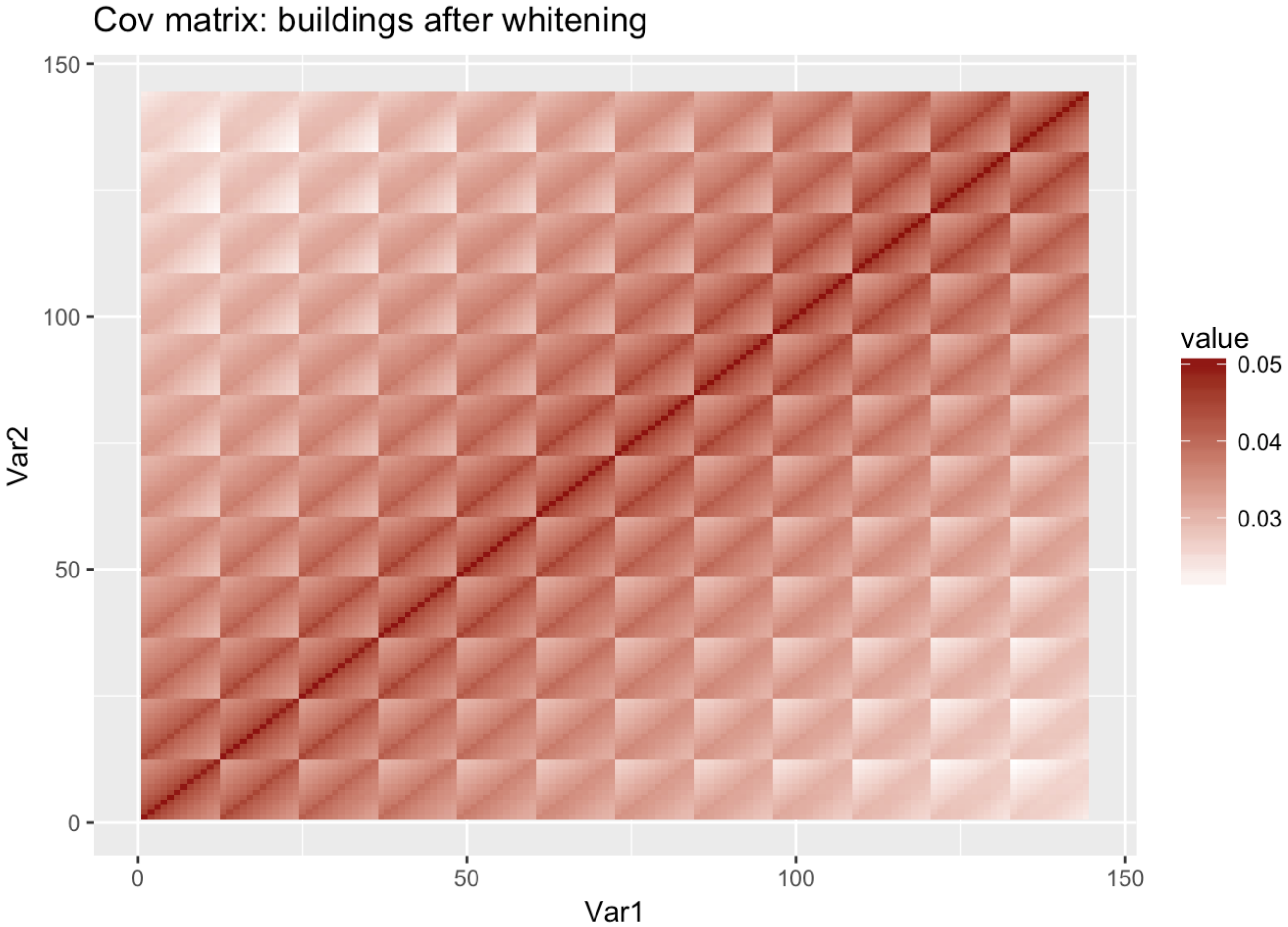
```
## Iteration 123 tol = 0.0001061615
```

```
## Iteration 124 tol = 0.0001015036
```

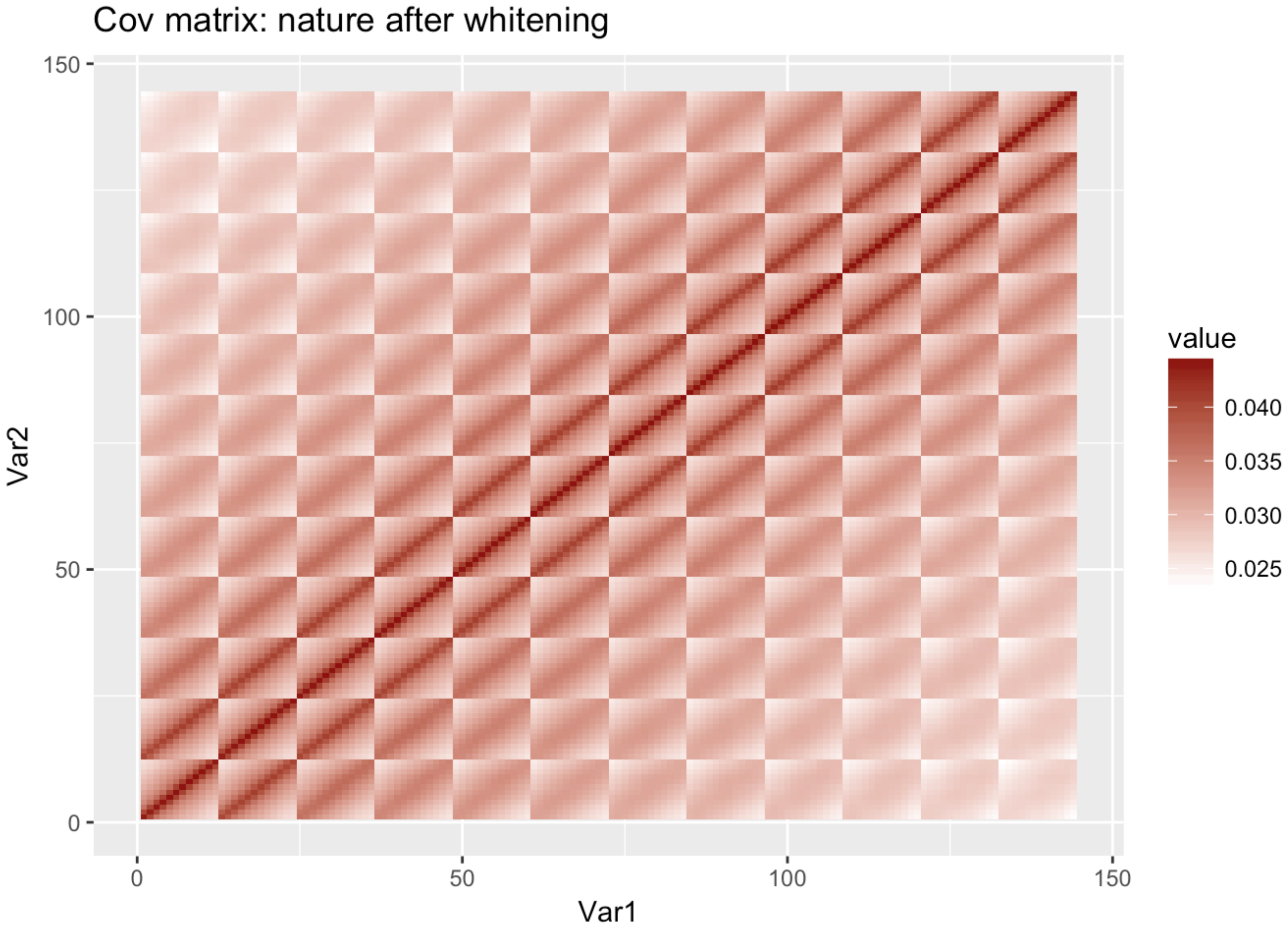
```
## Iteration 125 tol = 9.683414e-05
```

whitening: use pre-whitening matrix that projects data onto the first n.comp principal components and check

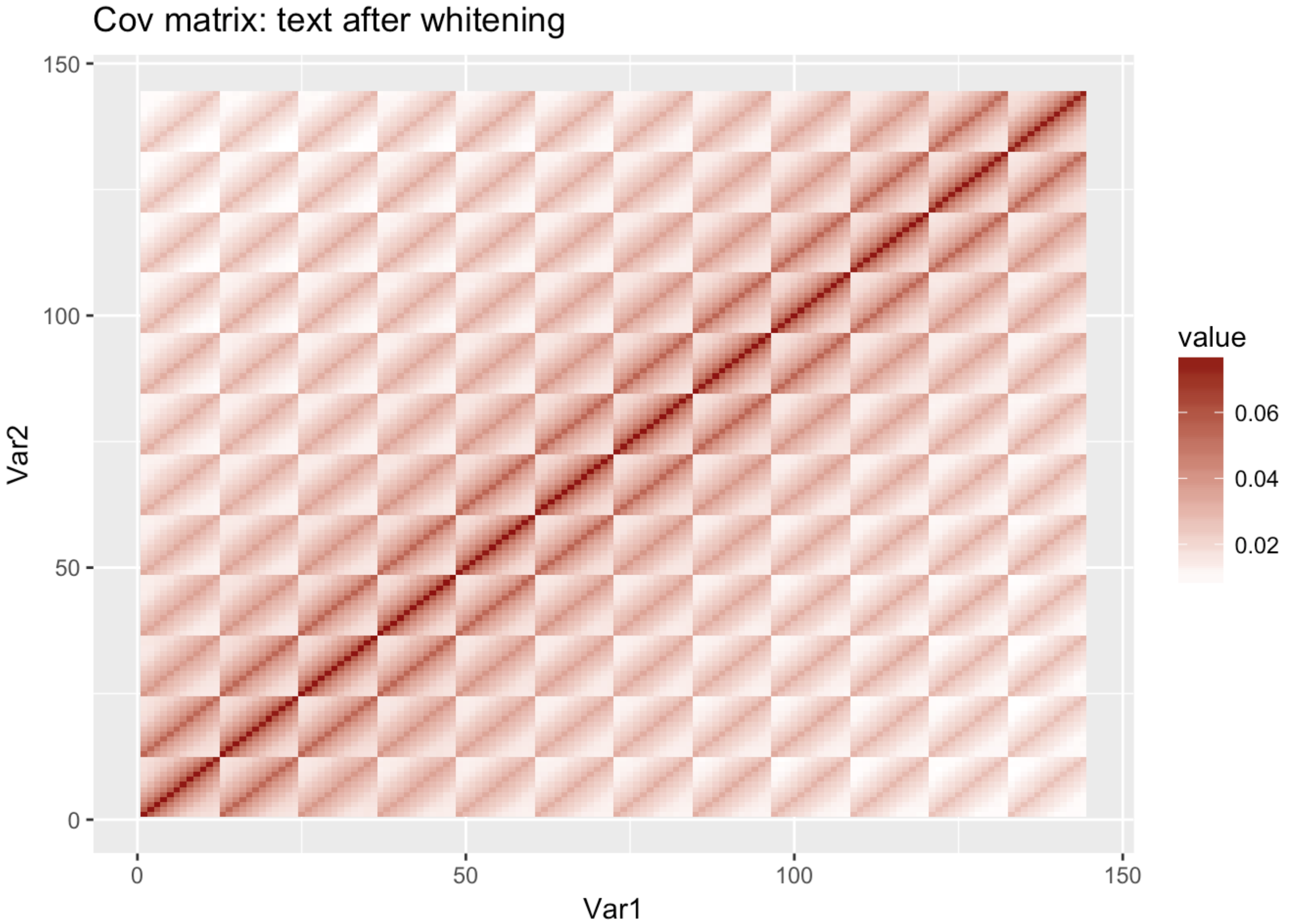
```
proj_b = matrix_b %*% a_b$K
proj_n = matrix_n %*% a_n$K
proj_t = matrix_t %*% a_t$K

# check results

# plot heatmaps
ggplot(data = melt(cov(matrix_b)), aes(x = Var1, y = Var2, fill = value)) +
    geom_tile() + scale_fill_gradient(high = "darkred", low = "white") +
    ggtitle("Cov matrix: buildings after whitening")
```




```
ggplot(data = melt(cov(matrix_n)), aes(x = Var1, y = Var2, fill = value)) +
    geom_tile() + scale_fill_gradient(high = "darkred", low = "white") +
    ggtitle("Cov matrix: nature after whitening")
```

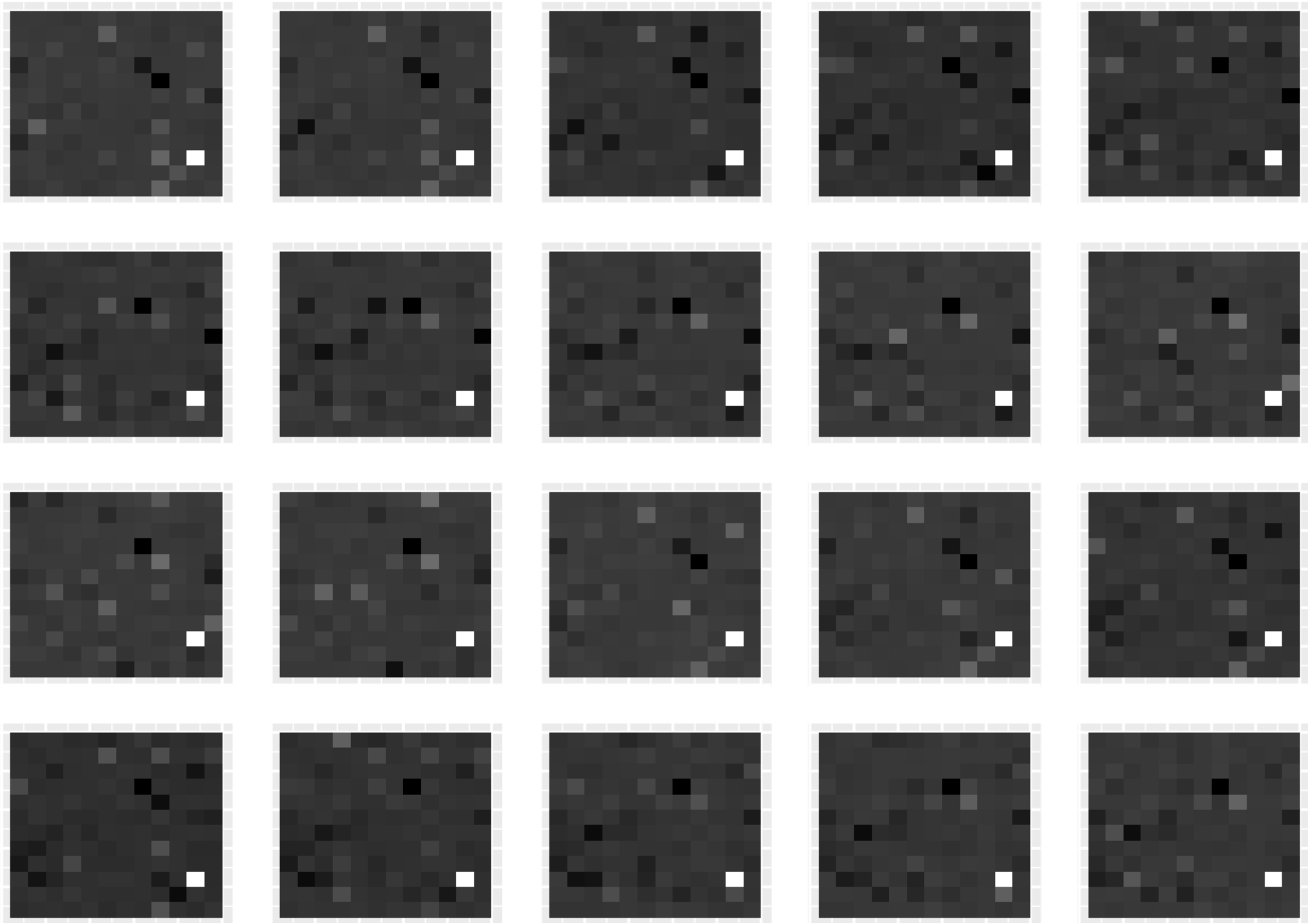



```
ggplot(data = melt(cov(matrix_t)), aes(x = Var1, y = Var2, fill = value)) +
    geom_tile() + scale_fill_gradient(high = "darkred", low = "white") +
    ggtitle("Cov matrix: text after whitening")
```

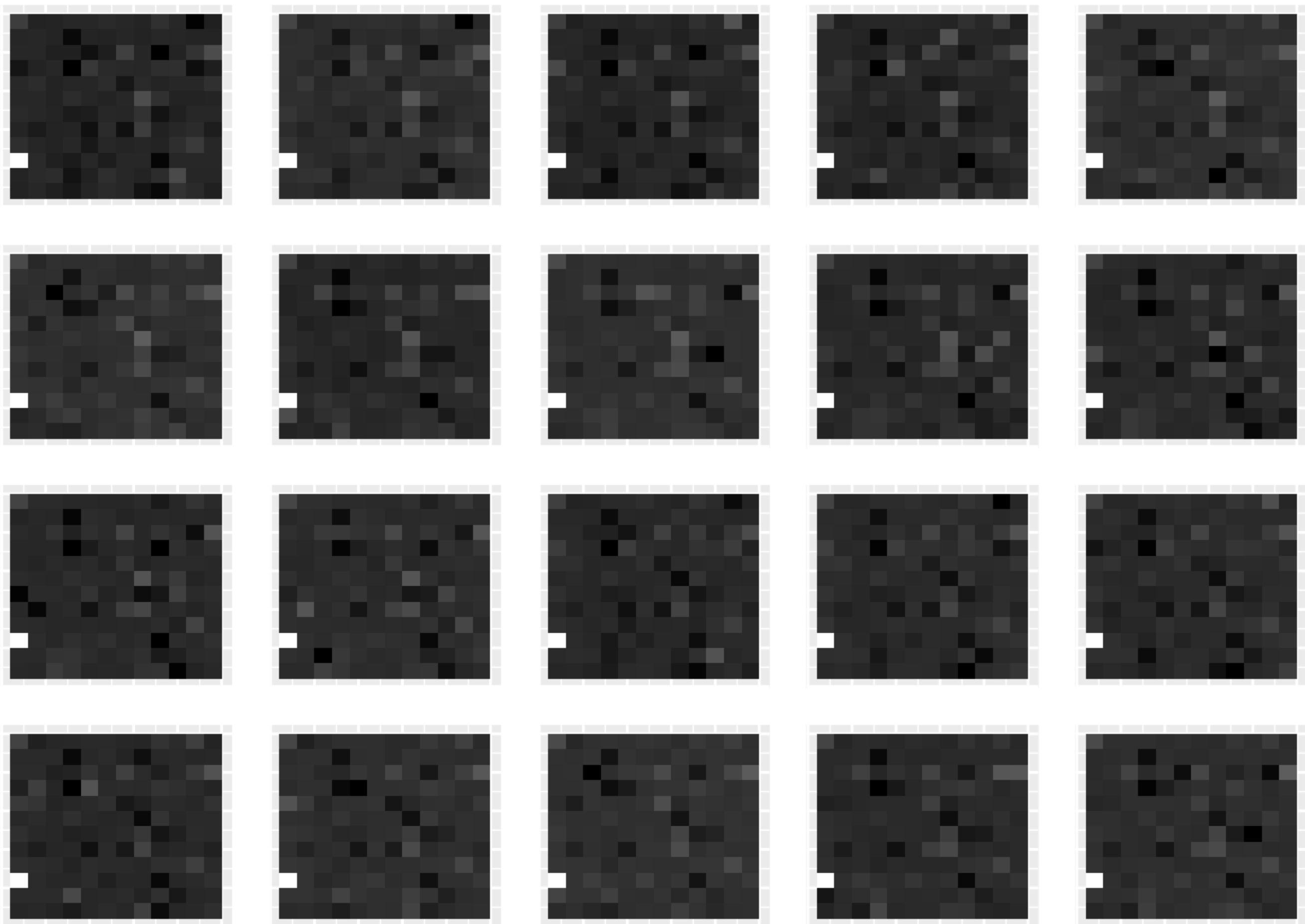


c. Show the first 20 independent features as (grayscale) image patches by rearranging the vec- √√ tors into N × N matrices and compare the results for the different categories. Order the independent features by decreasing negentropy, (such that the first feature has largest (approximated) negentropy etc).

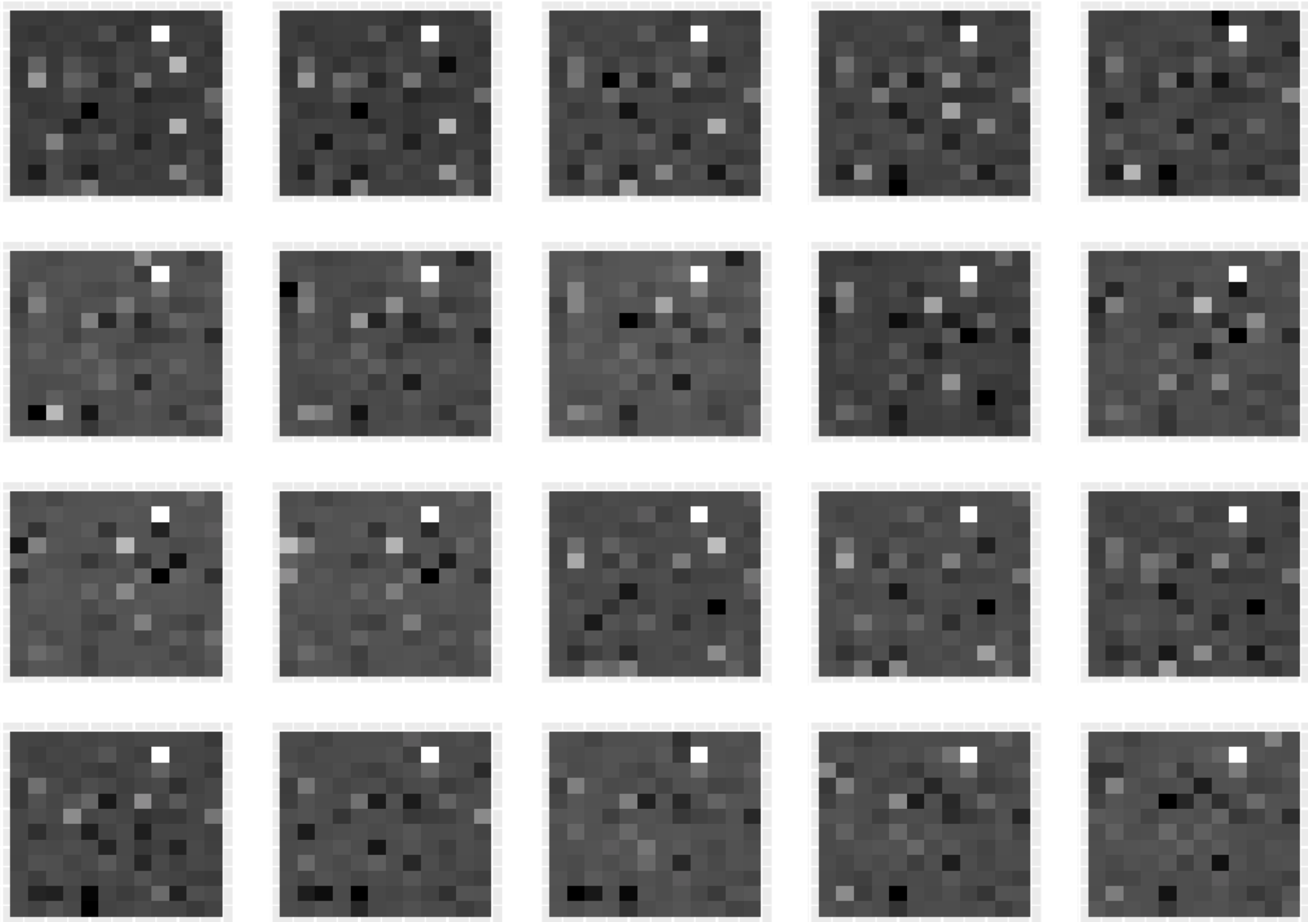
```
# the independent factors in the right direction
A_n <- a_n$A[, 1:20]
A_b <- a_b$A[, 1:20]
A_t <- a_t$A[, 1:20]

ica_patchesn <- ica_patchesb <- ica_patchest <- vector("list",
    length = 20)

for (i in 1:20) {
    ica_patchesn[[i]] <- melt(matrix(A_n[, i], N, N))
    ica_patchesb[[i]] <- melt(matrix(A_b[, i], N, N))
    ica_patchest[[i]] <- melt(matrix(A_t[, i], N, N))
}

plotlist_n <- lapply(ica_patchesn, heatmap_custom)
do.call("grid.arrange", c(plotlist_n, ncol = 5))
```



```
plotlist_b <- lapply(ica_patchesb, heatmap_custom)
do.call("grid.arrange", c(plotlist_b, ncol = 5))
```

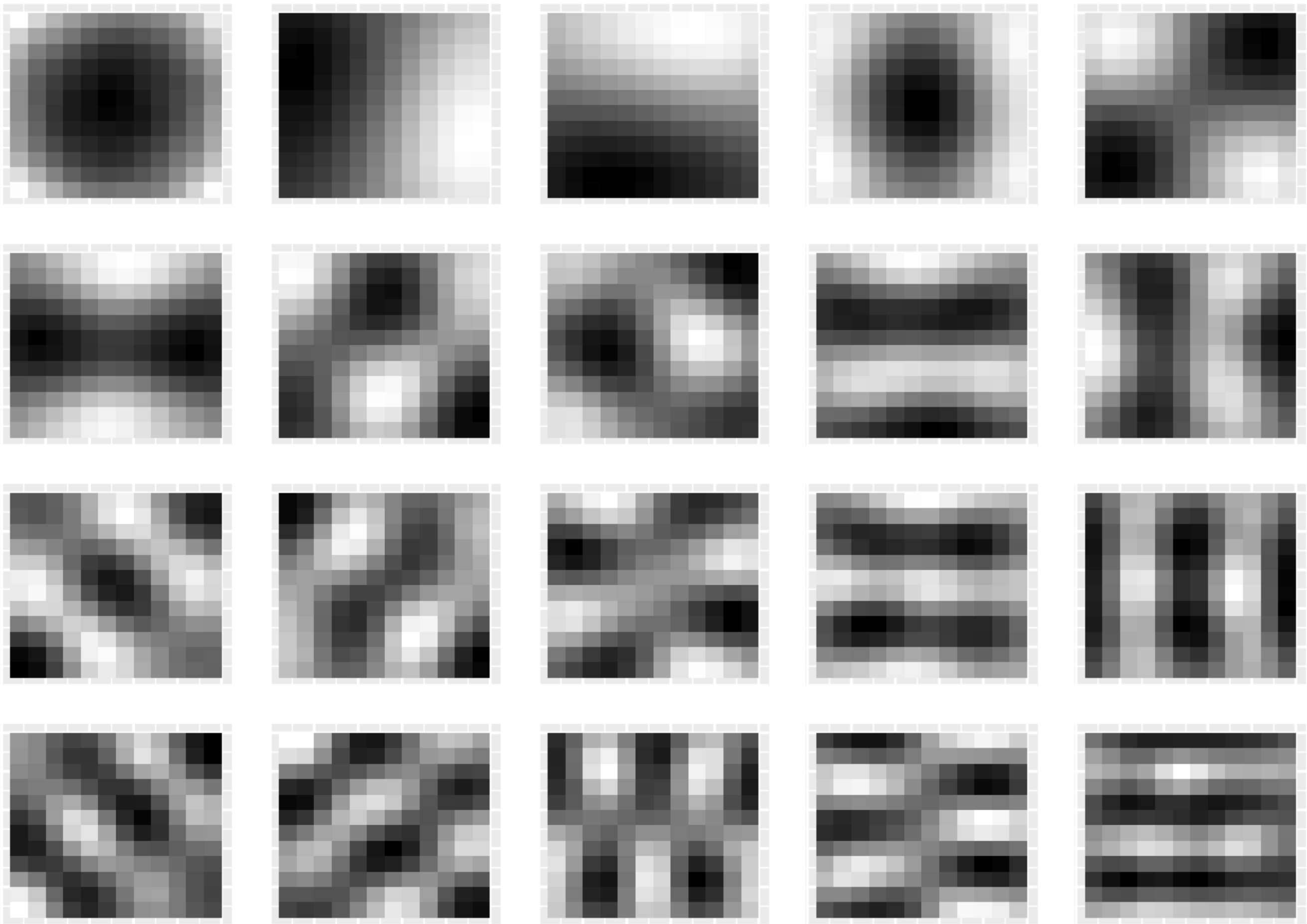


```
plotlist_t <- lapply(ica_patchest, heatmap_custom)
do.call("grid.arrange", c(plotlist_t, ncol = 5))
```

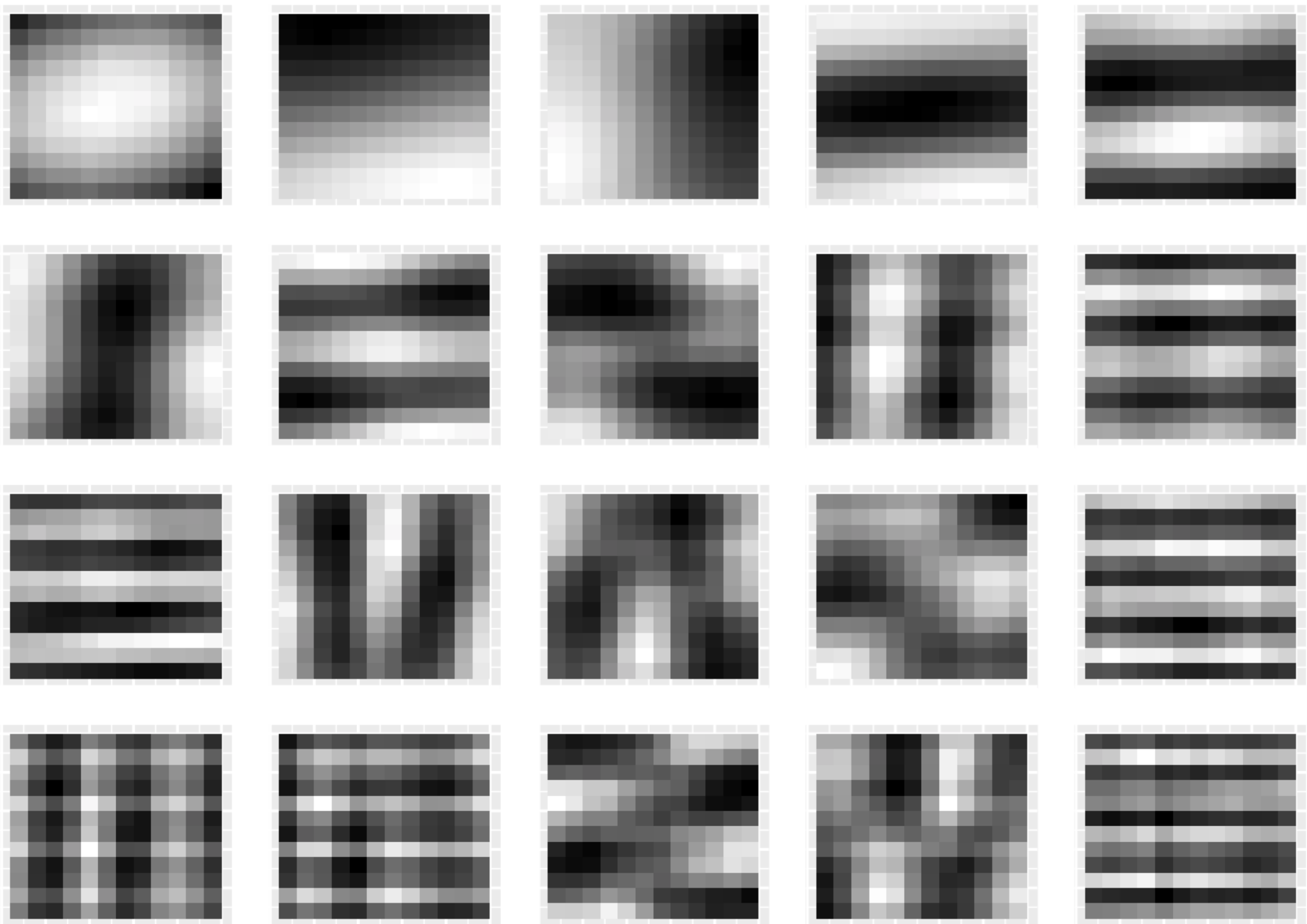


d. Perform PCA on the same set of patches, plot the the principal components (ordered by decreasing eigenvalue) as in (c) and compare them with the independent features.

```
# center
matrix_n_centered <- scale(matrix_n, center = TRUE, scale = FALSE)
matrix_b_centered <- scale(matrix_b, center = TRUE, scale = FALSE)
matrix_t_centered <- scale(matrix_t, center = TRUE, scale = FALSE)

# PCA's
pcs_n <- prcomp(matrix_n_centered)$rotation
pcs_b <- prcomp(matrix_b_centered)$rotation
pcs_t <- prcomp(matrix_t_centered)$rotation

pca_patchesn <- pca_patchesb <- pca_patchest <- vector("list",
    length = 20)

for (i in 1:20) {
    pca_patchesn[[i]] <- melt(matrix(pcs_n[, i], N, N))
    pca_patchesb[[i]] <- melt(matrix(pcs_b[, i], N, N))
    pca_patchest[[i]] <- melt(matrix(pcs_t[, i], N, N))
}


plotlist_n2 <- lapply(pca_patchesn, heatmap_custom)
do.call("grid.arrange", c(plotlist_n2, ncol = 5))
```



```
plotlist_b2 <- lapply(pca_patchesb, heatmap_custom)
do.call("grid.arrange", c(plotlist_b2, ncol = 5))
```



```
plotlist_t2 <- lapply(pca_patchest, heatmap_custom)
do.call("grid.arrange", c(plotlist_t2, ncol = 5))
```