Machine Intelligence II: Sheet 2 (PCA)

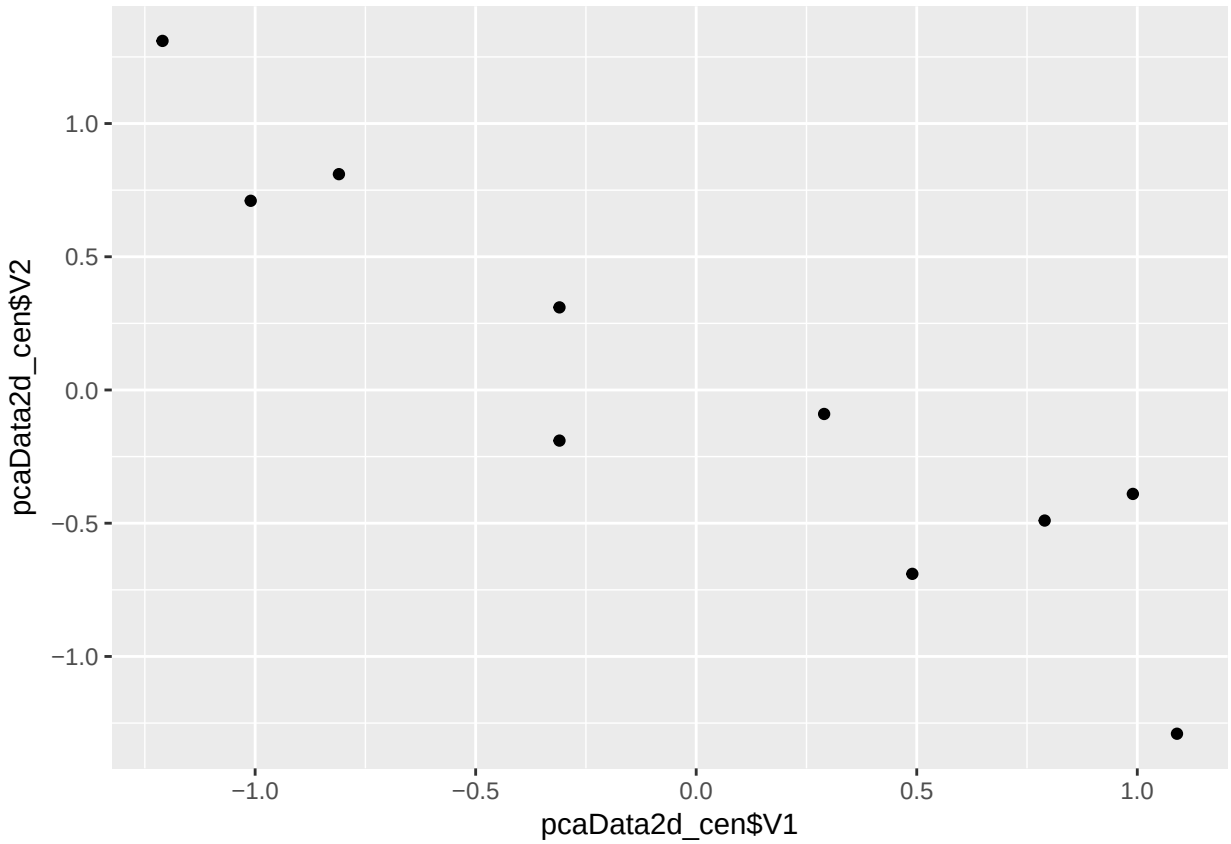
NoNames2

03.05.2017

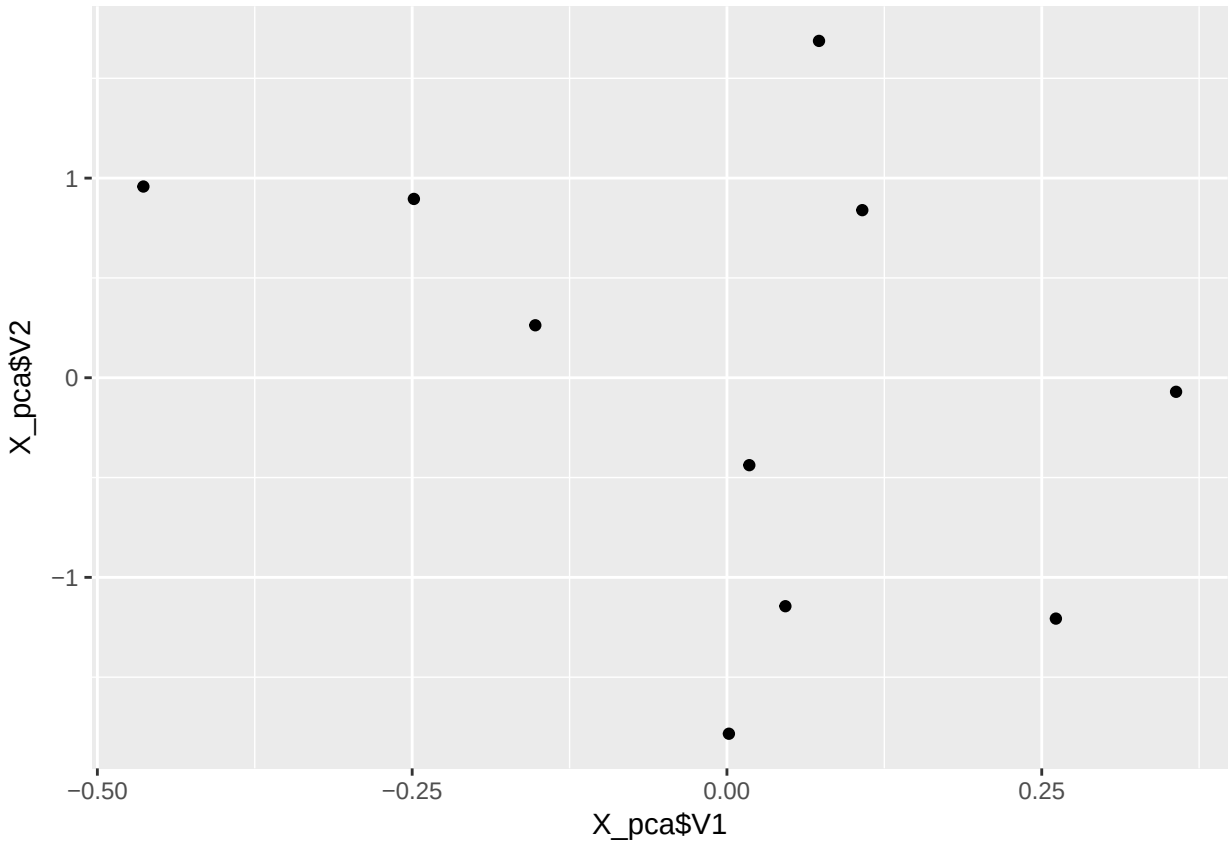
Machine Intelligence 2 Homework 1

```
# load packages
#install.packages("imager")
library(jpeg)

#install.packages("foreign")
library(foreign)

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

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

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

#install.packages("colorRamps")
library(colorRamps)

#setwd("~/Desktop/Uni/Statistik Master/courses/machine intelligence 2/data/natIMG.jpeg")
```

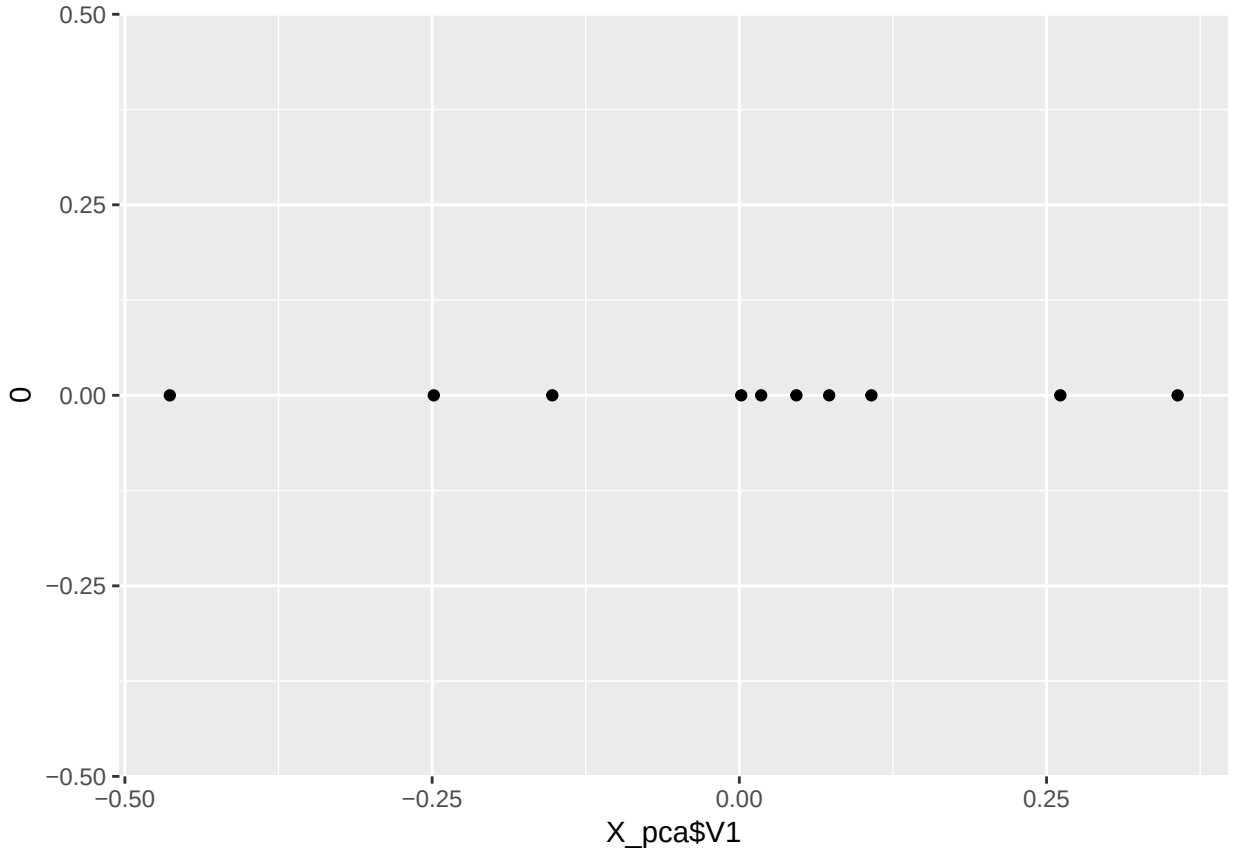
```
# 2.1 a)

# load data
pcaData2d <- read.csv("Ex2/pca-data-2d.txt", sep = "", header = FALSE)

# calcualte mean matrix
n <- nrow(pcaData2d)
M_mean <- matrix(data=1, nrow=n) %*% cbind(mean(pcaData2d[,1]),
                                           mean(pcaData2d[,2]))

# subtract mean from data set -> "Difference Matrix"
pcaData2d_cen <- pcaData2d - M_mean

# plot centered matrix
ggplot(pcaData2d_cen, aes(x=pcaData2d_cen$V1, y=pcaData2d_cen$V2)) + geom_point()
```



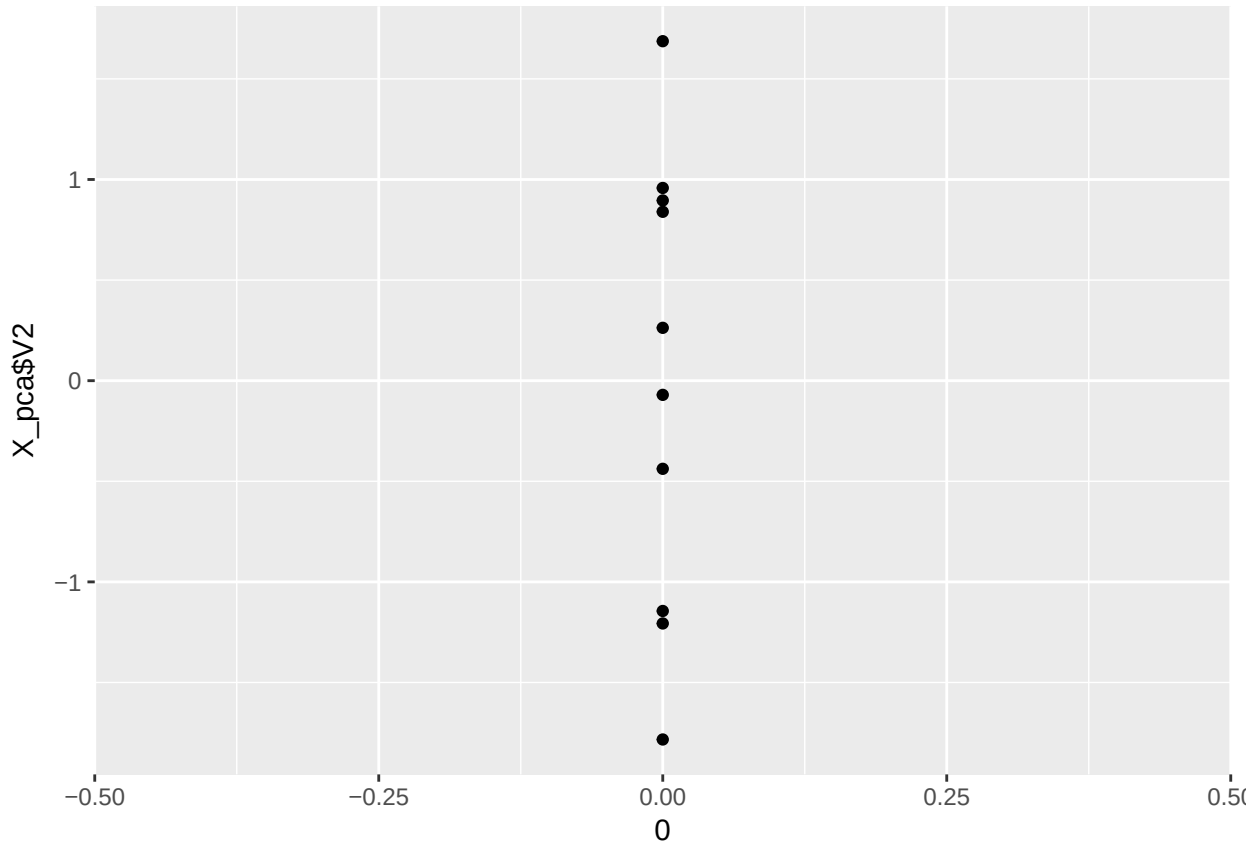

```r
# 2.1 b)

# convert df to matrix
D <- as.matrix(pcaData2d_cen)

# compute Covariance Matrix C
C <- (n-1)^-1 * t(D) %*% D

# Eigenvalues of C, Eigenvectors in P
ev <- eigen(C)
P  <- ev$vectors

# transform D
D_tr <- P %*% t(D)

# in ggplot2 plottable data frame
X_pca <- as.data.frame(t(D_tr))

# plot of the transformed set
ggplot(X_pca, aes(x=X_pca$V1, y=X_pca$V2)) + geom_point()
```



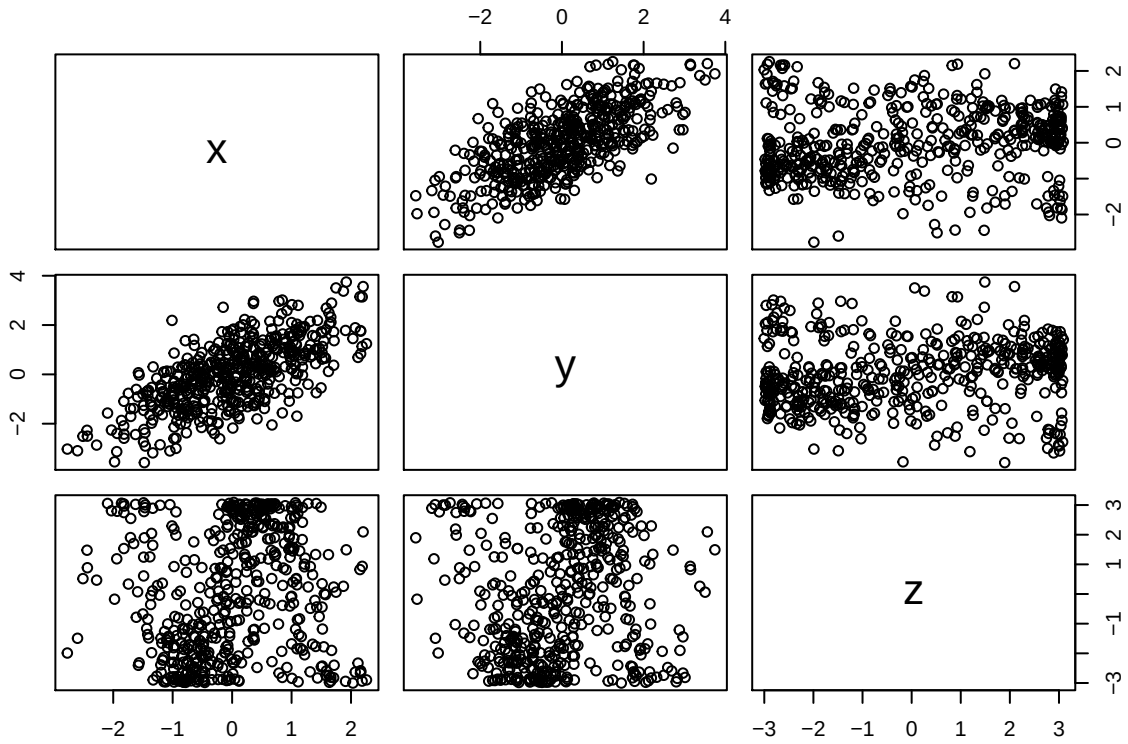

```
# 2.1 c)


# plot only PC1
ggplot(X_pca, aes(x=X_pca$V1, y=0)) + geom_point()
```



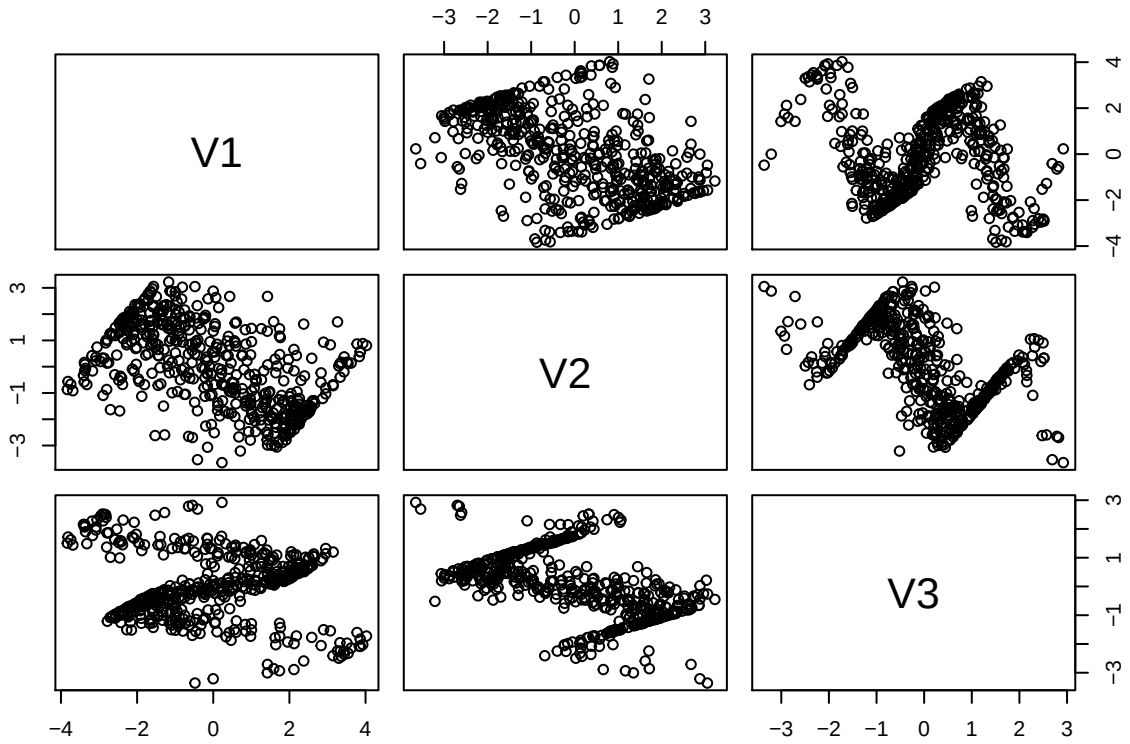

```
# plot only PC2
ggplot(X_pca, aes(x=0, y= X_pca$V2)) + geom_point()
```



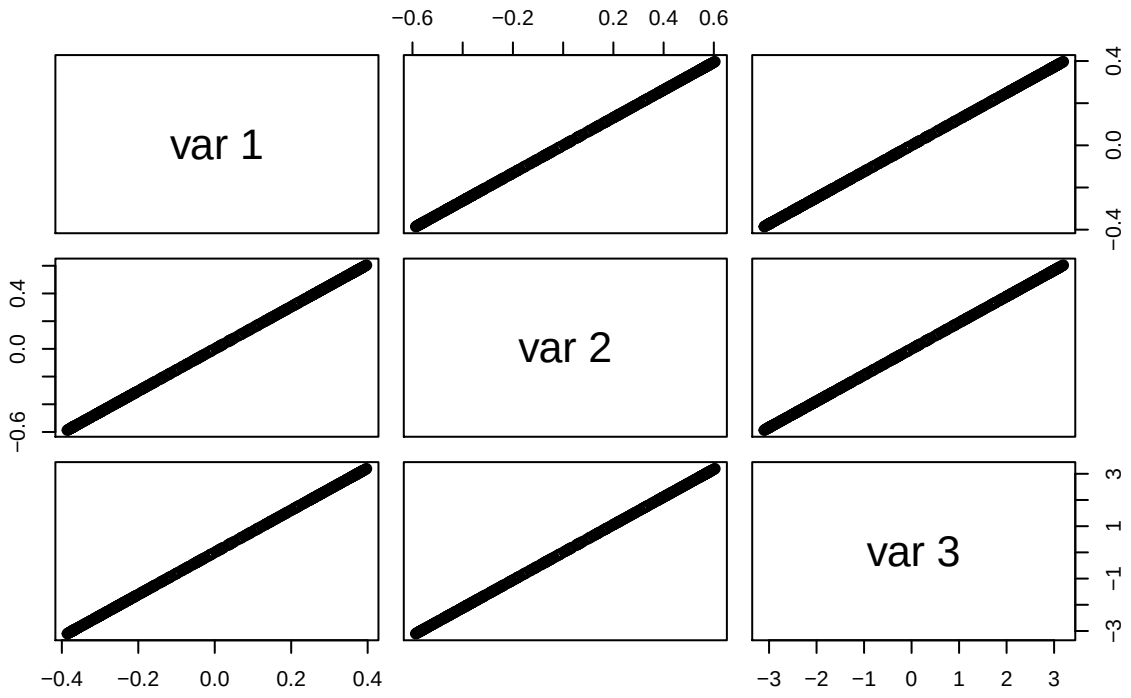

```
# 2.2 a)


# load data set
pcaData3d <- read.csv("Ex2/pca-data-3d.txt")

# centering
n_3 <- nrow(pcaData3d)
M3_mean <- matrix(data=1, nrow=n_3) %*% colMeans(pcaData3d)

# subtract mean from data set -> "Difference Matrix"
D3 <- pcaData3d - M3_mean

# scatterplot matrix
pairs(D3)
```



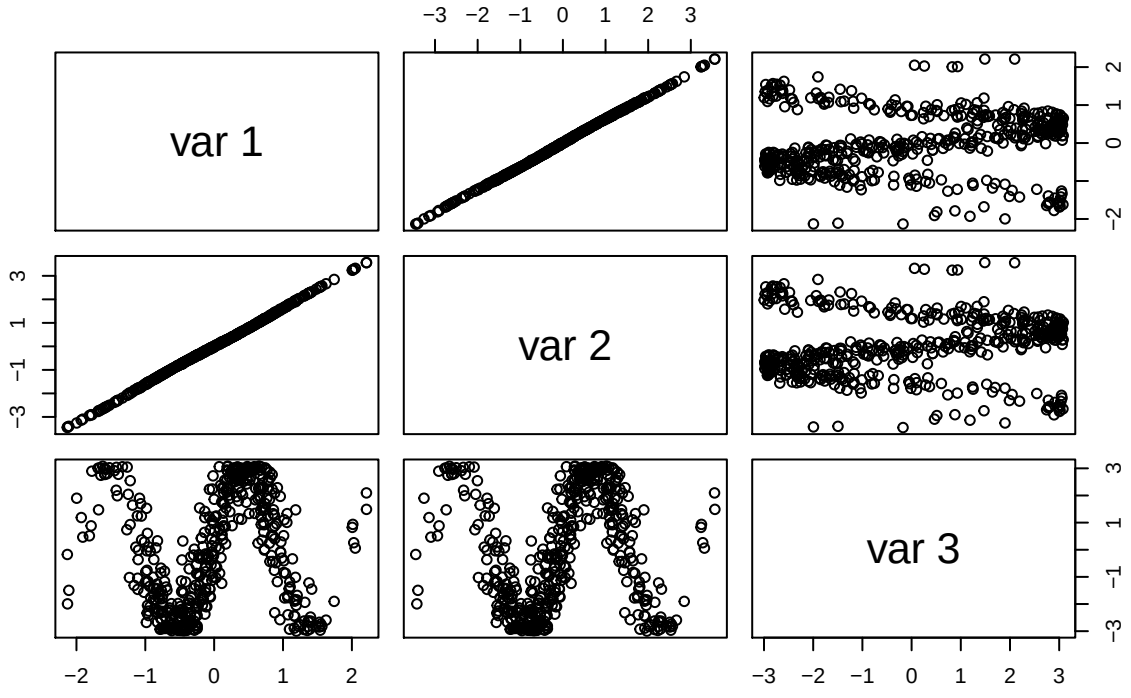

```r
# 2.2 b)

# convert df to matrix
D3 <- as.matrix(D3)

# compute Covariance Matrix C
C3 <- (n_3-1)^-1 * t(D3) %*% D3

# Eigenvalues of C3, Eigenvectors in P3
ev3 <- eigen(C3)
P3  <- ev3$vectors

# transform D3
D3 <- P3 %*% t(D3)
X_3pca <- as.data.frame(t(D3))
pairs(X_3pca)
```



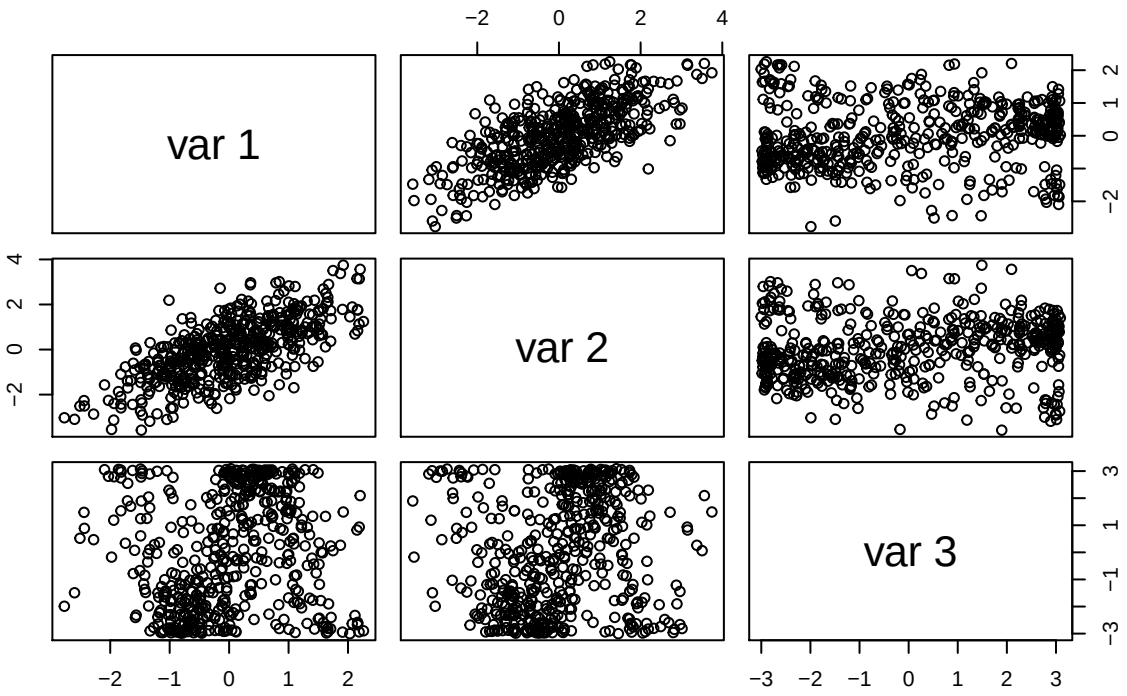

```r
# 2.2 c)


D3 <- pcaData3d - M3_mean

# convert df to matrix
D3 <- as.matrix(D3)

# compute Covariance Matrix C
C3 <- (n_3-1)^-1 * t(D3) %*% D3

eigen_vals <- eigen(C3)$values
eigen_vecs <- eigen(C3)$vectors

# compute pc scores
#pc_scores <- prcomp(D3)

pc_scores_1 <- D3 %*% eigen_vecs[, 1]
pc_scores_1_2 <- D3 %*% eigen_vecs[, 1:2]
pc_scores_1_2_3 <- D3 %*% eigen_vecs[, 1:3]


pca_reconstruction_1 <- pc_scores_1 %*% t(eigen_vecs[, 1])
pairs(pca_reconstruction_1, main = "PCs for reconstruction: 1")
```




```
pca_reconstruction_1_2 <- pc_scores_1_2 %*% t(eigen_vecs[, 1:2])
pairs(pca_reconstruction_1_2, main = "PCs for reconstruction: 1, 2")
```




```
pca_reconstruction_1_2_3 <- pc_scores_1_2_3 %*% t(eigen_vecs[, 1:3])
pairs(pca_reconstruction_1_2_3, main = "PCs for reconstruction: 1, 2, 3")
```




```
# for reconstruction of uncentered data, add mean veactor to each column (add M3)
```

Interpretation Obviously, the first two PCs are useful here. Plotting only the first PC alone does not yield a lot of information (only about the spread in this direction). When taking into account the first two PCs, one can see the (possibly) sinusoidal trend in the data. Using all three PCs yields the original centered COV matrix that was plotted in 2.2 a). This is oubvious because due to the orthonormality of the eigenvectors eigen_vecs %*% t(eigen_vecs) will yield an identity matrix, hence, when multiplying this matrix with the centered COV matrix D3, the COV matrix remains unchanged. However, one could argue that including the third PC adds too much noise to the data and the trends are harder to detect in the scatter plot matrix. Therefore, the first two PCs should be chosen. Looking at the eigenvalues supports this.

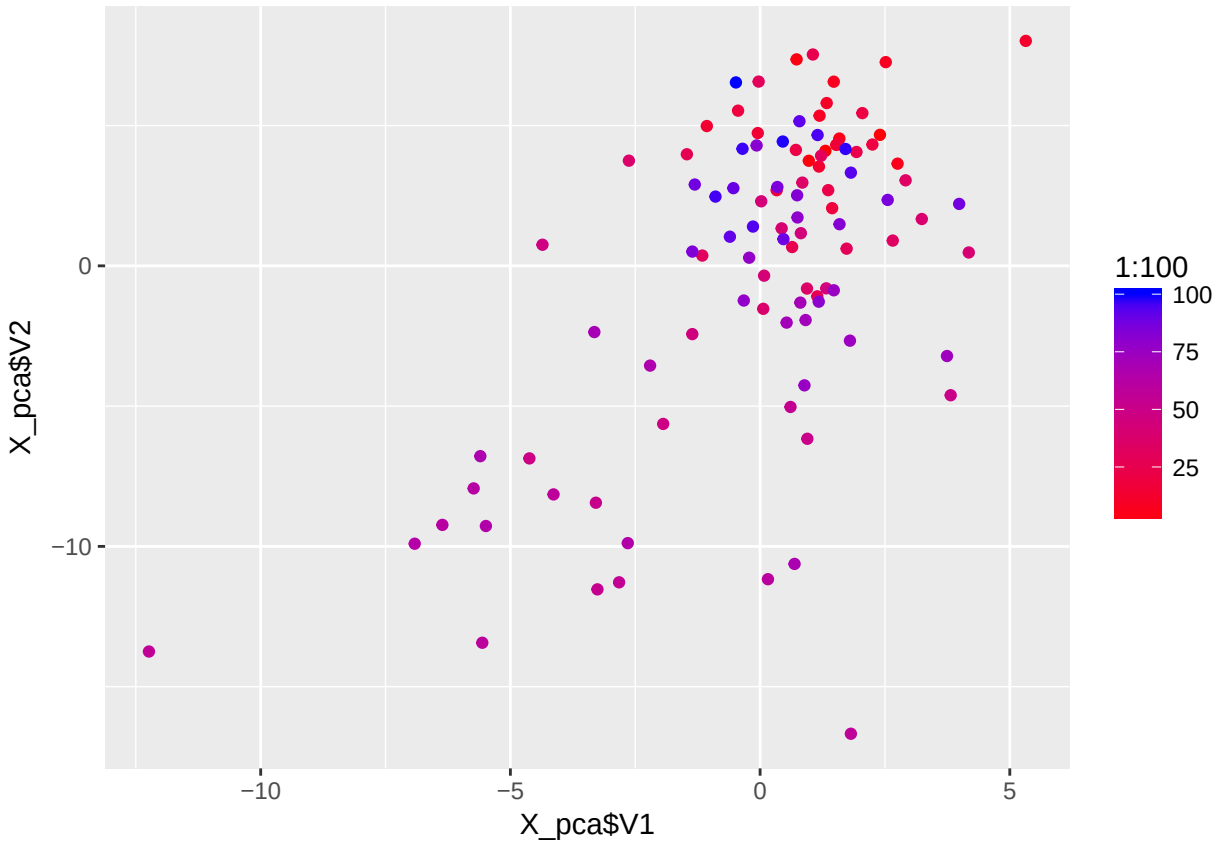
```
# 2.3 a)

# load data
expDat <- read.csv("Ex2/expDat.txt", sep = ",", header = TRUE)

# calculate mean matrix
n <- nrow(expDat[2:ncol(expDat)])
M_mean <- matrix(data=1, nrow=n) %*% colMeans(expDat[2:ncol(expDat)])
# subtract mean from data set
expDat_cen <- expDat[2:ncol(expDat)] - M_mean

# convert df to matrix
D <- as.matrix(expDat_cen)
# compute Covariance Matrix C
C <- (n-1)^-1 * t(D) %*% D
# Eigenvalues of C, Eigenvectors in P
ev <- eigen(C)
P  <- ev$vectors
```

```r
# print PCs
print(P)
```

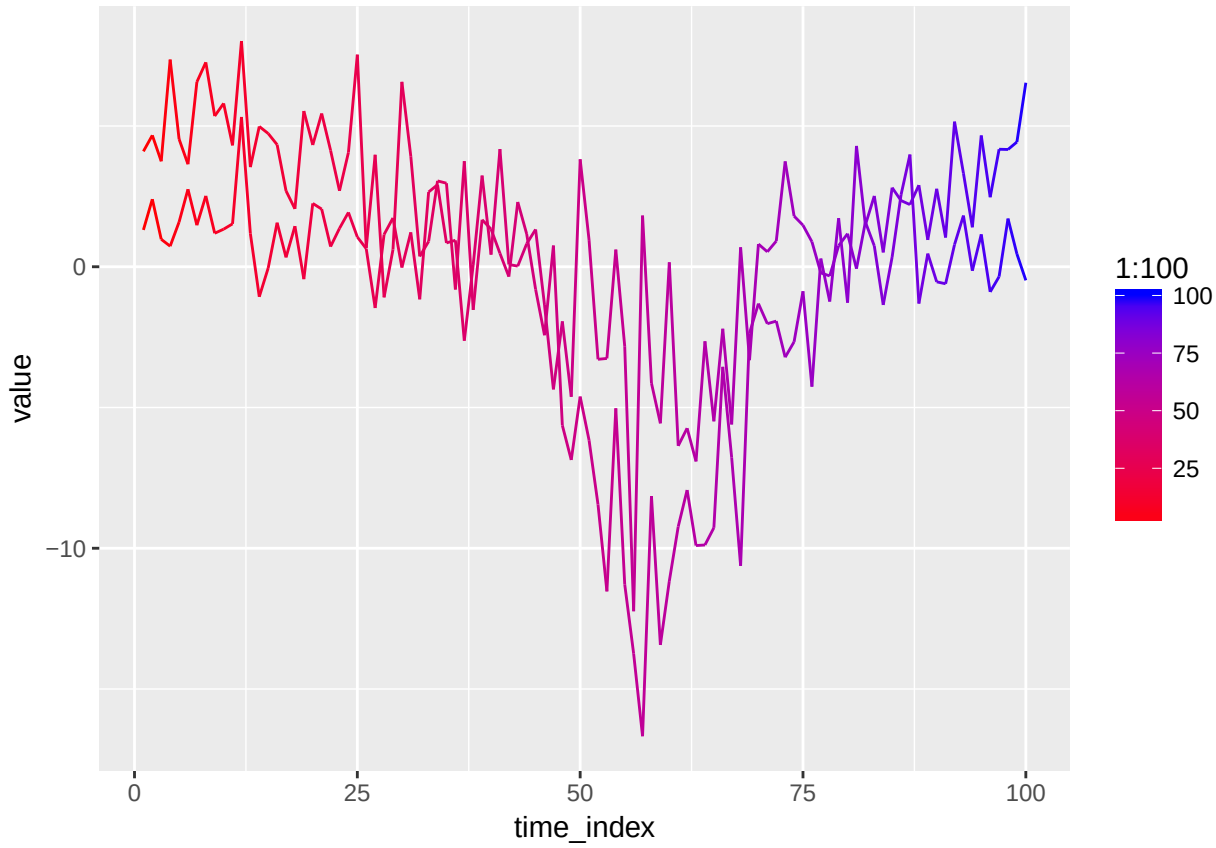
```
##              [,1]       [,2]         [,3]        [,4]        [,5]
##  [1,] -0.2317007 -0.2183058 -0.104406940  0.12313067 -0.05816656
##  [2,] -0.2036073 -0.1992018 -0.229536067 -0.31129876  0.26325604
##  [3,] -0.2500715 -0.2265650 -0.374488207 -0.06571507 -0.31672176
##  [4,] -0.2368622  0.2568714 -0.025102114  0.17449749  0.33486862
##  [5,] -0.2404493  0.1581708 -0.240553124 -0.45952533 -0.03231931
##  [6,] -0.2044047  0.2208849 -0.172278294 -0.02449325  0.13124224
##  [7,] -0.2405699 -0.1948711  0.103867339 -0.18318469  0.18237494
##  [8,] -0.2458614 -0.1618543  0.163636641 -0.08022182 -0.07824362
##  [9,] -0.1773770  0.2263329 -0.281368594  0.57605734 -0.12877602
## [10,] -0.1975186 -0.1274600  0.003365382  0.07196717  0.12468248
## [11,] -0.2335105 -0.2675393  0.415634996  0.21872101 -0.15874409
## [12,] -0.2592469 -0.1754496  0.161449401  0.26505012  0.12047236
## [13,] -0.2047987  0.2526093  0.061289268 -0.01330226  0.47409824
## [14,] -0.2092126  0.2547959 -0.076893580  0.16901742  0.05654813
## [15,] -0.2193636  0.2469967  0.226061570 -0.19404756 -0.42518262
## [16,] -0.1823281  0.2462823  0.238202082 -0.27542571 -0.01711058
## [17,] -0.2213010  0.2153870 -0.065437365  0.05010810 -0.15991627
## [18,] -0.2183011  0.2433842  0.236808160 -0.02942508 -0.31392855
## [19,] -0.2555495 -0.2849078 -0.338726258  0.01717171 -0.12004071
## [20,] -0.2169721 -0.2243653  0.315948092  0.01917147  0.19441001
##              [,6]         [,7]         [,8]        [,9]       [,10]
##  [1,] -0.41701805 -0.079755964 -0.122706458  0.21365060  0.48225894
##  [2,]  0.15196683 -0.166861444  0.210969621  0.02500722 -0.16714534
##  [3,]  0.27965257 -0.114889568 -0.058848387  0.01332672 -0.16350732
##  [4,]  0.17439629 -0.047887771 -0.184322378 -0.56762152  0.38835791
##  [5,] -0.13920332  0.122886480 -0.273905831  0.04766474 -0.08215378
##  [6,]  0.08235568 -0.016797923  0.425210440  0.27634383  0.37328985
##  [7,] -0.16507066  0.059014386  0.246297568  0.16692871  0.05966176
##  [8,] -0.23240601  0.125744688 -0.196736930 -0.06090582 -0.04672258
##  [9,]  0.02341982  0.044715545 -0.251951038  0.40568305 -0.13106109
## [10,]  0.33329940 -0.006701255 -0.011543046 -0.19803812 -0.12466131
## [11,]  0.35421132 -0.097437009  0.209857835  0.16022038 -0.08089921
## [12,] -0.49519020  0.074938497  0.242622153 -0.22701143 -0.25596695
## [13,] -0.02473018  0.148672811 -0.151908268  0.23908082 -0.41758305
## [14,] -0.01224016  0.056466783  0.199304862 -0.12212759 -0.17651475
## [15,] -0.13557693 -0.164717585 -0.153865142 -0.18665773 -0.15638906
## [16,]  0.09554217 -0.063825159  0.003042302  0.23679590  0.16507486
## [17,] -0.11832637 -0.642906097  0.175254378 -0.11021118 -0.04633290
## [18,]  0.16145203  0.531384602  0.179697415 -0.02530565  0.13155488
## [19,]  0.05956900  0.341343236 -0.015701967 -0.21166181  0.08222769
## [20,]  0.18099270 -0.172513448 -0.481194715  0.13987611  0.14784724
##             [,11]        [,12]       [,13]        [,14]       [,15]
##  [1,] -0.12841935  0.009884036 -0.07541543 -0.158702647  0.09538108
##  [2,] -0.27160826 -0.247943811 -0.01708109 -0.058458225  0.50524837
##  [3,]  0.04738420  0.318385611 -0.33235626  0.333338825  0.08876457
##  [4,] -0.03355926 -0.265485792 -0.19559042  0.168891339  0.05252197
##  [5,]  0.46780920 -0.249250614  0.02553488 -0.044642889 -0.01667929
##  [6,]  0.28687406  0.286826704 -0.09228037 -0.075240776 -0.24225429
##  [7,]  0.14326849 -0.268679279 -0.09181893  0.032420875 -0.08967208
```

```
##  [8,] -0.12243385  0.199808628  0.35861472  0.022548038  0.09161359
##  [9,] -0.07413833 -0.248253096 -0.06419500 -0.143818217  0.16307458
## [10,]  0.07389732  0.248500949  0.01034117 -0.788939703 -0.03339461
## [11,]  0.09829259 -0.392231722 -0.01814846  0.134958925 -0.21420827
## [12,]  0.06038913  0.128140380 -0.28408930  0.004774420  0.11459405
## [13,] -0.12229886  0.019158361  0.07795286  0.070422294 -0.25970137
## [14,]  0.01912344  0.317903444  0.07034506  0.299565735  0.04572182
## [15,] -0.04035176 -0.072934218 -0.38026652 -0.176110226 -0.21581065
## [16,] -0.60683532  0.158646350 -0.14623550  0.002367187 -0.04181756
## [17,]  0.03207794 -0.060009537  0.54936974 -0.004888801 -0.01417642
## [18,]  0.09060664 -0.018501548  0.21182332 -0.021283457  0.43935330
## [19,] -0.29542567 -0.089550629  0.27728607  0.058486059 -0.49115500
## [20,]  0.25244268  0.269329025  0.11515716  0.177708388  0.10266500
##             [,16]        [,17]       [,18]       [,19]       [,20]
##  [1,]  0.48243931  0.136201260 -0.27876938 -0.05387713  0.08482620
##  [2,] -0.13433412  0.019212268 -0.28975677 -0.06782089 -0.27340782
##  [3,]  0.19948029 -0.252567025  0.15622498 -0.07127451  0.23095758
##  [4,]  0.04181516 -0.089418084  0.13563406 -0.15429632  0.04362744
##  [5,] -0.12010616  0.387985818  0.02624764 -0.08720365  0.26724824
##  [6,] -0.28089553 -0.155426644 -0.12478545 -0.26861650 -0.17357503
##  [7,]  0.19953578 -0.201137295  0.55371922  0.40523332 -0.20591349
##  [8,] -0.07786338 -0.082496819  0.36112711 -0.58937230 -0.27665525
##  [9,] -0.24274930 -0.018469488  0.20017194  0.06147958 -0.14245425
## [10,]  0.15232997  0.108864557  0.15074337  0.05187791  0.08152009
## [11,]  0.07124244  0.267643192 -0.09575608 -0.30632164  0.09614173
## [12,] -0.37223125 -0.006914841 -0.08892737  0.04489347  0.31933364
## [13,]  0.32741798 -0.306957538 -0.24342906 -0.10574649  0.15015560
## [14,]  0.26866911  0.588973137  0.02562667  0.16703666 -0.34447252
## [15,]  0.01536157 -0.151038605 -0.23241725  0.06018054 -0.42780852
## [16,] -0.20984175  0.238830819  0.23035733  0.09873752  0.31021238
## [17,]  0.01051700 -0.163644626  0.02531709  0.13506865  0.21793281
## [18,]  0.10864709 -0.234731394 -0.16117843  0.14578169  0.13379092
## [19,] -0.21320168 -0.003315672 -0.17705218  0.23871744 -0.03966174
## [20,] -0.24833968 -0.024783821 -0.19342803  0.34788142 -0.12936741
```

```r
# 2.3 b)

# transform D
D_tr <- P %*% t(D)
# in ggplot2 plottable data frame
X_pca <- as.data.frame(t(D_tr))

#scatter plot
ggplot(X_pca, aes(x=X_pca$V1, y=X_pca$V2)) + scale_colour_gradient(low = "red", high = "blue") + geom_p
```

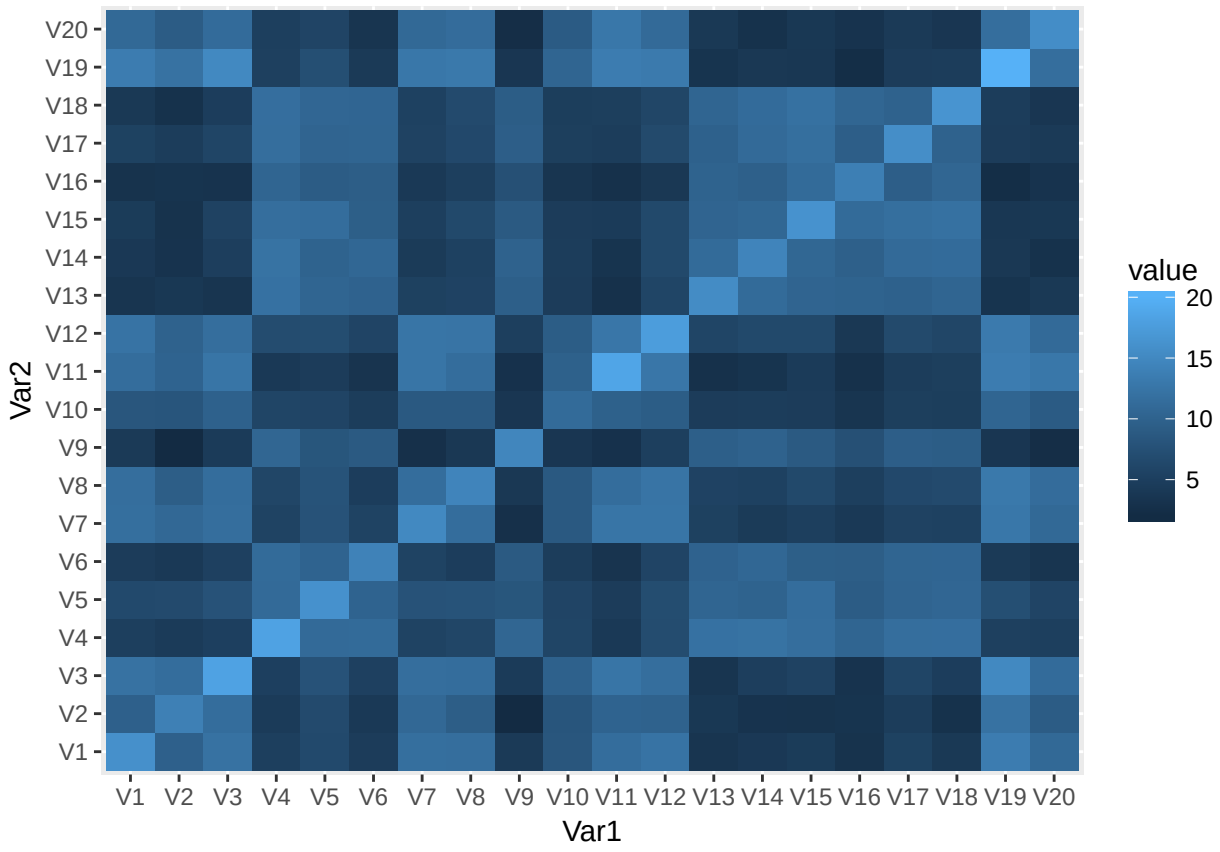



```
#line plot
df <- data.frame(expDat[,1], X_pca$V1, X_pca$V2)
colnames(df)[1] <- "time_index"

ggplot(df, aes(x=time_index, y=value)) +
    scale_colour_gradient(low = "red", high = "blue") +
    geom_line(aes(y=X_pca$V1, colour=1:100)) +
    geom_line(aes(y=X_pca$V2,colour=1:100))
```

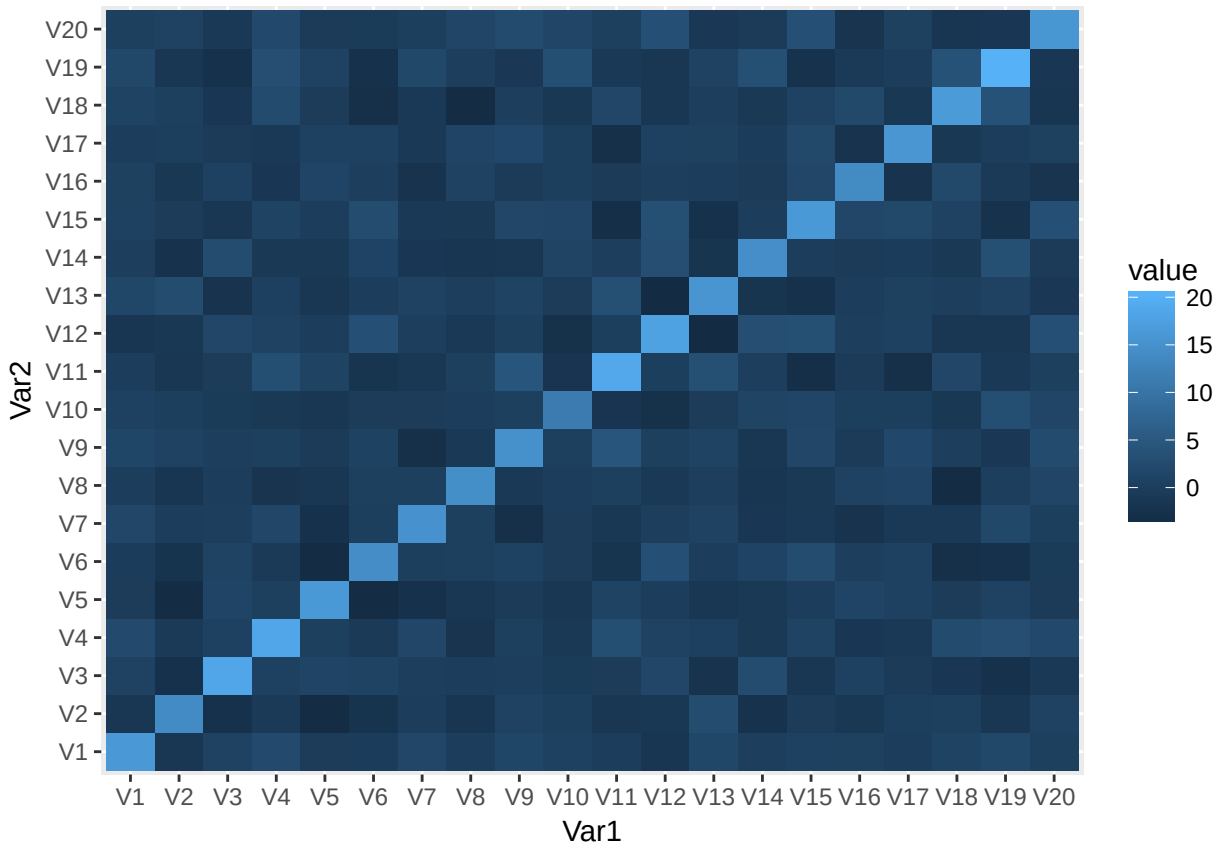



```r
# 2.3 c)

newDat <- apply(expDat[2:21], 2, sample)
```

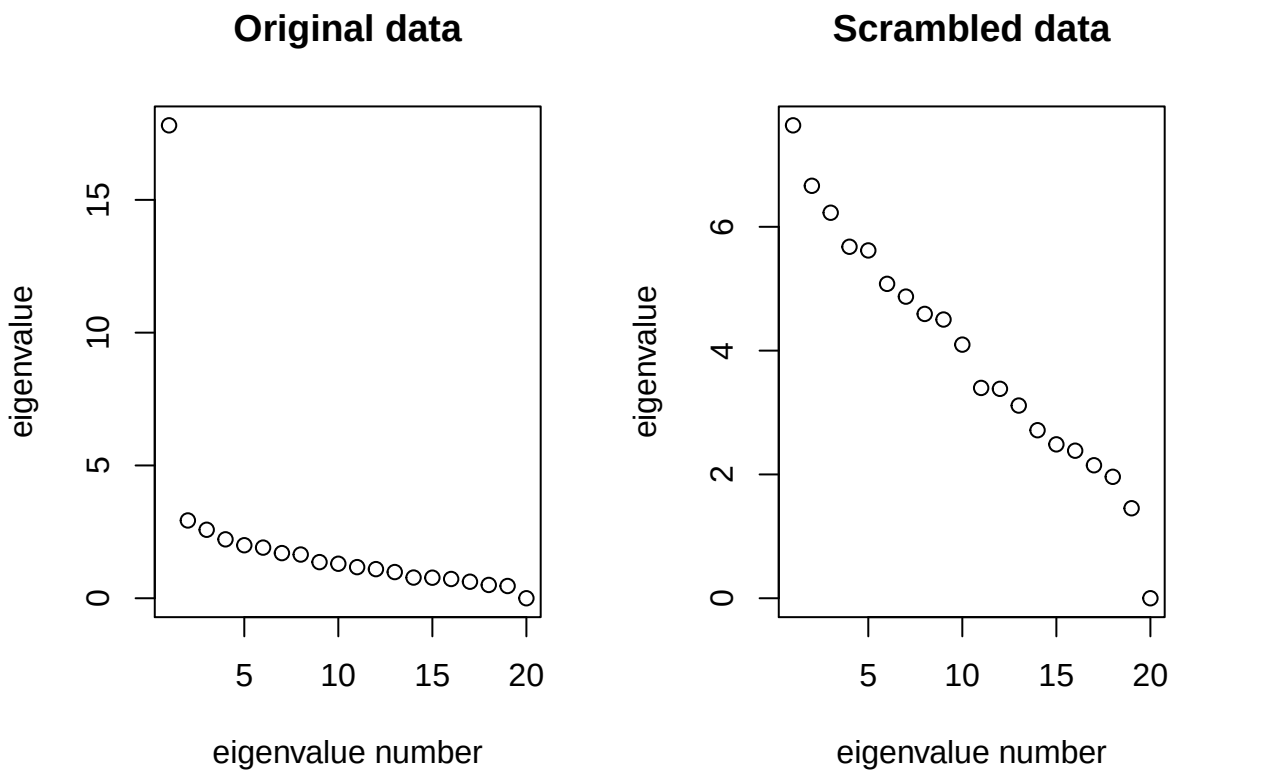
```r
# 2.3 d)

# shuffled dataset
# calculate mean matrix of new data set
n_new <- nrow(newDat)
M_mean_new <- matrix(data=1, nrow=n_new) %*% colMeans(newDat)
# subtract new mean from shuffled data set
Dat_cen_new <- newDat - M_mean_new
# compute Covariance Matrix C_new
C_new <- (n_new-1)^-1 * t(Dat_cen_new) %*% Dat_cen_new



# convert cov matrizes to data frame for easier plotting
C_melted <- melt(C)
C_melted_new <- melt(C_new)

# plot heatmaps
ggplot(data = C_melted, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```




```
ggplot(data = C_melted_new, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```




```
# scree plots

eigenvalues_original <- eigen(scale(C, center = TRUE))$values
eigenvalues_scrambled <- eigen(scale(C_new, center = TRUE))$values

par(mfrow = c(1, 2))
plot(1: length(eigenvalues_original), eigenvalues_original,
     main = "Original data", xlab = "eigenvalue number", ylab = "eigenvalue")
plot(1: length(eigenvalues_scrambled), eigenvalues_scrambled,
     main = "Scrambled data", xlab = "eigenvalue number", ylab = "eigenvalue")
```




```
dev.off()
```

```
## null device
##           1
```

Interpretation Due to the scrambling of the data, the PC structure is a lot less clear in the second plot, i.e. the decline of the eigenvalues is a lot more shallow compared to the plot of the original data, where the first eigenvalue is by far the largest. This is plausibe, since the data got randomized, i.e.the resulting covariance matrix is random as well.

2.3 e)

Shuffeling the data rowwise in the same sequence for all columns does not affect the resulting covariance matrix. This is easy to see without programming, since both, variances and covariances are based on element-wise sums of squared errors. For summing over a number of elements, the order is not relevant. Therefore, the variances remain unchanged even if the data was not shuffeled in the same sequence since they do not depend on the values of other variables. However, for the covariances this is important because otherwise the relationship between the features would change. But here, since the rows are shuffeled in the same way for all columns (i.e. features), the relationships (covariances) between the features remain the same as well.

helper function for patches

```
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]))
    }
}
```

```
# 2.4 a) buildings

path <- "/Users/Niko/Desktop/Uni/Statistik Master/courses/machine intelligence 2/mi2_homework/mi2_homew

f <- file.path(path, c("b1.jpg","b2.jpg", "b3.jpg", "b4.jpg", "b5.jpg", "b6.jpg", "b7.jpg", "b8.jpg", "

d <- lapply(f, readJPEG)


total_matrix_b <- matrix(nrow = 0, ncol = 256)

for(i in 1:length(d)){
    #set.seed(123)
    tmp1 <- t(replicate(500, get_patch(d[[i]], 16, 16, return_vector = TRUE), simplify = "vector"))
    #tmp2 <- replicate(10, get_patch(d[[i]], 16, 16, return_vector = TRUE))
    total_matrix_b <- rbind(total_matrix_b, tmp1)
}

# Show dimensions of matrix for building patches
dim(total_matrix_b)
```

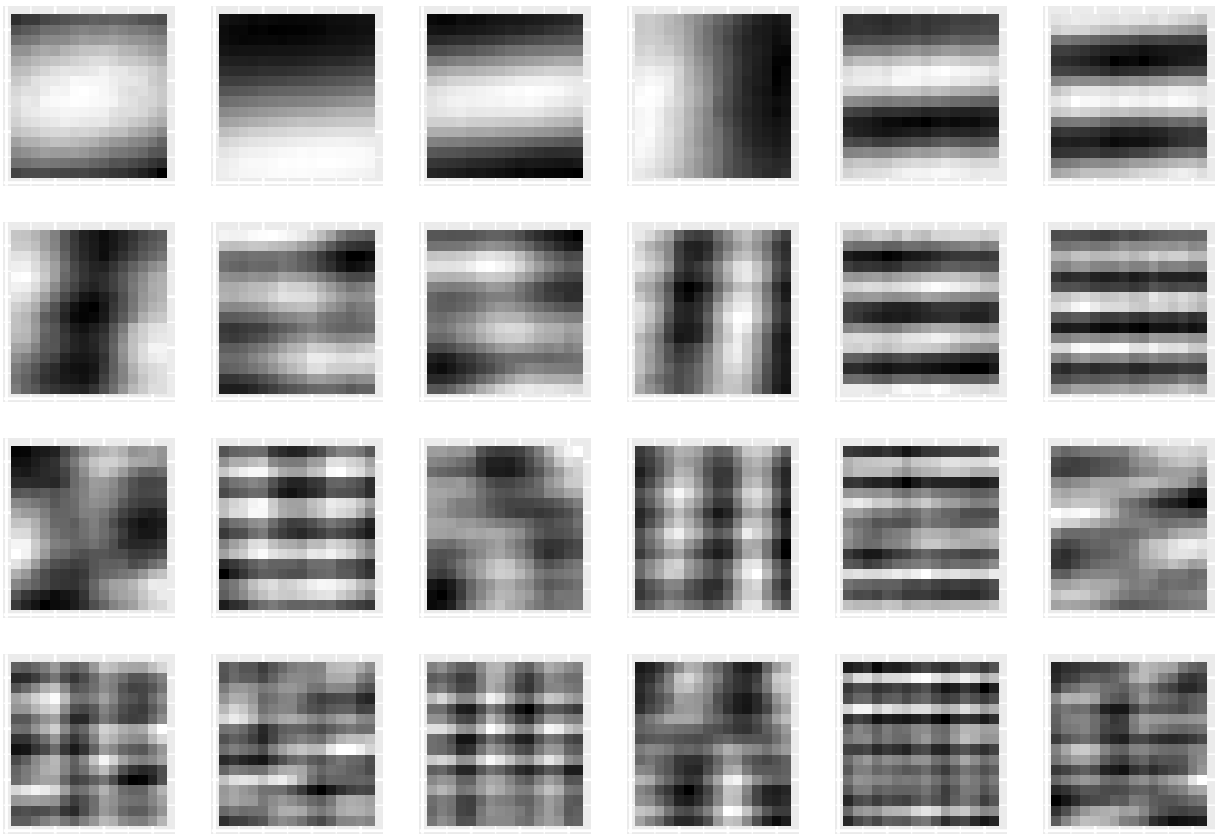
```
## [1] 5000  256
```

```
# 2.4. a) Nature

path <- "/Users/Niko/Desktop/Uni/Statistik Master/courses/machine intelligence 2/mi2_homework/mi2_homew

f <- file.path(path, c("n1.jpg","n2.jpg", "n3.jpg", "n4.jpg", "n5.jpg", "n6.jpg", "n7.jpg", "n8.jpg", "

nat <- lapply(f, readJPEG)


total_matrix_n <- matrix(nrow = 0, ncol = 256)
```

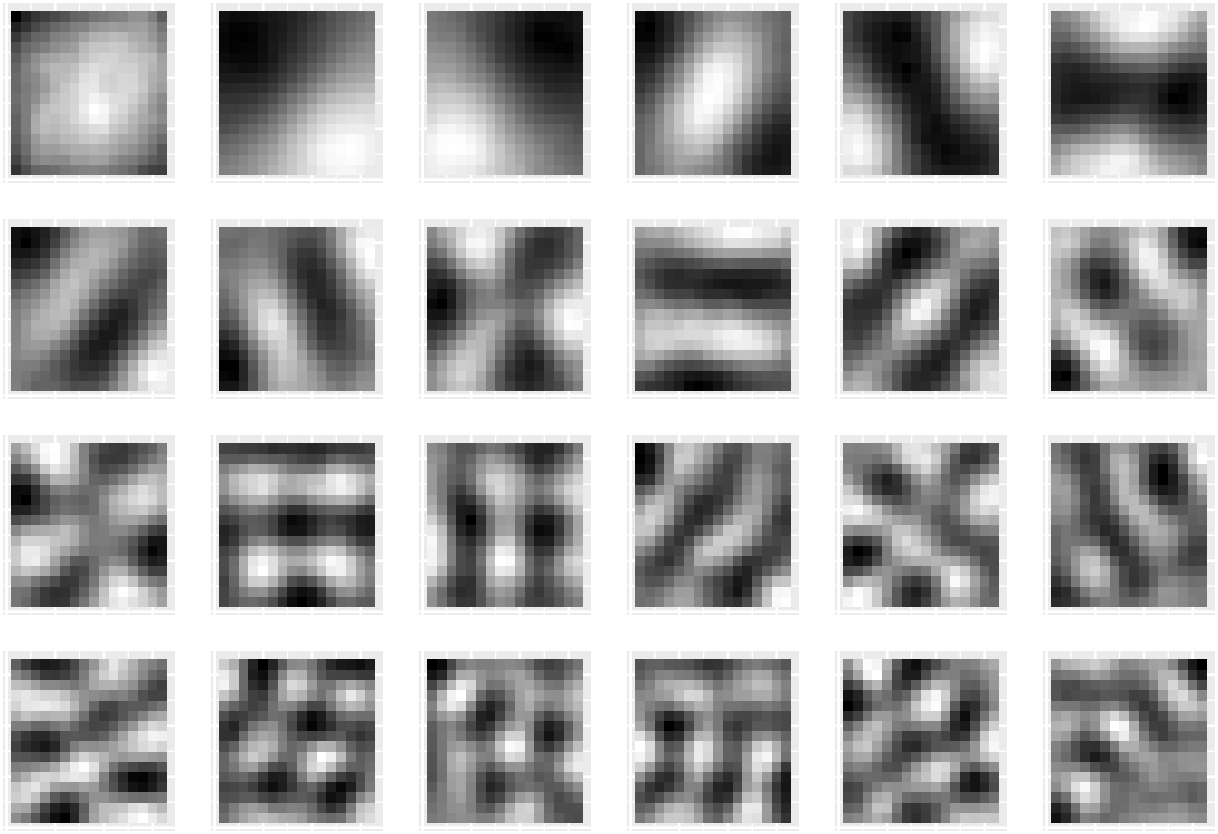
```
for(i in 1:length(nat)){
    #set.seed(123)
    tmp1 <- t(replicate(500, get_patch(nat[[i]], 16, 16, return_vector = TRUE), simplify = "vector"))
    #tmp2 <- replicate(10, get_patch(nat[[i]], 16, 16, return_vector = TRUE))
    total_matrix_n <- rbind(total_matrix_n, tmp1)
}

# Show dimensions of matrix for nature patches
dim(total_matrix_n)
```

```
## [1] 5000  256
```

```
# 2.4 b)

total_matrix_n_centered <- scale(total_matrix_n, center = TRUE, scale = FALSE)
total_matrix_b_centered <- scale(total_matrix_b, center = TRUE, scale = FALSE)

pcs_b <- eigen(cov(total_matrix_b_centered))$vectors
pcs_n <- eigen(cov(total_matrix_n_centered))$vectors


amount_pca <- 24
pca_patchesn <- pca_patchesb <- vector("list", length = amount_pca)

for(i in 1:24){

    pca_patchesn[[i]] <- melt(matrix(pcs_n[, i], 16, 16))
    pca_patchesb[[i]] <- melt(matrix(pcs_b[, i], 16, 16))

}


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)
}

plotlist_b <- lapply(pca_patchesb, heatmap_custom)
plotlist_n <- lapply(pca_patchesn, heatmap_custom)

do.call("grid.arrange", c(plotlist_b, ncol=6))
```

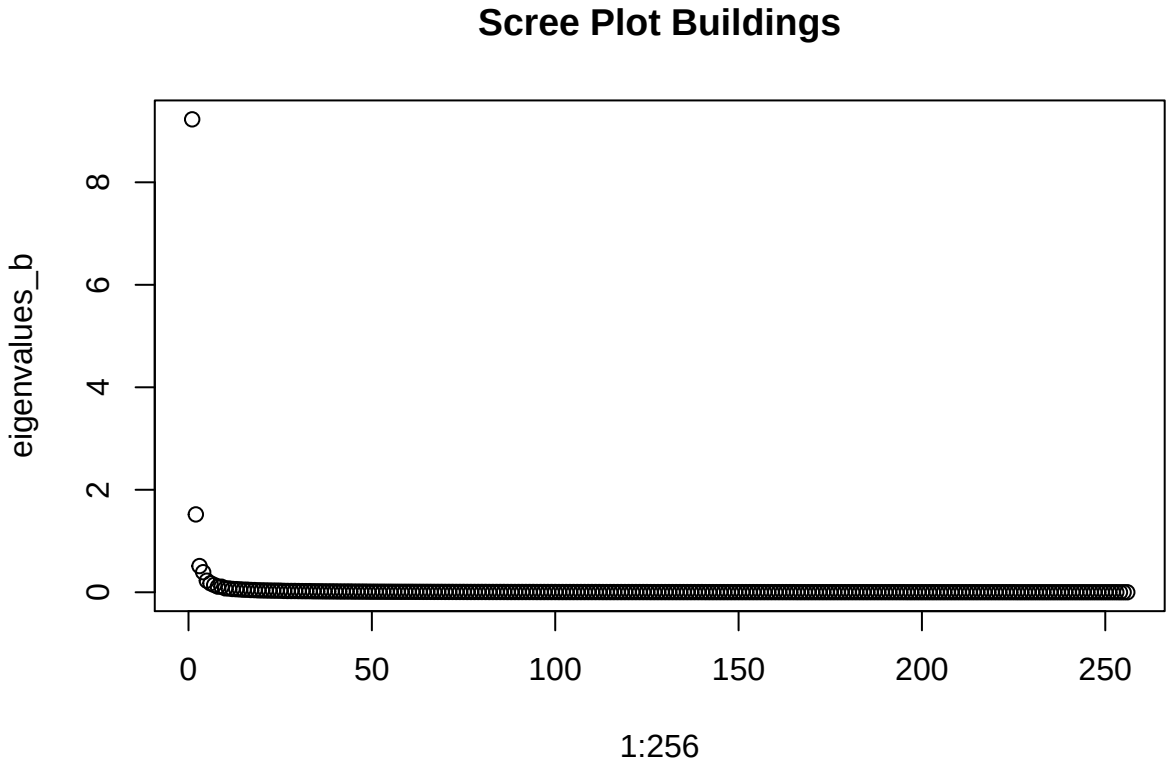


```
do.call("grid.arrange", c(plotlist_n, ncol=6))
```

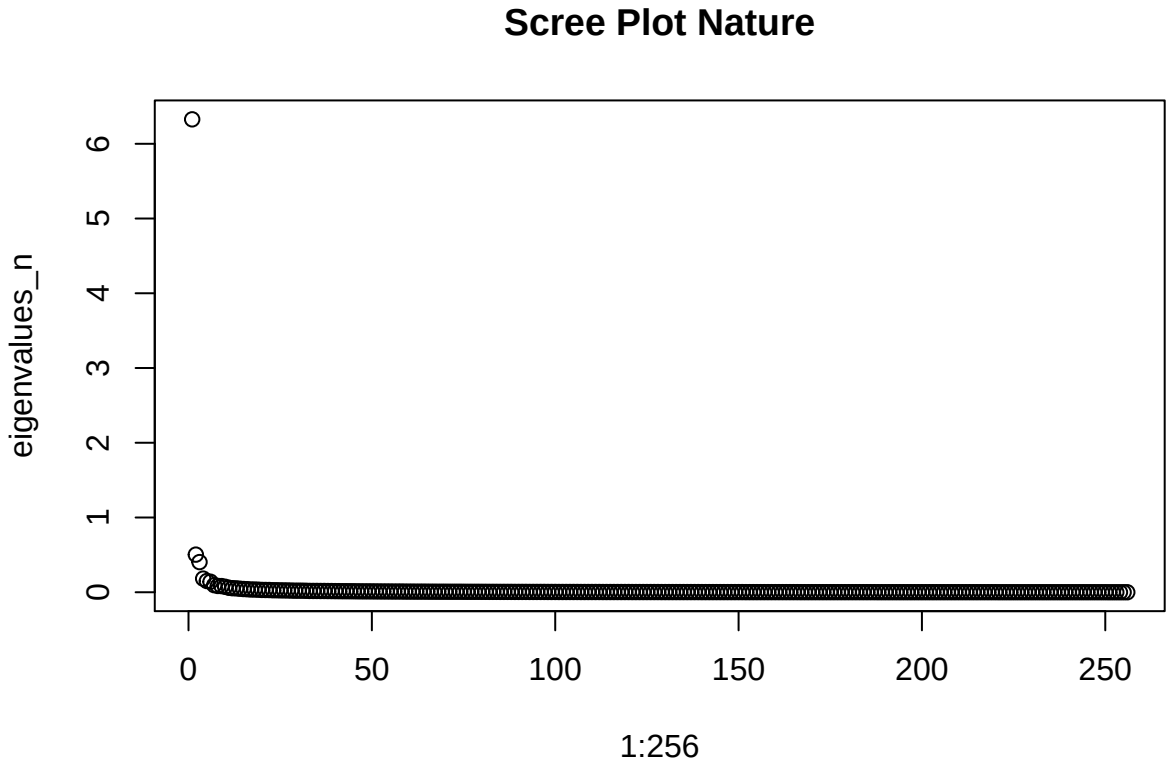


Comparison of PCs One can see that the PCs of both image groups are rather similar, in terms of the variance they account for (i.e. about 65% by the first PC in both cases). However, the second PC of the nature buildings accounts for about 10.7% of the variance while the second PC from the nature images only accounts for about 5.32%

```
# 2.4 c)

eigenvalues_b <- eigen(cov(total_matrix_b_centered))$values
plot(1:256, eigenvalues_b, main = "Scree Plot Buildings")
```




```r
cov_b <- cov(total_matrix_b)



eigenvalues_n <- eigen(cov(total_matrix_n_centered))$values
plot(1:256, eigenvalues_n, main = "Scree Plot Nature")
```



Buildings Based on the scree plot we would choose the first 4 PCs for the building images. Because of high eigenvalue and before the elobow of the Curve. Based on the Kaiser criterion (extract PCs with eigenvalues > 1) we would extract the first two PCs.

compression ratio: 4/256

Nature Based on the scree plot we would choose the first 3 PCs for the nature images. Because of high eigenvalue and before the elobow of the Curve. Based on the Kaiser criterion (see above) we would even extract only the first PC

compression ratio: 3/256