2 Part II: Programming Problems

2.1 The Method of PCA

Given a sample covariance matrix $\mathbf{S} = \mathbf{X}^T\mathbf{X}$, with $\mathbf{X}$ having dimension $n \times p$, eigen decomposition is performed such that

$$\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

where $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n)$ is the matrix of eigenvectors.

Further, for the ith case the jth principal component can be obtained by

$$y_{ij} = \mathbf{x}_i^\top \mathbf{q}_j$$

where $i = 1, 2, ..., n$ and $j = 1, 2, ..., p$.

2.2 Task 1

Set random seed for reproducibility.

```
set.seed(1337)
```

```
# convert vector to matrix
vec2mat = function(vec, dim) {
    return(matrix(unlist(vec), ncol=dim, byrow=T)[,dim:1])
}
```

Firstly, we fetch the data and extract the observations related to our favorite number 2.

```
# read data
dat = read.table('HA_2_zip.train')
dat_two <- dat[which(dat[,1] == 2), 2:257]
```

Then, we plotted the first 24 observations of our number.

```
# plot first 24 digits
par(mfrow = c(3, 4))
no = 1
for (n in 1:2) {
    for (i in 1:12) {
        img_i = vec2mat(dat_two[no, 256:1], dim=16)
        image(
            t(img_i), xaxt='n', yaxt='n',
            xlab='', ylab='', main=paste("Image", no)
        )
        no = no + 1
    }
}
```

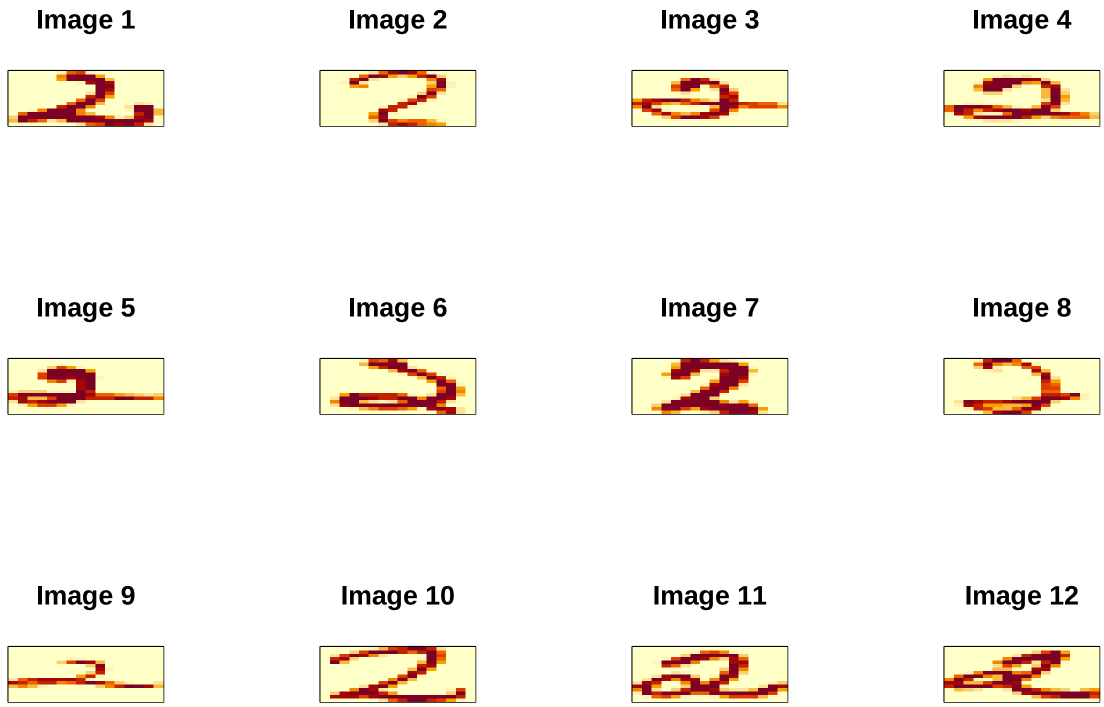
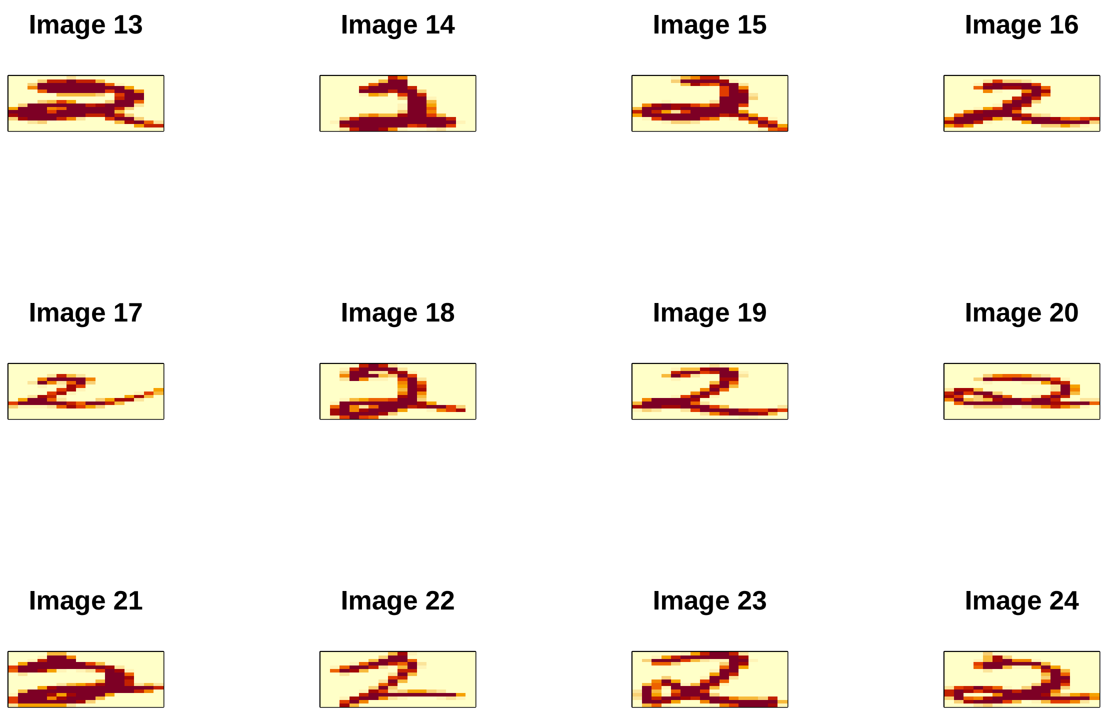


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Image 7
Image 8
Image 9
Image 10
Image 11
Image 12



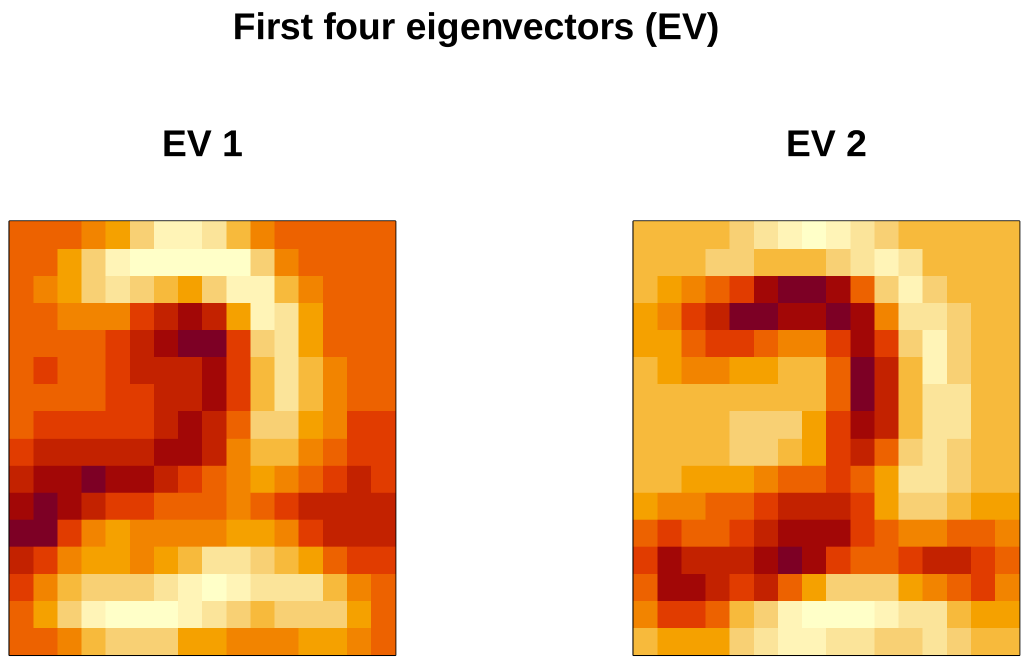
2.3 Task 2

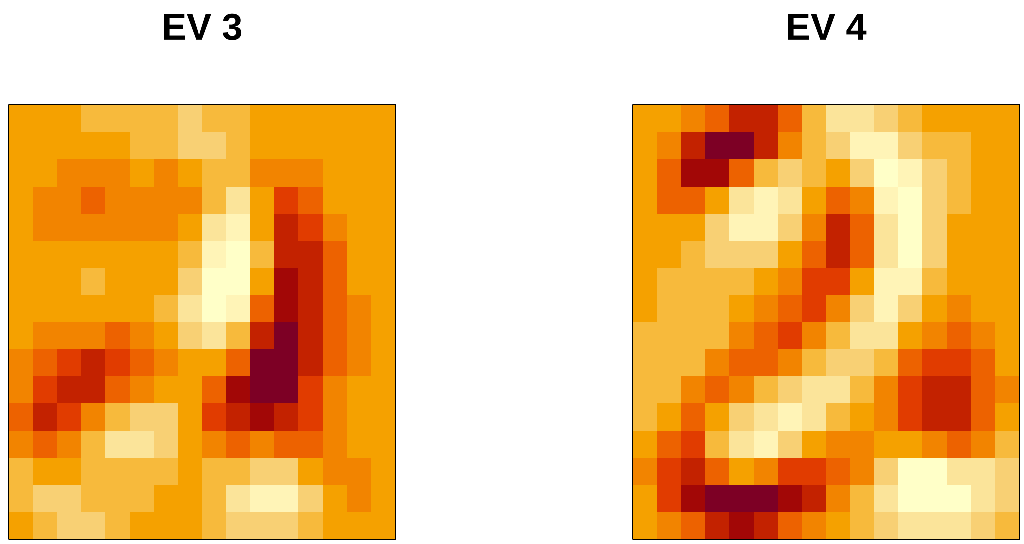
Consequently, we computed the principal components and plotted the four first and last, respectively, as 16x16 pixel images.

```
# convert data with digit 2 to matrix
dat_two_mat = matrix(
    unlist(dat_two),
    nrow=dim(dat_two)[1],
    ncol=dim(dat_two)[2]
)

# compute PC's and eigenvectors
dat_two_dem = demean(dat_two_mat)
S_two = t(dat_two_dem) %*% dat_two_dem
eigen_el = eigen(S_two)
Q_two = eigen_el$vectors
eigen_val = eigen_el$values
pc_two = dat_two_mat %*% Q_two
```

```
# first 4 PC's
par(mfrow=c(1, 2), oma=c(0, 0, 2, 0))

for (i in 1:4) {
    if (i == 3) {
```

```
        par(mfrow=c(1, 2))
    }
    pc = Q_two[, i]
    img = vec2mat(pc[256:1], dim=16)
    image(t(img), xaxt='n', yaxt='n', xlab='', ylab='', main=paste("EV", i))
    if (i == 2) {
        title("First four eigenvectors (EV)", outer=T)
    }
}
```

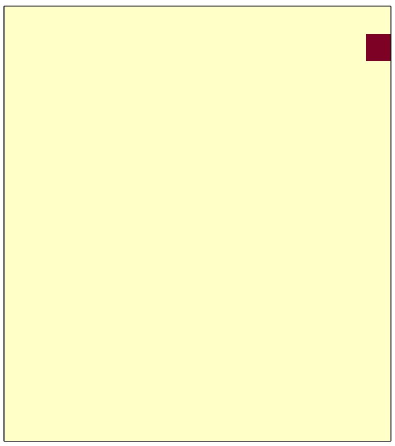




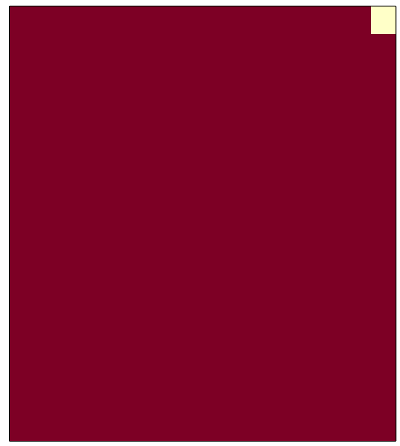

```r
# last 4 PC's
par(mfrow=c(1, 2), oma=c(0, 0, 2, 0))
for (i in 1:4) {
    if (i == 3) {
        par(mfrow=c(1, 2))
    }
    pc = Q_two[, (dim(dat_two_mat)[2] - 4) + i]
    img = vec2mat(pc[256:1], dim=16)
    image(t(img), xaxt='n', yaxt='n', xlab='', ylab='', main=paste("EV", i))
    if (i == 2) {
        title("Last four eigenvectors (EV)", outer=T)
    }
}
```

Last four eigenvectors (EV)

EV 3



EV 4

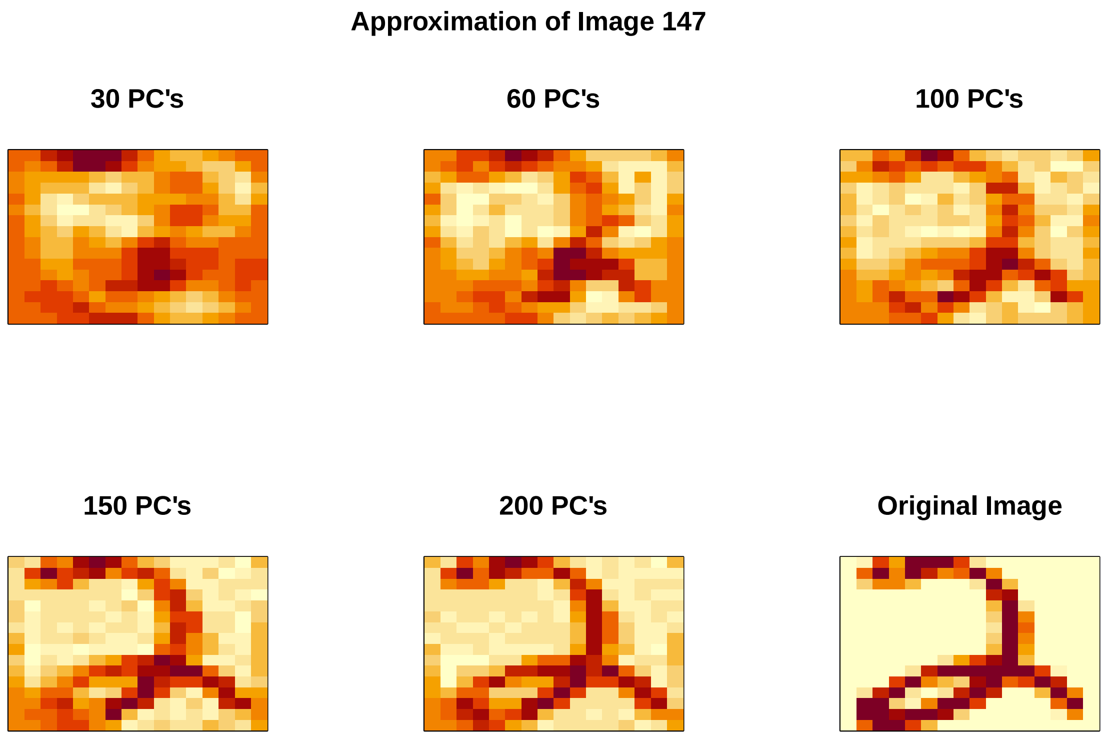


2.4 Task 3

By the principal components previously computed, a randomly selected image was selected and then reconstructed by different number of principal components. Alongside the image reconstruction, the mean-squared error (MSE) was calculated.

```
set.seed(1337)
par(mfrow=c(2, 3), oma=c(0, 0, 2, 0))
n = c(30, 60, 100, 150, 200)
rnd_idx = sample(1:dim(dat_two_mat)[1], 1)
real_vec = dat_two_mat[rnd_idx, 256:1]
real_img = vec2mat(real_vec, dim=16)
mse = vector()
j = 1
for (i in n) {
    q = Q_two[, 1:i]
    pc = real_vec %*% q
    est_vec = pc %*% t(q)
    img = vec2mat(est_vec, dim=16)
    image(t(img), xaxt='n', yaxt='n', xlab='', ylab='', main=paste(i, "PC's"))

    mse[j] = round(sum((real_vec - est_vec)^2) / length(real_vec), 4)
    j = j + 1
}
```

```
image(t(real_img), xaxt='n', yaxt='n', xlab='', ylab='', main="Original Image")
title(paste("Approximation of Image", rnd_idx), outer=T)
```




```
mse_df = data.frame(
    n_PC = c(n),
    MSE = c(mse)
)
print(mse_df)
```

```
##   n_PC    MSE
## 1   30 0.4180
## 2   60 0.2878
## 3  100 0.1808
## 4  150 0.1128
## 5  200 0.0679
```

2.5 Task 4

Related to the image reconstruction, the number of principal components needed to explain 85% of the variance was determined and also visualized by a bar plot. It was concluded that we need 38 principal components to explain 85% of the variance.

```r
tot_var = sum(eigen_val)
var_exp = cumsum(eigen_val / tot_var)
```

```r
barplot(
    var_exp[1:40] * 100,
    names.arg=1:40,
    xlab="Principal Component",
    ylab="Variance Epxlained (%)",
    ylim=c(0, 100),
    col=3
)
abline(h=85, col=2)
legend(x=38, y=100, legend="85%-limit", col=2, lty=1, cex=.75)
```

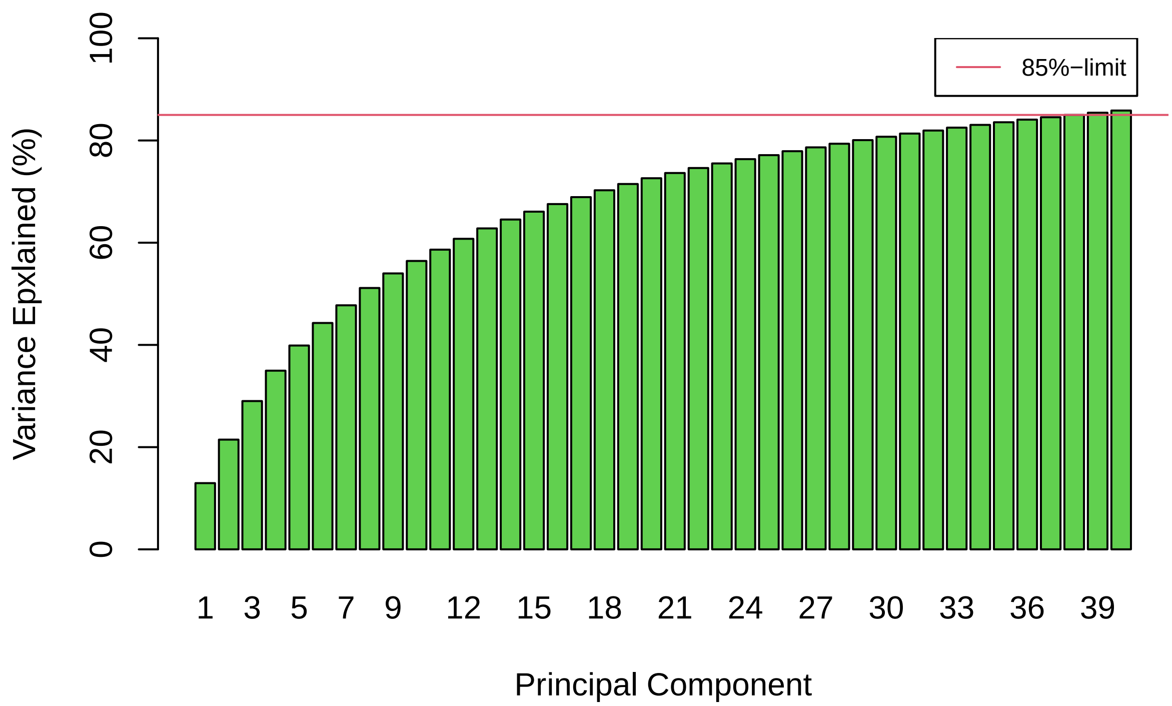


Figure 1: *The cumulative proportion of variance explained for the first 40 principal components.*

```r
paste("PC's needed to explain 85% of the variance:", sum(var_exp < 0.85) + 1)
```

```
## [1] "PC's needed to explain 85% of the variance: 38"
```

2.6 Task 5

2.6.1 Subtask 5.1

A new working data set was created by combining all observations of the digits 5 and 6.

```
dat_5 = dat[which(dat[,1] == 5), 1:257]
dat_6 = dat[which(dat[,1] == 6), 1:257]
c_dat = rbind(dat_5, dat_6)
```

2.6.2 Subtask 5.2

The data set was then divided into training set (80%) and testing set (20%) by random sampling.

```
set.seed(1337)

# randomly split data into train and test, 80%/20%
n = dim(c_dat)[1]
idx = sample(1:n, floor(0.8 * n))
train_dat = c_dat[idx, ]
test_dat = c_dat[-idx, ]

train = train_dat[, 2:257]
train_label = train_dat[, 1]

test = test_dat[, 2:257]
test_label = test_dat[, 1]
```

2.6.3 Subtask 5.3

A PCA, as presented in Section 1.2, was then performed on the training and testing set, respectively.

```
# compute eigen vectors
X_train = matrix(unlist(train), nrow=dim(train)[1], ncol=256)
X_test = matrix(unlist(test), nrow=dim(test)[1], ncol=256)

# compute PC's for train and test data
X_dem = demean(X_train)
S_train= t(X_dem) %*% X_dem
Q_train = eigen(S_train)$vectors
pc_train = X_train %*% Q_train
pc_test = X_test  %*% Q_train
```

2.6.4 Subtask 5.4

Consequently, a LDA classifier, as presented in Home Assignment 1, was fitted to the training data using only the first two principal components.

```
# format training data
train_data = as.data.frame(cbind(pc_train, train_label))

# fit LDA model
mdl = lda(
    train_label ~ V1 + V2,
    data=train_data
)
```

2.6.5 Subtask 5.5

Lastly, the LDA classifier was applied to the test set, and the confusion matrix along with the test accuracy were calculated. We obtain a test accuracy of approximately 95%, and the model tend to incorrectly classify slightly more 5's than 6's.

```
# predict test set
test_pred = predict(mdl, as.data.frame(pc_test))$class

# compute confusion matrix and test accuracy
test_res = confusionMatrix(
    data=factor(test_pred),
    reference=factor(test_label)
)
test_res$table
```

```
##            Reference
## Prediction   5   6
##          5 106   3
##          6   9 126
```

```
test_res$overall[1]
```

```
##  Accuracy
## 0.9508197
```

2.7 Conclusions

To summarize, considering the digit 2, we found that 38 principal components are needed to explain 85% of the variance. Further, the MSE was, trivially, decreasing as the number of components increased in the image reconstruction. Lastly, the LDA classifier resulted in a test accuracy of approximately 95% using only the first two components, which can be considered as a very good result. We would, however, not be even close to achieve these results with only two of the original variables, since these would not nearly explain as much as two "loaded" principal components.