Exercise 2 – Data Preprocessing

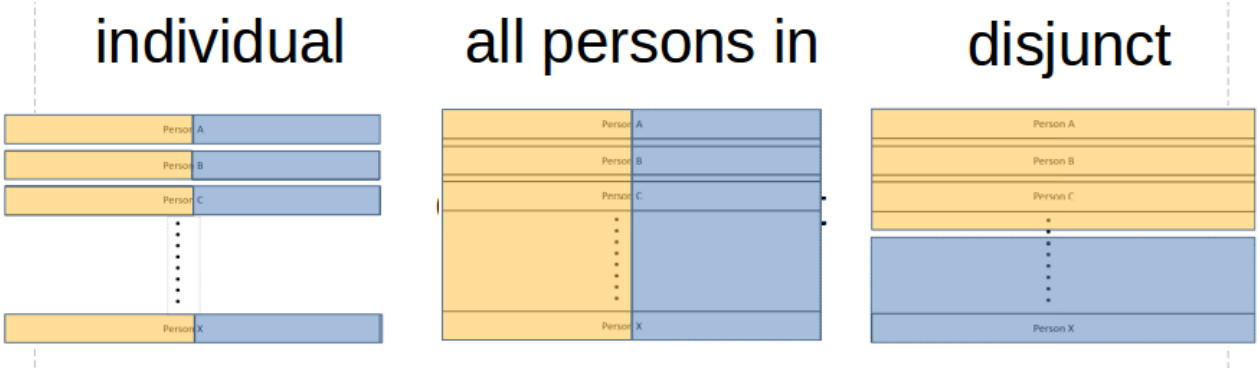
The goal of this exercise is to deepen your understanding of PCA as a tool for compressing the size of data. Furthermore, the exercise should enable you to better understand the effects of normalization and smoothing, and in the report you should discuss when and why these are important tools in general.

It is important to remember to **read** the provided material:

- Pages 72-74 of "Machine Learning with R" (First Edition) by Brett Lantz, Packt Publishing Ltd., 2013
- Chapter 6.3 (Dimension Reduction Methods) and 10.2 (Principal Component Analysis) of "An Introduction to Statistical Learning with Applications in R" from G. James, D. Witten, D., T. Hastie, R. Tibshirani. Springer 2013.
- Wikipedia: Eigen-Decomposition of a matrix (https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix)

It is important that you have an understanding of the theory behind the algorithms and you are able to describe the advantages and disadvantages of the different techniques.
During analysis of execution times, it is important to consider what part of the execution time would be an issue in real world analysis.

You are also welcome to use preprocessing, such as smoothing and down sampling. But remember to specify the methods you have used. To improve the results when using data from different people, you can consider thresholding the images into binary images, since this should improve the similarity of people drawing with differently saturated colors.



In R various functions can simplify the implementations such as:
prcomp: a simple tool for performing Eigenvalue decomposition PCA

- Remember to use ?prcomp in the console in Rstudio and read the description
- prcomp gives both the new values (scores), the PCA parameters (scaling, centering and loadings) and the eigenvalues
- you can use "predict" to evaluate new data with the found rotations

min, max, mean, sd: Simple tools for determining min,max, mean and the standard deviation. This should make normalization easy to implement.

Exercise 2.1: Principal Component Analysis (PCA)

Perform a PCA on the data for the "all persons in" and "disjunct" data set.

2.1.1 Show the standard deviation (From prcomp Eigenvalues), the proportion of variance and the cumulative sum of variance of the principal components. (In the report the first 10-20 principal components, should be sufficient to illustrate the tendencies.)

2.1.2 Show the performance of selecting enough principal components to represent 80%, 90%, 95%, 99% of the accumulated variance. For each test vary "k" in kNN, try 3 reasonable values.

2.1.3 Measure run times for the prediction step of the kNN-classifier with PCA based dimensionality reduction. How does the feature vector dimensionality effect performance?

2.1.4 Interpret the results.

Exercise 2.2: Normalization

Perform one of the two normalizations suggested in the lecture (min-max normalization and z-standardization) for the best parameter setting found under 2.1.3 and apply kNN with 10 fold cross-validation (10 runs, 90% training and 10% test set).

Apply the normalization before and after PCA independently and compare the results. Analyze the results

Exercise 2.3: Preprocessing: Apply gaussian smoothing function with different sigmas to the images.

Perform again the steps in task 2.2 (**Cross validation**). Describe and analyze the results for one of the smoothing methods depending on the amount of smoothing.

```
id_mat <- data.matrix(id, rownames.force = NA)
imageSize <- sqrt(ncol(id_mat) - 1)
rotate <- function(x) t(apply(x, 2, rev))

smoothImage <- function(grayImg){
  smoothed <- as.matrix(blur(as.im(grayImg), sigma = 0.5, normalise=FALSE, bleed = TRUE, varcov=NULL))
  return(smoothed)
```

```
}
# Smooth all images
for(i in 1:nrow(id_mat))
{
  rotated <- c(id_mat[i,2:ncol(id)])
  image <- matrix(rotated,nrow = imageSize,ncol = imageSize, byrow = FALSE)
  image <- smoothImage(image)
  id_mat[i,2:ncol(id_mat)] <- matrix(image,nrow = 1,ncol = ncol(id_mat) - 1, byrow = FALSE)
}
id <- as.data.frame(id_mat)
id[,1] <- factor(id[,1])
```

Exercise 2.4: Reconstruction using PCA

2.4.1 This task is about reconstructing data using PCA. First using these functions we can plot an image of a single cipher (for plotting images do not convert **id** to data frame):

```
id_mat <- data.matrix(id, rownames.force = NA)

rotate <- function(x) t(apply(x, 2, rev))
# Show first 10 images
for(i in 1:10)
{
  rotated <- c(id_mat[-400+i*400+1,2:ncol(id_mat)])
  rotated <- ((rotated - min(rotated)) / (max(rotated) - min(rotated)))

  image <- matrix(rotated,nrow = imageSize,ncol = imageSize, byrow = FALSE)
  image <- rotate(image)
  image( image,  zlim=c(0,1), col=gray(0:100/100) )
}
```

Plot one of each cipher.

2.4.2. Plot the first 10 eigenvectors/loadingvectors as images. Can you describe what you see? (The first eigenvector is retrieved as id_pca$rotation[,1])

2.4.3. Plot a reconstruction of the images you displayed in 2.4.1 using all PC's. This can be done by multiplying the loadings with the scores and adding the removed centering. (The 'cipherNumber' below is the row of the image for investigation.)

```
trunc <- id_pca$x[cipherNumber,cumsum(Proportion) < 0.99] %*%
t(id_pca$rotation[,cumsum(Proportion) < 0.99])
trunc <- scale(trunc, center = -1 * id_pca$center, scale=FALSE)
```

2.4.4. Now re-recreate using 80% of variance, 90% and 95%.
Can you describe what you see?
How much have you reduced the data size?

2.4.5. The last exercise is to compare the outcomes between two different ciphers. For instance, two different ciphers, (e.g. row 43 and row 456 represent a ’0’ and a ’1’), compare the 10 first scores and see if you can spot a difference. Try also to calculate the mean for all 400 instances of these ciphers and compare the first 10 scores. Can you spot a pattern when comparing with the loadings.