**#################################################**

**Unsupervised learning**

**Class 03 – Predictions and clustering of colours**

Katarzyna Kopczewska, Associate professor

[kkopczewska@wne.uw.edu.pl](mailto:kkopczewska@wne.uw.edu.pl)

**#################################################**

**library(cluster)**

**library(factoextra)**

**library(flexclust)**

**library(fpc)**

**library(clustertend)**

**library(ClusterR)**

**# changing the path for accessing the Working Directory**

**# warning: change \ on /**

**getwd() # checking current WD**

**setwd("D:/My all/&Wykłady/Wykłady - WNE Unsupervised Learning/01. Clustering/dane ceny")**

**getwd()**

**#import of data**

**price\_where<-read.csv("prices\_regions.csv", sep=";", dec=".", header=TRUE)**

**summary(price\_where)**

**dim(price\_where) # checking the dimensions of the dataset**

**price\_when<-read.csv("prices\_months.csv", sep=";", dec=".", header=TRUE)**

**summary(price\_when)**

**dim(price\_when) # checking the dimensions of the dataset**

**price\_what<-read.csv("prices\_products.csv", sep=";", dec=".", header=TRUE)**

**summary(price\_what)**

**dim(price\_what) # checking the dimensions of the dataset**

**# no labels at data – price\_when**

**region\_when<-price\_when[,1] # first column**

**product\_when<-price\_when[,2] # second column**

**months\_when<-colnames(price\_when[3:14]) # first row**

**price\_when<-as.matrix(price\_when[,3:14]) # data only**

Spis treści

[## 01. Predictions for kmeans and PAM 2](#_Toc84882709)

[5](#_Toc84882710)

[## 02. Clustering of colours from image 6](#_Toc84882711)

## 01. Predictions for kmeans and PAM

The essence of machine learning is the ability to automatically classify new data points on a basis of the predefined model. The aim is to obtain the allocation of new data to the established clusters. Such possibilities are offered by the flexclust:: package, which contains **predict()** function for most groupings from different packages. The syntax of the **predict()** command requires a grouping object in the *kcca* class and new data. The conversion to the *kcca* class is possible with the **as.kcca()** command, which specifies the grouping result and the original data.

**# let’s do well-known clustering**

**# clustering of dataset – by time (in rows after transposition)**

**km2<-eclust(t(price\_when), "kmeans", hc\_metric="euclidean",k=4)**

**x1<-apply(t(price\_when), 2, sd)**

**x2<-which(x1==0)**

**km2<-eclust(t(price\_when)[,-x2], "kmeans", hc\_metric="euclidean",k=4)**

**fviz\_cluster(km2, main="kmeans / Full data")**

Clustering with factoextra::eclust()

Classes "kmeans" "eclust"

**# let’s divide a dataset into two parts: training and test**

**# for test part we keep X05.2019 and X11.2019**

**set.full<-t(price\_when)[,-x2] # all non-zero data**

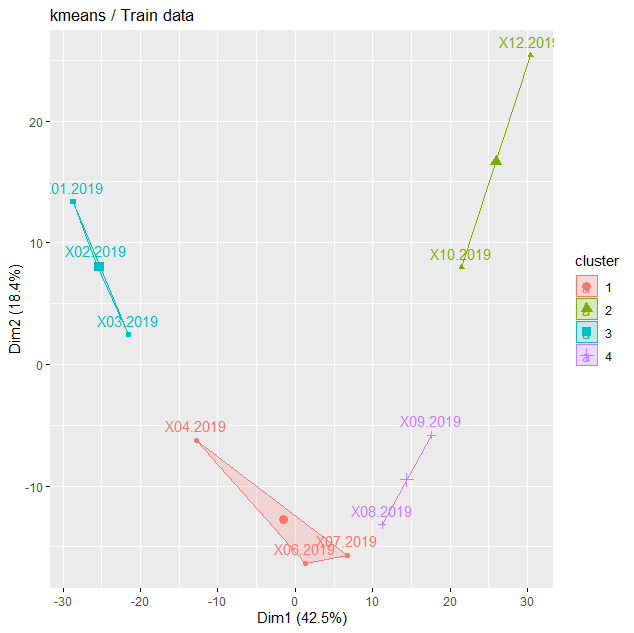
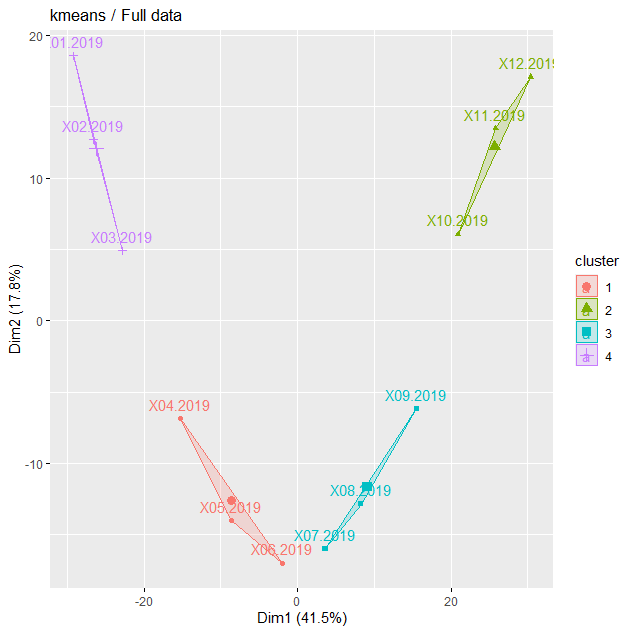
**set.train<-set.full[c(-5,-11), ] # full set without two rows**

**set.test<-set.full[c(5,11), ] # only two rows**

**# let’s cluster train data**

**km.train<-eclust(set.train, "kmeans", hc\_metric="euclidean",k=4)**

**fviz\_cluster(km.train, main="kmeans / Train data")**



The following example presents the classification of new points for k-means grouping, written above in object ***km.train*** in *kmeans* class. Example shows a conversion to the *kcca* class, preparation of new 2 rows of data, and visualisation of the result of the automatic classification. The figure of clustering shows the new points in clusters. As one can see, grouping and prediction are in fully consistent.

**library(flexclust)**

**km.train.kcca<-as.kcca(km.train, set.train) # conversion to kcca**

**km.pred<-predict(km.train.kcca, set.test) # prediction for k-means**

**km.pred**

#X05.2019 X11.2019

# 1 2

It is difficult to visualize the output as plot requires PCA reduction. However, cluster id’s were assigned well.

**# let’s do it once again – now with 2 variables only (for easy plotting)**

**# let’s divide a dataset into two parts: training and test**

**# for test part we keep X09.2019 and X11.2019**

**set.full<-t(price\_when)[,-x2] # all non-zero data**

**set.full<-set.full[,c(1,4)]**

**set.train<-set.full[c(-9,-11), ] # full set without two rows**

**set.test<-set.full[c(9,11), ] # only two rows**

**# let’s cluster full data**

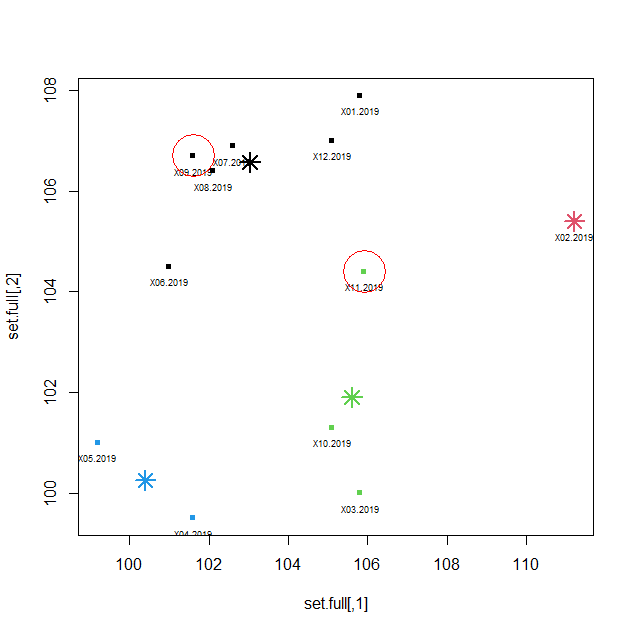
**km2<-kmeans(set.full, 4)**

**plot(set.full[,1], set.full[,2], col=km2$cluster, pch=".", cex=5) # figure has only 2D**

**points(km2$centers, col=1:4, pch=8, cex=2, lwd=2)**

**text(set.full[,1], set.full[,2]-0.3, labels=rownames(set.full), cex=0.6)**

**points(set.full[c(9,11),1], set.full[c(9,11),2], pch=21, cex=6, col="red")**



Clustering with kmeans()

class "kmeans"

**# let’s cluster train data**

**km.train<-kmeans(set.train, 4)**

**plot(set.train[,1], set.train[,2], col=km.train$cluster, pch=".", cex=5)**

**points(km.train$centers, col=1:4, pch=8, cex=2, lwd=2)**

**text(km.train$centers[,1], km.train$centers[,2]+1, labels=1:4, font=2, col=1:4)**

**text(set.train[,1], set.train[,2]-0.3, labels=rownames(set.train), cex=0.6)**

**# let’s make the prediction**

**km.train.kcca<-as.kcca(km.train, set.train) # conversion to kcca**

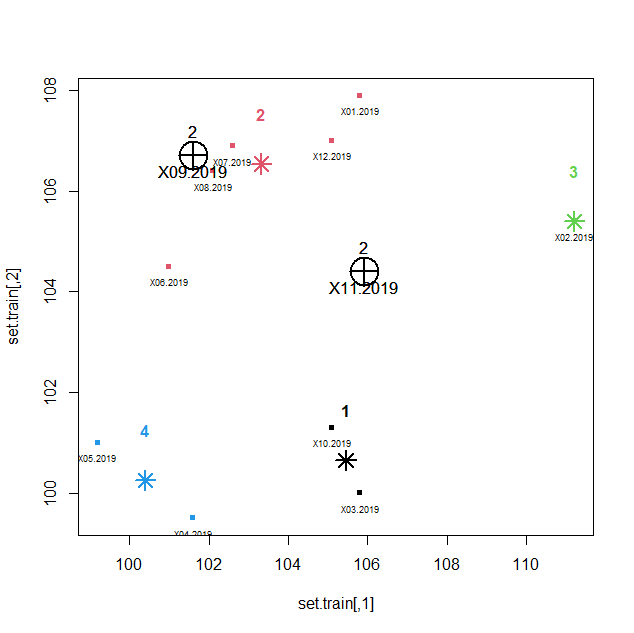
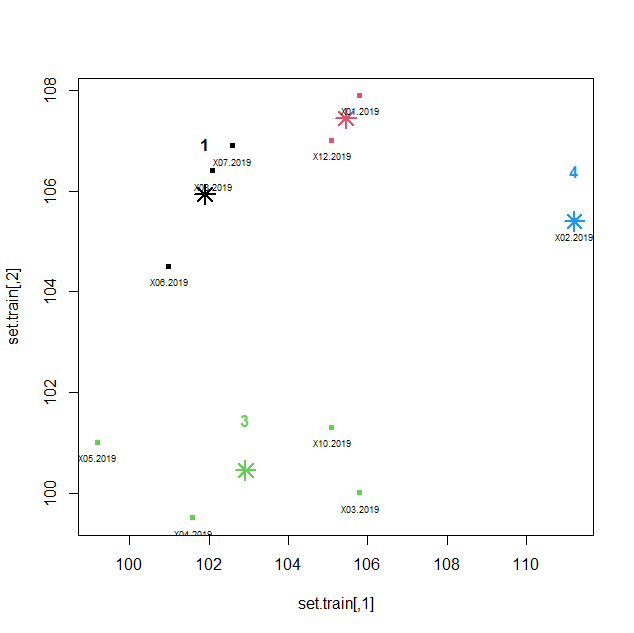
**km.pred<-predict(km.train.kcca, set.test) # prediction for k-means**

**km.pred**

**points(set.test[,1], set.test[,2], pch=10, cex=4, lwd=2)**

**text(set.test[,1], set.test[,2]+0.5, labels=km.pred)**

**text(set.test[,1], set.test[,2]-0.3, labels=names(km.pred))**



**# let’s try the same with PAM**

**# let’s do well-known clustering**

**# clustering of dataset – by time (in rows after transposition)**

**# let’s cluster train data**

**set.full<-t(price\_when)[,-x2] # all non-zero data**

**set.train<-set.full[c(-5,-11), ] # full set without two rows**

**set.test<-set.full[c(5,11), ] # only two rows**

Clustering with cluster::pam()

classes "pam" "partition"

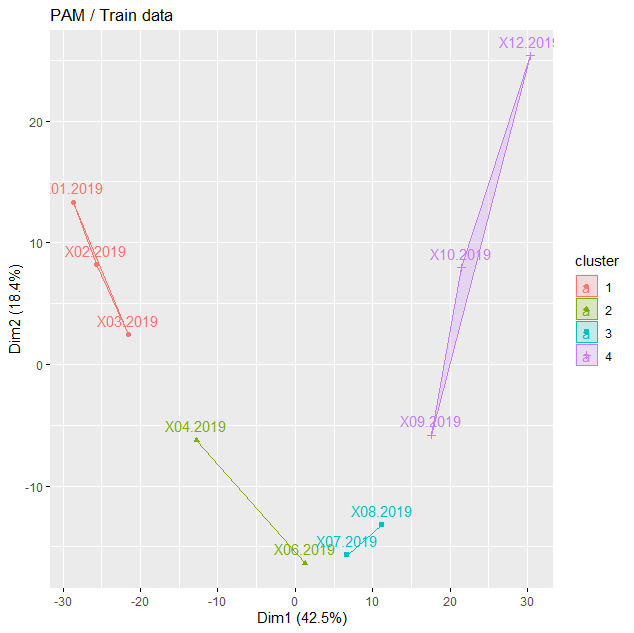
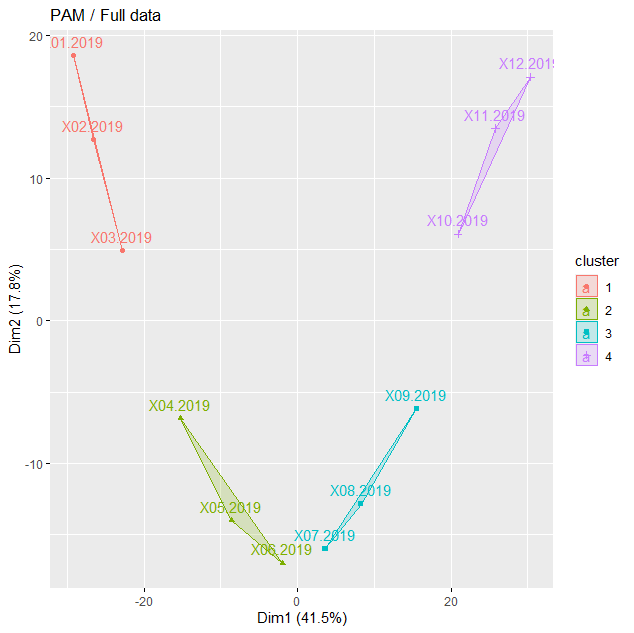
**library(cluster)**

**pam.full<-pam(set.full, k=4)**

**fviz\_cluster(pam.full, main="PAM / Full data")**

**pam.train<-pam(set.train, k=4)**

**fviz\_cluster(pam.train, main="PAM / Train data")**



**pam.train.kcca<-as.kcca(pam.train, set.train) # conversion to kcca**

**pam.pred<-predict(pam.train.kcca, set.test) # prediction for k-means**

**pam.pred**

# X05.2019 X11.2019

# 2 4

**Summary of predictions:**

- have train and test data

- run clustering on train data with any package: factoextra::eclust(), stats::kmeans(), cluster::pam(), cluster::clara() etc.

- convert clustering object into *kcca* class – use as.kcca() from flexclust::

- run predict() with *kcca* clustering object and test data

- in output you get cluster assignments for test data

**# EXTRA**

In case of binary classification (yes/no, true/false, dog/cat) one can check the quality of prediction by using Matthews correlation coefficient. Its input are two vectors (prediction vs. realization) or numbers from confusion matrix (2x2 matrix with true/false on both sides). Its value are interpreted similarly as Person correlation coefficient.

Very good explanation of Matthews coefficient and similar measures is here:

<https://en.wikipedia.org/wiki/Matthews_correlation_coefficient>

In R, there are two functions for that: mltools::mcc() and mccr::mccr().

## 02. Clustering of colours from image

**# libraries which read and plot jpg**

**library(jpeg)**

**library(rasterImage)**

**# importing jpg file to R**

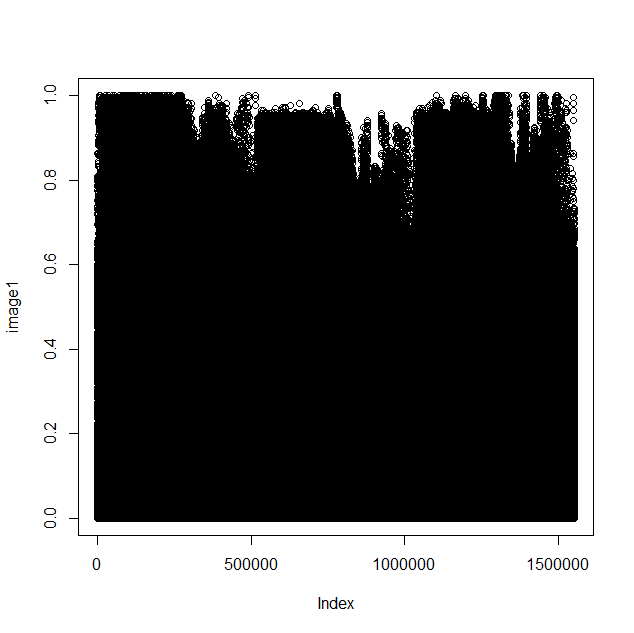
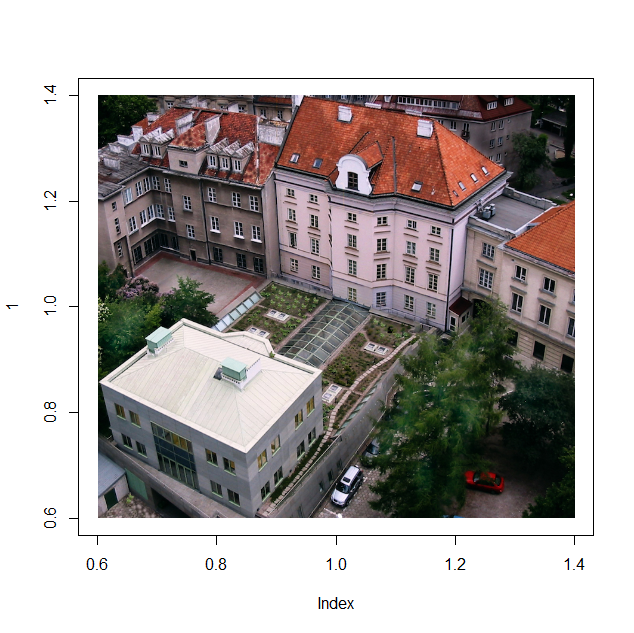
**image1<-readJPEG("wne-building.jpg") # class array**

**class(image1)**

**plot(1, type="n") # plotting the rasterImage**

**rasterImage(image1, 0.6, 0.6, 1.4, 1.4)**

#plot(image1) # does not work, do not try ☺



**# dimensions of dataset – it is 3D object**

**# “coordinates x”, “coordinates y” and RGB layer (3 columns of colours)**

**dm1<-dim(image1)**

**dm1**

# [1] 647 800 3

**# creating data frame: coordinates of each pixel + RGB three columns**

**rgbImage1<-data.frame(x=rep(1:dm1[2], each=dm1[1]), y=rep(dm1[1]:1, dm1[2]), r.value=as.vector(image1[,,1]), g.value=as.vector(image1[,,2]), b.value=as.vector(image1[,,3]))**

**head(rgbImage1)**

x y r.value g.value b.value

1 1 647 0.003921569 0.02745098 0.01176471

2 1 646 0.003921569 0.02745098 0.01960784

3 1 645 0.000000000 0.02352941 0.02352941

4 1 644 0.007843137 0.01568627 0.01176471

5 1 643 0.015686275 0.03529412 0.01960784

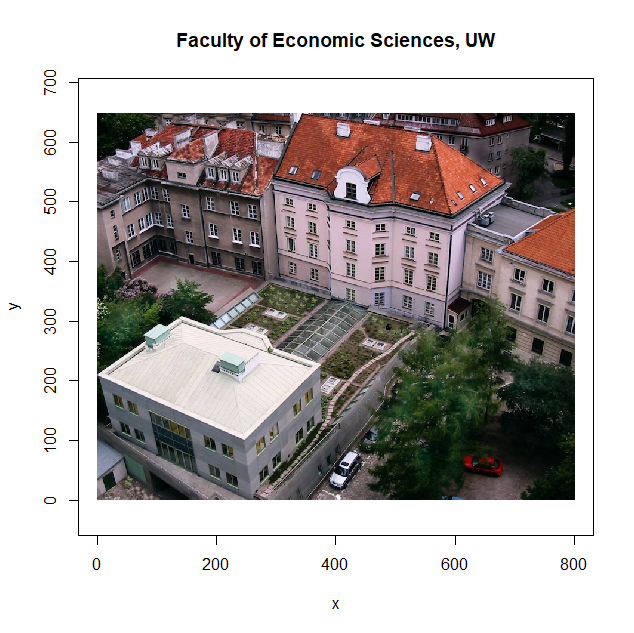
6 1 642 0.027450980 0.06666667 0.03529412

**dim(rgbImage1) # quite huge dataset**

#[1] 517600 5

**# plotting pixel by pixel in colours from RGB**

**plot(y~x, data=rgbImage1, main="Faculty of Economic Sciences, UW", col=rgb(rgbImage1[c("r.value", "g.value", "b.value")]), asp=1, pch=".")**



**# optimal number of clusters – it does not work, too big dataset**

library(factoextra)

library(cluster)

fviz\_nbclust(rgbImage1[,3:5], FUNcluster=cluster::clara) # factoextra::, not working

**# other solution to get silhouette – let’s use clara clustering from cluster::**

**library(cluster)**

**n1<-c() # empty vector to save results**

**for (i in 1:10) { # numer of clusters to consider**

**cl<-clara(rgbImage1[, c("r.value", "g.value", "b.value")], i)**

**n1[i]<-cl$silinfo$avg.width # saving silhouette to vector**

**}**

**plot(n1, type='l', main="Optimal number of clusters", xlab="Number of clusters", ylab="Average silhouette", col="blue")**

**points(n1, pch=21, bg="navyblue")**

**abline(h=(1:30)\*5/100, lty=3, col="grey50")**

**# silhouette for best number of clusters**

**# number of clusters in fact is number of colours**

**clara<-clara(rgbImage1[,3:5], 7) # set here number of clusters/colours**

**plot(silhouette(clara))**

**clara$medoids**

r.value g.value b.value

[1,] 0.05882353 0.1215686 0.08235294

[2,] 0.27058824 0.2745098 0.14901961

[3,] 0.56470588 0.5568627 0.57647059

[4,] 0.80392157 0.7215686 0.75686275

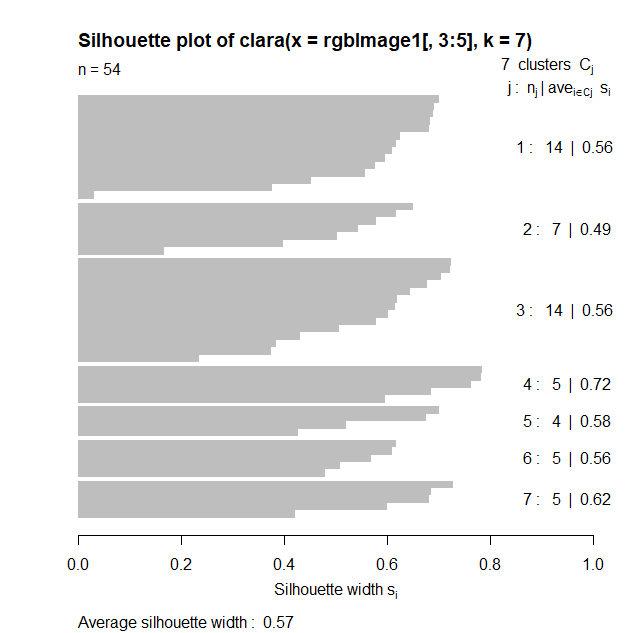
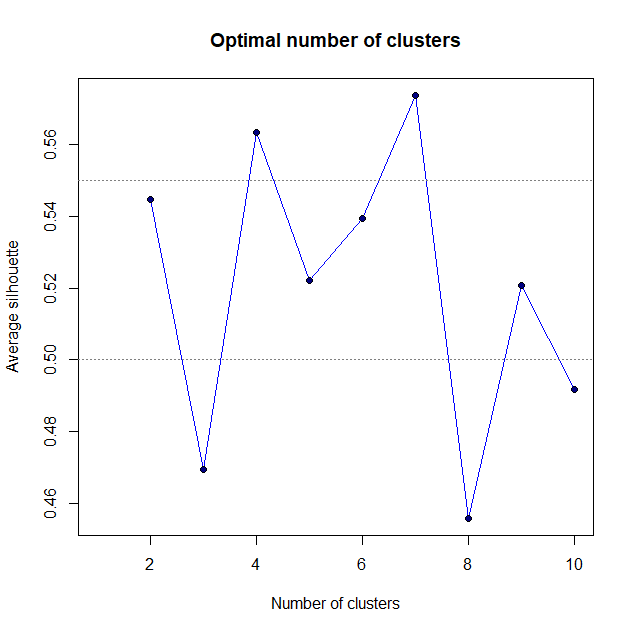
[5,] 0.34509804 0.3725490 0.39607843

[6,] 0.57647059 0.2941176 0.23529412

[7,] 0.90196078 0.8823529 0.86666667

rgb(clara$medoids)

[1] "#0F1F15" "#454626" "#908E93" "#CDB8C1" "#585F65" "#934B3C" "#E6E1DD"



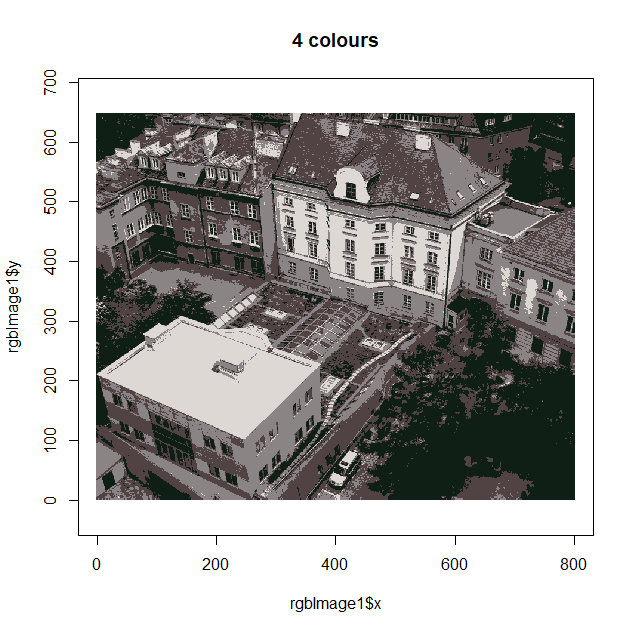
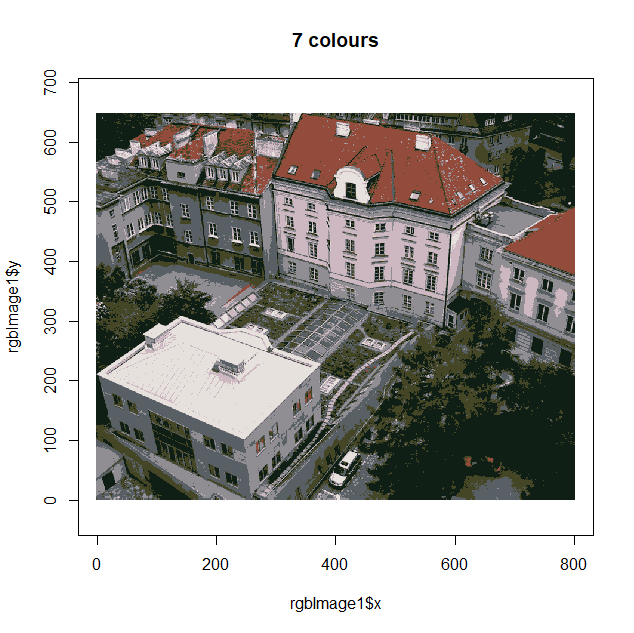
**# most important part: assigning medoid (“average” RGB values)**

**# to each cluster id and converting RGB into colour**

**colours<-rgb(clara$medoids[clara$clustering, ])**

**# plotting pixels in new colours**

**plot(rgbImage1$y~rgbImage1$x, col=colours, pch=".", cex=2, asp=1, main="4 colours")**



**# how many original colours were on picture**

**cols.org<-rgb(rgbImage1[,3:5])**

**head(cols.org)**

#[1] "#010703" "#010705" "#000606" "#020403" "#040905" "#071109"

**length(unique(cols.org))**

#[1] 114178

**Summary of clustering RGB colours**

- read your picture with jpeg:: readJPEG()

- check the dimensions of your imported picture with dim()

- create the data.frame with separate line for each pixel, where you store the RGB components

- cluster RGB components in selected number of clusters

- create new data frame, which for each pixel assigns cluster ID and medoids of clustering (which are “average” RGB components)

- plot pixel by pixel with new colours