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

**Unsupervised learning**

**Class 02 – Clustering with k-means /PAM / CLARA**

**Dr hab. Katarzyna Kopczewska**

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

**# Clustering**

# recommended packages for cluster analysis

# also other packages are available

install.packages("cluster")

install.packages("factoextra")

install.packages("flexclust")

install.packages("fpc")

install.packages("clustertend")

install.packages("ClusterR")

**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("E:/My all/&Wykłady/Wykłady - WNE Unsupervised Learning/01. Clustering")**

**getwd()**

**#import of data**

**dane<-read.csv("cluster.csv", sep=";", dec=",", header=TRUE)**

**dim(dane) # checking the dimensions of the dataset**

**# data selection**

**xxx<-dane[,c(25,32)] #XA06–unemployment rate, XA31–salaries (Poland=100%)**

**dane$population.total<-dane$XA19+dane$XA20+dane$XA21**

**dane$workforce.ratio<-dane$XA20/dane$population.total**

**names(dane)**

**yyy<-dane[,c(25,32, 36)]**

Spis treści

[## 01. Clustering / K-means 3](#_Toc526682622)

[## 02. Clustering / PAM (Partitioning Around Medoids) 7](#_Toc526682623)

[## 03. CLARA algorithms(CLustering LARge Applications) 10](#_Toc526682624)

[### TASK 1 12](#_Toc526682625)

## 01. Clustering / K-means

**# Using factoextra:: package:**

# One can use the eclust () function from the factoextra :: package.

# It allows for clusters using k-means, PAM, CLARA etc. methods,

# using Euclidean, Manhattan, Canberra, Minkowski distance etc.

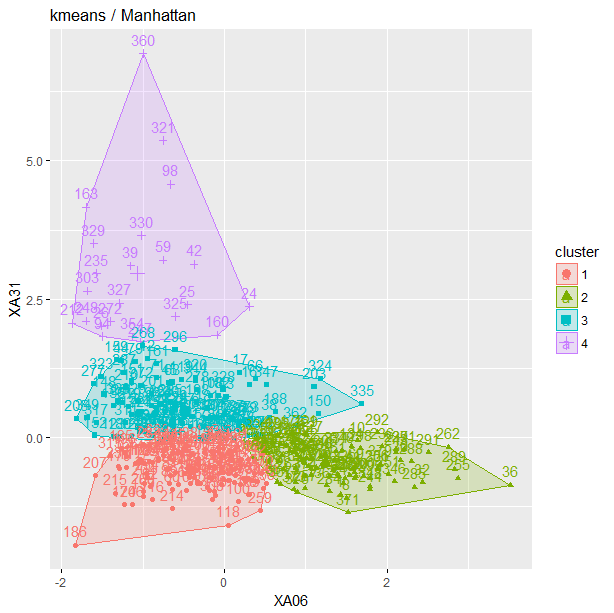
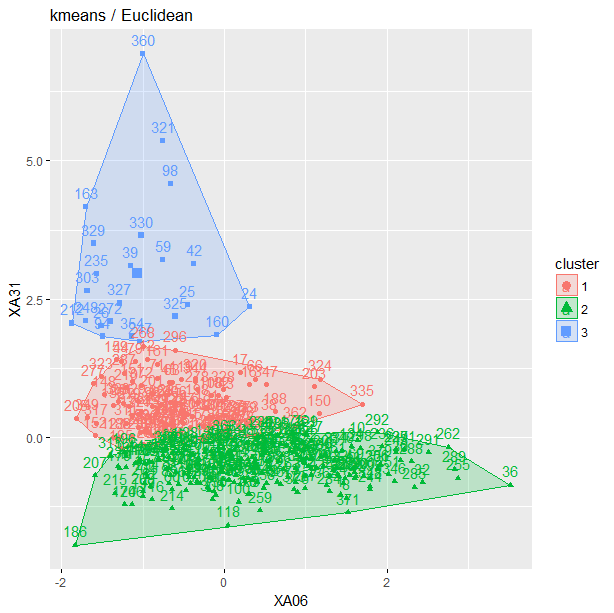
**# the same variables, different number of clusters and distance metrics**

**km1<-eclust(xxx, "kmeans", hc\_metric="euclidean",k=3)**

**fviz\_cluster(km1, main="kmeans / Euclidean")**

**km2<-eclust(xxx, "kmeans", hc\_metric="manhattan", k=4)**

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



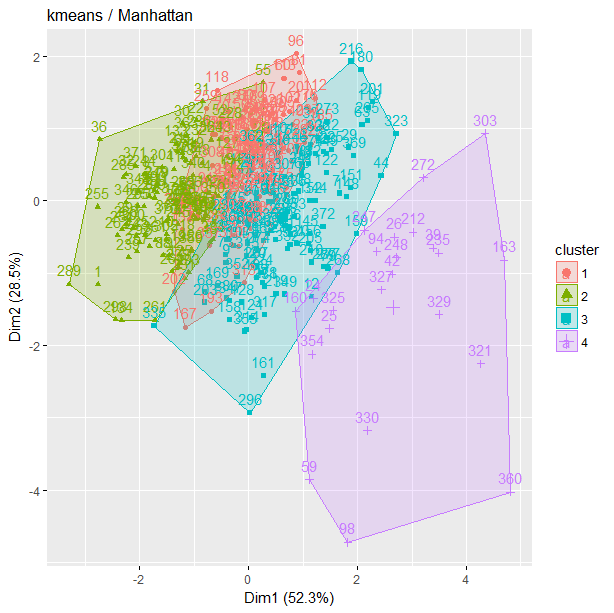
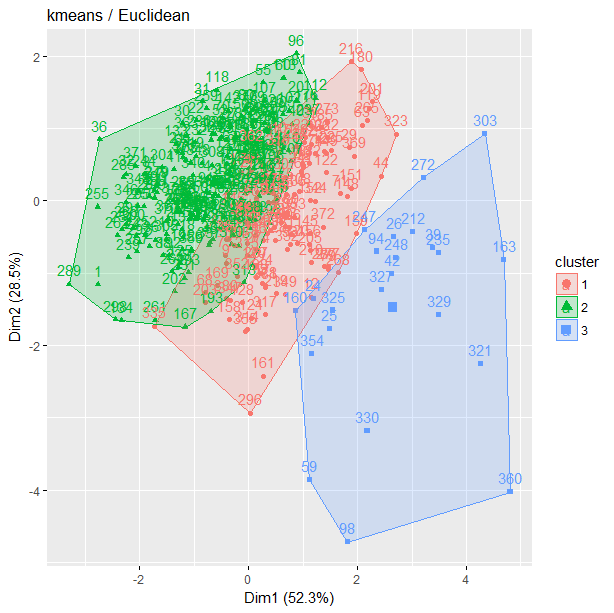
**# one variable extra, different number of clusters and distance metrics**

**km1<-eclust(yyy, "kmeans", hc\_metric="euclidean",k=3)**

**fviz\_cluster(km1, main="kmeans / Euclidean")**

**km2<-eclust(yyy, "kmeans", hc\_metric="manhattan", k=4)**

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



Observations / comments:

* One can easily change the analysed dataset (xxx & yyy) and distance metric (Euclidean, Manhattan)
* One can easily change the number of clusters (k)
* When two variables analysed (case 1), they are on the axes on the figure, when more variables analysed (case 2), the axes are the “dimensions” which explain the process (as in PCA).
* One extra variable changes clustering significantly

**# Let’s what is inside the clustering object**

**attributes(km1)**

$names

[1] "cluster" "centers" "totss" "withinss" "tot.withinss"

[6] "betweenss" "size" "iter" "ifault" "clust\_plot"

[11] "silinfo" "nbclust" "data"

$class

[1] "kmeans" "eclust"

**km1$cluster**

[1] 2 2 2 2 2 2 1 2 1 2 2 1 2 2 2 1 1 1 2 2 2 2 2 3 3 3 1 2 1 2 2 2 1 1 2 2 1

[38] 1 3 2 2 3 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 1 2 2 1 2 1 2 2 1 2 1 1

[75] 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 1 3 1 2 2 3 1 2 1 2 2 2 2 2 2 1 2 2 2

[112] 2 2 2 2 2 1 2 1 1 2 1 1 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 1

[149] 1 1 1 1 1 2 2 1 2 1 1 3 1 1 3 1 1 2 2 2 1 2 2 2 2 2 1 2 2 1 2 1 1 1 2 1 1

[186] 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 1 2 2 2 2 1 2 3 2 2 2 1 2 2 1 1 1 1

[223] 1 1 1 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 1 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2

[260] 2 2 2 2 2 1 2 1 1 2 1 2 3 1 1 1 2 1 1 1 1 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 1

[297] 2 2 2 2 2 2 3 2 2 1 1 2 2 2 1 2 2 1 1 1 1 2 1 1 3 1 1 1 3 1 3 1 3 3 1 1 2

[334] 2 1 1 2 2 1 2 2 2 2 1 2 2 1 2 1 2 2 1 1 3 1 2 1 1 2 3 2 1 2 2 2 2 2 2 1 2

[371] 2 1 2 2 2 2 2 2 2 1

**km1$centers**

XA06 XA31

1 9.561600 90.52720

2 14.284483 78.68707

3 6.647826 120.30000

**km1$silinfo**

$widths

cluster neighbor sil\_width

355 1 2 6.256970e-01

124 1 2 6.252728e-01

225 1 2 6.248356e-01

306 1 2 6.240719e-01

93 1 2 6.216066e-01

331 1 2 6.210798e-01

224 1 2 6.177074e-01

29 1 2 6.121028e-01

………………………………………………………………………………

$clus.avg.widths

[1] 0.4402580 0.4029323 0.3965386

$avg.width

[1] 0.4148235

Observations / comments:

* One see all slots available within the result object
* One can check the centers of clusters as well as silhouette statistics

**# there are also alternative commands for clustering**

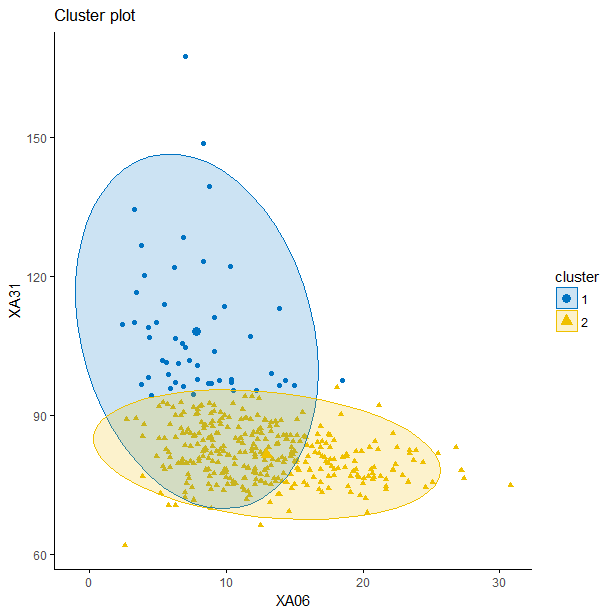
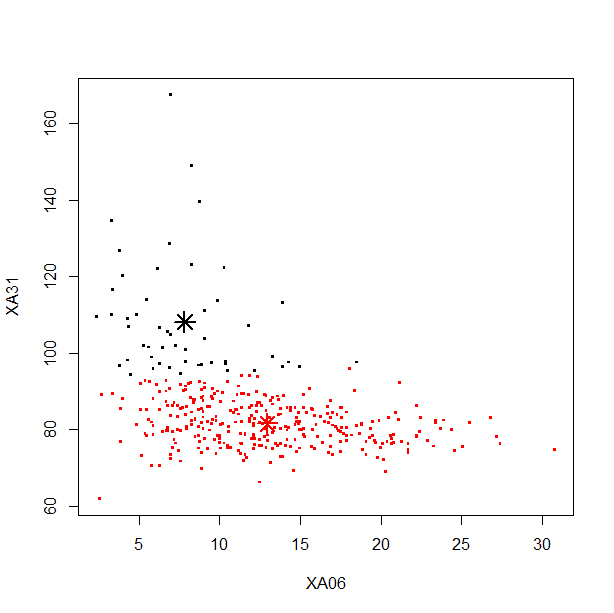
**# Alternatives with stats:: package**

**km10<-kmeans(xxx, 2) # stats::**

**plot(xxx, col = km10$cluster, pch=".", cex=3)**

**points(km10$centers, col = 1:5, pch = 8, cex=2, lwd=2)**

**fviz\_cluster(list(data=xxx, cluster=km10$cluster), ellipse.type="norm", geom="point", stand=FALSE, palette="jco", ggtheme=theme\_classic()) #factoextra::**



**# to get the silhouette plot for k-means one needs the dissimilarity matrix for given distance metric**

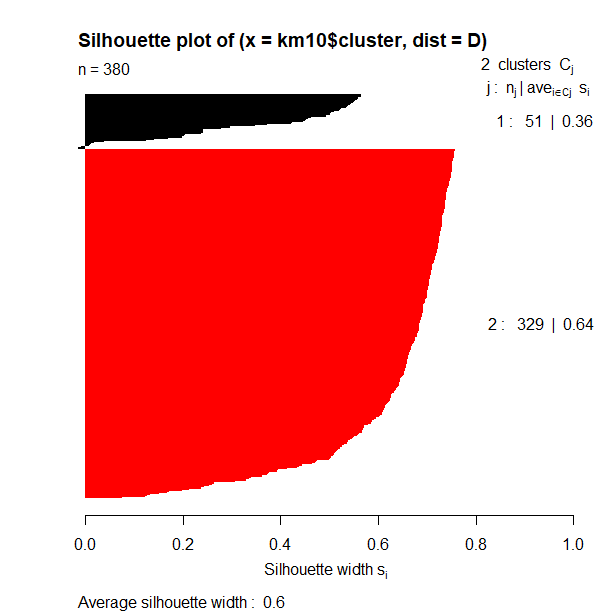
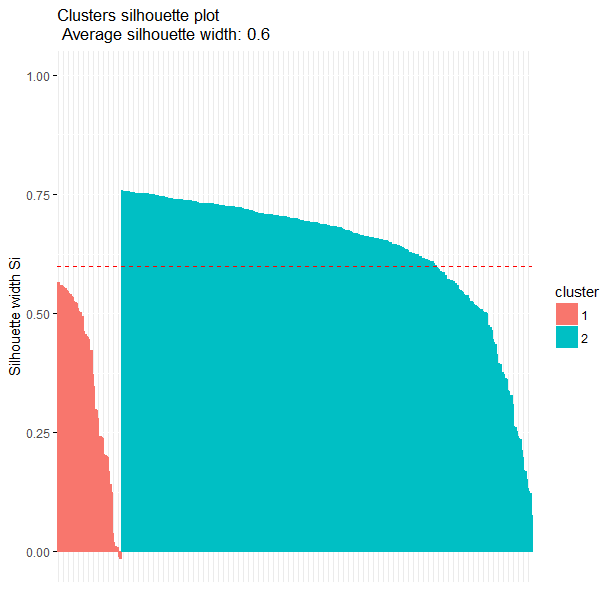
**D<-daisy(xxx) # calculates the dissimilarity matrix, cluster:: package**

**plot(silhouette(km10$cluster, D), col=1:2, border=NA)**

**# or apply dist() and fviz\_silhouette()**

**sil<-silhouette(km10$cluster, dist(xxx))**

**fviz\_silhouette(sil)**

**# Alternatives with flexclust:: package**

**km11<-cclust(xxx, k=4, simple=FALSE, save.data=TRUE) #flexclust:: class kcca**

**km11**

**plot(km11)**

**summary(km11)**

**attributes(km11) # checking the slots of output**

|  |  |
| --- | --- |
| call:  cclust(x = xxx, k = 4, simple = FALSE, save.data = TRUE)  cluster sizes:  Found more than one class "kcca" in cache; using the first, from namespace 'flexclust'  Also defined by ‘kernlab’  1 2 3 4  19 109 137 115 |  |

**# flexclust::kcca() - performs k-centroids clustering on a data matrix**

km13<-kcca(xxx, k=4, family=kccaFamily("kmedians"), control=list(initcent="kmeanspp"))

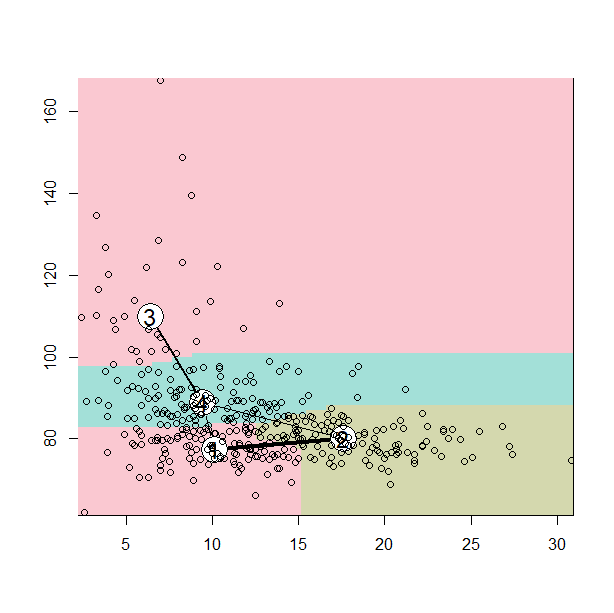
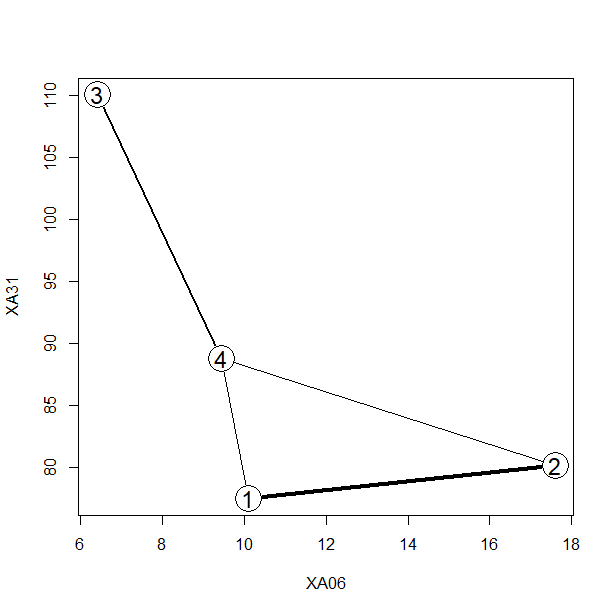
km13

parameters(km13) # to get centroids from KCCA object

plot(km13)

image(km13)

points(xxx)



Observations / comments:

* There are many packages available - each one generates different class of objects as well as different graphics
* Choose the method you like!

## 02. Clustering / PAM (Partitioning Around Medoids)

# PAM algorithm clusters data around medoids (real points from dataset)

# there are two phases of PAM:

# a) to select initial medoids and calculate objective function (BUILD)

# b) to change medoids for other and check the improvements (decrease) in fun. (SWAP)

# goal is to minimize the overall dissimilarity (mostly Manhattan distance)

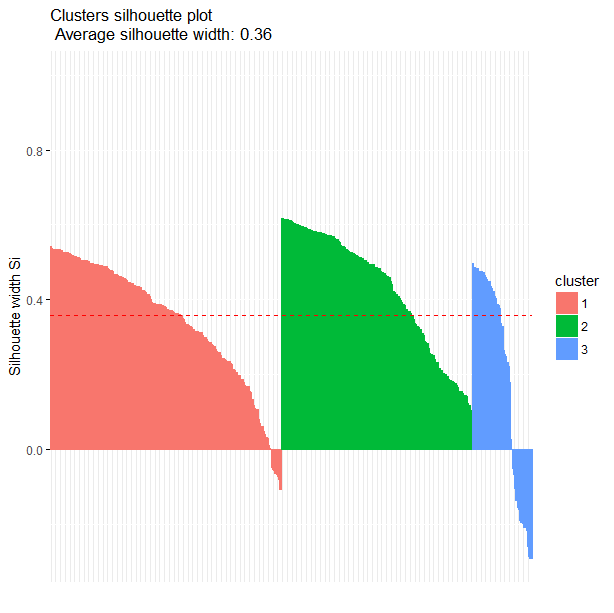
# between the representants of each cluster and its members

**# clustering with triangle graphics**

**pam1<-eclust(xxx, "pam", k=3) # factoextra::**

**fviz\_silhouette(pam1)**

**fviz\_cluster(pam1) #**

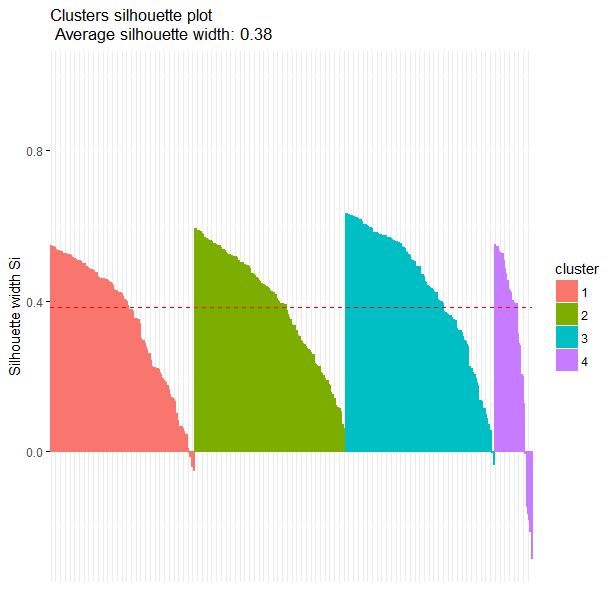
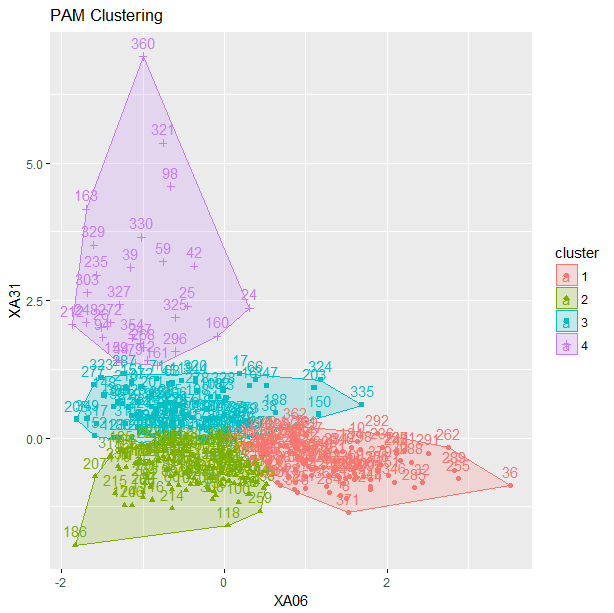


**# more clusters, other distance metric**

**pam2<-eclust(xxx, "pam", k=4, hc\_metric="manhattan") # factoextra::**

**fviz\_silhouette(pam2)**

**fviz\_cluster(pam2) #**



**attributes(pam1)**

$names

[1] "medoids" "id.med" "clustering" "objective" "isolation"

[6] "clusinfo" "silinfo" "diss" "call" "data"

[11] "clust\_plot" "nbclust"

$class

[1] "pam" "partition" "eclust"

**# Alternatives - one can use other functions to get the same result**

**pam3<-pam(xxx,3) #cluster::pam(), works for n<65536**

**summary(pam3)**

Medoids:

ID XA06 XA31

[1,] 192 15.3 78.2

[2,] 33 9.4 86.0

[3,] 268 7.0 104.7

Clustering vector:

[1] 1 1 1 1 1 1 2 1 2 1 2 3 2 1 2 3 3 2 1 2 1 1 1 3 3 3 2 2 2 1 1 1 2 2 1 1

[37] 2 2 3 1 1 3 2 3 3 2 1 1 2 1 2 1 1 2 1 1 1 1 3 1 1 1 3 1 1 3 1 2 1 1 3 1

[73] 2 2 2 2 1 1 3 1 1 1 1 2 1 1 1 1 2 2 1 2 2 3 2 1 1 3 2 1 2 1 1 1 2 2 1 2

[109] 1 2 1 2 1 1 1 1 2 1 3 2 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 2

[145] 2 2 2 2 2 2 3 2 2 1 2 2 1 2 3 3 3 2 3 2 2 2 2 1 2 1 1 2 1 1 2 2 2 2 2 2

[181] 2 2 2 2 2 1 2 2 1 1 1 1 2 1 1 2 2 2 1 1 2 1 2 1 2 1 2 2 1 3 1 3 2 1 1 2

[217] 1 1 2 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 3 1 2 1 1 2 2 1 1 1 1 1 3 3 1 1 1 1

[253] 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 3 2 2 1 3 2 2 2 1 3 2 2 2 2 2 2 1 1 2 3 1

[289] 1 1 1 1 2 2 1 3 1 1 1 1 1 2 3 1 1 2 2 1 2 2 2 1 2 2 2 2 2 1 2 3 3 2 3 3

[325] 3 2 3 2 3 3 2 2 1 2 2 2 1 2 2 1 1 1 2 3 1 1 3 1 2 1 1 2 2 3 2 1 2 2 1 3

[361] 1 2 1 1 1 1 1 2 2 1 1 3 1 1 1 1 1 1 1 2

Objective function:

build swap

6.206540 5.965357

Numerical information per cluster:

size max\_diss av\_diss diameter separation

[1,] 182 20.66349 5.528404 32.10685 0.500000

[2,] 151 13.32967 4.774162 23.88054 0.500000

[3,] 47 62.90000 11.484422 71.80843 1.649242

Isolated clusters:

L-clusters: character(0)

L\*-clusters: character(0)

Silhouette plot information:

cluster neighbor sil\_width

3 1 2 0.542118776

374 1 2 0.538231521

367 1 2 0.535453797

263 1 2 0.535213716

2 1 2 0.535074934

………………………..

129 1 2 -0.053199251

231 1 2 -0.105616991

241 2 1 0.617008252

………………………………

265 2 3 0.532297222

165 2 1 0.529268661

303 3 2 0.487714857

235 3 2 0.484640295

71 3 2 -0.103335194

Average silhouette width per cluster:

[1] 0.3434395 0.4284408 0.1904422

Average silhouette width of total data set:

[1] 0.358293

Available components:

[1] "medoids" "id.med" "clustering" "objective" "isolation"

[6] "clusinfo" "silinfo" "diss" "call" "data"

**class(pam3)**

**[1] "pam" "partition"**

Observations / comments:

* Clustering with PAM looks the same as k-means at the first glance, but the differences exist (especially in the small samples)

**# silhouette statistics can be extracted from pam objects as well as calculated with separate code**

**silhouette(pam3)**

cluster neighbor sil\_width

3 1 2 0.542118776

374 1 2 0.538231521

367 1 2 0.535453797

263 1 2 0.535213716

2 1 2 0.535074934

257 1 2 0.534896307

…………………………………………………………………

**silhouette(pam1) # also works with other class**

**# clustering vector can be extracted from pam object**

**pam3$clustering**

[1] 1 1 1 1 1 1 2 1 2 1 2 3 2 1 2 3 3 2 1 2 1 1 1 3 3 3 2 2 2 1 1 1 2 2 1 1 2 2 3 1 1 3 2 3 3 2 1 1

[49] 2 1 2 1 1 2 1 1 1 1 3 1 1 1 3 1 1 3 1 2 1 1 3 1 2 2 2 2 1 1 3 1 1 1 1 2 1 1 1 1 2 2 1 2 2 3 2 1

[97] 1 3 2 1 2 1 1 1 2 2 1 2 1 2 1 2 1 1 1 1 2 1 3 2 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2 1 1 1 1 1 1 2 2

[145] 2 2 2 2 2 2 3 2 2 1 2 2 1 2 3 3 3 2 3 2 2 2 2 1 2 1 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 1

[193] 2 1 1 2 2 2 1 1 2 1 2 1 2 1 2 2 1 3 1 3 2 1 1 2 1 1 2 2 2 2 2 2 2 1 2 1 1 2 1 1 1 1 3 1 2 1 1 2

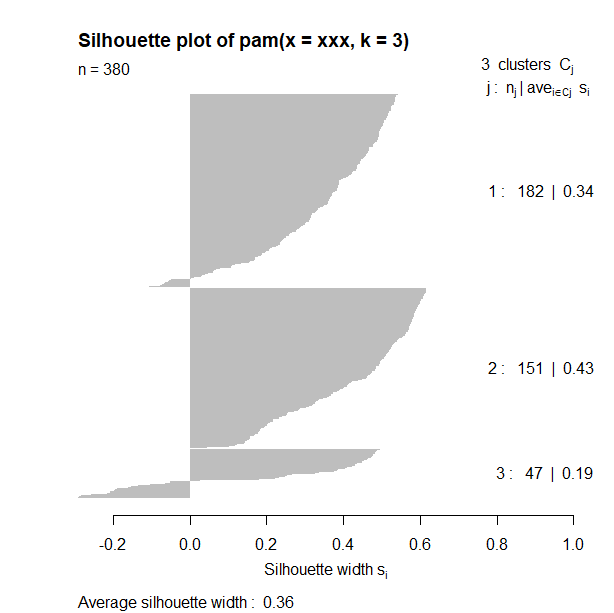
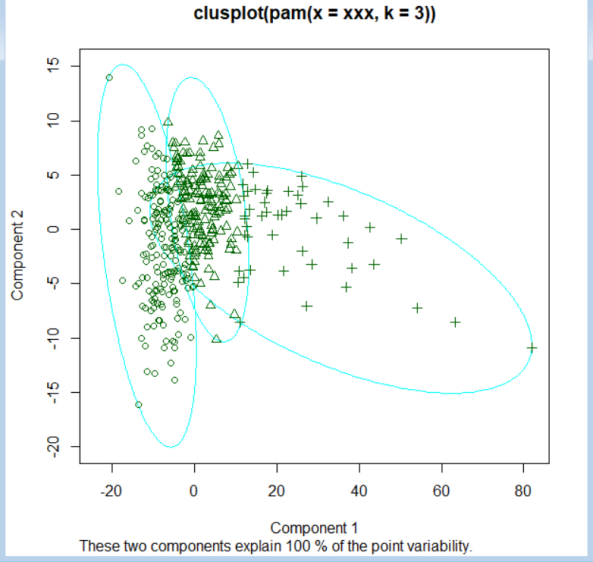
[241] 2 1 1 1 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 3 2 2 1 3 2 2 2 1 3 2 2 2 2 2 2 1 1 2 3 1

[289] 1 1 1 1 2 2 1 3 1 1 1 1 1 2 3 1 1 2 2 1 2 2 2 1 2 2 2 2 2 1 2 3 3 2 3 3 3 2 3 2 3 3 2 2 1 2 2 2

[337] 1 2 2 1 1 1 2 3 1 1 3 1 2 1 1 2 2 3 2 1 2 2 1 3 1 2 1 1 1 1 1 2 2 1 1 3 1 1 1 1 1 1 1 2

**# to run those figures please click the active Graphics Device**

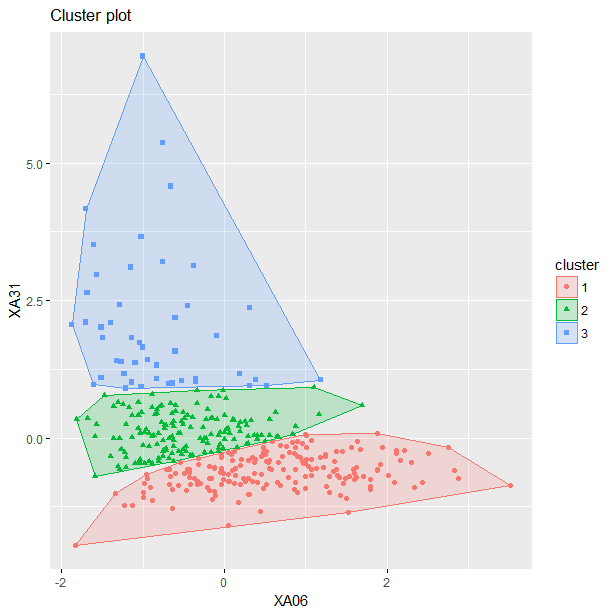
**plot(pam3) # in fact cluster::plot.partition() combined with cluster::clusplot()**



**# one can use ggplot() for pam object to get better graphics**

**fviz\_cluster(pam3, geom="point", ellipse.type="norm") # factoextra::**

**fviz\_cluster(pam3, geom="point", ellipse.type="convex") # factoextra::**

## 03. CLARA algorithms(CLustering LARge Applications)

# like PAM, but relies on the sampling approach and is designed for large data sets

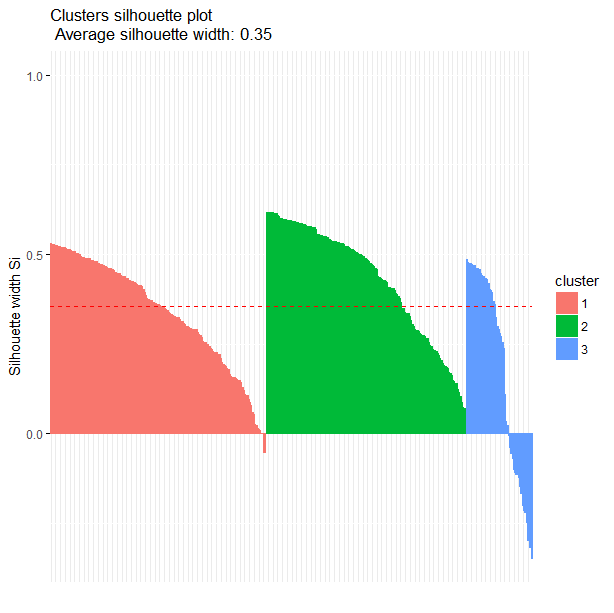
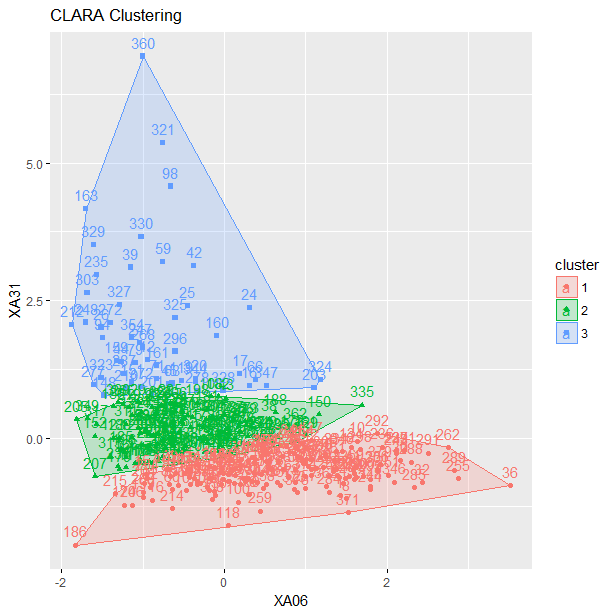
# PAM looks medoids for the entire data set and CLARA does the same for a sample

**cl2<-eclust(xxx, "clara", k=3) # factoextra**

**summary(cl2)**

**fviz\_cluster(cl2)**

**fviz\_silhouette(cl2)**



**# clara() from cluster::**

**cl1<-clara(xxx, 3, metric="euclidean", stand=FALSE, samples=5,**

**sampsize=50, trace=0, medoids.x=TRUE,**

**rngR=FALSE, pamLike=FALSE, correct.d=TRUE) #cluster::**

**class(cl1)**

[1] "clara" "partition"

**cl1**

Call: clara(x = xxx, k = 3, metric = "euclidean", stand = FALSE, samples = 5, sampsize = 50, trace = 0, medoids.x = TRUE, rngR = FALSE, pamLike = FALSE, correct.d = TRUE)

Medoids:

XA06 XA31

[1,] 13.0 77.0

[2,] 8.7 85.1

[3,] 6.5 101.3

Objective function: 6.207548

Clustering vector: int [1:380] 1 1 1 1 1 1 2 1 2 1 2 3 2 1 2 3 3 2 ...

Cluster sizes: 178 148 54

Best sample:

[1] 9 31 46 56 57 58 62 67 72 73 79 86 97 100 128 130 140 145

[19] 146 152 156 160 161 165 166 176 194 196 202 208 222 231 242 247 253 254

[37] 263 279 286 299 326 336 338 339 346 349 351 353 367 368

Available components:

[1] "sample" "medoids" "i.med" "clustering" "objective"

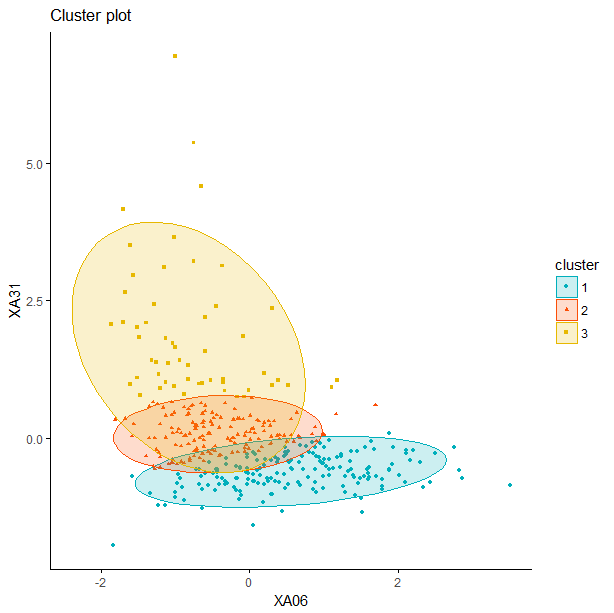
[6] "clusinfo" "diss" "call" "silinfo" "data"

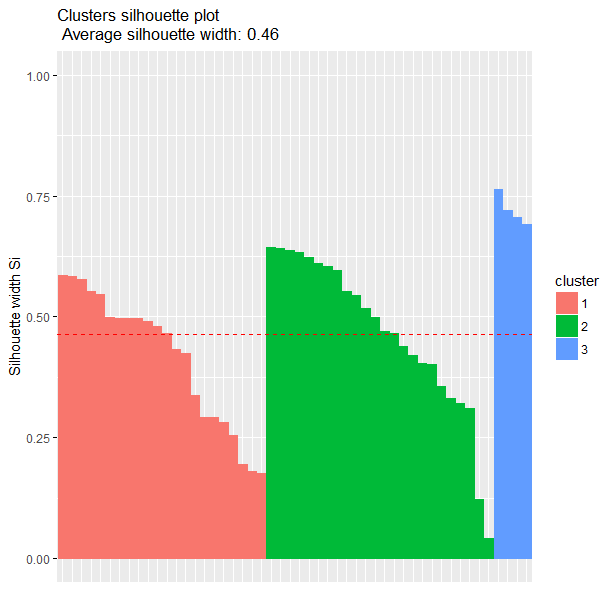
**# one can plot the result with ggplot::fviz\_cluster()**

**fviz\_cluster(cl1, geom = "point", ellipse.type = "norm") # factoextra::**

**fviz\_cluster(cl1, palette = c("#00AFBB", "#FC4E07", "#E7B800"), ellipse.type = "t", geom = "point", pointsize = 1, ggtheme = theme\_classic() )**

**fviz\_silhouette(cl1)**





Observations / comments:

* Result in general is the same as in PAM, as this is PAM solution for big data
* Depending on observations drawn, silhouette may be slightly different
* fviz\_cluster() allow for tailor-made skin of figure with graphics richness of ggplot2::

### TASK 1

Plot on single figure clustering points (from k-means and pam) and label them in legend. Prepare two figures:

1. for xxx subset
2. for yyy subset

For each subset test 2,3,4 and 5 clusters in k-means as well as in pam.

Instructions

km.xxx.2<-eclust(xxx, "kmeans", hc\_metric="euclidean",k=2)

km.xxx.2$centers

km.xxx.3<-eclust(xxx, "kmeans", hc\_metric="euclidean",k=3)

km.xxx.3$centers

plot(km.xxx.2$centers, col="black", pch=17, cex=1.1, xlim=c(0,31), ylim=c(60,170))

points(km.xxx.3$centers, col="red", pch=18, cex=1)

legend(20,160, c("KMeans k=2","KMeans k=3"), col=c("black","red"), pch=c(17,18), bty="n")

points(xxx, col="grey20", pch=".")

|  |  |
| --- | --- |
| Figure 1 – for xxx for different number of clusters for k-means and PAM | Figure 2 – for yyy for different number of clusters for k-means and PAM |
|  |  |