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

**Unsupervised learning**

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

Katarzyna Kopczewska, Associate professor

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

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

**# 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("ClusterR")

**library(cluster)**

**library(factoextra)**

**library(flexclust)**

**library(fpc)**

**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**

**# no labels at data – price\_what**

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

**months\_what<-price\_what[,2] # second column**

**product\_what<-colnames(price\_what[3:68]) # first row**

**price\_what<-as.matrix(price\_what[,3:68]) # data only**

Spis treści

[## 01. Clustering / K-means 3](#_Toc54548549)

[## 02. Clustering / PAM (Partitioning Around Medoids) 9](#_Toc54548550)

[## 03. CLARA algorithms (CLustering LARge Applications) 12](#_Toc54548551)

[## 04. Optimal number of clusters 14](#_Toc54548552)

[## 05. Statistics in clustered groups 15](#_Toc54548553)

## 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.

**# clustering of dataset – by individual obs. (in rows)**

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

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

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

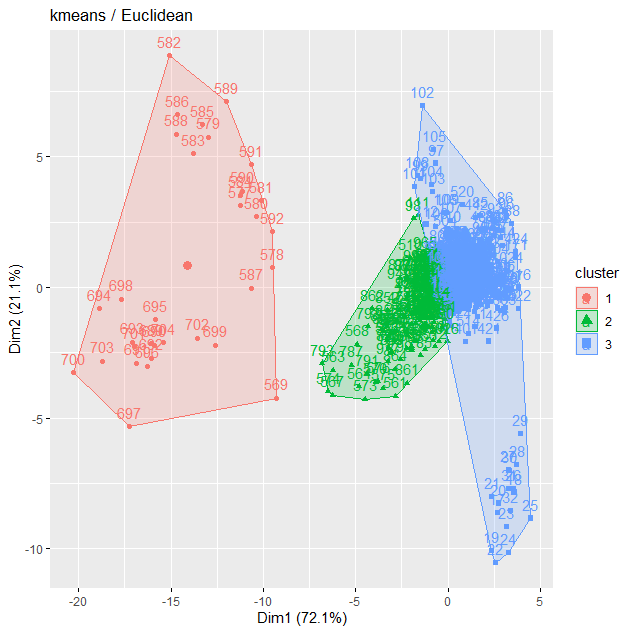
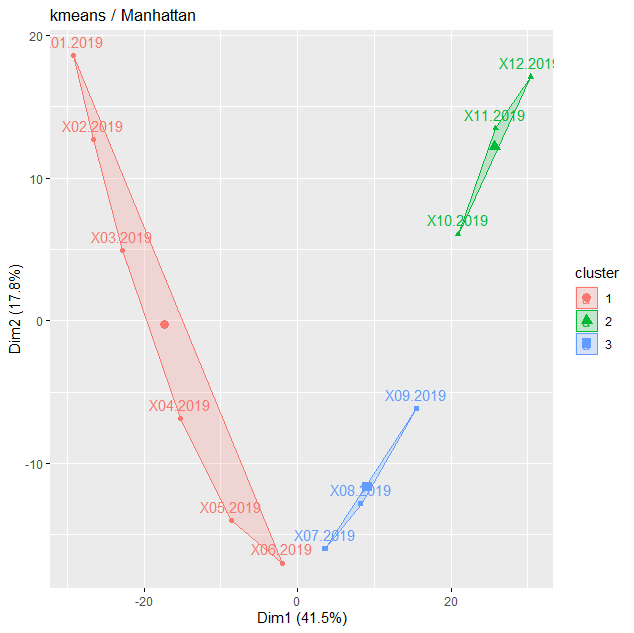
**km2<-eclust(t(price\_when), "kmeans", hc\_metric="euclidean",k=3)**

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

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

**km2<-eclust(t(price\_when)[,-x2], "kmeans", hc\_metric="euclidean",k=3)**

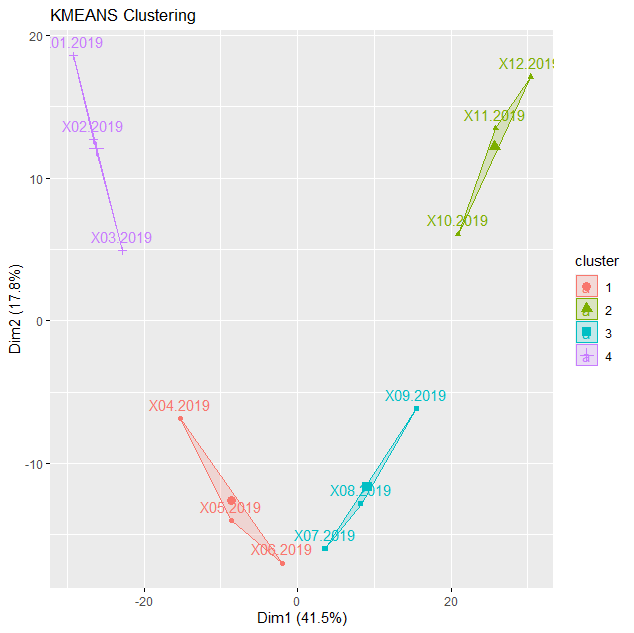
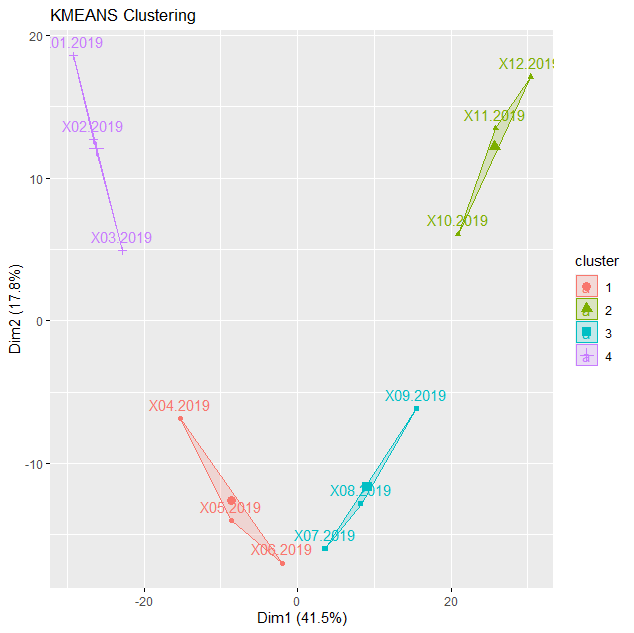
**fviz\_cluster(km2, main="kmeans / Manhattan")**

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

**km3<-eclust(t(price\_when)[,-x2], "kmeans", hc\_metric="euclidean",k=4)**

**km4<-eclust(t(price\_when)[,-x2],"kmeans", hc\_metric="manhattan", k=4)**



Observations / comments:

* One can easily change the analysed dataset and distance metric (Euclidean, Manhattan)
* One can easily change the number of clusters (k)
* When two variables analysed, they are on the axes on the figure, when more variables analysed, the axes are the “dimensions” which explain the process (as in PCA).

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

**attributes(km2)**

$names

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

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

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

$class

[1] "kmeans" "eclust"

**km2$cluster # each time period was assigned to cluster**

X01.2019 X02.2019 X03.2019 X04.2019 X05.2019 X06.2019 X07.2019 X08.2019 X09.2019

1 1 1 1 1 1 3 3 3

X10.2019 X11.2019 X12.2019

2 2 2

**km2$centers**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

1 104.1000 106.7667 99.96667 103.0500 104.93333 105.4 98.8500 103.4833 103.5333

2 105.3667 102.5333 101.03333 104.2333 98.13333 100.7 100.8667 106.4000 100.5667

3 102.1000 102.5667 98.40000 106.6667 99.36667 103.1 96.2000 104.4667 104.4000

[,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17]

1 105.2667 101.0333 98.01667 101.5833 103.7667 101.90000 105.0333 62.40000

2 105.1333 106.5667 104.43333 100.2000 103.0000 95.93333 100.7000 159.53333

3 103.8000 105.8000 102.30000 100.4667 109.7667 102.60000 103.3333 93.36667

**km2$silinfo**

$widths

cluster neighbor sil\_width

X03.2019 1 3 0.36310513

X02.2019 1 3 0.35506089

X04.2019 1 3 0.27966756

X01.2019 1 3 0.26242861

X05.2019 1 3 0.18657203

X06.2019 1 3 0.08092115

X11.2019 2 3 0.57448652

X12.2019 2 3 0.54485210

X10.2019 2 3 0.33111442

X08.2019 3 2 0.46913637

X07.2019 3 1 0.22288909

X09.2019 3 2 0.05194471

$clus.avg.widths

[1] 0.2546259 0.4834843 0.2479901

$avg.width

[1] 0.3101815

Observations / comments:

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

**Interpretation:**

Dataset: **price\_when**

**dim(price\_when)**

[1] 1040 12

**head(price\_when)**

Region Product.service X01.2019 X02.2019 X03.2019 X04.2019

1 DOLNOSLASKIE apple juice 105.8 111.2 105.8 101.6

2 KUJAWSKO-POMORSKIE apple juice 106.9 105.3 103.5 108.5

3 LODZKIE apple juice 101.8 101.8 99.5 100.8

4 LUBELSKIE apple juice 107.9 105.4 100.0 99.5

5 LUBUSKIE apple juice 109.7 107.1 98.1 111.0

6 MALOPOLSKIE apple juice 103.1 101.2 111.0 105.0

X05.2019 X06.2019 X07.2019 X08.2019 X09.2019 X10.2019 X11.2019 X12.2019

1 99.2 101.0 102.6 102.1 101.6 105.1 105.9 105.1

2 109.4 107.0 106.1 100.8 100.8 101.5 106.4 99.7

3 97.2 98.7 100.5 95.0 99.7 97.8 96.9 108.4

4 101.0 104.5 106.9 106.4 106.7 101.3 104.4 107.0

5 99.7 104.0 100.0 103.2 94.9 96.2 99.5 98.7

6 107.0 105.1 102.8 101.0 105.5 101.0 104.5 96.6

* In clustering: many points (as clustering looks at rows)
* In rows we have “time series” for each product in given place, so clusters collect similar time series (any product in any region) – we can say that e.g. apples in lubelskie behave similar as apple juice in Mazowieckie (if they are in the same cluster)
* Cluster means: in each of three clusters (which collect rows) we get the “middle” value of each period (column)
* Clustering vector is as long as many rows in dataset

**km1**

K-means clustering with 3 clusters of sizes 33, 175, 832

Cluster means:

X01.2019 X02.2019 X03.2019 X04.2019 X05.2019 X06.2019 X07.2019 X08.2019

1 153.1848 170.5848 182.7182 193.8424 220.2061 201.6970 186.9818 178.9545

2 104.8206 107.9971 109.8069 113.3554 115.3440 116.4263 117.8703 117.8429

3 101.1300 101.3755 101.6084 100.9415 100.9219 100.4756 100.6391 101.1913

X09.2019 X10.2019 X11.2019 X12.2019

1 170.0242 160.2394 148.2576 136.9000

2 118.6863 117.5583 117.8577 118.9366

3 101.4690 102.1661 102.8800 103.7731

Clustering vector:

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

[37] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

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

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

[145] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

**dim(t(price\_when))**

[1] 12 1040

**head(t(price\_when))**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]

X01.2019 105.8 106.9 101.8 107.9 109.7 103.1 102.7 110.8 104.5 110.8 101.1

X02.2019 111.2 105.3 101.8 105.4 107.1 101.2 97.2 103.3 104.8 104.3 103.2

X03.2019 105.8 103.5 99.5 100.0 98.1 111.0 101.3 112.8 100.2 111.9 100.8

X04.2019 101.6 108.5 100.8 99.5 111.0 105.0 100.5 98.9 103.5 108.6 95.0

X05.2019 99.2 109.4 97.2 101.0 99.7 107.0 94.8 94.5 103.0 94.4 105.3

X06.2019 101.0 107.0 98.7 104.5 104.0 105.1 96.6 100.6 105.2 101.6 100.8

[,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]

X01.2019 92.1 107.6 98.4 106.0 106.7 62.6 64.0 56.3 59.0 61.4 54.4

X02.2019 99.5 104.1 101.3 101.8 107.7 63.3 64.1 57.8 63.2 61.8 56.1

X03.2019 98.4 102.0 102.9 101.6 110.6 63.0 57.6 56.2 62.4 61.4 54.4

X04.2019 100.0 96.9 108.0 98.7 103.4 61.3 55.2 57.1 58.8 63.8 53.1

X05.2019 97.0 97.5 105.2 102.0 98.5 61.5 56.4 54.1 61.9 69.0 53.3

X06.2019 101.1 101.4 106.8 101.3 103.3 62.7 58.9 60.8 59.8 71.5 56.6

* In clustering: few points (exactly 12, for each month) (as clustering looks at rows)
* In rows we have periods for each product in given place, so clusters collect similar months (all products in all regions) – we can say that e.g. changes of prices in January (of all products in all regions) behave similar changes of prices in February (if they are in the same cluster)
* Cluster means: in each of three clusters (which collect rows) we get the value of each product in each region (columns)
* Clustering vector is as long as many rows in dataset

**km2**

K-means clustering with 3 clusters of sizes 6, 3, 3

Cluster means:

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]

1 104.1000 106.7667 99.96667 103.0500 104.93333 105.4 98.8500 103.4833

2 105.3667 102.5333 101.03333 104.2333 98.13333 100.7 100.8667 106.4000

3 102.1000 102.5667 98.40000 106.6667 99.36667 103.1 96.2000 104.4667

[,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]

1 103.5333 105.2667 101.0333 98.01667 101.5833 103.7667 101.90000 105.0333

2 100.5667 105.1333 106.5667 104.43333 100.2000 103.0000 95.93333 100.7000

3 104.4000 103.8000 105.8000 102.30000 100.4667 109.7667 102.60000 103.3333

[,17] [,18] [,19] [,20] [,21] [,22] [,23]

1 62.40000 59.36667 57.0500 60.8500 64.81667 54.6500 55.88333

2 159.53333 150.56667 167.6667 152.8333 152.33333 168.1667 157.53333

3 93.36667 90.10000 101.4333 100.3667 100.80000 102.7667 98.33333

Clustering vector:

X01.2019 X02.2019 X03.2019 X04.2019 X05.2019 X06.2019 X07.2019 X08.2019

1 1 1 1 1 1 3 3

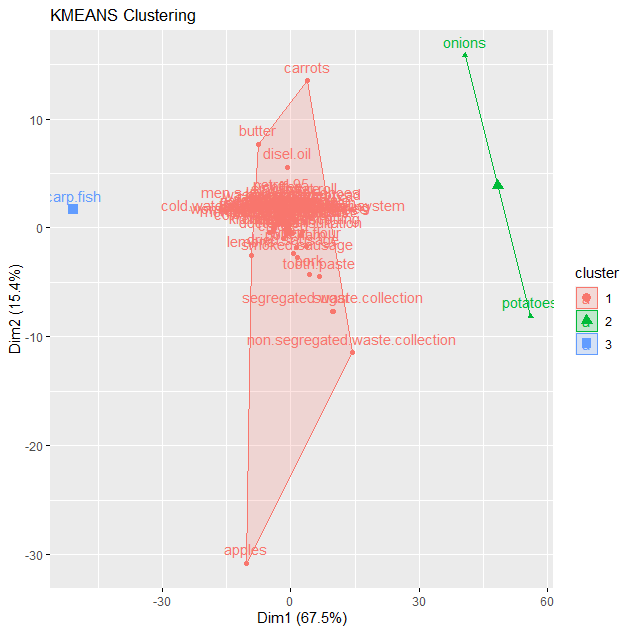
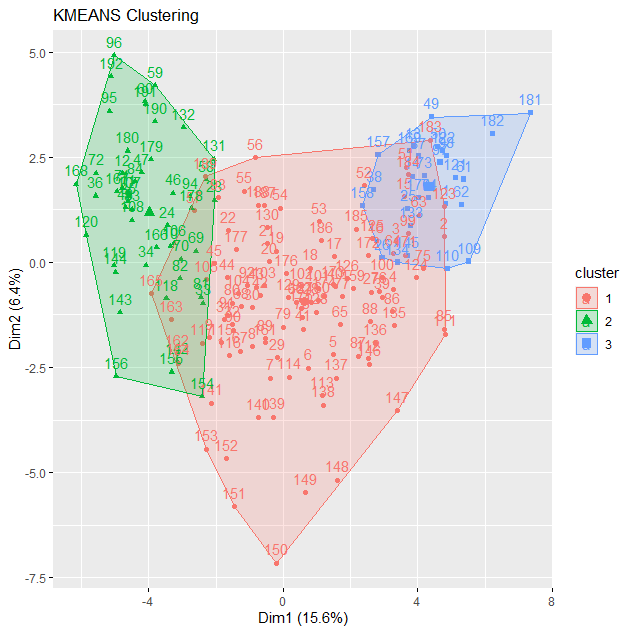
X09.2019 X10.2019 X11.2019 X12.2019

3 2 2 2

**# clustering on other dataset**

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

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

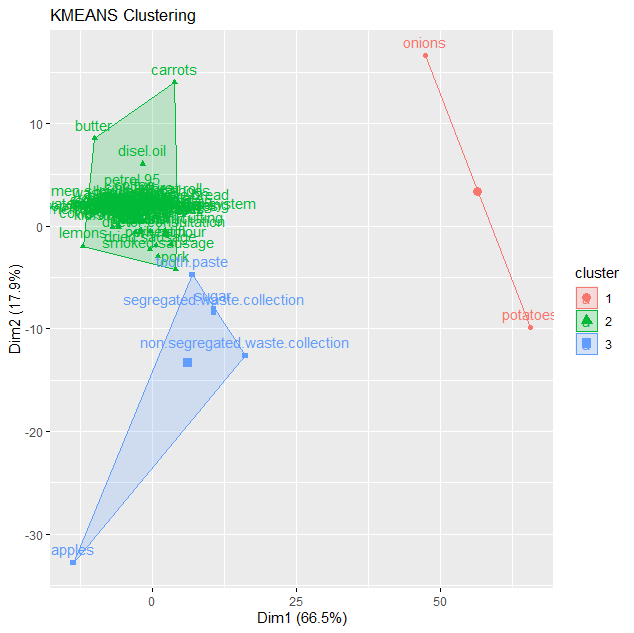


**rownames(t(price\_what))**

**# carp.fish in row 14**

**# analisis without carp fish**

**km1<-eclust(t(price\_what)[-14,], "kmeans", hc\_metric="euclidean",k=3)**



**# alternative commands for clustering\***

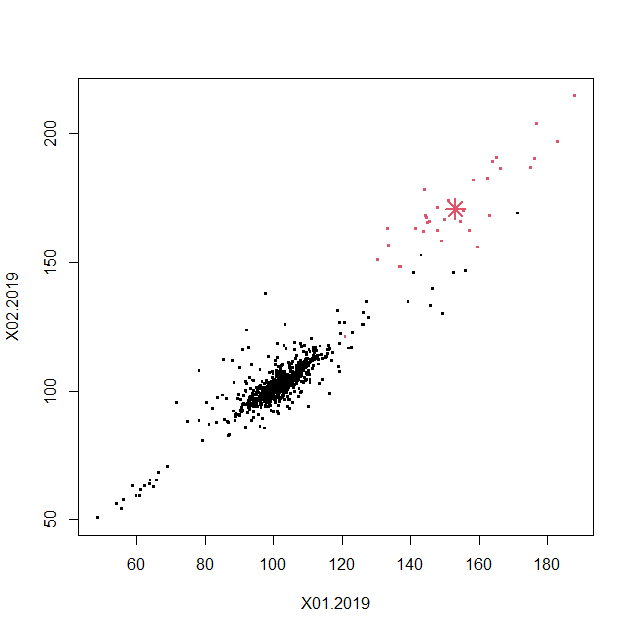
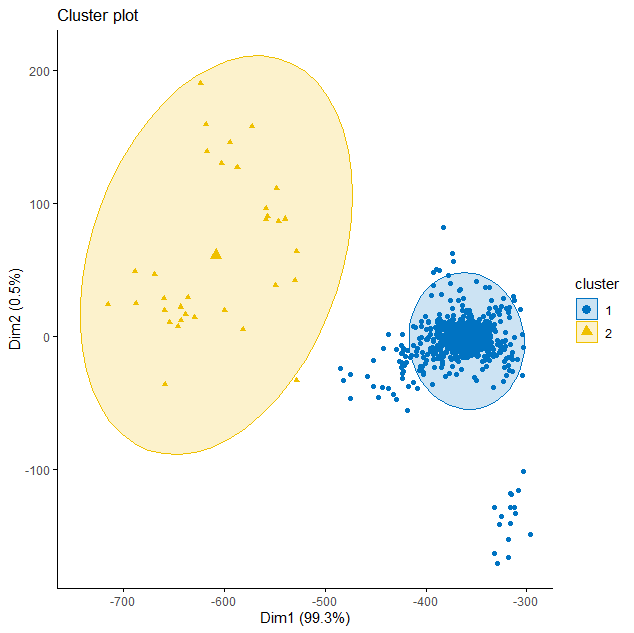
**# Alternatives with stats:: package & factoextra:: package**

**km10<-kmeans(price\_when, 2) # stats::**

**plot(price\_when, col = km10$cluster, pch=".", cex=3) # figure has only 2D**

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

**fviz\_cluster(list(data=price\_when, 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**

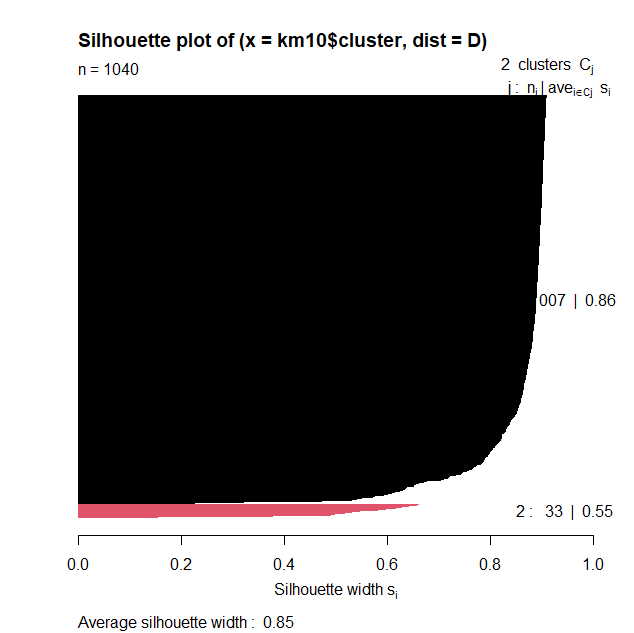
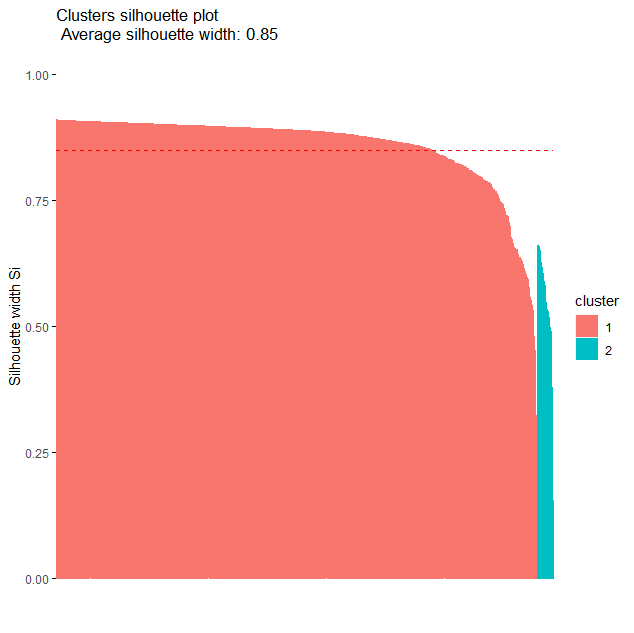
**D<-daisy(price\_when) # calculates the dissimilarity matrix, cluster::**

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

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

**sil<-silhouette(km10$cluster, dist(price\_when))**

**fviz\_silhouette(sil)**

**# Alternatives with flexclust:: package\***

**km11<-cclust(t(price\_when)[,-x2], k=4, simple=FALSE, save.data=TRUE) #flexclust:: class kcca**

**km11**

**plot(km11)**

**summary(km11)**

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

|  |  |
| --- | --- |
| kcca object of family ‘kmeans’  call:  cclust(x = t(price\_when)[, -x2], k = 4, simple = FALSE, save.data = TRUE)  cluster sizes:  1 2 3 4  3 3 3 3 |  |

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

**km13<-kcca(t(price\_when)[,-x2], k=4, family=kccaFamily("kmedians"), control=list(initcent="kmeanspp"))**

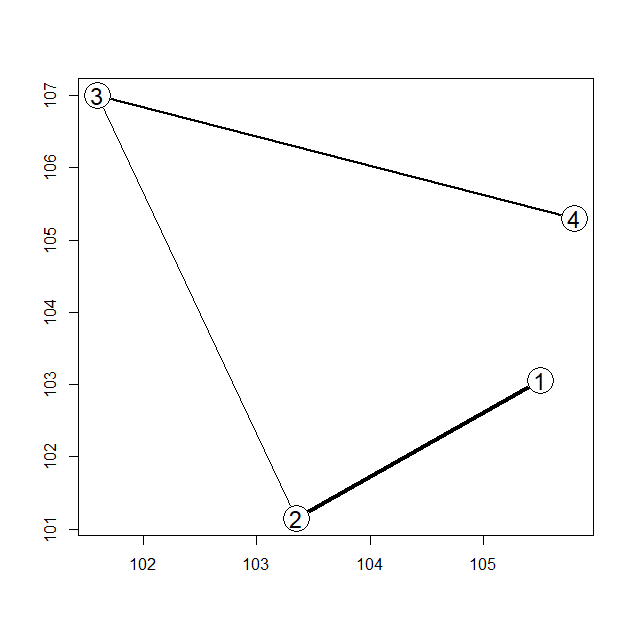
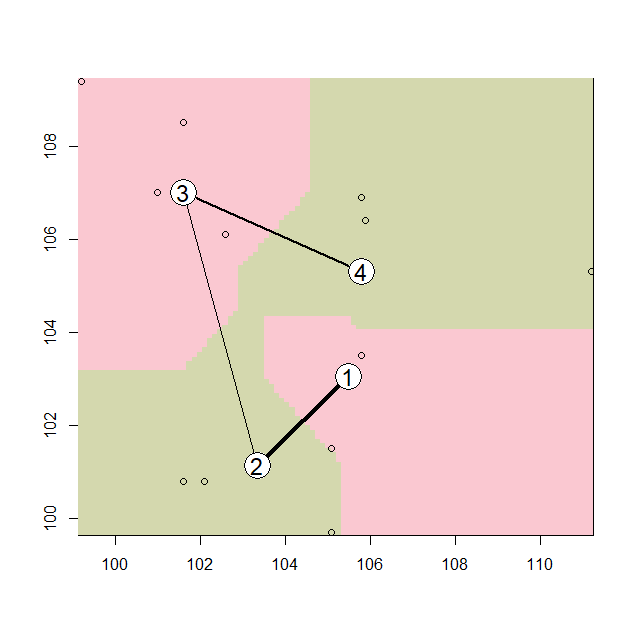
**km13**

**parameters(km13) # to get centroids from KCCA object**

**plot(km13)**

**image(km13)**

**points(t(price\_when)[,-x2])**

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)

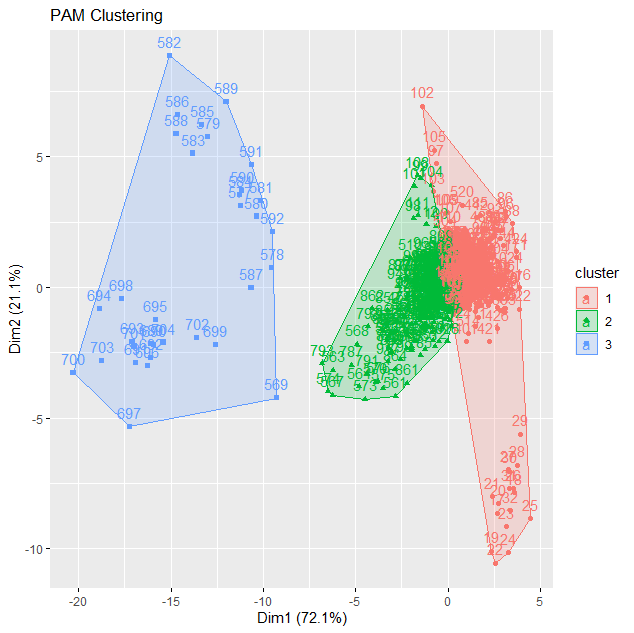
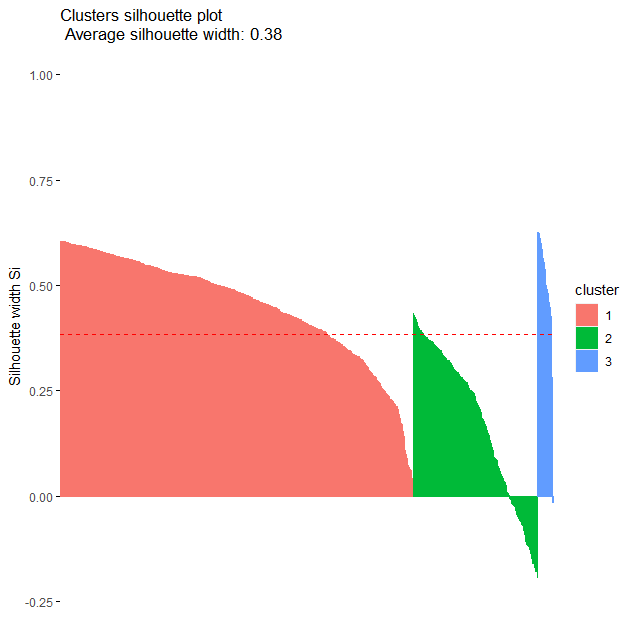
The goal is to minimize the overall dissimilarity (mostly Manhattan distance) between the representatives of each cluster and its members

**# clustering with triangle graphics**

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

**fviz\_silhouette(pam1)**

**fviz\_cluster(pam1) #**

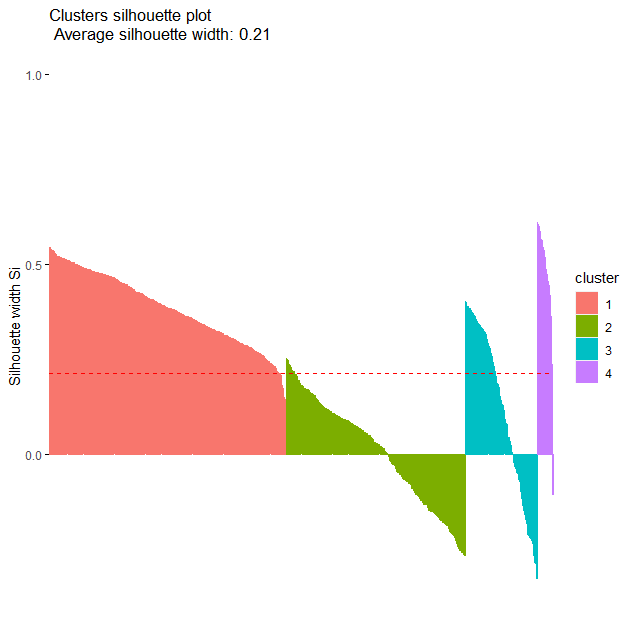
 

**# more clusters, other distance metric**

**pam2<-eclust(price\_when, "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(price\_when,3) #cluster::pam(), works for n<65536**

**summary(pam3)**

Medoids:

ID X01.2019 X02.2019 X03.2019 X04.2019 X05.2019 X06.2019 X07.2019 X08.2019

[1,] 889 100.6 101.6 101.6 101.6 101.6 101.6 101.6 101.6

[2,] 731 108.2 107.8 106.9 108.5 112.0 111.5 111.1 111.1

[3,] 695 154.7 165.6 186.7 194.6 199.4 206.9 203.2 204.7

X09.2019 X10.2019 X11.2019 X12.2019

[1,] 101.6 101.6 101.6 101.6

[2,] 111.1 111.5 111.5 111.5

[3,] 189.7 173.6 167.3 154.1

Clustering vector:

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[40] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

[79] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 2 1 1 1 2 1 1 2 2 1 2 1 2 1

[118] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1

[157] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1

Objective function:

build swap

23.84151 23.03219

Numerical information per cluster:

size max\_diss av\_diss diameter separation

[1,] 744 174.7276 18.45646 274.6563 3.612478

[2,] 263 113.2918 27.03515 160.8869 3.612478

[3,] 33 176.5256 94.29175 253.7035 46.624886

Isolated clusters:

L-clusters: character(0)

L\*-clusters: character(0)

Silhouette plot information:

cluster neighbor sil\_width

66 1 2 0.6045498632

68 1 2 0.6045498632

77 1 2 0.6045498632

157 1 2 0.6042785064

76 1 2 0.6038029112

364 1 2 0.6037239750

145 1 2 0.6036651161

494 1 2 0.6035808345

Average silhouette width per cluster:

[1] 0.4527316 0.1759953 0.4949754

Average silhouette width of total data set:

[1] 0.3840897

Available components:

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

[7] "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

66 1 2 0.6045498632

68 1 2 0.6045498632

77 1 2 0.6045498632

157 1 2 0.6042785064

76 1 2 0.6038029112…………………………………………………………………

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

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

**pam3$clustering**

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[40] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

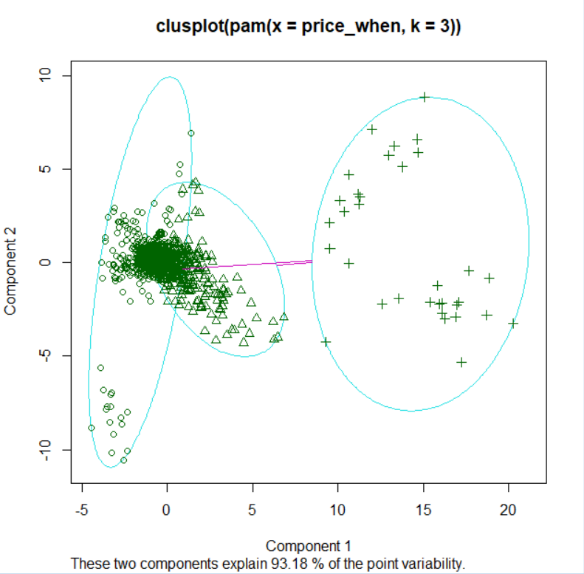
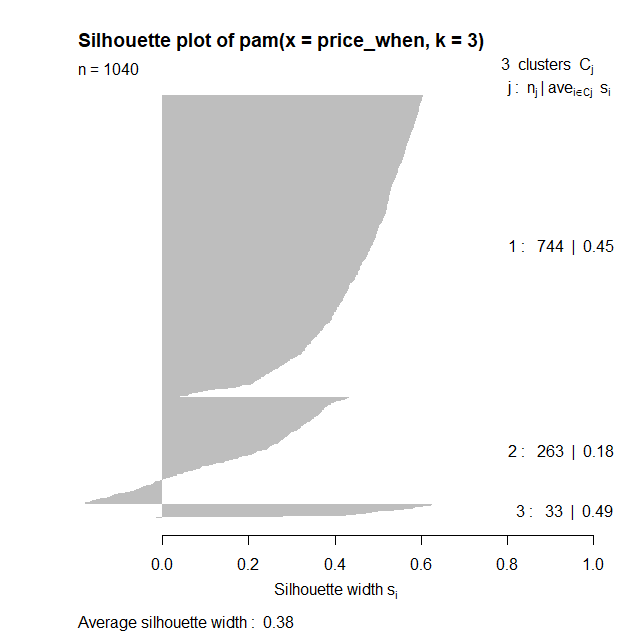
[79] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 2 1 1 1 2 1 1 2 2 1 2 1 2 1

[118] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1

[157] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1

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

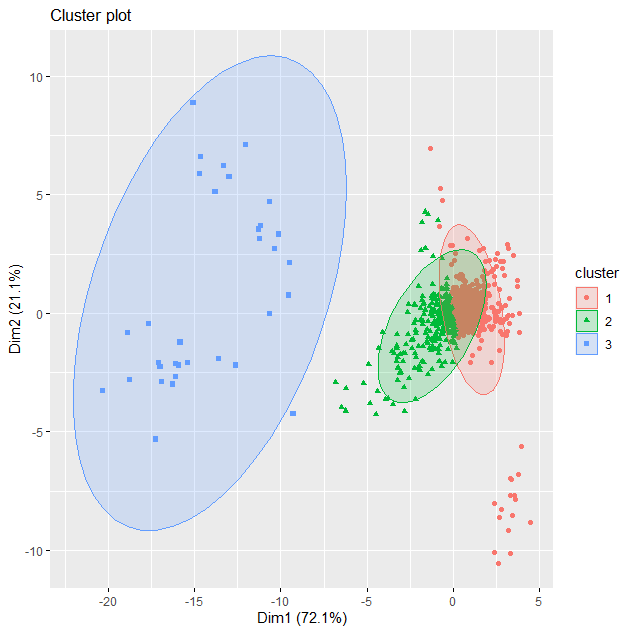
**plot(pam3) # in fact cluster::plot.partition() combined with cluster::clusplot()**

**# 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)

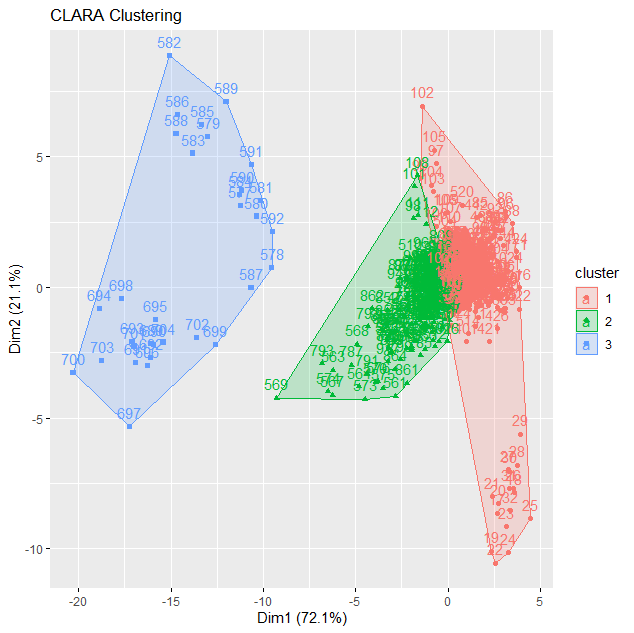
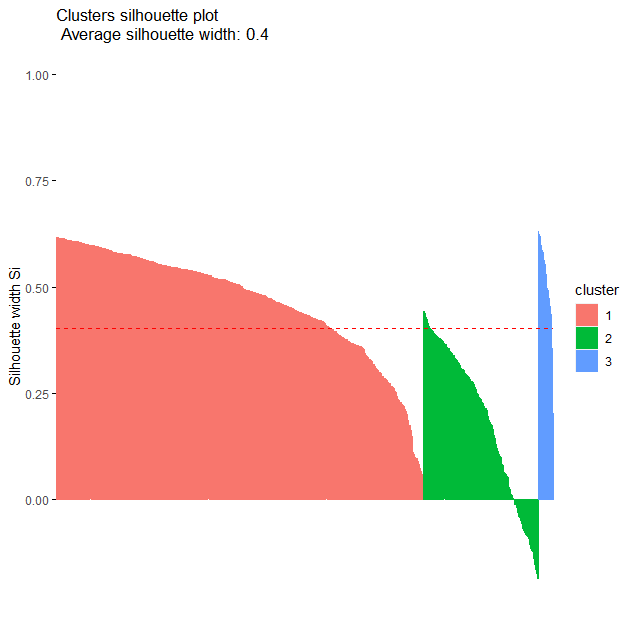
CLARA is like PAM, but relies on the sampling approach and is designed for large data sets. PAM looks medoids for the entire data set and CLARA does the same for a sample.

**cl2<-eclust(price\_when, "clara", k=3) # factoextra**

**summary(cl2)**

**fviz\_cluster(cl2)**

**fviz\_silhouette(cl2)**

# cluster size ave.sil.width

#1 1 769 0.47

#2 2 239 0.19

#3 3 32 0.51

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

**cl1<-clara(price\_when, 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 = price\_when, k = 3, metric = "euclidean", stand = FALSE, samples = 5, sampsize = 50, trace = 0, medoids.x = TRUE, rngR = FALSE, pamLike = FALSE, correct.d = TRUE)

Medoids:

X01.2019 X02.2019 X03.2019 X04.2019 X05.2019 X06.2019 X07.2019 X08.2019

[1,] 102.1 102.1 102.1 102.1 101.7 101.0 100.0 101.4

[2,] 102.2 108.6 109.7 109.7 109.7 109.7 111.3 111.3

[3,] 176.4 189.9 199.1 205.7 240.2 209.4 191.1 192.3

X09.2019 X10.2019 X11.2019 X12.2019

[1,] 101.9 101.9 101.9 102.6

[2,] 111.3 111.3 111.3 108.9

[3,] 183.6 186.1 174.3 166.7

Objective function: 23.56615

Clustering vector: int [1:1040] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 ...

Cluster sizes: 744 264 32

Best sample:

[1] 11 16 35 62 68 98 133 137 155 193 217 228 286 300 318 327 337 348 452 458

[21] 494 495 501 558 580 630 637 642 675 693 698 706 707 727 750 785 798 833 837 883

[41] 895 900 907 923 926 944 953 970 989 990

Available components:

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

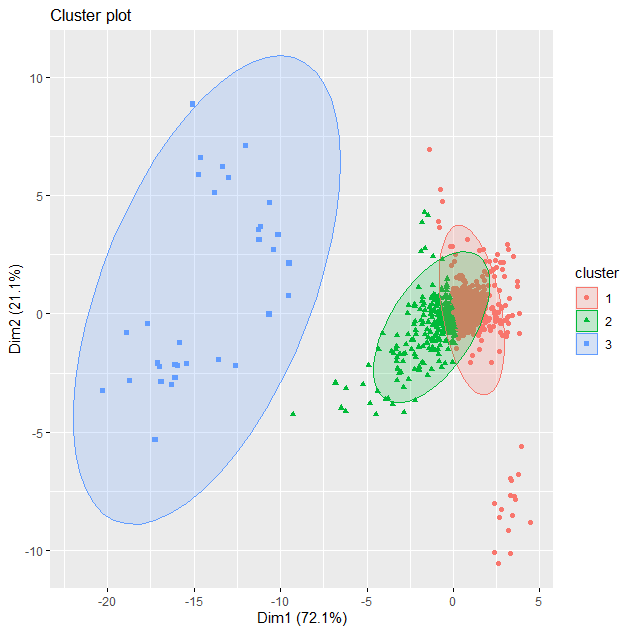
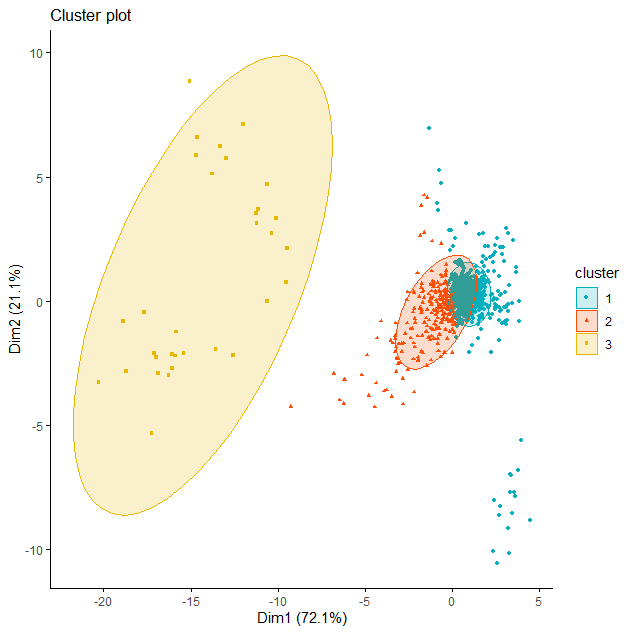
[7] "diss" "call" "silinfo" "data"

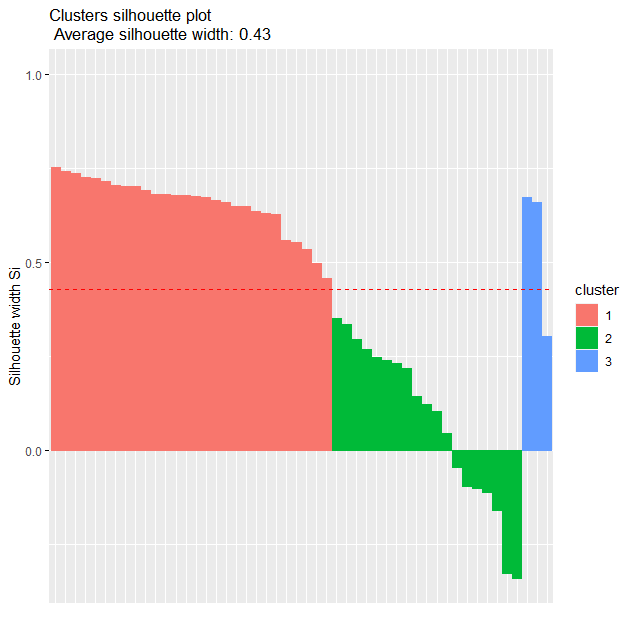
**# one can plot the result with ggplot::fviz\_cluster()**

**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::

## 04. Optimal number of clusters

**checking how many clusters should be**

**a) defines temporary clusters with kmeans, pam, clara, funny, hcut**

**b) calculates the statistics for clusters number:**

**- silhouette, - total within sum of square (wss) and - gap statistics**

**c) plots the result**

**Shilhouette**

Method of interpretation and validation of consistency within clusters of data

Shilhouette statistics s=(bi–ai)/max(ai,bi)

ai 🡪 average distance to all other objects in the cluster

bi 🡪 minimum of average distance to other clusters (cluster by cluster)

Statistics is between -1 and 1

negative s – undesirable, a>b other clusters are closer than “our” cluster

positive s – desirable, good when a~0 (distance in our cluster), then s~1

More at: <https://cs.fit.edu/~pkc/classes/ml-internet/silhouette.pdf>

**Gap statistic**

Method which compares empirical pattern with theoretical (uniform) distribution

One derives for comparisons twice “the log of within cluster dispersion” in empirical and theoretical case

One looks for biggest GAP between those two values – this indicates the optimal number of clusters

More at: <https://statweb.stanford.edu/~gwalther/gap>

**Options: FUNcluster 🡪 kmeans, cluster::pam, cluster::clara, hcut, etc.**

**method 🡪 "silhouette", "wss", "gap\_stat"**

**fviz\_nbclust(price\_when, FUNcluster=cluster::pam) # factoextra::**

**fviz\_nbclust(price\_when, FUNcluster=cluster::clara, method="gap\_stat")+ theme\_classic() # factoextra::**

|  |  |
| --- | --- |
|  |  |

Observations / comments:

* The best division is for maximum silhouette
* One should be aware that the differences in silhouette between groupings may be small
* One should note that different measures can give different indications
* Non-clusterable data may have very different info on best k

## 05. Statistics in clustered groups

Clustering methods are often used in segmentation. They are to split the sample into heterogeneous groups. In consequence, as a post-action we want to see the descriptive statistics in the defined groups. By assumption, they should be different.

**# stripes for k-means**

**# to plot with stripes the distance of data points to cluster centroids**

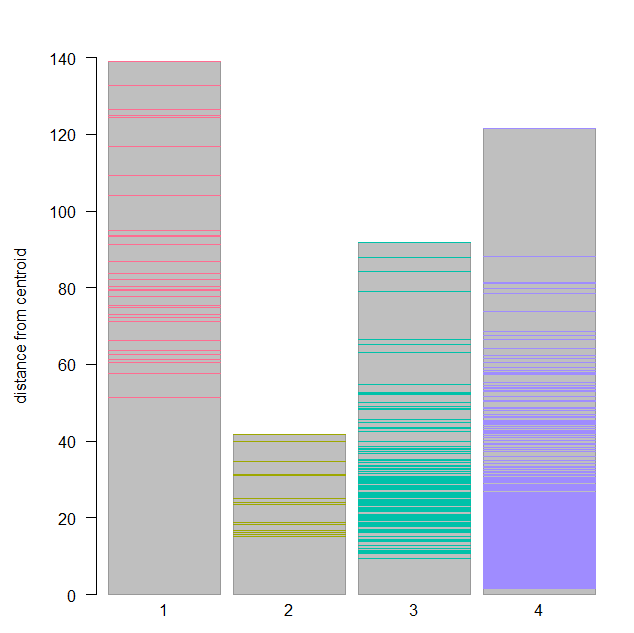
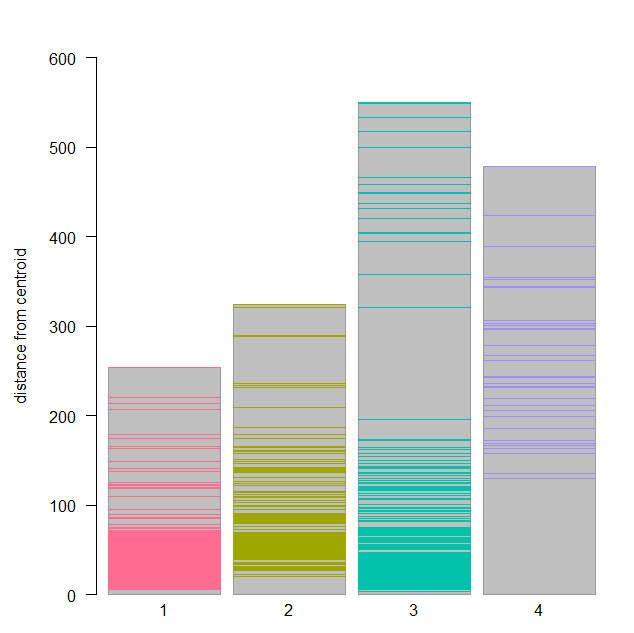
**# stripes() works only with class kcca (from cclust())**

**d1<-cclust(price\_when, 4, dist="euclidean") #flexclust::**

**stripes(d1) #flexclust::**

**d2<-cclust(price\_when, 4, dist="manhattan") #flexclust::**

**stripes(d2) #flexclust::**

Observations / comments:

* On stripes chart we see the distance of every single observation from the centroid of cluster. The higher bin the more distant locations of points within given cluster (undesirable)

**# boxplots for variables in groups**

**# the code below links to kmeans() command and operates on clustering object**

**# for k-means**

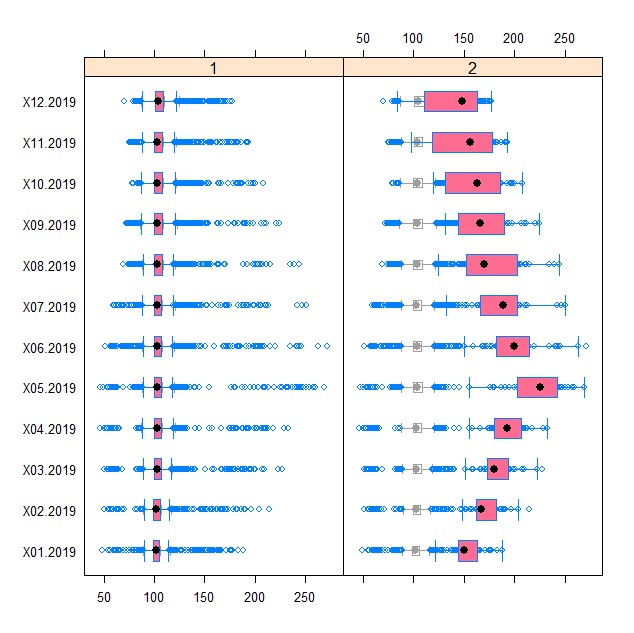
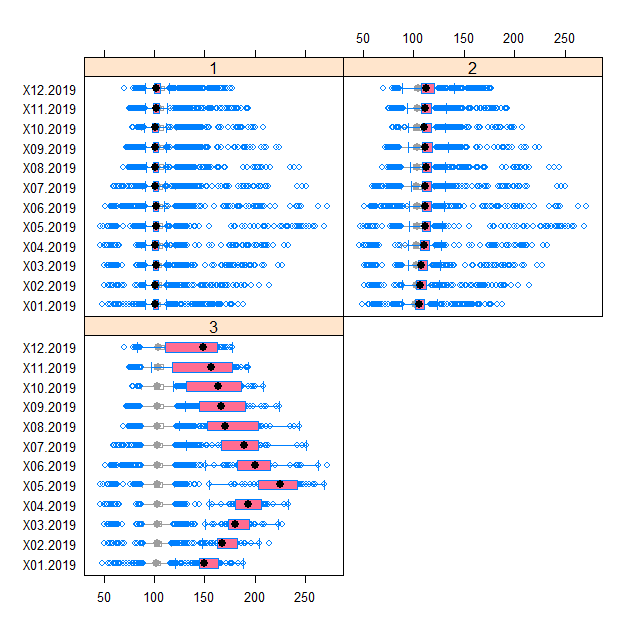
**km1<-kmeans(price\_when, 2) # stats::**

**groupBWplot(price\_when, km1$cluster, alpha=0.05) #flexclust::**

**# for PAM**

**c1<-pam(price\_when,3)#cluster::**

**groupBWplot(price\_when, as.factor(c1$clustering), alpha=0.05) #flexclust::**

**# One can use also the standard commands for boxplots and other statistical analyses.**

**# We need only to add to the subset the clustering vector obtained from k-means procedure.**

**# Than we use this as a grouping variable.**

**# data subset with information on cluster**

**xxc<-as.data.frame(cbind(price\_when, km1$cluster))**

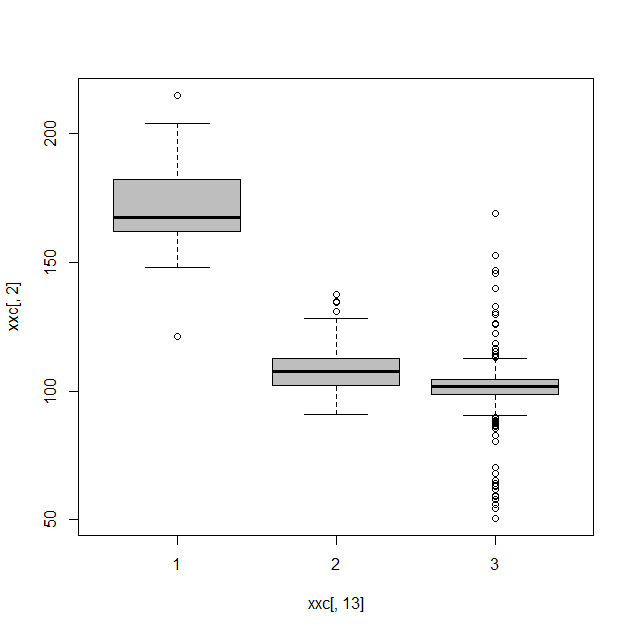
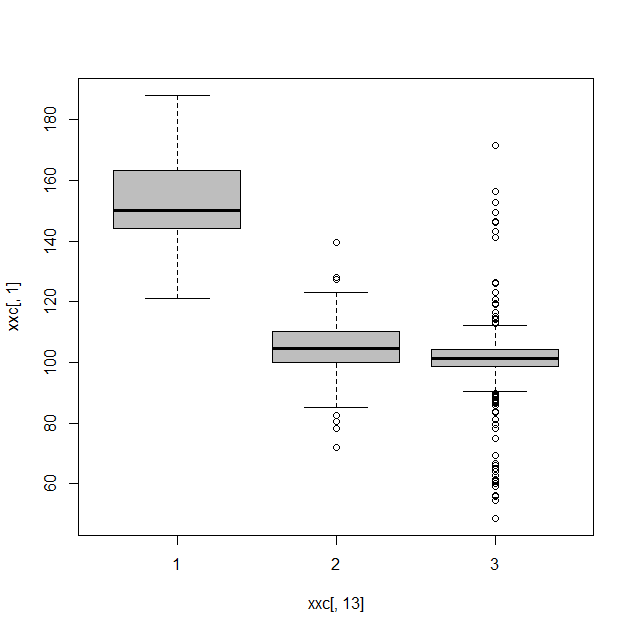
**colnames(xxc)[1:12]<-months\_when**

**colnames(xxc)[13]<-"clust"**

**# boxplot in a typical way – each graph is for separate variable**

**boxplot(xxc[,1]~xxc[,13], vertical=TRUE, col="grey")**

**boxplot(xxc[,2]~xxc[,13], vertical=TRUE, col="grey")**

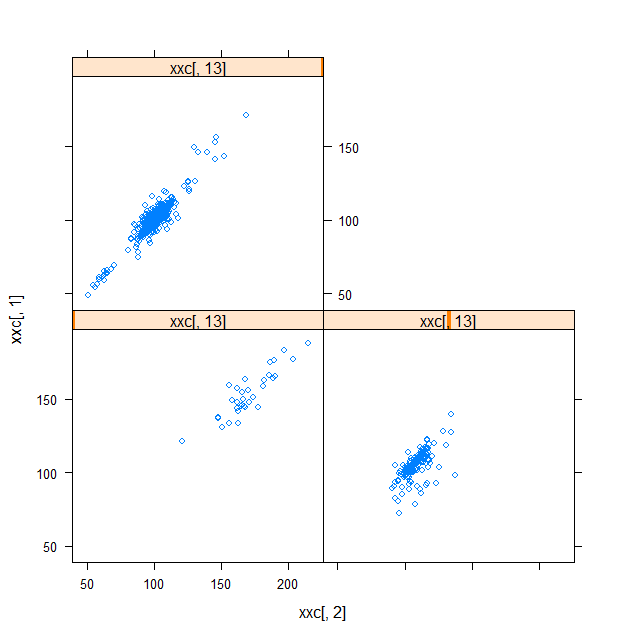


**# dotplots in groups**

**install.packages("lattice")**

**library(lattice)**

**xyplot(xxc[,1] ~ xxc[,2] | xxc[,13], data=xxc)**



Observations / comments:

* Histograms and dotplots prove that values of observations on both groups are different.

**Beyond graphical summaries, one can easily summarize the data with descriptive statistics. There are few methods for this.**

**# typical statistics**

**xxc<-as.data.frame(cbind(price\_when, km1$cluster))**

**colnames(xxc)[1:12]<-months\_when**

**colnames(xxc)[13]<-"clust"**

**#install.packages("psych")**

**library(psych)**

**describeBy(xxc[,1:12], xxc[,13])**

Descriptive statistics by group

INDICES: 1

vars n mean sd median trimmed mad min max range skew kurtosis se

X01.2019 1 33 153.18 15.71 150.0 152.68 14.23 121.0 187.9 66.9 0.30 -0.51 2.73

X02.2019 2 33 170.58 18.15 167.3 170.26 15.72 121.0 214.6 93.6 0.04 0.59 3.16

X03.2019 3 33 182.72 19.15 179.9 182.50 14.83 132.6 227.4 94.8 0.07 0.54 3.33

X04.2019 4 33 193.84 18.54 193.3 194.02 19.57 154.7 232.7 78.0 -0.06 -0.50 3.23

X05.2019 5 33 220.21 27.73 225.2 221.11 28.32 154.7 269.2 114.5 -0.31 -0.72 4.83

X06.2019 6 33 201.70 29.95 200.0 199.91 26.09 150.3 271.8 121.5 0.57 -0.47 5.21

X07.2019 7 33 186.98 28.70 189.1 185.52 25.50 132.1 250.4 118.3 0.30 -0.32 5.00

X08.2019 8 33 178.95 32.33 170.5 177.63 36.77 124.4 244.1 119.7 0.29 -1.06 5.63

X09.2019 9 33 170.02 28.11 166.1 168.51 34.40 130.8 224.4 93.6 0.31 -1.32 4.89

X10.2019 10 33 160.24 29.26 163.2 160.05 37.66 119.1 208.1 89.0 0.01 -1.64 5.09

X11.2019 11 33 148.26 31.34 156.6 148.20 43.00 97.3 193.0 95.7 -0.04 -1.67 5.46

X12.2019 12 33 136.90 27.77 148.3 137.56 32.32 83.5 177.4 93.9 -0.16 -1.59 4.83

----------------------------------------------------------------------------

INDICES: 2

vars n mean sd median trimmed mad min max range skew kurtosis se

X01.2019 1 175 104.82 9.04 104.7 105.10 7.56 71.9 139.4 67.5 -0.21 1.86 0.68

X02.2019 2 175 108.00 8.07 107.5 107.64 7.86 90.9 137.5 46.6 0.72 1.26 0.61

Observations / comments:

* With describeBy() we get in a simple way all statistics in both groups for both variables.

**#install.packages("doBy")**

**library(doBy)**

**fun<-function(x) {c(m=mean(x), s=sd(x), n=length(x))}**

**summaryBy(cbind(X01.2019, X02.2019, X03.2019)~clust, data=xxc, FUN=fun)**

clust X01.2019.m X01.2019.s X01.2019.n X02.2019.m X02.2019.s X02.2019.n

1 1 153.1848 15.710668 33 170.5848 18.149313 33

2 2 104.8206 9.037440 175 107.9971 8.074064 175

3 3 101.1300 9.230835 832 101.3755 8.923320 832

X03.2019.m X03.2019.s X03.2019.n

1 182.7182 19.153467 33

2 109.8069 7.735666 175

3 101.6084 8.717552 832

Observations / comments:

* With summaryBy() we define in formula as the LHS the variables to be analysed and as the RHS the grouping variable, while in FUN we specify statistics to be calculated

**# with aggregate function**

**s1<-aggregate(xxc[,1], by=list(xxc[,13]), mean)**

**s2<-aggregate(xxc[,1], by=list(xxc[,13]), sd)**

Group.1 x

1 1 153.1848

2 2 104.8206

3 3 101.1300

**s2**

Group.1 x

1 1 15.710668

2 2 9.037440

3 3 9.230835

Observations / comments:

* The aggregate() command always outputs the column Group.1 and x and whole object is *data.frame*.
* In this command one can run in one line only a single statistics for a single variable (multi grouping criteria are allowed)

**# summary with data.table**

**#install.packages("data.table")**

**library(data.table)**

**dt<-data.table(xxc)**

**dt[,list(mean=mean(X01.2019),sd=sd(X01.2019)),by=clust]**

clust mean sd

1: 3 101.1300 9.230835

2: 2 104.8206 9.037440

3: 1 153.1848 15.710668

**# statistics in a loop**

**# for each variable by groups, for many statistics**

**stats<-matrix(0, nrow=6, ncol=4)**

**colnames(stats)<-c("mean","sd","min", "max")**

**rownames(stats)<-rep(c("cluster1","cluster2", "cluster3"),times=2)**

**rownames(stats)<-paste(rownames(stats), rep(c("var1","var2"), each=3))**

**funs<-c("mean","sd","min", "max")**

**for(i in 1:2){ # iterating by variables**

**for(j in 1:4){ # iterating by functions**

**temp<-aggregate(xxc[,i], by=list(xxc[,13]), funs[j])**

**stats[(3\*i-2):(3\*i),j]<-temp$x}}**

mean sd min max

cluster1 var1 153.1848 15.710668 121.0 187.9

cluster2 var1 104.8206 9.037440 71.9 139.4

cluster3 var1 101.1300 9.230835 48.7 171.4

cluster1 var2 170.5848 18.149313 121.0 214.6

cluster2 var2 107.9971 8.074064 90.9 137.5

cluster3 var2 101.3755 8.923320 50.7 168.9

**# significance tests for differences between groups**

**# reading data once again and keeping headers**

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

summary(price\_when)

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

library(rstatix)

library(ggpubr)

options(tibble.print\_max = 50)

km<-kmeans(**price\_when[,3:14]**, 3) # k-means clustering

price\_when$group<-km$cluster # clustering vector added

price\_when %>% anova\_test(X07.2019 ~ group)

#Coefficient covariances computed by hccm()

#ANOVA Table (type II tests)

#

# Effect DFn DFd F p p<.05 ges

#1 group 1 1038 1619.309 4.19e-214 \* 0.609

res.aov<-price\_when %>% anova\_test(X07.2019 ~ group)

get\_anova\_table(res.aov, correction="GG")

#ANOVA Table (type II tests)

#

# Effect DFn DFd F p p<.05 ges

#1 group 1 1038 1619.309 4.19e-214 \* 0.609

**# individual in pairs**

pwc<-price\_when %>%

pairwise\_t\_test(X07.2019 ~ group, p.adjust.method="bonferroni")

pwc

# A tibble: 3 x 9

.y. group1 group2 n1 n2 p p.signif p.adj p.adj.signif

\* <chr> <chr> <chr> <int> <int> <dbl> <chr> <dbl> <chr>

1 X07.2019 1 2 832 175 4.27e- 91 \*\*\*\* 1.28e- 90 \*\*\*\*

2 X07.2019 1 3 832 33 3.61e-295 \*\*\*\* 1.08e-294 \*\*\*\*

3 X07.2019 2 3 175 33 2.51e-208 \*\*\*\* 7.53e-208 \*\*\*\*

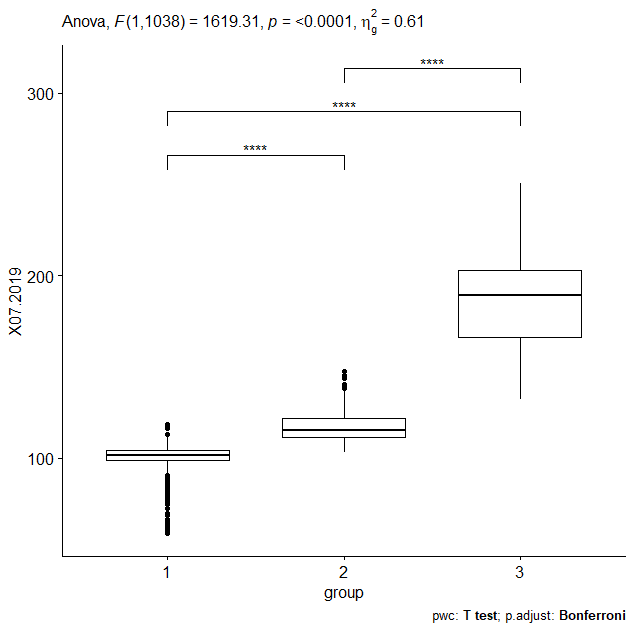
pwc <- pwc %>% add\_xy\_position(x="group")

ggboxplot(price\_when, x="group", y="X07.2019") +

stat\_pvalue\_manual(pwc, hide.ns=TRUE, label="p.adj.signif") +

labs(subtitle=get\_test\_label(res.aov, detailed=TRUE),

caption=get\_pwc\_label(pwc))



**# comparison of distances**

**dd1<-get\_dist(price\_when, method="euclidean")**

**dd1.mat<-as.matrix(dd1)**

**dd1.mat[1:6, 1:6]**

1 2 3 4 5 6

1 0.00000 16.90887 18.91005 13.22422 20.07262 18.19396

2 16.90887 0.00000 24.02332 17.18168 17.43732 12.47437

3 18.91005 24.02332 0.00000 19.76284 20.61019 23.89791

4 13.22422 17.18168 19.76284 0.00000 21.44598 19.59490

5 20.07262 17.43732 20.61019 21.44598 0.00000 22.64443

6 18.19396 12.47437 23.89791 19.59490 22.64443 0.00000

**dd2<-get\_dist(price\_when, method="manhattan")**

**dd2.mat<-as.matrix(dd2)**

**dd3<-get\_dist(price\_when, method="minkowski")**

**dd3.mat<-as.matrix(dd3)**

**dd4<-get\_dist(price\_when, method="canberra")**

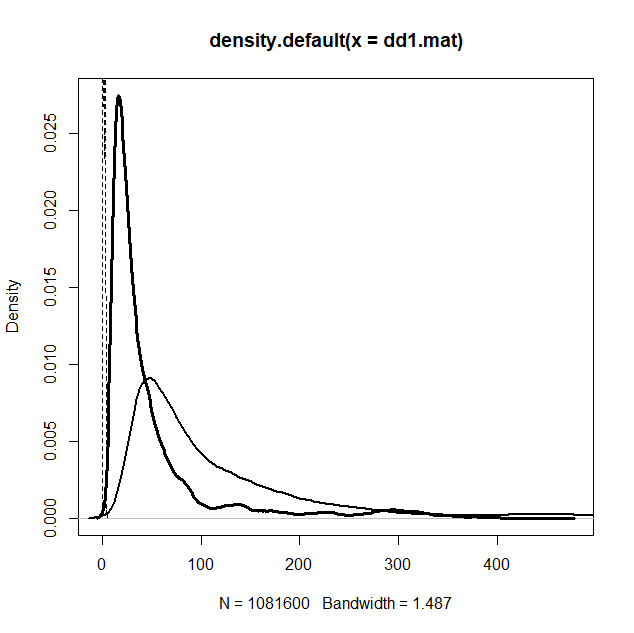
**dd4.mat<-as.matrix(dd4)**

**plot(density(dd1.mat))**

**lines(density(dd2.mat), lwd=2)**

**lines(density(dd3.mat), lwd=3)**

**lines(density(dd4.mat), lty=2, lwd=1)**



Summary:

- First think what information you want to get from clustering – then adjust the data

- There are many commands that you can use – choose your favourite way!

- Mostly decide about number of clusters and distance metric – and clustering is ready

- When data clustered – check their features with statistics. Centres of cluster (or medoids) give the average values of variables which were clustered

- Clustering vector gives you the information, to which cluster given observation was classified