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

**Unsupervised learning**

**Class 03 – Quality measures in clustering**

Katarzyna Kopczewska, Associate professor

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

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

**# Clustering**

**library(cluster)**

**library(factoextra)**

**library(flexclust)**

**library(fpc)**

**library(hopkins)**

**library(ClusterR)**

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

**# warning: change \ on /**

**setwd("C:/Users/Admin/Desktop/Wykłady - Unsupervised learning/01. Clustering/dane ceny")**

**getwd()# checking current WD**

**#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. Optimal number of clusters (once again) 2](#_Toc84882687)

[## 02. Automatic selection of number of clusters 4](#_Toc84882688)

[## 03. Assessing Clustering Tendency using Hopkins' statistic 5](#_Toc84882689)

[## 04. Visual assessment of ordered dissimilarity matrix (ODM) 6](#_Toc84882690)

[## 05. Rand Index & Jaccard similarity – for how much two clusterings of the same differ 7](#_Toc84882691)

[## 06. Variable importance in clustering – what drives our clusters 10](#_Toc84882692)

[## 07. Calinski-Harabsasz and Duda-Hart measures of clustering quality 11](#_Toc84882693)

[## 08. Shadow statistics (as silhouette) 12](#_Toc84882694)

## 01. Optimal number of clusters (once again)

1. Previously, we used **factoextra::fviz\_nbclust()**, which calculated silhouette, wss and gap\_stat.
2. Much more (30 measures) one can get with package **NbClust::**. Full description is available at <https://www.jstatsoft.org/article/view/v061i06/v61i06.pdf> . Functions may work with distance matrix or dataset.

**install.packages("NbClust")**

**library(NbClust)**

**library(help="NbClust") # only one function in a package**

**?NbClust # comprehensive description on distances, methods, tests**

**c3<-NbClust(price\_when, distance="euclidean", min.nc=2, max.nc=8, method="complete", index="ch")**

**c3 # it chooses the best partition**

**c3$All.index**

**c3$Best.nc**

**c3$Best.partition**

1. There is also function **Optimal\_Clusters\_KMeans()** from ClusterR:: package. It has few available criteria:

a) variance\_explained (default)

b) WCSSE (within-cluster-sum-of-squared-error)

c) dissimilarity,

d) silhouette,

e) distortion\_fK,

f), g), h), AIC, BIC and Adjusted\_Rsquared.

**#price\_when.s<-center\_scale(price\_when) # ClusterR:: to scale or center the data**

**opt<-Optimal\_Clusters\_KMeans(price\_when, max\_clusters=10, plot\_clusters = TRUE)**

**opt<-Optimal\_Clusters\_KMeans(price\_when, max\_clusters=10, plot\_clusters=TRUE, criterion="silhouette")**

**opt<-Optimal\_Clusters\_KMeans(price\_when, max\_clusters=10, plot\_clusters=TRUE, criterion="AIC")**

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

Observations / comments:

* Methods give different results – sometimes consistent, sometimes opposite
* Most often used is silhouette, however the others are also interesting

1. There is also **ClusterR::Optimal\_Clusters\_Medoids()**, which includes a nice selection of distance metrics, possibility of sampling. On the basis of dissimilarity figure one can decide about clusters, however there is no automatic decision. It displays a plot to decide how many clusters to make – type a number to get shilhouette.

**opt\_md<-Optimal\_Clusters\_Medoids(price\_when, 10, 'euclidean', plot\_clusters=TRUE)**

## 02. Automatic selection of number of clusters

One can also use automatic tool to select best number of clusters in PAM with **fpc::pamk()**. Most popular criterion – **asw** – is average silhouette width – uses data. Second popular criterion – **ch** - Calinski-Harabasz – uses dissimilarity matrix.

**pamk.best<-pamk(price\_when, krange=2:10,criterion="asw", usepam=TRUE, scaling=FALSE, alpha=0.001, diss=inherits(price\_when, "dist"), critout=FALSE) # fpc::pamk()**

**class(pamk.best)**

**pamk.best**

$pamobject

Medoids:

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

[1,] 175 102.7 102.6 102.6 103.2 103.2 102.9 102.9 103.5

[2,] 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,] 104.1 104.1 104.3 104.3

[2,] 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

[38] 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

[75] 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

[112] 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

[149] 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

……….

Objective function:

build swap

26.87465 26.87465

Available components:

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

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

$nc

[1] 2

$crit

[1] 0.0000000 0.8489796 0.3840897 0.2138637 0.2345737 0.2184029 0.2189183

[8] 0.2140758 0.1483818 0.1702148

**# announcement printed out**

**cat("number of clusters estimated by optimum average silhouette width:", pamk.best$nc, "\n")**

**# plot of the result**

**plot(pam(price\_when, pamk.best$nc))**

## 03. Assessing Clustering Tendency using Hopkins' statistic

In simple words: “how well the data can be clustered”

Hopkins statistics: total y / (total x + total y)

total x 🡪 average distance to nearest neighbor between real data

total y 🡪 average distance to nearest neighbor between real point

and uniformly generated random point (with the same variance as real data)

Because of randomly generated data statistics may differ.

Null hypothesis: the dataset is uniformly distributed (i.e., no meaningful clusters)

Alternative hypothesis: the dataset is not uniformly distributed (i.e., contains meaningful clusters)

Usually the interpretation is as follows:

|  |  |  |
| --- | --- | --- |
| h~0 | h~0.5 | h~1 |
| - accept the null hypothesis  - unlikely that there are statistically significant clusters  - no clusters are visible, uniformely distributed data | Random data | - reject the null hypothesis  - dataset is significantly a clusterable data  - some clusters are visible |

# note that results from get\_clust\_tendency() may somehow differ from hopkins()

# more on (skip part using clustertend:: as it was depreciated and switched to hopkins::) <http://www.sthda.com/english/articles/29-cluster-validation-essentials/95-assessing-clustering-tendency-essentials/>

**hopkins::hopkins(price\_when, m=nrow(price\_when)/10) # clustertend::**

$H

[1] 1

**get\_clust\_tendency(price\_when, 2, graph=TRUE, gradient=list(low="red", mid="white", high="blue"), seed = 123) # factoextra:: #interpretation as in table**

|  |  |
| --- | --- |
| $hopkins\_stat  [1] 0.9658775  **Data are clusterable (clusters visible).** | **$plot** |

Observations / comments:

* In Hopkins statistic we interpret mainly the value of statistics, figure which appears automatically is a dissimilarity matrix, which is analysed below
* We do not need any significance test, as the comparison with the reference distribution is inherited in the function itself.

## 04. Visual assessment of ordered dissimilarity matrix (ODM)

The figure as below appears also as an accompanying graphics to the Hopkins statistics. It can be obtained in at least two ways:

* with **factoextra::get\_clust\_tendency()**
* with **factoextra::fviz\_dist()**

**Plotting the Ordered Dissimilarity Matrix**

1.Compute the dissimilarity matrix (DM) between the objects in the data set (Euclidean distance or other)

2. Reorder the DM to put similar objects close to one another and plot - one gets an ordered dissimilarity matrix (ODM)

result blue (violet): high distance 🡪 high dissimilarity 🡪 low similarity

result red (pink): low distance 🡪 low dissimilarity 🡪 high similarity

interpretation: clustering tendency is present when blocks of colours are visible

random data - when ordered data look like unordered data

**# two commands for distance (dissimilarity) are possible**

**stats:: dist()** is basic function, output in class dist

**factoextra:: get\_dist()** is tailored fuction, output also in class dist

**dist()** allows for distances as "euclidean", "maximum",  "manhattan", "canberra", "binary" or "minkowski"

**get\_dist()** allows for distances as dist() and extra "pearson", "spearman" or "kendall"

**d<-dist(price\_when)**

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

**fviz\_dist(d, show\_labels = FALSE)+ labs(title="our data") #factoextra::**

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

Observations / comments:

* One can see the blocks of colours at the figure, what confirms that data are clusterable and clustering is feasible.

## 05. Rand Index & Jaccard similarity – for how much two clusterings of the same differ

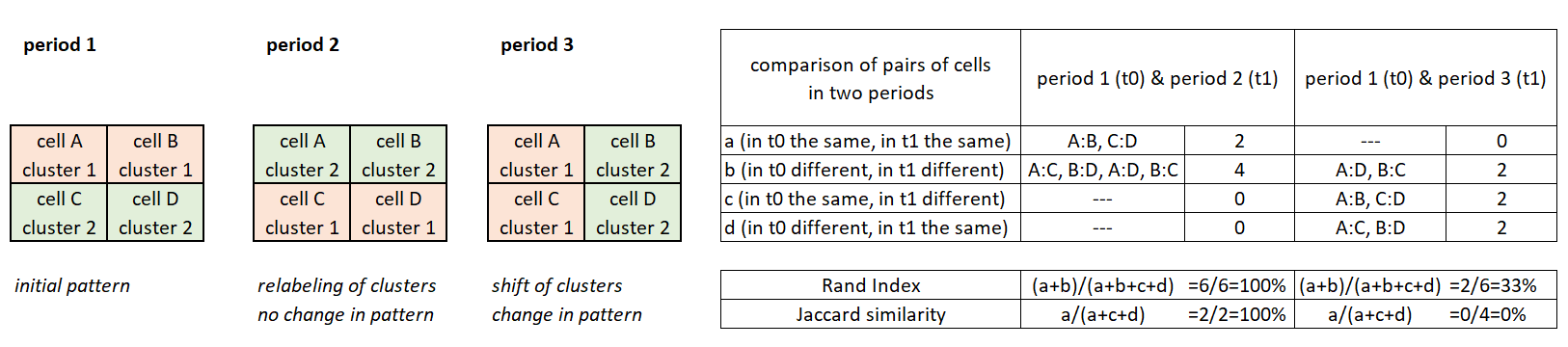
Measures to compare how cluster id changed (because of new clustering, because of new period etc.) – so called analysis of several partitioning. In general, they check how many points changed the cluster in new partition.

They are measuring the so called “agreement of partitions”

**# Most popular: Rand Index and Jaccard similarity**

**# All measures are calculated with one command flexlust::randIndex()**

|  |  |
| --- | --- |
| **# Rand index 🡪** all pairs of observations are compared in two periods (t0 and t1), and checked if they are in the same or different clusters:  a - in t0 the same, in t1 the same,  b - in t0 different, in t1 different,  c - in t0 the same, in t1 different,  d - in t0 different, in t1 the same;  thus, the counter is always the same (a) and always different (b) clusters, and denominator are all possible outcomes (a,b,c,d). Rand Index=1 means that partitions always agree (c and d are NULL) and clusterings are the same, while Rand Index=0 means that partitions migrate and do not agree for even a single pair. **Worth to remember, it checks pairs of pairs of points. It is insensitive to relabelling.** |  |
| # Adjusted Rand Index - as the random observations may be clustered in the same partitions, the pure Rand Index may not reach zero – for this reason, one uses the **Adjusted Rand Index (ARI)**, rescaled in a way that eliminates random assignments. |  |
| **# Jaccard similarity 🡪** it omits a number of events which are always in different clusters (b), both in the counter and denominator. It measures the ratio of overlap and union of two vectors, shapes or any other datasets. Using notation presented for Rand Index, Jaccard similarity is interpreted in a similar way to the Rand Index, but it is concentrated only on pairs that are connected, being a zoom compared to the Rand Index. |  |
| **# Fowlkes-Mallows 🡪** almost the same as Rand Index, but A is number of pairs in the same cluster divided by the geometric mean of the sums of the number of pairs in each cluster of the two partitions. |  |



**# let’s check if partitioning in selected months is stable**

**set.per1<-price\_when[,1:3] # months 1,2,3**

**set.per3<-price\_when[,7:9] # months 7,8,9**

**d1<-cclust(set.per1, 4, dist="euclidean") # flexclust::**

**d2<-cclust(set.per3, 4, dist="manhattan")**

**d1**

#kcca object of family ‘kmeans’

#call:

#cclust(x = set.per1, k = 4, dist = "euclidean")

#cluster sizes:

# 1 2 3 4

# 16 210 38 776

**d2**

#kcca object of family ‘kmedians’

#call:

#cclust(x = set.per3, k = 4, dist = "manhattan")

#cluster sizes:

# 1 2 3 4

#184 45 423 388

**randIndex(d1, d2) # flexclust::**

# ARI

#0.1295739

**comPart(d1, d2)**

# ARI RI J FM

#0.1295739 0.5368050 0.3378446 0.5260188

Observations / comments:

* Partitionings differed in two selected periods – these are not homogenous months
* It shows that cluster compositions differ under both scenarios – overlap is rather low

**# let’s try to compare four quarters of the year**

**set.per1<-price\_when[,1:3] # months 1,2,3**

**set.per2<-price\_when[,4:6] # months 4,5,6**

**set.per3<-price\_when[,7:9] # months 7,8,9**

**set.per4<-price\_when[,10:12] # months 10,11,12**

**d1<-cclust(set.per1, 4, dist="euclidean") # flexclust::**

**d2<-cclust(set.per2, 4, dist="euclidean")**

**d3<-cclust(set.per3, 4, dist="euclidean")**

**d4<-cclust(set.per4, 4, dist="manhattan")**

**par(mar=c(5,5,5,5)) # setting the margins**

**library(RColorBrewer)**

**install.packages("unikn")**

**library(unikn)**

**p6<-brewer.pal(n=9, name="YlGn")**

**mix6<-usecol(pal=c("white", p6)) # vector of colours with white**

**vec.coef<-c("d1", "d2", "d3", "d4") # set of outputs**

**tab.ri<-matrix(0, nrow=4, ncol=4)**

**for(i in 1:4){**

**for(j in 1:4){**

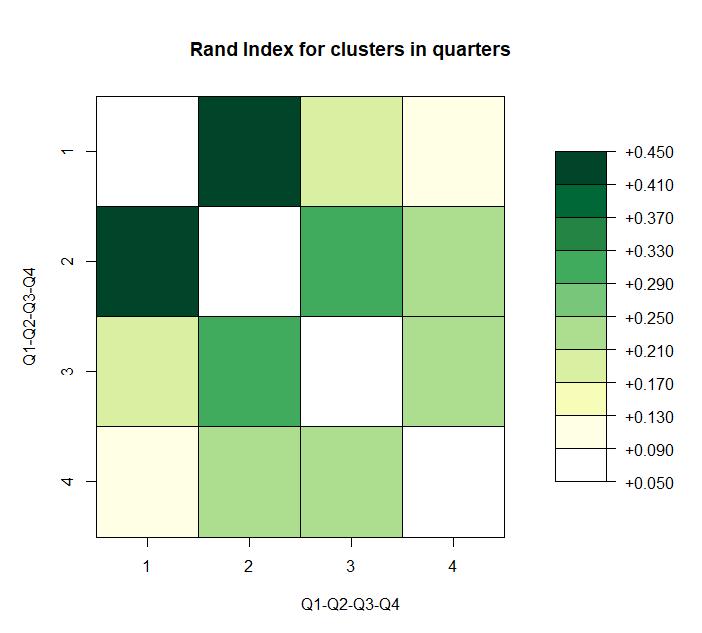
**rix<-randIndex(get(vec.coef[i]), get(vec.coef[j]))**

**tab.ri[i,j]<-rix}}**

**diag(tab.ri)<-NA # for better plotting, NA on diagonal**

**library(plot.matrix)**

**plot(tab.ri, col=mix6, main="Rand Index for clusters in quarters", xlab="Q1-Q2-Q3-Q4", ylab="Q1-Q2-Q3-Q4")**



* We clustered changes of prices of all products in all regions over three months – each observation (product in region) was classified into cluster.
* With Rand Index we check for how much the observations get to the same clusters in next three months
* Q1 and Q2 are similar – ca. every second observation goes to the same cluster as before
* There is very weak relation between Q1 and Q4 – every sixth observation is classified in the same cluster

## 06. Variable importance in clustering – what drives our clusters

Web-material: <https://cran.r-project.org/web/packages/FeatureImpCluster/readme/README.html>

Easy concept based on permutation: it shuffles (permutates) values of given variable and checks for how much cluster assignment changed with this mixed variable. Misclassification rate is the number of wrongly assigned observations divided by number of observations.

**# reading package**

install.packages("FeatureImpCluster")

library(FeatureImpCluster) # for FeatureImpCluster()

**# clustering + feature importance**

km<-kcca(**price\_when**, k=3)

FeatureImp\_km<-FeatureImpCluster(km, as.data.table(price\_when))

plot(FeatureImp\_km)

km<-kcca(**price\_what[,3:68]**, k=3)

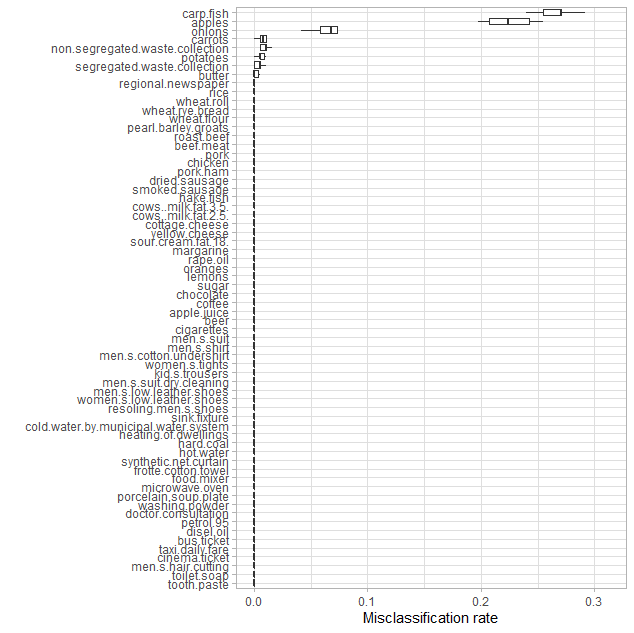
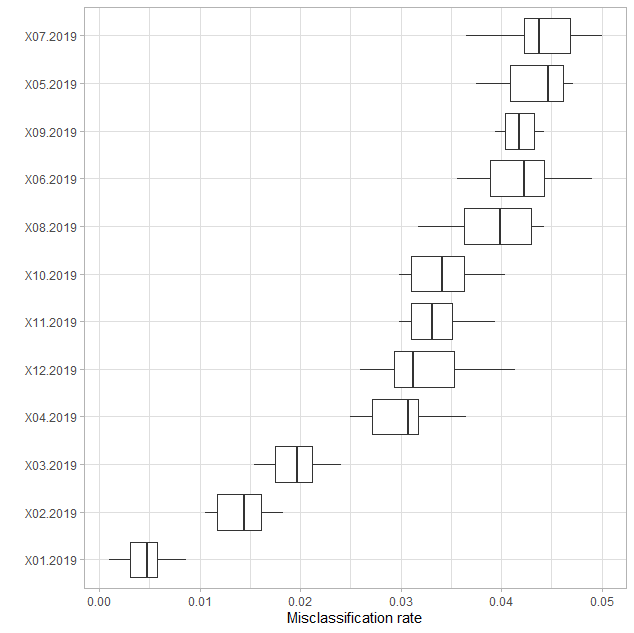
FeatureImp\_km<-FeatureImpCluster(km, as.data.table(price\_what[,3:68]))

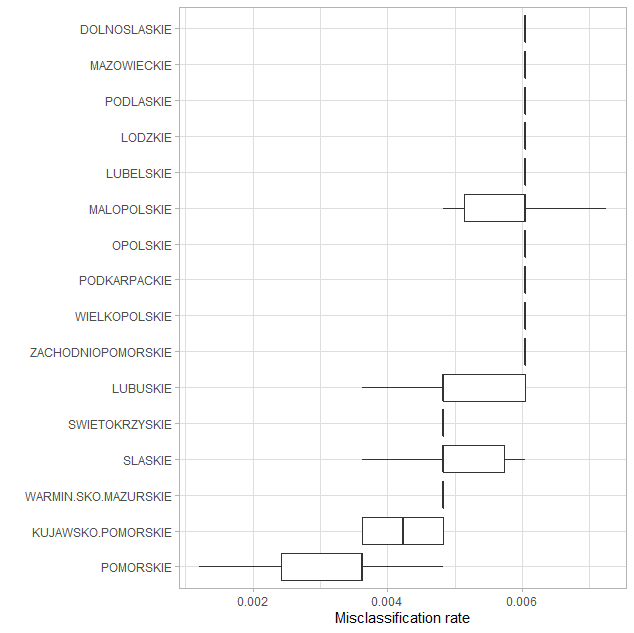
plot(FeatureImp\_km)

km<-kcca(**price\_where[,3:18]**, k=3)

FeatureImp\_km<-FeatureImpCluster(km, as.data.table(price\_where[,3:18]))

plot(FeatureImp\_km)





## 07. Calinski-Harabsasz and Duda-Hart measures of clustering quality

1. There is special package **clusterCrit::** with many measures of clustering quality:

<https://cran.r-project.org/web/packages/clusterCrit/vignettes/clusterCrit.pdf>

Overview of clustering stat is also available in fpc::cluster.stats(). All are based on distance matrix.

**d<-dist(price\_when)**

**complete3<-cutree(hclust(d),3)**

**c.stat<-cluster.stats(d,complete3)**

**c.stat**

1. Very popular measure of quality of clustering for k-means is **Calinski-Harabasz index (CH).** Its construction is as follows:

counter: BGSS/(K-1) 🡪 between-group sum of squares (for K clusters)

nominator: WGSS/(N-K) 🡪 within-cluster sum of squares (sum of the within-cluster dispersions for all clusters) (for N observations)

The higher statistics the better. It is usually used for comparing solutions for alternative number of clusters.

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

**round(calinhara(price\_when, km1$cluster),digits=2) #fpc::calinhara()**

[1] 109.19

**km2<-kmeans(price\_when, 3) # stats::**

**round(calinhara(price\_when, km2$cluster),digits=2) #fpc::calinhara()**

[1] 1125.36

# **Calinski-Harabasz is also available in fpc::cluster.stats()**

**d<-dist(price\_when)**

**complete3<-cutree(hclust(d),3)**

**c.stat<-cluster.stats(d,complete3)**

**c.stat$ch**

1. There is also Duda-Hart test whether a data set should be split into two clusters. It is well defined for *kmeans* class. Its hypotheses are as follows:

H0: homogeneity of cluster (data within cluster as similar)

H1: heterogeneity of cluster (one can easily split the cluster)

Statistics dh: ratio of within-cluster sum of squares for two clusters and overall sum of squares.

verification: cluster1=FALSE (H0 of homogeneity rejected, accept H1)

verification: when dh statistics is lower than “compare” (critical value), accept H1

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

**dudahart2(price\_when, km1$cluster) #fpc::**

$p.value #[1] 0

$dh #[1] 0.4049874

$compare #[1] 0.9091727

$cluster1 #[1] FALSE

$alpha #[1] 0.001

$z #[1] 3.090232

Observations / comments:

* Following the Duda-Hart test we should split the clusters (as cluster1=FALSE )
* Following Calinski-Harabasz test, we should prefer k=3 over k=2

## 08. Shadow statistics (as silhouette)

Shadow statistics very close to silhouette. More in Leisch F (2009) Neighborhood Graphs, Stripes and Shadow Plots for Cluster Visualization<https://pdfs.semanticscholar.org/4d41/34253b34c3425b6cf29595d3f84c96a76892.pdf>

“*The main difference between silhouette values and shadow values is that we replace average dissimilarities to points in a cluster by dissimilarities to point averages (=centroids).*” <https://rdrr.io/cran/flexclust/man/shadow.html>

**definition of shadow:** twice the distance to the closest centroid divided by the sum of distances to closest and second-closest centroid.

interpretation: shadow~0 🡪 points are close to their centroids

shadow~1 🡪 points are equidistant to the two centroids

Good cluster: many points with small shadow values

**# k-means clustering**

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

**shadow(d1) # flexclust::**

**plot(shadow(d1))**

|  |  |
| --- | --- |
| **1 2 3 4**  **0.5376759 0.4719773 0.7134725 0.2893842** |  |

Observations / comments:

* In some special cases shadow statistics performs better than shillouette.

Summary:

- Choose properly the number of clusters - using measures and simply understanding what you want to show. There are many criteria, mostly used are: silhouette, gap, Calinski-Harabasz and Duda-Hart

- When finding appropriate number of clusters is difficult, check with Hopkins if data are clusterable

- When comparing in time the stability of clustering, use Rand Index or Jaccard similarity – you can easily draw a matrix of results to compare the periods

- check with Feature importance what are the most important variables that matter most for your clustering