Package 'kmed'

October 13, 2022

Type Package

Title Distance-Based k-Medoids

Version 0.4.2

Date 2022-08-29

Description Algorithms of distance-based k-medoids clustering: simple and fast k-medoids, ranked k-medoids, and increasing number of clusters in k-medoids. Calculate distances for mixed variable data such as Gower, Podani, Wishart, Huang, Harikumar-PV, and Ahmad-Dey. Cluster validation applies internal and relative criteria. The internal criteria includes silhouette index and shadow values. The relative criterium applies bootstrap procedure producing a heatmap with a flexible reordering matrix algorithm such as complete, ward, or average linkages. The cluster result can be plotted in a marked barplot or pca biplot.

Depends R (>= 2.10)

License GPL-3

LazyData TRUE

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports ggplot2

NeedsCompilation no

Author Weksi Budiaji [aut, cre]

Maintainer Weksi Budiaji <budiaji@untirta.ac.id>

Repository CRAN

Date/Publication 2022-08-29 06:40:02 UTC

R topics documented:

`barplotnum` *Barplot of each cluster for numerical variables data set*

Description

This function creates a barplot from a cluster result. A barplot indicates the location and dispersion of each cluster. The x-axis of the barplot is variable's mean, while the y-axis is the variable's name.

Usage

```
barplotnum(dataori, clust, nc = 1, alpha = 0.05)
```

Arguments

`dataori`	An original data set.
`clust`	A vector of cluster membership (see **Details**).
`nc`	A number of columns for the plot of all cluster (see **Details**).
`alpha`	A numeric number to set the significant level (between 0 and 0.2).

Details

This is a marked barplot because some markers are added, i.e. a significant test, a population mean for each (numerical) variable. The significance test applies t-test between the population's mean and cluster's mean in every variable. The alpha is set in between 0 to 20%. If the population mean differs to the cluster's mean, the bar shade in the barplot also differs.

`clust` is a vector with the length equal to the number of objects (n), or the function will be an error otherwise. `nc` controls the layout (grid) of the plot. If `nc = 1`, the plot of each cluster is placed in a column. When the number of clusters is 6 and `nc = 2`, for example, the plot has a layout of 3-row and 2-column grids.

Value

Function returns a barplot.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Leisch, F. (2008). Handbook of Data Visualization, Chapter Visualizing cluster analysis and finite mixture models, pp. 561-587. Springer Handbooks of Computational Statistics. Springer Verlag.

Dolnicar, S. and F. Leisch (2014). Using graphical statistics to better understand market segmentation solutions. International Journal of Market Research 56, 207-230.

Examples

```
dat <- iris[,1:4]
memb <- cutree(hclust(dist(dat)),3)
barplotnum(dat, memb)
barplotnum(dat, memb, 2)
```

---

clust4 *4-clustered data set*

Description

A dataset containing two variables of 300 objects and their class memberships generated by the **clusterGeneration** package.

Usage

```
clust4
```

Format

A data frame with 300 rows and 3 variables:

x1 X1.

x2 X2.

class Class membership.

Source

Data is generated via the `genRandomClust` function in the **clusterGeneration** package. The code to generate this data set is

set.seed(2016)

randclust <- clusterGeneration::genRandomClust(4, sepVal = 0.001, numNonNoisy = 2, numReplicate = 1, clustszind = 3, clustSizes = as.numeric(table(sample(1:4, 300, replace = TRUE))), outputDatFlag=FALSE, outputLogFlag=FALSE, outputEmpirical=FALSE, outputInfo=FALSE)

clust4 <- as.data.frame(randclust$datList$test_1)

clust4$class <- randclust$memList$test_1

References

Qiu, W., and H. Joe. 2015. ClusterGeneration: Random Cluster Generation (with Specified Degree of Separation).

Qiu, W., and H. Joe. 2006a. Generation of Random Clusters with Specified Degree of Separation. Journal of Classification 23 pp. 315-34.

Qiu, W., and H. Joe. 2006b. Separation Index and Partial Membership for Clustering. Computational Statistics and Data Analysis 50 pp. 585-603.

`clust5` *5-clustered data set*

Description

A dataset containing two variables of 800 objects and their class memberships generated by the **clusterGeneration** package.

Usage

```
clust5
```

Format

A data frame with 800 rows and 3 variables:

x1 X1.

x2 X2.

class Class membership.

Source

Data is generated via the genRandomClust function in the **clusterGeneration** package. The code to generate this data set is

set.seed(2016)

randclust <- clusterGeneration::genRandomClust(5, sepVal = 0.2, numNonNoisy = 2, numReplicate = 1, clustszind = 3, clustSizes = as.numeric(table(sample(1:5, 800, replace = TRUE))), outputDatFlag=FALSE, outputLogFlag=FALSE, outputEmpirical=FALSE, outputInfo=FALSE)

clust5 <- as.data.frame(randclust$datList$test_1)

clust5$class <- randclust$memList$test_1

References

Qiu, W., and H. Joe. 2015. ClusterGeneration: Random Cluster Generation (with Specified Degree of Separation).

Qiu, W., and H. Joe. 2006a. Generation of Random Clusters with Specified Degree of Separation. Journal of Classification 23 pp. 315-34.

Qiu, W., and H. Joe. 2006b. Separation Index and Partial Membership for Clustering. Computational Statistics and Data Analysis 50 pp. 585-603.

---

clustboot *Bootstrap replications for clustering alorithm*

Description

This function does bootstrap replications for a clustering algorithm. Any hard clustering algorithm is valid.

Usage

```
clustboot(distdata, nclust = 2, algorithm = fastclust, nboot = 25, diss = TRUE)
```

Arguments

distdata	A distance matrix (n x n)/ dist object or a data frame.
nclust	A number of clusters.
algorithm	A clustering algorithm function (see **Details**).
nboot	A number of bootstrap replicates.
diss	A logical if distdata is a distance matrix/ object or a data frame.

Details

This is a function to obtain bootstrap evaluation for cluster results. The `algorithm` argument is a function where this function has two input arguments. The two input arguments are a distance matrix/ object or a data frame, and number of clusters. Then the output is only a vector of cluster memberships.

The default `algorithm` is `fastclust` applying the `fastkmed` function. The code of the `fastclust` is

fastclust <- function(x, nclust) {

res <- fastkmed(x, nclust, iterate = 50)

return(res$cluster)

}

For other examples, see **Examples**. It applies ward and kmeans algorithms. When kmeans is applied, for example, `diss` is set to be `FALSE` because the input of the `kmclust` and `clustboot` is a data frame instead of a distance.

Value

Function returns a matrix of bootstrap replicates with a dimension of n x b, where n is the number of objects and b is the number of bootstrap replicates.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Dolnicar, S. and Leisch, F. 2010. Evaluation of structure and reproducibility of cluster solutions using the bootstrap. Marketing Letters 21 pp. 83-101.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
ward.D2 <- function(x, nclust) {
res <- hclust(as.dist(x), method = "ward.D2")
member <- cutree(res, nclust)
return(member)
}
kmclust <- function(x, nclust) {
res <- kmeans(x, nclust)
return(res$cluster)
}
irisfast <- clustboot(mrwdist, nclust=3, nboot=7)
head(irisfast)
irisward <- clustboot(mrwdist, nclust=3, algorithm = ward.D2, nboot=7)
head(irisward)
iriskmeans <- clustboot(num, nclust=3, algorithm = kmclust, nboot=7, diss = FALSE)
```

```
head(iriskmeans)
```

`clustheatmap` *Consensus matrix heatmap from A consensus matrix*

Description

This function creates a consensus matrix heatmap from a consensus/ agreement matrix. The values of the consensus/ agreement matrix are transformed in order to plot the heatmap.

Usage

```
clustheatmap(consmat, title = "")
```

Arguments

`consmat`	A matrix of consensus/ agreement matrix (see **Details**).
`title`	A title of the plot.

Details

This is a function to produce a consensus matrix heatmap from a consensus/ agreement matrix. A matrix produced by the `consensusmatrix` function can be directly provided in the `consmat` argument. The values of the consensus matrix, **A**, are then transformed via a non-linear transformation by applying

$$a_{ij}^{trf} = \frac{a_{ij} - min(a_{..})}{max(a_{..}) - min(a_{..})}$$

where a_{ij} is the value of the consensus matrix in row i and column j, and $a_{..}$ is the all values of the matrix ($\forall$**A**).

Value

Function returns a heatmap plot.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Monti, S., P. Tamayo, J. Mesirov, and T. Golub. 2003. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning 52 pp. 91-118.

Hahsler, M., and Hornik, K., 2011. Dissimilarity plots: A visual exploration tool for partitional clustering. Journal of Computational and Graphical Statistics 20(2) pp. 335-354.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
irisfast <- clustboot(mrwdist, nclust=3, nboot=7)
complete <- function(x, nclust) {
res <- hclust(as.dist(x), method = "complete")
member <- cutree(res, nclust)
return(member)
}
consensuscomplete <- consensusmatrix(irisfast, nclust = 3, reorder = complete)
clustheatmap(consensuscomplete)
```

`consensusmatrix` *Consensus matrix from A matrix of bootstrap replicates*

Description

This function creates a consensus matrix from a matrix of bootstrap replicates. It transforms an n x b matrix into an n x n matrix, where n is the number of objects and b is the number of bootstrap replicates.

Usage

```
consensusmatrix(bootdata, nclust, reorder = fastclust)
```

Arguments

`bootdata`	A matrix of bootstrap replicate (n x b) (see **Details**).
`nclust`	A number of clusters.
`reorder`	Any distance-based clustering algorithm function (see **Details**).

Details

This is a function to obtain a consensus matrix from a matrix of bootstrap replicates to evaluate the clustering result. The `bootdata` argument can be supplied directly from a matrix produced by the `clustboot` function. The values of the consensus matrix, **A**, are calculated by

$$a_{ij} = a_{ji} = \frac{\#n\ of\ objects\ i\ and\ j\ in\ the\ same\ cluster}{\#n\ of\ objects\ i\ and\ j\ sampled\ at\ the\ same\ time}$$

where a_{ij} is the agreement index between objects i and j. Note that due to the agreement between objects i and j equal to the agreement between objects j and i, the consensus matrix is a symmetric matrix.

Meanwhile, the `reorder` argument is a function to reorder the objects in both the row and column of the consensus matrix such that similar objects are close to each other. This task can be solved by applying a clustering algorithm in the consensus matrix. The `reorder` has to consist of two

input arguments. The two input arguments are a distance matrix/ object and number of clusters. The output is only a vector of cluster memberships. Thus, the algorihtm that can be applied in the `reorder` argument is the distance-based algorithm with a distance as the input.

The default `reorder` is `fastclust` applying the `fastkmed` function. The code of the `fastclust` is

fastclust <- function(x, nclust) {

res <- fastkmed(x, nclust, iterate = 50)

return(res$cluster)

}

For other examples, see **Examples**. It applies centroid and complete linkage algorithms.

Value

Function returns a consensus/ agreement matrix of n x n dimension.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Monti, S., P. Tamayo, J. Mesirov, and T. Golub. 2003. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning 52 pp. 91-118.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
irisfast <- clustboot(mrwdist, nclust=3, nboot=7)
consensusfast <- consensusmatrix(irisfast, nclust = 3)
centroid <- function(x, nclust) {
res <- hclust(as.dist(x), method = "centroid")
member <- cutree(res, nclust)
return(member)
}
consensuscentroid <- consensusmatrix(irisfast, nclust = 3, reorder = centroid)
complete <- function(x, nclust) {
res <- hclust(as.dist(x), method = "complete")
member <- cutree(res, nclust)
return(member)
}
consensuscomplete <- consensusmatrix(irisfast, nclust = 3, reorder = complete)
consensusfast[c(1:5,51:55,101:105),c(1:5,51:55,101:105)]
consensuscentroid[c(1:5,51:55,101:105),c(1:5,51:55,101:105)]
consensuscomplete[c(1:5,51:55,101:105),c(1:5,51:55,101:105)]
```

cooccur *Co-occurrence distance for binary/ categorical variables data*

Description

This function calculates the co-occurrence distance proposed by Ahmad and Dey (2007).

Usage

```
cooccur(data)
```

Arguments

data	A matrix or data frame of binary/ categorical variables (see **Details**).

Details

This function computes co-occurrence distance, which is a binary/ categorical distance, that based on the other variable's distribution (see **Examples**). In the **Examples**, we have a data set:

object	x	y	z
1	1	2	2
2	1	2	1
3	2	1	2
4	2	1	2
5	1	1	1
6	2	2	2
7	2	1	2

The co-occurrence distance transforms each category of binary/ categorical in a variable based on the distribution of other variables, for example, the distance between categories 1 and 2 in the x variable can be different to the distance between categories 1 and 2 in the z variable. As an example, the transformed distance between categories 1 and 2 in the z variable is presented.

A cross tabulation between the z and x variables with corresponding (column) proportion is

	1	2	‖	1	2
1	2	1	‖	1.0	0.2
2	0	4	‖	0.0	0.8

A cross tabulation between the z and y variables with corresponding (column) proportion is

	1	2	‖	1	2
1	1	3	‖	0.5	0.6
2	1	2	‖	0.5	0.4

Then, the maximum values of the proportion in each row are taken such that they are 1.0, 0.8, 0.6,

and 0.5. The new distance between categories 1 and 2 in the z variable is

$$\delta^{z}_{1,2} = \frac{(1.0 + 0.8 + 0.6 + 0.5) - 2}{2} = 0.45$$

The constant 2 in the formula applies because the z variable depends on the 2 other variable distributions, i.e the x and y variables. The new distances of each category in the for the x and y variables can be calculated in a similar way.

Thus, the distance between objects 1 and 2 is 0.45. It is only the z variable counted to calculate the distance between objects 1 and 2 because objects 1 and 2 have similar values in both the x and y variables.

The `data` argument can be supplied with either a matrix or data frame, in which the class of the element has to be an integer. If it is not an integer, it will be converted to an integer class. If the `data` of a variable only, a simple matching is calculated. The co-occurrence is absent due to its dependency to the distribution of other variables and a `warning` message appears.

Value

Function returns a distance matrix (n x n).

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Ahmad, A., and Dey, L. 2007. A K-mean clustering algorithm for mixed numeric and categorical data. Data and Knowledge Engineering 63, pp. 503-527.

Harikumar, S., PV, S., 2015. K-medoid clustering for heterogeneous data sets. JProcedia Computer Science 70, 226-237.

Examples

```
set.seed(1)
a <- matrix(sample(1:2, 7*3, replace = TRUE), 7, 3)
cooccur(a)
```

---

csv *Centroid shadow value (CSV) index and plot*

Description

This function computes centroid shadow values and shadow value plots of each cluster. The plot presents the mean of the shadow values as well.

Usage

```
csv(distdata, idmedoid, idcluster, title = "")
```

Arguments

distdata	A distance matrix (n x n) or dist object.
idmedoid	A vector of id medoids (see **Details**).
idcluster	A vector of cluster membership (see **Details**).
title	A title of the plot.

Details

The origin of the centroid shadow value is calculated in the shadow function of the **flexclust** package, in which it is based on the first and second closest centroid. The csv function in this package modifies the centroid into medoid such that the formula to compute shadow value of object i is

$$csv(i) = \frac{2d(i, m(i))}{d(i, m(i)) + d(i, m'(i))}$$

where $d(i, m(i))$ is the distance between object i to the first closest medoid and d(i, m'(i)) is the distance between object i to the second closest medoid.

The idmedoid argument corresponds to the idcluster argument. If the length of idmedoid is 3, for example, the idcluster has to have 3 unique cluster memberships, or it returns Error otherwise. The length of the idcluster has also to be equal to n (the number of objects). In contrast to the silhoutte value, the centoird shadow value is interpreted that lower value is the better cluster separation.

Value

Function returns a list with following components:

result is a data frame of the shadow values for all objects

plot is the shadow value plots of each cluster.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

F. Leisch. 2010 Neighborhood graphs, stripes and shadow plots for cluster visualization. Statistics and Computing. vol. 20, pp. 457-469

W. Budiaji. 2019 Medoid-based shadow value validation and visualization. International Journal of Advances in Intelligent Informatics. Vol 5 No 2 pp. 76-88

Examples

```
distiris <- as.matrix(dist(iris[,1:4]))
res <- fastkmed(distiris, 3)
sha <- csv(distiris, res$medoid, res$cluster)
sha$result[c(1:3,70:75,101:103),]
sha$plot
```

`distmix` *Distances for mixed variables data set*

Description

This function computes a distance matrix for a mixed variable data set applying various methods.

Usage

```
distmix(data, method = "gower", idnum = NULL, idbin = NULL, idcat = NULL)
```

Arguments

`data`	A data frame or matrix object.
`method`	A method to calculate the mixed variables distance (see **Details**).
`idnum`	A vector of column index of the numerical variables.
`idbin`	A vector of column index of the binary variables.
`idcat`	A vector of column index of the categorical variables.

Details

There are six methods available to calculate the mixed variable distance. They are `gower`, `wishart`, `podani`, `huang`, `harikumar`, `ahmad`.

`gower`

The Gower (1971) distance is the most common distance for a mixed variable data set. Although the Gower distance accommodates missing values, a missing value is not allowed in this function. If there is a missing value, the Gower distance from the `daisy` function in the **cluster** package can be applied. The Gower distance between objects i and j is computed by $d_{ij} = 1 - s_{ij}$, where

$$s_{ij} = \frac{\sum_{l=1}^{p} \omega_{ijl} s_{ijl}}{\sum_{l=1}^{p} \omega_{ijl}}$$

ω_{ijl} is a weight in variable l that is usually 1 or 0 (for a missing value). If the variable l is a numerical variable,

$$s_{ijl} = 1 - \frac{|x_{il} - x_{jl}|}{R_l}$$

$s_{ijl} \in \{0, 1\}$, if the variable l is a binary/ categorical variable.

`wishart`

Wishart (2003) has proposed a different measure compared to Gower (1971) in the numerical variable part. Instead of a range, it applies a variance of the numerical variable in the s_{ijl} such that the distance becomes

$$d_{ij} = \sqrt{\sum_{l=1}^{p} \omega_{ijl} \left(\frac{x_{il} - x_{jl}}{\delta_{ijl}} \right)^2}$$

where $\delta_{ijl} = s_l$ when l is a numerical variable and $\delta_{ijl} \in \{0, 1\}$ when l is a binary/ categorical variable.

`podani`

Podani (1999) has suggested a different method to compute a distance for a mixed variable data set. The Podani distance is calculated by

$$d_{ij} = \sqrt{\sum_{l=1}^{p} \omega_{ijl} \left(\frac{x_{il} - x_{jl}}{\delta_{ijl}} \right)^2}$$

where $\delta_{ijl} = R_l$ when l is a numerical variable and $\delta_{ijl} \in \{0, 1\}$ when l is a binary/ categorical variable.

`huang`

The Huang (1997) distance between objects i and j is computed by

$$d_{ij} = \sum_{r=1}^{P_n} (x_{ir} - x_{jr})^2 + \gamma \sum_{s=1}^{P_c} \delta_c(x_{is} - x_{js})$$

where P_n and P_c are the number of numerical and categorical variables, respectively,

$$\gamma = \frac{\sum_{r=1}^{P_n} s_r^2}{P_n}$$

and $\delta_c(x_{is} - x_{js})$ is the mismatch/ simple matching distance (see `matching`) between object i and object j in the variable s.

`harikumar`

Harikumar-PV (2015) has proposed a distance for a mixed variable data set:

$$d_{ij} = \sum_{r=1}^{P_n} |x_{ir} - x_{jr}| + \sum_{s=1}^{P_c} \delta_c(x_{is} - x_{js}) + \sum_{t=1}^{p_b} \delta_b(x_{it}, x_{jt})$$

where P_b is the number of binary variables, $\delta_c(x_{is}, x_{js})$ is the co-occurrence distance (see `cooccur`), and $\delta_b(x_{it}, x_{jt})$ is the Hamming distance.

`ahmad`

Ahmad and Dey (2007) has computed a distance of a mixed variable set via

$$d_{ij} = \sum_{r=1}^{P_n} (x_{ir} - x_{jr})^2 + \sum_{s=1}^{P_c} \delta_c(x_{is} - x_{js})$$

where $\delta_c(x_{it}, x_{jt})$ are the co-occurrence distance (see cooccur). In the Ahmad and Dey distance, the binary and categorical variables are not separable such that the co-occurrence distance is based on the combined these two classes, i.e. binary and categorical variables. Note that this function applies standard version of Squared Euclidean, i.e without any weight.

At leas two arguments of the idnum, idbin, and idcat have to be provided because this function calculates the mixed distance. If the method is harikumar, the categorical variables have to be at least two variables such that the co-occurrence distance can be computed. It also applies when method = "ahmad". The idbin + idcat has to be more than 1 column. It returns to an Error message otherwise.

Value

Function returns a distance matrix (n x n).

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Ahmad, A., and Dey, L. 2007. A K-mean clustering algorithm for mixed numeric and categorical data. Data and Knowledge Engineering 63, pp. 503-527.

Gower, J., 1971. A general coefficient of similarity and some of its properties. Biometrics 27, pp. 857-871

Harikumar, S., PV, S., 2015. K-medoid clustering for heterogeneous data sets. JProcedia Computer Science 70, pp. 226-237.

Huang, Z., 1997. Clustering large data sets with mixed numeric and categorical values, in: The First Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 21-34.

Podani, J., 1999. Extending gower's general coefficient of similarity to ordinal characters. Taxon 48, pp. 331-340.

Wishart, D., 2003. K-means clustering with outlier detection, mixed variables and missing values, in: Exploratory Data Analysis in Empirical Research: Proceedings of the 25th Annual Conference of the Gesellschaft fur Klassifikation e.V., University of Munich, March 14-16, 2001, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 216-226.

Examples

```
set.seed(1)
a <- matrix(sample(1:2, 7*3, replace = TRUE), 7, 3)
a1 <- matrix(sample(1:3, 7*3, replace = TRUE), 7, 3)
mixdata <- cbind(iris[1:7,1:3], a, a1)
colnames(mixdata) <- c(paste(c("num"), 1:3, sep = ""),
                       paste(c("bin"), 1:3, sep = ""),
                       paste(c("cat"), 1:3, sep = ""))
distmix(mixdata, method = "gower", idnum = 1:3, idbin = 4:6, idcat = 7:9)
```

`distNumeric` *A pair distance for numerical variables*

Description

This function computes a pairwise numerical distance between two numerical data sets.

Usage

```
distNumeric(x, y, method = "mrw", xyequal = TRUE)
```

Arguments

`x`	A first data matrix (see **Details**).
`y`	A second data matrix (see **Details**).
`method`	A method to calculate the pairwise numerical distance (see **Details**).
`xyequal`	A logical if `x` is equal to `y` (see **Details**).

Details

The `x` and `y` arguments have to be matrices with the same number of columns where the row indicates the object and the column is the variable. This function calculate all pairwise distance between rows in the `x` and `y` matrices. Although it calculates a pairwise distance between two data sets, the default function computes all distances in the `x` matrix. If the `x` matrix is not equal to the `y` matrix, the `xyequal` has to be set `FALSE`.

The `method` available are `mrw` (Manhattan weighted by range), `sev` (squared Euclidean weighted by variance), `ser` (squared Euclidean weighted by range), `ser.2` (squared Euclidean weighted by squared range) and `se` (squared Euclidean). Their formulas are:

$$mrw_{ij} = \sum_{r=1}^{p_n} \frac{|x_{ir} - x_{jr}|}{R_r}$$

$$sev_{ij} = \sum_{r=1}^{p_n} \frac{(x_{ir} - x_{jr})^2}{s_r^2}$$

$$ser_{ij} = \sum_{r=1}^{p_n} \frac{(x_{ir} - x_{jr})^2}{R_r}$$

$$ser.2_{ij} = \sum_{r=1}^{p_n} \frac{(x_{ir} - x_{jr})^2}{R_r^2}$$

$$se_{ij} = \sum_{r=1}^{p_n} (x_{ir} - x_{jr})^2$$

where p_n is the number of numerical variables, R_r is the range of the r-th variables, s_r^2 is the variance of the r-th variables.

Value

Function returns a distance matrix with the number of rows equal to the number of objects in the x matrix (n_x) and the number of columns equals to the number of objects in the y matrix (n_y).

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
mrwdist[1:6,1:6]
```

---

fastkmed *Simple and fast k-medoid algorithm*

Description

This function runs the simple and fast k-medoid algorithm proposed by Park and Jun (2009).

Usage

```
fastkmed(distdata, ncluster, iterate = 10, init = NULL)
```

Arguments

distdata	A distance matrix (n x n) or dist object.
ncluster	A number of clusters.
iterate	A number of iterations for the clustering algorithm.
init	A vector of initial objects as the cluster medoids (see **Details**).

Details

The simple and fast k-medoids, which sets a set of medoids as the cluster centers, adapts the k-means algorithm for medoid up-dating. The new medoids of each iteration are calculated in the within cluster only such that it gains speed.

init = NULL is required because the Park and Jun (2009) has a particular method to select the initial medoids. The initial medoids are selected by

$$v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}}, \quad j = 1, 2, 3, \ldots, n$$

where the first k of the v_j is selected if the number of clusters is k.

`init` can be provided with a vector of id objects. The length of the vector has to be equal to the number of clusters. However, assigning a vector in the `init` argument, the algorithm is no longer the simple and fast k-medoids algorithm. The `inckmed` function, for example, defines a different method to select the initial medoid though it applies the `fastkmed` function.

Value

Function returns a list of components:

`cluster` is the clustering memberships result.

`medoid` is the id medoids.

`minimum_distance` is the distance of all objects to their cluster medoid.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Park, H., Jun, C., 2009. A simple and fast algorithm for k-medoids clustering. Expert Systems with Applications 36, pp. 3336-3341.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
result <- fastkmed(mrwdist, ncluster = 3, iterate = 50)
table(result$cluster, iris[,5])
```

---

globalfood *Global food security index*

Description

A dataset containing four variables of 113 countries for their food security index based on panelists evaluation in 2017.

Usage

```
globalfood
```

Format

A data frame with 113 rows and 4 variables:

affordability Index of food affordability.

availability Index of food availability.

safety Index of food quality and safety.

resilience Index of natural resources and resilience.

Source

The original indicator variables consist of 27 variables. Then, they are summarized into four pillars of food security; they are affordability, availability, quality and safety, and natural resources and resilience. Food-security expertise panelists evaluate the score of each country from 0 to 100, where 0 is the least favorable towards food security.

https://impact.economist.com/sustainability/project/food-security-index/

---

heart *Heart Disease data set*

Description

A mixed variable dataset containing 14 variables of 297 patients for their heart disease diagnosis.

Usage

```
heart
```

Format

A data frame with 297 rows and 14 variables:

age Age in years (numerical).

sex Sex: 1 = male, 0 = female (logical).

cp Four chest pain types: (1) typical angina, (2) atypical angina (3)non-anginal pain, (4) asymptomatic (categorical).

trestbps Resting blood pressure (in mm Hg on admission to the hospital) (numerical).

chol Serum cholestoral in mg/dl (numerical).

fbs Fasting blood sugar more than 120 mg/dl (logical).

restecg Resting electrocardiographic results: (0) normal, (1) having ST-T wave abnormality, (2) showing probable or definite left ventricular hypertrophy by Estes' criteria (categorical).

thalach Maximum heart rate achieved (numerical).

exang Exercise induced angina (logical).

oldpeak ST depression induced by exercise relative to rest (numerical).

slope The slope of the peak exercise ST segment: (1) upsloping, (2) flat, (3) downsloping (categorical).

ca Number of major vessels (0-3) colored by flourosopy (numerical).

thal (3) normal, (6) fixed defect, (7) reversable defect (categorical).

class Diagonosis of heart disease (4 classes). It can be 2 classes by setting 0 for 0 values and 1 for non-0 values.

Source

The data set is taken from machine learning repository of UCI. The original data set consists of 303 patients with 6 NA's. Then, the missing values are omitted such that it reduces into 297 patients.

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

References

Lichman, M. (2013). UCI machine learning repository.

`inckmed` *Increasing number of clusters in k-medoids algorithm*

Description

This function runs the increasing number of clusters in the k-medoids algorithm proposed by Yu et. al. (2018).

Usage

```
inckmed(distdata, ncluster, iterate = 10, alpha = 1)
```

Arguments

`distdata`	A distance matrix (n x n) or dist object.
`ncluster`	A number of clusters.
`iterate`	A number of iterations for the clustering algorithm.
`alpha`	A stretch factor to determine the range of initial medoid selection (see **Details**).

Details

This algorithm is claimed to manage with the weakness of the simple and fast-kmedoids (`fastkmed`). The origin of the algorithm is a centroid-based algorithm by applying the Euclidean distance. Then, Bbecause the function is a medoid-based algorithm, the object mean (centroid) and variance are redefined into medoid and deviation, respectively.

The `alpha` argument is a stretch factor, i.e. a constant defined by the user. It is applied to determine a set of medoid candidates. The medoid candidates are calculated by $O_c = \{X_i | \ \sigma_i \leq \alpha\sigma, i =$

$1, 2, \ldots, n$ }, where σ_i is the average deviation of object i, and σ is the average deviation of the data set. They are computed by

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} d(O_i, v_1)}$$

$$\sigma_i = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} d(O_i, O_j)}$$

where n is the number of objects, O_i is the object i, and v_1 is the most centrally located object.

Value

Function returns a list of components:

`cluster` is the clustering memberships result.

`medoid` is the id medoids.

`minimum_distance` is the distance of all objects to their cluster medoid.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Yu, D., Liu, G., Guo, M., Liu, X., 2018. An improved K-medoids algorithm based on step increasing and optimizing medoids. Expert Systems with Applications 92, pp. 464-473.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
result <- inckmed(mrwdist, ncluster = 3, iterate = 50, alpha = 1.5)
table(result$cluster, iris[,5])
```

`matching` *A pair distance for binary/ categorical variables*

Description

This function computes the simple matching distance from two data frames/ matrices.

Usage

```
matching(x, y)
```

Arguments

x	A first data frame or matrix (see **Details**).
y	A second data frame or matrix (see **Details**).

Details

The x and y arguments have to be data frames/ matrices with the same number of columns where the row indicates the object and the column is the variable. This function calculates all pairwise distance between rows in the x and y data frames/ matrices. If the x data frame/ matrix is equal to the y data frame/ matrix, it explicitly calculates all distances in the x data frame/ matrix.

The simple matching distance between objects i and j is calculated by

$$d_{ij} = \frac{\sum_{s=1}^{P}(x_{is} - x_{js})}{P}$$

where P is the number of variables, and $x_{is} - x_{js} \in \{0, 1\}$. $x_{is} - x_{js} = 0$, if $x_{is} = x_{js}$ and $x_{is} - x_{js} = 1$, when $x_{is} \neq x_{js}$.

As an example, the distance between objects 1 and 2 is presented.

object	x	y	z
1	1	2	2
2	1	2	1

The distance between objects 1 and 2 is

$$d_{12} = \frac{\sum_{s=1}^{3}(x_{is} - x_{js})}{3} = \frac{0 + 0 + 1}{3} = \frac{1}{3} = 0.33$$

Value

Function returns a distance matrix with the number of rows equal to the number of objects in the x data frame/ matrix (n_x) and the number of columns equals to the number of objects in the y data frame/ matrix (n_y).

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

Examples

```
set.seed(1)
a <- matrix(sample(1:2, 7*3, replace = TRUE), 7, 3)
matching(a, a)
```

msv *Medoid shadow value (MSV) index and plot*

Description

This function computes medoid shadow values and shadow value plots of each cluster. The plot presents the mean of the shadow values as well.

Usage

```
msv(distdata, idmedoid, idcluster, title = "")
```

Arguments

distdata	A distance matrix (n x n) or dist object.
idmedoid	A vector of id medoids (see **Details**).
idcluster	A vector of cluster membership (see **Details**).
title	A title of the plot.

Details

The origin of the shadow value is calculated in the shadow function of the **flexclust** package, in which it is based on the first and second closest centroid. The msv function in this package modifies the centroid into medoid such that the formula to compute shadow value of object i is

$$msv(i) = \frac{d(i, m'(i)) - d(i, m(i))}{d(i, m'(i))}$$

where $d(i, m(i))$ is the distance between object i to the first closest medoid and d(i, m'(i)) is the distance between object i to the second closest medoid.

The idmedoid argument corresponds to the idcluster argument. If the length of idmedoid is 3, for example, the idcluster has to have 3 unique cluster memberships, or it returns Error otherwise. The length of the idcluster has also to be equal to n (the number of objects). In contrast to the centroid shadow value, the medoid shadow value is interpreted likewise a silhoutte value, the higher value the better separation.

Value

Function returns a list with following components:

result is a data frame of the shadow values for all objects

plot is the shadow value plots of each cluster.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

F. Leisch. 2010 Neighborhood graphs, stripes and shadow plots for cluster visualization. Statistics and Computing. vol. 20, pp. 457-469

W. Budiaji. 2019 Medoid-based shadow value validation and visualization. International Journal of Advances in Intelligent Informatics. Vol 5 No 2 pp. 76-88

Examples

```
distiris <- as.matrix(dist(iris[,1:4]))
res <- fastkmed(distiris, 3)
sha <- msv(distiris, res$medoid, res$cluster)
sha$result[c(1:3,70:75,101:103),]
sha$plot
```

`pcabiplot` *Biplot of a PCA object*

Description

This function creates a biplot from a pca object, which is generated by the `prcomp` function from the **stats** package.

Usage

```
pcabiplot(
  PC,
  x = "PC1",
  y = "PC2",
  var.line = TRUE,
  colobj = rep(1, nrow(PC$x)),
  o.size = 1
)
```

Arguments

`PC`	A pca object generated by `prcomp` function.
`x`	X axis (see **Details**).
`y`	Y axis (see **Details**).
`var.line`	A logical input, if variable lines are plotted.
`colobj`	A vector to provide color in the objects (see **Details**).
`o.size`	A numeric number to set the object size.

Details

This is a function to plot a pca biplot from a pca object. The x and y axes can be supplied with any principle component. The length of the `colobj` vector has to be equal to the number of objects. This argument controls the color of the objects and is very convenient to explore the clustering result. The default value is that all object have the same color.

Value

Function returns a plot of pca.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

Examples

```
pcadat <- prcomp(iris[,1:4], scale. = TRUE)
pcabiplot(pcadat)
```

`rankkmed` *Rank k-medoid algorithm*

Description

This function runs the rank k-medoids algorithm proposed by Zadegan et. al. (2013).

Usage

```
rankkmed(distdata, ncluster, m = 3, iterate = 10, init = NULL)
```

Arguments

`distdata`	A distance matrix (n x n) or dist object.
`ncluster`	A number of clusters.
`m`	A number of objects to compute hostility (see **Details**).
`iterate`	A number of iterations for the clustering algorithm.
`init`	A vector of initial objects as the cluster medoids (see **Details**).

Details

This algorithm is claimed to cope with the local optima problem of the simple and fast-kmedoids algorithm (fastkmed). The m argument is defined by the user and has to be $1 < m \leq n$. The m is a hostility measure computed by

$$m_i = \sum_{X_j \in Y} r_{ij}$$

where x_j is the object j, Y is the set of objects as many as m, and r_{ij} is the rank distance, i.e. sorted distance, between object i and j.

init can be provided with a vector of id objects. The length of the vector has to be equal to the number of clusters. However, assigning a vector in the init argument, the algorithm is no longer the rank k-medoids algorithm.

Value

Function returns a list of components:

cluster is the clustering memberships result.

medoid is the id medoids.

minimum_distance is the distance of all objects to their cluster medoid.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

Zadegan, S.M.R, Mirzaie M, and Sadoughi, F. 2013. Ranked k-medoids: A fast and accurate rank-based partitioning algorithm for clustering large datasets. Knowledge-Based Systems 39, 133-143.

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
result <- rankkmed(mrwdist, ncluster = 3, iterate = 50)
table(result$cluster, iris[,5])
```

---

sil *Silhouette index and plot*

Description

This function creates silhouette indices and silhouette plots of each cluster. The plot presents also the mean of the silhouette indices per cluster.

Usage

```
sil(distdata, idmedoid, idcluster, title = "")
```

Arguments

`distdata`	A distance matrix (n x n) or dist object.
`idmedoid`	A vector of id medoids (see **Details**).
`idcluster`	A vector of cluster membership (see **Details**).
`title`	A title of the plot.

Details

The silhouette index of object i is calculated by

$$si(i) = \frac{b_i - a_i}{max(a_i, b_i)}$$

where a_i is the average distance of object i to all objects within the cluster, and b_i is the average distance of object i to all objects within the nearest cluster.

The `idmedoid` argument corresponds to the `idcluster` argument. If the length of `idmedoid` is 3, for example, the `idcluster` has to have 3 unique memberships, or it returns `Error` otherwise. The length of the `idcluster` has also to be equal to n (the number of objects).

Value

Function returns a list with following components:

`result` is a data frame of the silhouette indices for all objects

`plot` is the silhouette plots of each cluster.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

P. J. Rousseeuw. 1987 Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, vol. 20, pp. 53-65

Examples

```
distiris <- as.matrix(dist(iris[,1:4]))
res <- fastkmed(distiris, 3)
silhouette <- sil(distiris, res$medoid, res$cluster)
silhouette$result[c(1:3,70:75,101:103),]
silhouette$plot
```

skm *Simple k-medoid algorithm*

Description

This function runs the simple k-medoid algorithm proposed by Budiaji and Leisch (2019).

Usage

```
skm(distdata, ncluster, seeding = 20, iterate = 10)
```

Arguments

distdata	A distance matrix (n x n) or dist object.
ncluster	A number of clusters.
seeding	A number of seedings to run the algorithm (see **Details**).
iterate	A number of iterations for each seeding (see **Details**).

Details

The simple k-medoids, which sets a set of medoids as the cluster centers, adapts the simple and fast k-medoid algoritm. The best practice to run the simple and fast k-medoid is by running the algorithm several times with different random seeding options.

Value

Function returns a list of components:

`cluster` is the clustering memberships result.

`medoid` is the id medoids.

`minimum_distance` is the distance of all objects to their cluster medoid.

Author(s)

Weksi Budiaji
Contact: <budiaji@untirta.ac.id>

References

W. Budiaji, and F. Leisch. 2019. Simple K-Medoids Partitioning Algorithm for Mixed Variable Data. Algorithms Vol 12(9) 177

Examples

```
num <- as.matrix(iris[,1:4])
mrwdist <- distNumeric(num, num, method = "mrw")
result <- skm(mrwdist, ncluster = 3, seeding = 50)
table(result$cluster, iris[,5])
```

Index