Package: RelativeDistClust (via r-universe)

October 31, 2026

Type Package

Title Clustering with a Novel Non Euclidean Relative Distance

Version 0.1.0

Author Irene Creus Martí [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7962-4478>)

Maintainer Irene Creus Martí <ircrmar@mat.upv.es>

Description Using the novel Relative Distance to cluster datasets. Implementation of a clustering approach based on the k-means algorithm that can be used with any distance. In addition, implementation of the Hartigan and Wong method to accommodate alternative distance metrics. Both methods can operate with any distance measure, provided a suitable method is available to compute cluster centers under the chosen metric. Additionally, the k-medoids algorithm is implemented, offering a robust alternative for clustering without the need of computing cluster centers under the chosen metric. All three methods are designed to support Relative distances, Euclidean distances, and any user-defined distance functions. The Hartigan and Wong method is described in Hartigan and Wong (1979) <doi:10.2307/2346830> and an explanation of the k-medoids algorithm can be found in Reynolds et al (2006) <doi:10.1007/s10852-005-9022-1>.

License GPL-3

Encoding UTF-8

Imports compositions, proxy, utils, ggpubr, factoextra, ggplot2

Suggests testthat (>= 3.0.0), clusterSim, fpc, gtools, cluster

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Config/pak/sysreqs cmake libfreetype6-dev make libicu-dev libpng-dev libuv1-dev zlib1g-dev

Repository https://creus-marti.r-universe.dev

Date/Publication 2025-09-22 13:01:07 UTC

RemoteUrl https://github.com/cran/RelativeDistClust

RemoteRef HEAD

RemoteSha 09779eaa93ae9690ff6969185b716fb4e8476b0c

Contents

add_unique_numbers . . . 3
add_unique_numbers2 . . . 3
AitchisonDistance . . . 4
BrayCurtisDissimilarity . . . 5
centers_function_mean . . . 5
centers_function_RelativeDistance . . . 6
ClustPlot . . . 7
d_i_other_group . . . 8
DaviesBouldinIndex . . . 9
Dist_IC1_IC2 . . . 10
DistanceBetweenGroups . . . 11
DistanceSameGroup . . . 12
DosMinimos . . . 13
DunnIndex . . . 13
ECDentroCluster . . . 14
ECDentroCluster3 . . . 16
encontrar_componente . . . 17
Euclideandistance . . . 17
Hartigan_and_Wong . . . 18
Hartigan_and_Wong_total . . . 19
init_centers_hw . . . 21
init_centers_random . . . 22
kmedois_distance . . . 23
ManhattanDistance . . . 24
NEC . . . 25
NEC_total . . . 27
Number_of_failes . . . 28
RelativeDistance . . . 29
Silhouette . . . 30
Step4 . . . 31
Step6 . . . 33
to_minimize . . . 36
vector_a_lista . . . 37

Index **38**

add_unique_numbers *Add values to a vector if they are not already in it*

Description

This function adds two values to a vector if the values are not already in the vector.

Usage

```
add_unique_numbers(vector, num1, num2)
```

Arguments

vector	Vector with values
num1	Number. Value that will be added to the vector it it is no already in it.
num2	Number. Value that will be added to the vector it it is no already in it.

Value

Returns the vector with the values added if they are not alredy in the vector.

Examples

```
mi_vector <- c(1, 2, 3, 4, 5)
num1 <- 8
num2 <- 10

mi_vector <- add_unique_numbers(mi_vector, num1, num2)
```

add_unique_numbers2 *Add one value to a vector if it is not already there*

Description

This function adds one value to a vector if it is not already in the vector.

Usage

```
add_unique_numbers2(vector, num1)
```

Arguments

vector	Vector with values
num1	Number. Value that will be added to the vector it it is no already in it.

Value

Returns the vector with the value added if it is not already in the vector.

Examples

```
mi_vector <- c(1, 2, 3, 4, 5)
num1 <- 8

mi_vector <- add_unique_numbers2(mi_vector, num1)
```

AitchisonDistance *Aitchison distance*

Description

This function calculates the Aitchison distance between two vectors.

Usage

```
AitchisonDistance(vect1, vect2)
```

Arguments

vect1	vector
vect2	vector

Value

A number with the distance between `vect1` and `vect2`.

Examples

```
AitchisonDistance(c(1,2,3), c(4,5,6))
```

BrayCurtisDissimilarity *Bray-Curtis dissimilarity*

Description

This function calculates the Bray-Curtis dissimilarity between two vectors

Usage

```
BrayCurtisDissimilarity(x, y)
```

Arguments

x	vector
y	vector

Value

A number with the Bray-Curtis dissimilarity between x and y.

Examples

```
BrayCurtisDissimilarity(c(1,2,3), c(4,5,6))
```

centers_function_mean *Center of a cluster using the mean*

Description

This function calculates the center of a group using the mean of its components.

Usage

```
centers_function_mean(data, grouping)
```

Arguments

data	Matrix. The points that we want to group are in the rows.
grouping	List. List with the number of the rows of the data matrix that are in the group [[i]].

Value

A matrix. The row i contains the centers of the group in [[i]].

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
grouping=list(c(1,2), c(3,4),c(5,6))
centers_function_mean(data, grouping)
```

`centers_function_RelativeDistance`
Center of a cluster when the Relative distance is used.

Description

This function calculates the center of a group when the Relative distance is used to group.

Usage

```
centers_function_RelativeDistance(data, grouping)
```

Arguments

`data`	Matrix. The points that we want to group are in the rows.
`grouping`	List. List with the number of the rows of the data matrix that are in the group `[[i]]`.

Value

A matrix. The row i contains the centers of the group in `[[i]]`.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))

grouping=list(c(1,2), c(3,4),c(5,6))
centers_function_RelativeDistance(data, grouping)
```

`ClustPlot` *Plotting the clustring results*

Description

This function performs a PCA to reduce the dataset to two dimensions. Then, it draws the points, marks the center of the groups, the exact groups and the obtained groups.

Usage

```
ClustPlot(data, grouping, exact_grouping, centers, k)
```

Arguments

`data`	Matrix with `dim(data)[1]` points of `dim(data)[2]` dimensions.
`grouping`	List with information of the groups obtained using some clustering method. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to the group i are.
`exact_grouping`	List with the information of the real groups present in the data. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to the group i are.
`centers`	Matrix. Each row contains the center of each group. The groups are obtained using some clustering methods.
`k`	Number. Number of groups.

Value

Returns a plot where it is possible to visualize the he points, the center of the groups, the exact groups (represented in the type of point used to represent the data) and the obtained groups (observed in the geometric froms that join the points).

Examples

```
data=iris[,-5]
exact_grouping=list(which(iris[,5]=="setosa"),
                    which(iris[,5]=="versicolor"),
                    which(iris[,5]=="virginica"))

grouping=list(c(1:40),c(41:90),c(91:150))
k=3
centers=rbind(c(1,2,3,4),c(2,3,4,5),c(4,5,6,7))

ClustPlot(data, grouping, exact_grouping,centers, k)
```

d_i_other_group *Distance between a point and a group*

Description

This function calculates the distance between the point i of the data matrix and all the components in the group num.

Usage

```
d_i_other_group(data, i, distance, FHW_output, num)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
i	Number. Number of the row of data where the point is.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
FHW_output	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
num	Number. Number of the group from FHW_output$grouping.

Value

Returns a vector. The component j contains the distance between the point in the row i of the data matrix and the point j of the group num.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
         matrix(runif(20,20,30), nrow = 2, ncol = 10),
         matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5


FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
                             centers_function_mean,
                             init_centers_random,
                             seed=seed,
```

```
                              10)

d_i_other_group(data, 1, Euclideandistance, FHW_output,2)
```

DaviesBouldinIndex *Davies-Bouldin index*

Description

This function calculates the Davies-Bouldin index as is defined by Davies and Bouldin (1979) without imposing that the use of the euclidean distance. This function allows calculating the Davies-Bouldin index using different distances.

Usage

```
DaviesBouldinIndex(data, FHW_output, distance)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
FHW_output	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a number, the value of the Davies-Bouldin index.

References

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence, (2), 224-227.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5


FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
                             centers_function_mean,
                             init_centers_random,
                             seed=seed,
                             10)

DaviesBouldinIndex(data, FHW_output, Euclideandistance)
```

Dist_IC1_IC2 *Finding IC1 and IC2 from a distance matrix*

Description

This function finds the IC1 and IC2 from a distance matrix. IC1 and IC2 are the closets and second closest cluster centers.

Usage

```
Dist_IC1_IC2(Dist_e_cent)
```

Arguments

Dist_e_cent	Matrix. The position (i,j) contains the distance between the taxa i and the center j.

Value

Returns a matrix. The first column contain the IC1 and the second column contain the IC2.

Examples

```
dist=rbind(c(1,2,3),c(6,19,2),c(2,4,1),c(2,3,9))
Dist_IC1_IC2(dist)
```

DistanceBetweenGroups *Distance between groups*

Description

This function calculates the distance between points in two groups. For each point in the first group, it calculates the distance from that point to all points in the second group. Finally, it takes the minimum distance obtained.

Usage

```
DistanceBetweenGroups(group1, group2, FHW_output, distance, data)
```

Arguments

group1	Number. Number of the first group.
group2	Number. Number of the second group.
FHW_output	List. Output of the Hartigan_and_Wong function. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.

Value

Returns a number, the value of the minimum distance between pair of points of the two groups.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
         matrix(runif(20,20,30), nrow = 2, ncol = 10),
         matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5


FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
                             centers_function_mean,
                             init_centers_random,
                             seed=seed,
```

```
                            10)

DistanceBetweenGroups(1, 2, FHW_output, Euclideandistance, data)
```

`DistanceSameGroup` *Distance between points in the same group*

Description

This function calculates the distance between points in the same group. This function calculates the distance between the pair of points in the group. Then, takes the maximum distance.

Usage

```
DistanceSameGroup(group1, FHW_output, data, distance)
```

Arguments

`group1`	Number. Number of the group.
`FHW_output`	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
`data`	Matrix with `dim(data)[1]` points of `dim(data)[2]` dimensions.
`distance`	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a number, the value of the maximum distance between pair of points of the group.

Examples

```
set.seed(451)
data=rbind(matrix(runif(30,1,5), nrow = 3, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5


FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
```

```
                            centers_function_mean,
                            init_centers_random,
                            seed=seed,
                            10)

DistanceSameGroup(2,  FHW_output, data, Euclideandistance)
```

DosMinimos *Finding the two smallest values for each row of a matrix*

Description

This function finds the two smallest values for each row of a matrix `matriz`.

Usage

```
DosMinimos(matriz)
```

Arguments

`matriz`	Matrix

Value

Returns a matrix. The row i contains the two minimum values of the row i of the matrix `matriz`. The first column of the matriz contains the smallest value.

Examples

```
ma=rbind(c(5,4,3,2,1), c(10,9,8,7,6), c(120,119,103,104,105))
DosMinimos(ma)
```

DunnIndex *Dunn's index*

Description

This function calculates the Dunn's index as is defined in Bezdek and Pal (1995) without imposing that the use of the euclidean distance. This function allows calculating the Dunn's index using different distances.

Usage

```
DunnIndex(data, FHW_output, distance)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
FHW_output	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a number, the value of the Dunn's index.

References

Bezdek, J. C., & Pal, N. R. (1995, November). Cluster validation with generalized Dunn's indices. In Proceedings 1995 second New Zealand international two-stream conference on artificial neural networks and expert systems (pp. 190-193). IEEE.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
         matrix(runif(20,20,30), nrow = 2, ncol = 10),
         matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5

FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
                             centers_function_mean,
                             init_centers_random,
                             seed=seed,
                             10)

DunnIndex(data, FHW_output, Euclideandistance)
```

ECDentroCluster *Sum of squared errors within the cluster*

Description

The sum of squared errors within the cluster (also known as inertia) is calculated. We calculate the squared distance between the points that belong to a cluster and the cluster centroid. Then, we sum all the squared distances obtained. In this function the user can choose the distance that want to use to calculate the sum of squared errors within the cluster.

Usage

```
ECDentroCluster(data, FHW_output, distance)
```

Arguments

`data`	Matrix with `dim(data)[1]` points of `dim(data)[2]` dimensions.
`FHW_output`	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
`distance`	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a vector. The component i contains the sum of squared errors value of group i.

Examples

```
set.seed(231)
data1=gtools::rdirichlet(10,c(1,1,1,4,4))
data=t(data1)
grouping=list(c(1,2,3),c(4,5))
centers=centers_function_mean(data, grouping)
FHW_output=list(centers=centers,   grouping=grouping)
distance=Euclideandistance

ECDentroCluster(data, FHW_output, distance)
```

`ECDentroCluster3` *Sum of errors within the cluster*

Description

We calculate the distance between the points that belong to a cluster and the cluster centroid. Then, we sum all the distances obtained. In this function the user can choose the distance that want to use to calculate the sum of errors within the cluster.

Usage

```
ECDentroCluster3(data, FHW_output, distance)
```

Arguments

`data`	Matrix with `dim(data)[1]` points of `dim(data)[2]` dimensions.
`FHW_output`	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
`distance`	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a vector. The component i contains the sum of squared errors value of group i.

Examples

```
#'set.seed(231)
data1=gtools::rdirichlet(10,c(1,1,1,4,4))
data=t(data1)
grouping=list(c(1,2,3),c(4,5))
centers=centers_function_mean(data, grouping)
FHW_output=list(centers=centers,   grouping=grouping)
distance=Euclideandistance

ECDentroCluster3(data, FHW_output, distance)
```

encontrar_componente *Finding the component in the list that contains a value*

Description

This function finds in which component of the list `lista` the number `valor` is.

Usage

```
encontrar_componente(lista, valor)
```

Arguments

`lista`	List. Each component of the list has a vector. The different vector can not contain the same number.
`valor`	Number. We want to know in which component of the list `lista` the number `valor` is.

Value

Returns a number. Return the number of the component of `list` that contains the number `valor`.

Examples

```
mi_lista <- list(
a = c(1, 2, 3),
b = c(6,7,8,9),
c = c(4,5)
)

valor=7

encontrar_componente(mi_lista, valor)
```

Euclideandistance *Euclidean distance*

Description

This function calculates the euclidean distance between two vectors

Usage

```
Euclideandistance(vect1, vect2)
```

Arguments

vect1	vector
vect2	vector

Value

A number with the distance between vect1 and vect2.

Examples

```
Euclideandistance(c(1,2,3), c(4,5,6))
```

Hartigan_and_Wong *Flexibilization of the Hartigan and Wong algorithm*

Description

This function implements the Hartigan and Wong algorithm (Hartigan and Wong, 1979) without imposing the use of the euclidean distance and without imposing that the centers of the groups are calculated by averaging the points. This function allow the use of other distances and different ways to calculate the centers of the groups.

Usage

```
Hartigan_and_Wong(
  data,
  distance,
  k,
  centers_function,
  init_centers,
  seed = NULL,
  ITER
)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
k	Number. Number of groups into which we are going to group the different points.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.

init_centers	Function. This function designs how we are going to calculate the initial centers. The input must be the data, distance and k and the output must be a matrix where each row has the center of one group.
seed	Number. Number to fix a seed and be able to reproduce your results.
ITER	Number. Maximum number of iterations.

Value

Returns a list with:

- centers: the information of the centers updated. Matrix with dim(centers)[1] centers of dim(centers)[2] dimensions.
- grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.

References

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
         matrix(runif(20,20,30), nrow = 2, ncol = 10),
         matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5

Hartigan_and_Wong(data,
                  Euclideandistance,
                  k,
                  centers_function_mean,
                  init_centers_random,
                  seed=seed,
                  10)
```

Hartigan_and_Wong_total

Hartigan and Wong algorithm

Description

This function apply the Hartigan_and_Wong to different number of groups and calculates quality metrics as Silhouette.

Usage

```
Hartigan_and_Wong_total(
  data,
  distance,
  centers_function,
  init_centers,
  seed = NULL,
  ITER,
  KK = 10,
  index = "DaviesBouldin",
  k = NULL
)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
init_centers	Function. This function designs how we are going to calculate the initial centers. The input must be the data, distance and k and the output must be a matrix where each row has the center of one group.
seed	Number. Number to fix a seed and be able to reproduce your results.
ITER	Number. Maximum number of iterations.
KK	Number. Calculates the algorithm for the number of groups 2,3,...,KK. Default KK=10.
index	Character. If index="Silhouette" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Silhouette index. If index="DaviesBouldin" the function returns the results obtained with the number of groups (between 2 and KK) that minimize the Davies Bouldin index. If index="Dunn" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Dunn index. Default: "DaviesBouldin".
k	Number. If k is not NULL the function returns the results obtained with k groups.

Value

Returns a list with:

- Number_of_groups: Number of groups took into account to cluster.

- Output_of_grouping: list with the centers and the clusters.
- Quality: vector with the Silhouette index, Davies Bouldin Index, the Dunn index, the Within Cluster Sum (WCS) and the time (in seconds) that the function Hartigan_and_Wong needs to be executed. The WCS is equal to the sum of the distance of each point to the center of its group.

References

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))



RES=Hartigan_and_Wong_total(data,
                           RelativeDistance,
                           centers_function_RelativeDistance,
                           init_centers_random,
                           seed=10,
                           ITER=10,
                           KK=4,
                           index="DaviesBould",
                           k=NULL)
```

---

init_centers_hw *Initializing the centers*

Description

This function initializes the cluster centers following the procedure described in the 'Additional Comments' section of Hartigan and Wong (1979), without restricting the method to the use of Euclidean distance.

Usage

```
init_centers_hw(data, distance, k, centers_function)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
k	Number. Number of groups into which we are going to group the different points.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.

Value

Returns a matrix where each row is the center of a group.

References

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5

centr=init_centers_hw(data, Euclideandistance,k,centers_function_mean)
```

---

init_centers_random *Initializing the centers*

Description

This function initializes the centers of the groups randomly.

Usage

```
init_centers_random(data, distance, k, centers_function)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
k	Number. Number of groups into which we are going to group the different points.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.

Value

Returns a matrix where each row is the center of a group.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5

centr=init_centers_random(data, EuclideanDistance,k,centers_function_mean)
```

---

kmedois_distance *K-Medoids*

Description

This function apply the K-Medoids with any distance to different number of groups and calculates quality metrics as Silhouette.

Usage

```
kmedois_distance(data, distance, KK = 10, index = "DaviesBouldin", k = NULL)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
KK	Number. Calculates the K-Medoids for the number of groups 2,3,...,KK. Default KK=10.
index	Character. If index="Silhouette" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Silhouette index. If index="DaviesBouldin" the function returns the results obtained with the number of groups (between 2 and KK) that minimize the Davies Bouldin index. If index="Dunn" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Dunn index. Default: "DaviesBouldin".
k	Number. If k is not NULL the function returns the results obtained with the K-Medoids for k groups.

Value

Returns a list with:

- Number_of_groups: Number of groups took into account to cluster.
- Output_of_grouping: list with the centers and the clusters.
- Quality: vector with the Silhouette index, Davies Bouldin Index, the Dunn index, the Within Cluster Sum (WCS) and the time (in seconds) that the algorithm needs to be executed. The WCS is equal to the sum of the distance of each point to the center of its group.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))

kmedois_distance(data, RelativeDistance, KK=4, index="Silhouette", k=NULL)

kmedois_distance(data, RelativeDistance, k=2)
```

ManhattanDistance *Manhattan distance*

Description

This function calculates the Manhattan distance between two vectors

Usage

```
ManhattanDistance(x, y)
```

Arguments

x	vector
y	vector

Value

A number with the distance between x and y.

Examples

```
ManhattanDistance(c(1,2,3), c(4,5,6))
```

NEC *Non Euclidean Algorithm to Cluster*

Description

We give initial centers, calculate the distance between each point and each center and assign each point to the center with minimum distance. Calculate the center of the group and repeat the process. The process is stopped when the distance between a center and the previous one is small than COTA or the maximum number of iterations is reached.

Usage

```
NEC(data, distance, k, centers_function, init_centers, seed = NULL, ITER, COTA)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
k	Number. Number of groups into which we are going to group the different points.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
init_centers	Function. This function designs how we are going to calculate the initial centers. The input must be the data, distance and k and the output must be a matrix where each row has the center of one group.

seed	Number. Number to fix a seed and be able to reproduce your results.
ITER	Number. Maximum number of iterations.
COTA	Number. The process is stopped when the distance between a center and the previous one is smaller than COTA.

Value

Returns a list with:

- FHW_output; is a list with
  - centers: the information of the centers updated. Matrix with `dim(centers)[1]` centers of `dim(centers)[2]` dimensions.
  - grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
- Stop_Criteria: returns the distance between one center and the previous one for all the iterations
- Chanche_yes_no: matrix, in the position `[i,j]` returns "yes" if the point i have changed its group in the iteration j and return "no" if the point have not changed.
- all_output: is a list with the information of the center and the groups of each iteration of the process

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5


o2=NEC(data,
      RelativeDistance,
      k,
      centers_function_RelativeDistance,
      init_centers_random,
      seed=seed,
      10,
      0.01)
```

NEC_total *NEC algorithm*

Description

This function apply the NEC to different number of groups and calculates quality metrics as Silhouette.

Usage

```
NEC_total(
  data,
  distance,
  centers_function,
  init_centers,
  seed = NULL,
  ITER,
  COTA,
  KK = 10,
  index = "DaviesBouldinIndex",
  k = NULL
)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers. With grouping we mean a list. The list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
init_centers	Function. This function designs how we are going to calculate the initial centers. The input must be the data, distance and k and the output must be a matrix where each row has the center of one group.
seed	Number. Number to fix a seed and be able to reproduce your results.
ITER	Number. Maximum number of iterations.
COTA	Number. The process is stopped when the distance between a center and the previous one is smaller than COTA.
KK	Number. Calculates the algorithm for the number of groups 2,3,...,KK. Default KK=10.

index	Character. If index="Silhouette" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Silhouette index. If index="DaviesBouldin" the function returns the results obtained with the number of groups (between 2 and KK) that minimize the Davies Bouldin index. If index="Dunn" the function returns the results obtained with the number of groups (between 2 and KK) that maximize the Dunn index. Default: "DaviesBouldin".
k	Number. If k is not NULL the function returns the results obtained with k groups.

Value

Returns a list with:

- Number_of_groups: Number of groups took into account to cluster.
- Output_of_grouping: list with the centers and the clusters.
- Quality: vector with the Silhouette index, Davies Bouldin Index, the Dunn index, the Within Cluster Sum (WCS) and the time (in seconds) that algorithm needs to be executed. The WCS is equal to the sum of the distance of each point to the center of its group.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
         matrix(runif(20,20,30), nrow = 2, ncol = 10),
         matrix(runif(20,50,70), nrow = 2, ncol = 10))

RES=NEC_total(data,
             RelativeDistance,
             centers_function_RelativeDistance,
             init_centers_random,
             seed=10,
             ITER=10,
             0.01,
             KK=4,
             index="DaviesBould",
             k=NULL)
```

Number_of_failes *Comparison of groupings*

Description

This function compares the real clustering with a clustering obtained with some mathematical method. For each group, this function calculates the number of components that are in the expected grouping that are not in the real grouping. This function adds this value for all groups. It calculates it for all possible combinations of groups and returns the minimum value.

Usage

```
Number_of_failes(grouping_exact, grouping_obtained)
```

Arguments

grouping_exact	List. Each component of the list contains a vector with the components of one group. This list represents the actual grouping of the data.
grouping_obtained	List. Each component of the list contains a vector with the components of one group. This list represents the grouping obtained by some mathematical method.

Value

Returns a number with the quantity of points that are misclassified in the grouping_obtained.

Examples

```
grouping_exact=list(c(1,2,3,4,5),c(6,7),c(8,9))
grouping_obtained=list(c(1,3,7),c(2,4,6),c(8,9,5))

Number_of_failes(grouping_exact, grouping_obtained)
```

RelativeDistance *Relative Distance*

Description

This function calculates the Relative Distance between two vectors.

Usage

```
RelativeDistance(vect1, vect2)
```

Arguments

vect1	vector
vect2	vector

Value

A number with the distance between vect1 and vect2.

Examples

```
RelativeDistance(c(1,2,3), c(4,5,6))
```

Silhouette *Silhouette*

Description

This function calculates the Silhouette as is defined in Rousseeuw (1987) without imposing that the use of the euclidean distance. This allows calculating the Silhouette using different distances. Note that the Silhouette must be calculated using a distance that is a a ratio scale (Rousseeuw, 1987).

Usage

```
Silhouette(data, FHW_output, distance)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
FHW_output	List. List with: • centers: the information of the centers updated. • grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a vector. The component i contains the Silhouette value of the point in the row i of the data matrix.

References

Rousseeuw, P.J. (1987) Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math., 20, 53–65.

Examples

```
set.seed(451)
data=rbind(matrix(runif(20,1,5), nrow = 2, ncol = 10),
          matrix(runif(20,20,30), nrow = 2, ncol = 10),
          matrix(runif(20,50,70), nrow = 2, ncol = 10))
k=3
seed=5

FHW_output=Hartigan_and_Wong(data,
                             Euclideandistance,
                             k,
```

```
                              centers_function_mean,
                              init_centers_random,
                              seed=seed,
                              10)

Silhouette(data, FHW_output, Euclideandistance)
```

Step4 *Step 4 of the Hartigan and Wong algorithm*

Description

This function implements the Step 4 of the Hartigan and Wong (Hartigan and Wong, 1979) algorithm without imposing that the use of the euclidean distance and without imposing that the centers of the groups are calculated by averaging the points. This function allows other distances to be used and allows the centers of the groups to be calculated in different ways.

Usage

```
Step4(
  data,
  centers,
  grouping,
  LIVE_SET_original,
  distance,
  centers_function,
  Ic12_change,
  index
)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
centers	Matrix with dim(centers)[1] centers of dim(centers)[2] dimensions.
grouping	List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
LIVE_SET_original	Vector that contains the groups that have been modified in the previous Step 6. The Step 6 is described in Hartigan and Wong (1979).
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers.

Ic12_change	Matrix. The first row contains the IC1 of each point. The second column contains the IC2 of each point. IC1 and IC2 are the closets and second closest cluster centers.
index	Number. When a point is reallocated, index becomes zero.

Value

Returns a list with:

- centers: the information of the centers updated. Matrix with `dim(centers)[1]` centers of `dim(centers)[2]` dimensions.
- IC1andIC2: the information of the IC1 and IC2 updated. Matrix. The first row contains the IC1 of each point. The second column contains the IC2 of each point. IC1 and IC2 are the closets and second closest cluster centers.
- grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
- Live_set: Vector. Contains the groups that have been modified during the Step 4.
- no_Change: vector with the points that do not change its group. More specifically, contains the row of the matrix `data` where these points are.

References

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

Examples

```
set.seed(231)
data1=gtools::rdirichlet(10,c(1,1,4,4,20,20))
data=t(data1)
k=3
seed=5



if(!is.null(seed)){
 set.seed(seed)
}
centers <- data[sample(1:nrow(data), k), ]

#We calculate the distance between each row of the data matrix and the centers
Dist_e_cent=matrix(0,dim(data)[1],dim(centers)[1])
for (i in 1:(dim(data)[1])){
for (j in 1:(dim(centers)[1])){
 Dist_e_cent[i,j]=Euclideandistance(data[i,],centers[j,])
}
}
```

```
Ic12=Dist_IC1_IC2(Dist_e_cent)
Ic12_change=Ic12
Group=Ic12[,1]
grouping<-list()
for(i in 1:(max(Group))){
grouping[[i]]=which(Group==i)
}

#Update the clusters centers.
centers=centers_function_mean(data, grouping)

#Live set.
LIVE_SET_original1=c(1:length(grouping))

index=0

P1=Step4(data,
        centers,
        grouping,
        LIVE_SET_original1,
        Euclideandistance,
        centers_function_mean,
        Ic12_change,
        index)
```

Step6 *Step 6 of the Hartigan and Wong algorithm*

Description

This function implements the Step 6 of the Hartigan and Wong (Hartigan and Wong, 1979) algorithm without imposing that the use of the euclidean distance and without imposing that the centers of the groups are calculated by averaging the points. This function allows other distances to be used and allows the centers of the groups to be calculated in different ways.

Usage

```
Step6(
  data,
  centers,
  grouping,
  distance,
  centers_function,
  Ic12_change,
  Ic12,
```

```
  index
)
```

Arguments

data	Matrix with dim(data)[1] points of dim(data)[2] dimensions.
centers	Matrix with dim(centers)[1] centers of dim(centers)[2] dimensions.
grouping	List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
distance	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.
centers_function	Function. This function designs how the centers of the groups will be calculated. It must have as input data and grouping and as output a matrix that has the centers. This matrix will have as many rows as centers.
Ic12_change	Matrix. Contains IC1 and IC2 after the Step 4 is carried out. The first row contains the IC1 of each point. The second column contains the IC2 of each point. IC1 and IC2 are the closets and second closest cluster centers.
Ic12	Matrix. Contains IC1 and IC2 before the Step 4 is carried out. The first row contains the IC1 of each point. The second column contains the IC2 of each point. IC1 and IC2 are the closets and second closest cluster centers.
index	Number. When a point is reallocated, index becomes zero.

Value

Returns a list with:

- centers: the information of the centers updated. Matrix with dim(centers)[1] centers of dim(centers)[2] dimensions.
- IC1andIC2: the information of the IC1 and IC2 updated. Matrix. The first row contains the IC1 of each point. The second column contains the IC2 of each point. IC1 and IC2 are the closets and second closest cluster centers.
- grouping: the information of the groups updated. List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix data where the points belonging to group i are.
- Live_set: Vector. Contains the groups that have been modified during the Step 6.
- index: number. The information of index updated.

References

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 28(1), 100-108.

Examples

```
set.seed(231)
data12=gtools::rdirichlet(10,c(1,1,4,4,20,20))
data1=t(data12)
k=3
seed=5

distance<- function(vect1, vect2){
 sqrt(sum((vect1-vect2)^2))
}

centers_function<-function(data, grouping){
 center=matrix(0,length(grouping), dim(data)[2])
 for (i in 1:(length(grouping))){

   if(length(grouping[[i]])==1){
     center[i,]=data[grouping[[i]],]
   }else{
     center[i,]=apply(data[grouping[[i]],],2,mean)
   }
 }
 return(center)
}

if(!is.null(seed)){
 set.seed(seed)
}
centers <- data1[sample(1:nrow(data1), k), ]

#We calculate the distance between each row of the data matrix and the centers
Dist_e_cent=matrix(0,dim(data1)[1],dim(centers)[1])
for (i in 1:(dim(data1)[1])){
 for (j in 1:(dim(centers)[1])){
   Dist_e_cent[i,j]=distance(data1[i,],centers[j,])
 }
}

#We obtain the IC1 and IC2 for each taxa
Ic12_change=Dist_IC1_IC2(Dist_e_cent)
Group=Ic12_change[,1]
grouping<-list()
for(i in 1:(max(Group))){
 grouping[[i]]=which(Group==i)
}


#Update the clusters centers.
centers=centers_function(data1, grouping)

Ic12=cbind(c(1,1,3,3,2,2),c(1,2,1,2,3,3))

P1=Step6(data1, centers, grouping, distance, centers_function, Ic12_change,Ic12, 0)
```

`to_minimize` *Sum of the distance between the points in a group and a given center.*

Description

This function calculates the sum of the distance between the points in a group and a given center of the group. The function calculates these values for all groups and then adds them together. The user can choose which distance to choose.

Usage

```
to_minimize(inicenters_v, data, grouping, distance)
```

Arguments

`inicenters_v`	Vector. Vector with the centers of the groups that has more than one point. The centres are arranged by the number of the group. If a group has only one component, this center is not included in the vector. The vector contain all the components of the center of the first group (if this group has more than one point, otherwise the vector will start with the components of the center of the second group), then all the components of the center of the second group (if this group has more than one point), then all the components of the third group (if this group has more than one point), and so on until the center of all groups with more than one point are introduced.
`data`	Matrix with `dim(data)[1]` points of `dim(data)[2]` dimensions.
`grouping`	List. Each component of the list contains a vector with the points that belong to that group. More specifically, the list component i has a vector with the numbers of the row of the matrix `data` where the points belonging to group i are.
`distance`	Function. This function designs how the distance is going to be calculated. It must have as input two vectors and as output the distance of these vectors.

Value

Returns a number. First this function calculates the distance between each point of a group and its given center and sum these values. Then, the function sum the values obtained for each group. This is the output.

Examples

```
grouping=list(c(1,2,3),c(4,5),c(6,7))
set.seed(451)
data=t(gtools::rdirichlet(10, c(1,1,1,4,4,9,9)))
inicenters=runif(dim(data)[2]*length(grouping), 0.1, 0.9)
inicenters_v=as.vector(inicenters)
to_minimize(inicenters_v, data, grouping, Euclideandistance)
```

vector_a_lista *Vector to list*

Description

This function returns a list. The component of the list i contains the positions of the vector that are equal to i.

Usage

```
vector_a_lista(clustering_vector)
```

Arguments

clustering_vector
: Vector

Value

Returns a list. The component of the list i contains the positions of the vector that are equal to i.

Examples

```
vect=c(1,1,1,1,1,2,2,2,2,2,3,3,3,3,3)
vector_a_lista(vect)
```

Index

add_unique_numbers, 3
add_unique_numbers2, 3
AitchisonDistance, 4

BrayCurtisDissimilarity, 5

centers_function_mean, 5
centers_function_RelativeDistance, 6
ClustPlot, 7

d_i_other_group, 8
DaviesBouldinIndex, 9
Dist_IC1_IC2, 10
DistanceBetweenGroups, 11
DistanceSameGroup, 12
DosMinimos, 13
DunnIndex, 13

ECDentroCluster, 14
ECDentroCluster3, 16
encontrar_componente, 17
Euclideandistance, 17

Hartigan_and_Wong, 18
Hartigan_and_Wong_total, 19

init_centers_hw, 21
init_centers_random, 22

kmedois_distance, 23

ManhattanDistance, 24

NEC, 25
NEC_total, 27
Number_of_failes, 28

RelativeDistance, 29

Silhouette, 30
Step4, 31
Step6, 33

to_minimize, 36

vector_a_lista, 37