Package 'clustergas'

July 2, 2014

Type Package

Title A hierarchical clustering method based on genetic algorithms

Version 1.0

Date 2012-03-01

Depends cluster

Author Jose A. Castellanos-Garzon <jantonio_cu@ieee.org>, Fernando Diaz <fdiaz@infor.uva.es>

Maintainer Jose A. Castellanos-Garzon <jantonio_cu@ieee.org>

Description This package develops a genetic algorithm to build hierarchical clusterings from a dataset. It additionally provides a set of functions applied to cluster analysis.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2012-06-13 19:26:06

NeedsCompilation no

R topics documented:

`clustergas-package` *A hierarchical clustering method based on genetic algorithms.*

Description

This package develops a genetic algorithm to build hierarchical clusterings from a dataset. It additionally provides a set of functions applied to cluster analysis. This package most be initialized before its use, to do this, function "init.variables" most be call to create the variables of internal use. Finally, the main function of this package is "agnes.gas" (also known as HCGA or EMHC) that runs the genetic algorithm to build dendrograms from a dataset. A dendrogram of this package is represented as a list of lists of vectors. That is, each clustering of the dendrogram is represented as list of vectors, where each vector is a cluster and each list is a clustering. Hence, all nested clusterings of the dendrogram are collected in lists to form a new list representing a dendrogram.

Depends: R (>= 2.10.0), cluster

Details

Package:	clustergas
Type:	Package
Version:	1.0
Date:	2012-03-01
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Jose A. Castellanos-Garzon <jantonio_cu@ieee.org>, Fernando Diaz <fdiaz@infor.uva.es>.

Maintainer: Jose A. Castellanos-Garzon <jantonio_cu@ieee.org>

`a.rand.index2` *Adjusted Rand Index.*

Description

Adjusted Rand Index measures the extent of agreement between two clusterings. When Adjusted Rand Index value increases, the agreement grows.

Usage

```
a.rand.index2(refpartition, clustering)
```

Arguments

`refpartition`	A reference partition in form of a list of vectors representing a clustering. Vector c(0,0) most be added at the end of the list.
`clustering`	As the above argument. A list of vectors representing the clustering that would be compared with the reference partition. Every vector of the list represents a cluster of the clustering. Vector c(0,0) most be added at the end of the list. If the clustering (as a list) is part of a dendrogram, then it is not necessary to add vector c(0,0) at the end of the list.

Details

This function compares a reference partition of a dataset with a clustering using the Adjusted Rand Index as a similarity measure. Note that the object-data of the dataset are represented as numbers starting at 1. Therefore, object-data in each cluster (vector) are represented as numbers.

Value

A numeric value representing the similarity digree between both clusterings.

Note

init.variables function most be executed before using any other function of this package. It initializes the internal variables.

References

Rand WM: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 1971, 66:846-850.

See Also

degree.similarity, rand.index, jacard.coef, minkowski

Examples

```
refpart <- list(c(1, 3, 4), c(5, 2), c(6, 7, 8), c(0, 0))
clust <- list(c(8, 1, 2, 7, 3), c(5, 4, 6), c(0, 0))
a.rand.index2(refpart, clust)
```

---

ag.coef.mean1 *Agglomerative coefficient for function "clustering.fitness".*

Description

This function computes the level whose clustering has the maximum fitness value in a dendrogram according to fitness function "clustering.fitness".

Usage

```
ag.coef.mean1(dendogram)
```

Arguments

dendogram	A dendrogram given as list of lists of vectors.

Details

This function applies "clustering.fitness" to each clustering of the passed dendrogram to obtain the level where the fitness is maximum. This function spends less runtime than function "ag.coef1".

Value

A number representing the highest fitness level of dendrogram.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

ag.coef1, ag.coef.sigma

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
ag.coef.mean1(dendro)
```

---

ag.coef.sigma *Agglomerative coefficient for function "sigma.clust.fitness".*

Description

This function computes the level whose clustering has the maximum fitness value of the dendrogram according to fitness function "sigma.clust.fitness".

Usage

```
ag.coef.sigma(dendogram)
```

Arguments

dendogram	A dendrogram given by a list of lists of vectors.

Details

This function applies "sigma.clust.fitness" to each clustering of the dendrogram to obtain the level where the fitness is maximum. This function spends less runtime than function "ag.coef1".

Value

A number representing the level with the highest fitness of the dendrogram.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

ag.coef1, ag.coef.mean1

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4, fitness.f = "sigma.dendo.fitness1")
ag.coef.sigma(dendro)
```

---

ag.coef1 *Agglomerative coefficient for a fitness function.*

Description

This function computes the level whose clustering has the maximum fitness value in a dendrogram according to a given fitness function.

Usage

```
ag.coef1(dendogram, fitness.clust = "clustering.fitness")
```

Arguments

dendogram	A dendrogram given as list of lists of vectors.
fitness.clust	Fitness function for applying to the dendrogram.

Details

This function applies the passed fitness function to each clustering of the passed dendrogram to obtain the level where the fitness is maximum.

Value

A number representing the highest fitness level of dendrogram.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

ag.coef.mean1, ag.coef.sigma

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
ag.coef1(dendro, fitness.clust = "clustering.fitness")
```

agnes.gas *Genetic algorithm for data hierarchical clustering.*

Description

This is the main function of the package. This function runs a genetic algorithm that is a hierarchical clustering method building dendrograms from a dataset. It builds or reads from parameter "init.pop", an initial population of individuals (dendrogram population) to mate them a given number of generations through of the used genetic operators.

Usage

```
agnes.gas(dist.matrix, pop.size = 10, gen.numb = 10, part = 1/2,
          fitness.f = "fitness.mean", homog.f = "cluster.fitness",
          crossover = "crossover2", mutation = "mutation2",
          p.cross = 0.4, p.mut = 0.1, return.pop = FALSE,
          init.pop = NULL)
```

Arguments

dist.matrix	Proximity matrix (distance matrix) of the dataset to be clustered.
pop.size	Size of the population that will be created or loaded from a file.
gen.numb	Number of generations (iterations) to mate the individuals.
part	A fraction representing the part of the dendrograms that will be removed. By default "part = 1/2", which means that the first half of the dendrograms is removed. In other words, the size of the dendrograms is reduced to the half.
fitness.f	Name of the fitness function that measures the quality of the dendrograms of the genetic algorithm. We provide several fitness functions: fitness.mean, sigma.dendo.fitness1, fitness.mean.silhouette, fitness.meanD.HS and sigma.meanD.HS.
homog.f	Name of the homogeneity function used by the fitness function of the above parameter to measure the homogeneity of a cluster. All fitness functions use homogeneity function "cluster.fitness". Small values of homogeneity represent clusters of high quality whereas large values represent clusters of low quality.
crossover	Name of the crossover operator used by the genetic algorithm. We provide three types of operators, "crossover", "crosssover2" and "crossover3", all of them cross two dendrograms to produce a new dendrogram. The "crossover" operator chooses randomly a level on two dendrograms and the best clusters (according to homogeneity) of each dendrogram in the chosen level are taken out to form a new clustering. Which is used to build the child dendrogram. An agglomerative strategy that randomly merges two clusters of the current clustering in order to form the next upper levels is used to build the top part of the dendrogram. A divisive strategy that divides a cluster (randomly chosen) of the current level into two clusters to form the next lower levels is used to build the down part of the dendrogram.

The "crossover2" operator performs as the "crossover" one, but in this case, to build the child dendrogram from the new clustering, the agglomerative strategy repeatedly merges two clusters (usually, the 30% of all combinations of two clusters taken of the clustering are selected) and the most homogeny merge of them is taken out to form the next upper levels. On the other hand, the divisive strategy that builds the down part of the dendrogram just divides the less homogeneous cluster to form the next lower level, and so on.

The "crossover3" operator performs as the "crossover2" one, but in this case, before building the child dendrogram from the new clustering, an improvement strategy of this clustering which exchanges object-data between the clusters of this one is executed. Usually, this evolutionary strategy carries out 200 exchanges between randomly selected clusters.

`mutation` Name of the mutation operator used by the genetic algorithm. We provide three types of operators, "mutation", "mutation2" and "mutation3", all of them mute a dendrogram to produce another dendrogram.

The "mutation" operator chooses randomly a level of the dendrogram, afterwards, it merges two randomly selected clusters of the current level to form the next upper level and so on, until it reaches the top of the new dendrogram.

The "mutation2" operator performs as the "mutation" one. In this case, it repeatedly merges two randomly selected clusters (usually, the 30% of all combinations of two clusters are selected) of the chosen level. The most homogenous merge is chosen to form the next upper level, and so on, until it reaches the top of the new dendrogram.

The "mutation3" operator exchanges two object-data between the two clusters of the penultimate level of the dendrogram (the level that has only two clusters). Afterwards, that change is propagated to the lower levels of the dendrogram to obtain the new one.

`p.cross` Crossover likelihood applied to the crossover operator.

`p.mut` Mutation likelihood applied to the mutation operator.

`return.pop` If it is FALSE, then this function returns the best individual of the whole evolutionary process. That is, the best dendrogram. If it is TRUE, then this function returns a list with two named elements. The first element, named "elite", has the best individual (the best dendrogram) of the whole evolutionary process, and the second one, named "lastpop", has the last population of individuals, generated in the last generation of the algorithm.

`init.pop` It represents the initial population of individuals (dendrograms) to be used as a starting point by this function. If "init.pop" is NULL, then this function creates a random initial population. Otherwise, the initial population of this function is taken from the argument assigned by the user to "init.pop".

Details

Argument "part" is used to define the new size of the dendrograms in the whole evolutionary process of the method. Thus, it reduces the level number of the dendrograms to a size defined by the user. That is, "dendrogram size = n - 2 - |part * n|", where n is the size of the used dataset. That is, if the size of a dataset is 20 objects, then a dendrogram of this one will have 20 levels. But if for example,

we want to delete the fourth part of the dendrograms, that is, part = 1/4, then "20 - 2 - |1/4 * 20| = 13" is the new size of the dendrograms.

Value

If parameter "return.pop" has FALSE, then this function returns a list (a dendrogram) of lists (clusterings in the dendrogram) of vectors (clusters) representing to the best dendrogram of the whole evolutionary process in the genetic algorithm. Otherwise, it returns a list with two named arguments, where the first element is the best dendrogram and second one is the last population of individuals as explained for parameter "return.pop". Additionally, the time taken by the execution of the algorithm and the fitness value of the best dendrogram are shown.

Note

Keep in mind that returning the last population of invidividuals through parameter "return.pop = TRUE", allows to the user running this function (the genetic algorithm) over again but taking as initial population the last population returned by the above running, through parameter "init.pop". This way, the current solutions can be improved. Moreover, for that purpose, the last population of individuals generated by "agnes.gas" can also be given as initial population to function `local.search`, which performs an evolutionary local search.

See Also

`local.search`, `improve.dendo`, `clustering.dendo`, `complete.tree`, `dendrogram.graph`

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
# Example 1
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)

# Example 2
result <- agnes.gas(vr.matrix, 5, 10, part = 3/4, return.pop = TRUE)
best.ind <- result$elite
dendro1 <- agnes.gas(vr.matrix, 5, 50, part = 3/4, init.pop = result$lastpop)
fitness.mean(best.ind)
```

---

clustering.dendo *Evolutionary strategy to improve a clustering.*

Description

This function improves a clustering by exchanging or moving object-data between the clusters of such a clustering. At the end, it builds a dendrogram in the way as crossover operator "crossover3". See function `agnes.gas` for more information on this operator.

Usage

```
clustering.dendo(clustering, level, times = 500, count = 10,
                 fitness.f = "clustering.fitness", homog.f = "cluster.fitness",
                 print.fitness = FALSE)
```

Arguments

clustering	Clustering to improve (a list of vectors, each vector represents a cluster).
level	Level at which this clustering belongs.
times	Exchange number of each iteration.
count	Iteration number.
fitness.f	Name of the fitness function that measures the quality of the dendrograms. We provide several fitness functions: fitness.mean, sigma.dendo.fitness1, fitness.mean.silhouette, fitness.meanD.HS and sigma.meanD.HS.
homog.f	Name of the homogeneity function used by the fitness function of the above parameter to measure the homogeneity of a cluster. All fitness functions use homogeneity function "cluster.fitness". Small values of homogeneity represent clusters of high quality whereas large values represent clusters of low quality.
print.fitness	If it is TRUE, the fitness values (before and after improving the clustering) and the time spent by the function execution are printed on the screen.

Value

A list (a dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters) representing the dendrogram built from the improved clustering.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

agnes.gas, local.search, improve.dendo, complete.tree, dendrogram.graph

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
dendro2 <- clustering.dendo(clust, 6, times = 10, count = 20, print.fitness = TRUE)
```

clustering.fitness *Main fitness function of a clustering.*

Description

This function computes the clustering fitness based on the difference between the separation and homogeneity mean of the clusters.

Usage

```
clustering.fitness(clustering)
```

Arguments

clustering	A clustering represented by a list of vectors.

Value

A numerical value representing the clustering fitness.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

sigma.clust.fitness, silhouette.clust, fitness.meanC.HS, sigma.clust.HS.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
clust <- dendro[[5]]
fitness <- clustering.fitness(clust)
fitness
```

`complete.tree` *Adds the missing levels of a dendrogram to reach its original size.*

Description

This function complete the levels of a dendrogram whose size was reduced by parameter "part" (see function `agnes.gas`) to its original size.

Usage

```
complete.tree(dendogram)
```

Arguments

`dendogram`	A dendrogram whose size is less than its original size. That is, the original size of a dendrogram is the size of the used dataset less 2.

Details

To build the down part of the input dendrogram, a divisive strategy that divides a cluster (chosen on the last level of the dendrogram) into two ones is executed in order to form the next lower level. And so on until it reaches the first level of the new dendrogram. This function is usually used in conjunction with function `dendrogram.graph`, which converts a dendrogram to an object of class "agnes" (see package "cluster"). Class "agnes" represents an agglomerative hierarchical clustering of a dataset.

Value

A dendrogram whose size is the size of the dataset less 2.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`agnes.gas`, `local.search`, `improve.dendo`, `clustering.dendo`, `dendrogram.graph`

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
length(dendro)
fulldendro <- complete.tree(dendro)
length(fulldendro)
```

`degree.similarity` *Builds the vector of agreement and disagreement between the two clusterings passed as arguments.*

Description

This function builds a vector of four components, where each one has the number of object pairs together (or not) in both clusterings.

Usage

```
degree.similarity(clustering, groundtruth)
```

Arguments

`clustering`	The clustering to be compared with the other argument.
`groundtruth`	A clustering which acts as a reference partition of the used data set.

Details

The vector returned by this function is used to measure the agreement and disagreement through functions "rand.index", "jacard.coef" and "minkowski". Therefore, this vector is the input to these functions.

Value

A vector representing in each component, the number of object pairs together or separated into each cluster of both clusterings.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

References

Jiang D, Tang C, Zhang A: Cluster Analysis for Gene Expression Data: A Survey. IEEE Transactions on Knowledge and Data Engineering 2004, 16(11):1370-1386.

See Also

`rand.index`, `jacard.coef`, `minkowski`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 100, part=3/4)
clust <- dendro[[6]]
dendro <- agnes.gas(vr.matrix, 5, 100, part=3/4)
ref.part <- dendro[[7]]
nij <- degree.similarity(clust, ref.part)
nij
```

---

dendrogram.graph *Converts a full dendrogram to an object of class "agnes".*

Description

This function converts a full dendrogram (with all levels) to an object of class "agnes".

Usage

```
dendrogram.graph(dendogram, datam, fitness.clust = "clustering.fitness",
                 homog.f = "cluster.fitness")
```

Arguments

dendogram	The dendrogram with all levels to convert it to an object of class "agnes".
datam	used dataset.
fitness.clust	Name of the fitness function that measures the quality of a clustering. We provide several fitness functions: clustering.fitness, sigma.clust.fitness, silhouette.clust, fitness.meanC.HS, sigma.clust.HS.
homog.f	Name of the homogeneity function used by the fitness function of the above parameter to measure the homogeneity of a cluster. All fitness functions use homogeneity function "cluster.fitness". Small values of homogeneity represent clusters of high quality whereas large values represent clusters of low quality.

Details

The dendrogram is converted to an object of class "agnes", which allows to use R functions that handle this type of object. The objects of class "agnes" represent an agglomerative hierarchical clustering of a dataset. Namely, we can use function "plot" that displays the graph of the dendrogram on a dataset. See package "cluster" of R for more information on "agnes.object".

Value

An object of class "agnes" representing a dendrogram.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`agnes.gas`, `local.search`, `improve.dendo`, `clustering.dendo`, `complete.tree`, `transf.tree`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
agn <- dendrogram.graph(complete.tree(dendro), votes.repub)
plot(agn)
```

---

evol.cluster *Evolutionary strategy to improve a clustering.*

Description

This function is an evolutionary strategy that improves a clustering according to a fitness function by exchanging two objects between two different clusters or moving one object from a cluster to another.

Usage

```
evol.cluster(clustering, gen.numb, fitness.f = "clustering.fitness",
             mop = "gene.change1", print.fitness = FALSE)
```

Arguments

`clustering`	The clustering to improve.
`gen.numb`	Number of generations. Number of times that the operator is applied.
`fitness.f`	Fitness function of the clustering.
`mop`	Type of operator to use, that can be "gene.change1" or "gene.change2".
`print.fitness`	If it is TRUE, the fitness values of the initial and final clustering as well as the runtime are screen printed.

Details

Operator "gene.change1" moves one object from a cluster to another cluster. Operator "gene.change2" exchanges two objects located in different clusters.

Value

A new clustering as a result of the applied evolution process.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`exec.strag`, `clustering.dendo`, `improve.dendo`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
clust2 <- evol.cluster(clust, 100, mop = "gene.change2", print.fitness = TRUE)
```

---

exec.strag *Evolutionary strategy to improve a clustering.*

Description

This function is an evolutionary strategy that improves a clustering according to a fitness function by applying two genetic operators that exchanging two objects between two different clusters and moving one object from a cluster to another.

Usage

```
exec.strag(clustering, times = 1000, count = 10, fitness.f = "clustering.fitness",
           print.fitness = FALSE)
```

Arguments

`clustering`	The clustering to improve.
`times`	Number of times that each operator is applied.
`count`	Number of times that the "times" parameter is applied. That is, number of times that both operators are applied each "times" times.
`fitness.f`	Fitness function applied to the clustering.
`print.fitness`	If it is TRUE, the fitness values of the initial and final clustering as well as the runtime are screen printed.

Details

The operators used are "gene.change1" and "gene.change2", which move one object from a cluster to another and exchange two objects located in different clusters, respectively.

Value

A new clustering as a result of the applied evolution process.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`evol.cluster`, `clustering.dendo`, `improve.dendo`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
clust2 <- exec.strag(clust, times = 100, count = 5, print.fitness = TRUE)
```

`fitness.mean` *Main fitness function of a dendrogram.*

Description

This is the main fitness function of a dendrogram in the package. This function computes the dendrogram fitness based on the mean value of the fitness of each clustering of the dendrogram. So, the dendrogram fitness is computed by the mean fitness of the dendrogram clusterings using function `clustering.fitness`.

Usage

```
fitness.mean(dendogram)
```

Arguments

`dendogram`	A dendrogram represented by a list (dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters).

Value

A value representing the dendrogram fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`sigma.dendo.fitness1`, `fitness.mean.silhouette`, `fitness.meanD.HS`, `sigma.meanD.HS`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
fitness <- fitness.mean(dendro)
fitness
```

---

fitness.mean.silhouette

Fitness function measuring the dendrogram silhouette width.

Description

This function computes the dendrogram fitness based on the mean value of the silhouette width of each clustering (`silhouette.clust`) of the one.

Usage

```
fitness.mean.silhouette(dendogram)
```

Arguments

`dendogram`	A dendrogram represented by a list (dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters).

Value

A value representing the dendrogram fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

References

L. Kaufman, P. J. Rousseeuw, Finding Groups in Data. An Introduction to Clustering Analysis, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

See Also

fitness.mean, sigma.dendo.fitness1, fitness.meanD.HS, sigma.meanD.HS.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
fitness <- fitness.mean.silhouette(dendro)
fitness
```

---

fitness.meanC.HS *Clustering fitness function based on separation and homogeneity.*

Description

This function computes the clustering fitness based on the difference between the separation and homogeneity average of the clusters.

Usage

```
fitness.meanC.HS(clustering)
```

Arguments

clustering	A clustering represented by a list of vectors.

Value

A value representing the clustering fitness.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

References

D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: A survey, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1370-1386.

See Also

clustering.fitness, sigma.clust.fitness, silhouette.clust, sigma.clust.HS.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
clust <- dendro[[5]]
fitness <- fitness.meanC.HS(clust)
fitness
```

`fitness.meanD.HS` *Fitness function of a dendrogram based on function "fitness.meanC.HS".*

Description

This function computes the fitness of a dendrogram based on the difference between homogeneity and separation given by function "fitness.meanC.HS". Which computes the fitness of each clustering of the dendrogram passed as parameter. Finally, "fitness.meanD.HS" computes the mean value of all values given by "fitness.meanC.HS".

Usage

```
fitness.meanD.HS(dendogram)
```

Arguments

`dendogram`	A dendrogram represented as a list of lists of vectors.

Value

A numerical value representing the fitness value of the dendrogram.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`sigma.dendo.fitness1`, `fitness.mean.silhouette`, `fitness.mean`, `sigma.meanD.HS`

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
fitness <- fitness.meanD.HS(dendro)
fitness
```

H.ave *Clustering homogeneity measure.*

Description

This function measures the homogeneity of a clustering.

Usage

```
H.ave(clustering)
```

Arguments

clustering	A clustering in form of a list of vectors.

Value

A numerical value representing the homogeneity of the clustering.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

References

Jiang D, Tang C, Zhang A, Cluster Analysis for Gene Expression Data: A Survey. IEEE Transactions on Knowledge and Data Engineering 2004, 16(11):1370-1386.

See Also

S.ave, silhouette.mean.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]]
H.ave(clust)
```

improve.dendo *Evolutionary strategy to improve a dendrogram.*

Description

This function carries out an improvement of a dendrogram by using a mutation operator.

Usage

```
improve.dendo(dendo, ite = 10, operator = "mutation2",
              fitness.f = "fitness.mean", homog.f = "cluster.fitness",
              print.fitness = FALSE)
```

Arguments

dendo	Dendrogram to improve (as a list of lists of vectors).
ite	Iteration number of the strategy.
operator	Name of the mutation operator used by the evolutionary strategy. We provide three types of operators, "mutation", "mutation2" and "mutation3", all of them mute a dendrogram to produce another dendrogram. These operators are explained in function agnes.gas.
fitness.f	Name of the fitness function that measures the quality of the dendrograms. We provide several fitness functions: fitness.mean, sigma.dendo.fitness1, fitness.mean.silhouette, fitness.meanD.HS and sigma.meanD.HS.
homog.f	Name of the homogeneity function used by the fitness function of the above parameter to measure the homogeneity of a cluster. All fitness functions use homogeneity function "cluster.fitness". Small values of homogeneity represent clusters of high quality whereas large values represent clusters of low quality.
print.fitness	If it is TRUE, the fitness values (before and after improving the input dendrogram) and the time spent by the function execution are printed on the screen.

Value

A list (a dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters) representing a dendrogram as a improvement of the input dendrogram.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

agnes.gas, local.search, clustering.dendo, complete.tree, dendrogram.graph

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
dendro2 <- improve.dendo(dendro, ite = 20, print.fitness = TRUE)
```

---

init.variables *Initialization of the global variables of the package.*

Description

This function creates and initializes all global variables used by the functions defined in this package. It most be call before using any other function.

Usage

```
init.variables(dist.matrix, pop.size = 10, gen.numb = 10, p.cross = 0.4,
               p.mut = 0.1, part = 1/2)
```

Arguments

dist.matrix	Proximity matrix of the dataset to be clustered.
pop.size	Size of the population to be used by the genetic algorithm functions.
gen.numb	Number of generations (iterations) of the genetic algorithm.
p.cross	Value of the crossover likelihood of the genetic algorithm.
p.mut	Value of the mutation likelihood of the genetic algorithm.
part	It is a fraction representing the part of the dendrogram that we want to remove (example: part = 3/4).

Details

Argument "part" is used to remove the firts part of the dendrogram if we do not want that it has all its levels. That is, if a dendrogram has 20 levels and part= 2/3, then it really will have 20 - 2 - |2/3 * 20| = 5 levels, removing the 15 first levels. By default "part = 1/2", which means that the first half of the dendrogram is removed.

Value

No return value.

Note

This is the firts function that most be called before using any other function different of "agnes.gas". Because it initializes important global variables.

See Also

agnes.gas and local.search.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
init.variables(vr.matrix, part = 3/4)
```

---

jacard.coef *Jacard Coefficient measure.*

Description

Jacard Coefficient measures the extent of agreement between two clusterings. When the Jacard Coefficient value increases, the agreement grows.

Usage

```
jacard.coef(nij)
```

Arguments

nij	it is a vector returned by function degree.similarity.

Details

This function compares a reference partition of a dataset with a clustering using Jacard Coefficient as a similarity measure. Note that the object-data of the dataset are represented as numbers starting at 1. Therefore, object-data in each cluster (vector) are represented as numbers. The reference partition and the clustering are passed as input to degree.similarity, which returns a vector nij.

Value

A numeric value representing the similarity digree between two clusterings.

Note

init.variables function most be executed before using any other function of this package. It initializes the internal variables.

References

Halkidi M, Batistakis Y, Vazirgiannis M: On Clustering Validation Techniques. In Intelligent Information Systems J. 2001.

Sokal RR: Clustering and Classification: Background and Current Directions. Classification and Clustering, Academic Press 1977.

See Also

degree.similarity, rand.index, a.rand.index2, minkowski

Examples

```
refpart <- list(c(1, 3, 4), c(5, 2), c(6, 7, 8), c(0, 0))
clust <- list(c(8, 1, 2, 7, 3), c(5, 4, 6), c(0, 0))
nij <- degree.similarity(refpart, clust)
jacard.coef(nij)
```

---

local.search *Evolutionary strategy of local search.*

Description

This function is an evolutionary strategy of local search that improves the individuals (dendrograms) from a initial population. The initial population can randomly be created or read from parameter "init.pop". Usually, this function is used to improve the individuals returned in a population of individuals by function agnes.gas.

Usage

```
local.search(dist.matrix, pop.size = 10, gen.numb, part = 1/2,
             mutation = "mutation2", fitness.f = "fitness.mean",
             homog.f = "cluster.fitness", return.pop = FALSE,
             init.pop = NULL)
```

Arguments

dist.matrix	Proximity matrix (distance matrix) of the dataset to be clustered.
pop.size	Size of the population that will be created or loaded from a file.
gen.numb	Number of generations (iterations) to mate the individuals.
part	A fraction representing the part of the dendrograms that will be removed. By default part = 1/2, which means that the first half of the dendrograms are removed. In other words, the size of the dendrograms is reduced to the half. See section details of function agnes.gas for more information.
mutation	Name of the mutation operator used by the evolutionary strategy. We provide three types of operators, "mutation", "mutation2" and "mutation3", all of them mute a dendrogram to produce another dendrogram. These operators are explained in function agnes.gas.
fitness.f	Name of the fitness function that measures the quality of the dendrograms. We provide several fitness functions: fitness.mean, sigma.dendo.fitness1, fitness.mean.silhouette, fitness.meanD.HS and sigma.meanD.HS.

homog.f	Name of the homogeneity function used by the fitness function of the above parameter to measure the homogeneity of a cluster. All fitness functions use homogeneity function "cluster.fitness". Small values of homogeneity represent clusters of high quality whereas large values represent clusters of low quality.
return.pop	If it is FALSE, then this function returns the best individual of the whole evolutionary process. That is, the best dendrogram. If it is TRUE, then this function returns a list with two named elements. The first element, named "elite", has the best individual (the best dendrogram) of the whole evolutionary process, and the second one, named "lastpop", has the last population of individuals, generated in the last generation of the algorithm.
init.pop	It represents the initial population of individuals (dendrograms) to be used as a starting point by this function. If "init.pop" is NULL, then this function creates a random initial population. Otherwise, the initial population of this function is taken from the argument assigned by the user to "init.pop".

Details

This evolutionary strategy mutes all individuals in each generation by using the passed mutation operator. It only replaces the individuals that improve their fitness.

Value

If parameter "return.pop" has FALSE, then this function returns a list (a dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters) representing the best dendrogram of the whole evolutionary process. Otherwise, it returns a list with two named arguments, where the first element is the best dendrogram and second one is the last population of individuals as explained for parameter "return.pop". Additionally, the time taken by the execution of the algorithm and the fitness value of the best dendrogram are shown.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

agnes.gas, improve.dendo, clustering.dendo, complete.tree, dendrogram.graph

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))

# Example 1
dendro1 <- local.search(vr.matrix, pop.size = 5, 10, part=3/4)

# Example 2
result.list <- agnes.gas(vr.matrix, 5, 10, part = 3/4, return.pop = TRUE)
dendro2 <- result.list$elite
```

```
population <- result.list$lastpop
# Improving the individuals of "population".
dendro3 <- local.search(vr.matrix, pop.size = 5, 50, part=3/4,
                        init.pop = population)
fitness.mean(dendro2)
```

`minkowski` *Minkowski Measure.*

Description

Minkowski Measure illustrates the disagreement proportion between two clusterings. When the Minkowski Measure value decreases, the disagreement becomes smaller.

Usage

```
minkowski(nij)
```

Arguments

`nij`	It is a vector returned by function `degree.similarity`.

Details

This function compares a reference partition of a dataset with a clustering using the Minkowski Measure as a disimilarity measure. Note that the object-data of the dataset are represented as numbers starting at 1. Therefore, object-data in each cluster (vector) are represented as numbers. The reference partition and the clustering are passed as input to `degree.similarity`, which returns a vector nij.

Value

A numeric value representing the disimilarity digree between two clusterings.

Note

`init.variables` function most be executed before using any other function of this package. It initializes the internal variables.

References

Halkidi M, Batistakis Y, Vazirgiannis M: On Clustering Validation Techniques. In Intelligent Information Systems J. 2001.

Sokal RR: Clustering and Classification: Background and Current Directions. Classification and Clustering, Academic Press 1977.

See Also

`degree.similarity`, `rand.index`, `a.rand.index2`, `jacard.coef`

Examples

```
refpart <- list(c(1, 3, 4), c(5, 2), c(6, 7, 8), c(0, 0))
clust <- list(c(8, 1, 2, 7, 3), c(5, 4, 6), c(0, 0))
nij <- degree.similarity(refpart, clust)
minkowski(nij)
```

`missvalue.knn` *Replacing missing values using k-nearest neighbors.*

Description

This function replaces missing values by values computed from the technique of k-nearest neighbors.

Usage

```
missvalue.knn(datamat, K = 20)
```

Arguments

`datamat`	Data matrix (dataset) to remove missing values.
`K`	The number of neighbors used to compute the new value that replaces the missing value.

Value

A data matrix without missing values.

References

Jain A. K., Dubes RC: Algorithms for Clustering Data. Prentice Hall Englewood Cliffs, New Jersey 07632 1998.

See Also

`standard`.

Examples

```
library(cluster)
data(votes.repub)
newdata <- missvalue.knn(votes.repub, K = 5)
```

`rand.index` *Rand Index measure.*

Description

Rand Index measures the extent of agreement between two clusterings. When the Rand Index value increases, the agreement grows.

Usage

```
rand.index(nij)
```

Arguments

`nij`	it is a vector returned by function `degree.similarity`.

Details

This function compares a reference partition of a dataset with a clustering using the Rand Index as a similarity measure. Note that the object-data of the dataset are represented as numbers starting at 1. Therefore, object-data in each cluster (vector) are represented as numbers. The reference partition and the clustering are passed as input to `degree.similarity`, which returns a vector nij.

Value

A numeric value representing the similarity digree between two clusterings.

Note

`init.variables` function most be executed before using any other function of this package. It initializes the internal variables.

References

Halkidi M, Batistakis Y, Vazirgiannis M: On Clustering Validation Techniques. In Intelligent Information Systems J. 2001.

Sokal RR: Clustering and Classification: Background and Current Directions. Classification and Clustering, Academic Press 1977.

See Also

`degree.similarity`, `a.rand.index2`, `jacard.coef`, `minkowski`

Examples

```
refpart <- list(c(1, 3, 4), c(5, 2), c(6, 7, 8), c(0, 0))
clust <- list(c(8, 1, 2, 7, 3), c(5, 4, 6), c(0, 0))
nij <- degree.similarity(refpart, clust)
rand.index(nij)
```

S.ave *Clustering separation measure.*

Description

This function measures the separation of a clustering.

Usage

```
S.ave(clustering)
```

Arguments

clustering	A clustering represented by a list of vectors.

Value

A numerical value representing the separation of the clustering.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

References

Jiang D, Tang C, Zhang A: Cluster Analysis for Gene Expression Data: A Survey. IEEE Transactions on Knowledge and Data Engineering 2004, 16(11):1370-1386.

See Also

H.ave, silhouette.mean.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
S.ave(clust)
```

`sigma.clust.fitness` *Clustering fitness function considering the size of the clusters.*

Description

This fitness function is similar to function `clustering.fitness`, but in this case, it adds a new objective (sigma) that measures the number of data-object of each clusters.

Usage

```
sigma.clust.fitness(clustering)
```

Arguments

`clustering`	A clustering represented by a list of vectors.

Details

This function additionally checks that the size of the clusters is not very different from each other.

Value

A value representing the clustering fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`clustering.fitness`, `silhouette.clust`, `fitness.meanC.HS`, `sigma.clust.HS`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
clust <- dendro[[5]]
fitness <- sigma.clust.fitness(clust)
fitness
```

`sigma.clust.HS` *Clustering fitness function based on separation, homogeneity and the size of the clusters.*

Description

This function performs as `fitness.meanC.HS` but in this case, a new objective was added, a check of the size of the clusters.

Usage

```
sigma.clust.HS(clustering)
```

Arguments

`clustering`	A clustering represented by a list of vectors.

Details

This function additionally checks that the size of the clusters is not very different from each other.

Value

A value representing the clustering fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

References

D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: A survey, IEEE Transactions on Knowledge and Data Engineering 16 (11) (2004) 1370-1386.

See Also

`clustering.fitness`, `sigma.clust.fitness`, `silhouette.clust`, `sigma.clust.HS`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
clust <- dendro[[5]]
fitness <- sigma.clust.HS(clust)
fitness
```

`sigma.dendo.fitness1` *Dendrogram fitness function checking the cluster size.*

Description

This fitness function is similar to function `fitness.mean`, but in this case, it adds a new objective (sigma) that measures the number of object-data of each clusters in the dendrogram.

Usage

```
sigma.dendo.fitness1(dendogram)
```

Arguments

`dendogram`	A dendrogram represented by a list (dendrogram) of lists (clusterings of the dendrogram) of vectors (clusters).

Value

A value representing the dendrogram fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`fitness.mean`, `fitness.mean.silhouette`, `fitness.meanD.HS`, `sigma.meanD.HS`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
fitness <- sigma.dendo.fitness1(dendro)
fitness
```

`sigma.meanD.HS` *Dendrogram fitness function based on homogeneity and separation, checking the cluster size.*

Description

This fitness function is similar to function `fitness.meanD.HS`, but in this case, it adds a new objective (sigma), checking that the number of object-data in each cluster of the dendrogram is not very different.

Usage

```
sigma.meanD.HS(dendogram)
```

Arguments

`dendogram`	The dendrogram to compute the fitness.

Value

A numerical value representing the dendrogram fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`fitness.mean`, `fitness.mean.silhouette`, `fitness.meanD.HS`, `sigma.dendo.fitness1`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
fitness <- sigma.meanD.HS(dendro)
fitness
```

`silhouette.clust` *Fitness function measuring the clustering silhouette width.*

Description

This function computes the clustering fitness based on the silhouette width measure.

Usage

```
## S3 method for class 'clust'
silhouette(...)
```

Arguments

`...`	A clustering represented by a list of vectors.

Value

A value representing the clustering fitness.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

References

L. Kaufman, P. J. Rousseeuw, Finding Groups in Data. An Introduction to Clustering Analysis, John Wiley & Sons, Inc., Hoboken, New Jersey, 2005.

See Also

`clustering.fitness`, `sigma.clust.fitness`, `fitness.meanC.HS`, `sigma.clust.HS`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part=3/4)
clust <- dendro[[5]]
fitness <- silhouette.clust(clust)
fitness
```

`silhouette.mean` *Clustering silhouette measure.*

Description

This function measures silhouette with of a clustering, relating homogeneity and separation.

Usage

```
## S3 method for class 'mean'
silhouette(...)
```

Arguments

`...`	A clustering represented by a list of vectors.

Value

A numerical value representing the silhouette width of the clustering.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

References

Kaufman L, Rousseeuw PJ: Finding Groups in Data. An Introduction to Clustering Analysis. John Wiley & Sons, Inc., Hoboken, New Jersey 2005.

See Also

`H.ave`, `S.ave`.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
silhouette.mean(clust)
```

standard *Normalizing a dataset.*

Description

This function normalizes a dataset to mean 0 and variance 1.

Usage

```
standard(mat)
```

Arguments

mat	Dataset to normalize.

Value

The normalized dataset.

See Also

missvalue.knn.

Examples

```
library(cluster)
data(votes.repub)
newdata <- standard(votes.repub)
```

transf.tree *Transforms a dendrogram of class "twins" to a dendrogram as a list of lists of vectors.*

Description

This function transform a dendrogram of class "twins" to a list of lists of vectors. This last dendrogram structure is the one used by function "agnes.gas" and all functions of this package.

Usage

```
transf.tree(agn, part = 2)
```

Arguments

agn	A dendrogram of the structure of class "twins".
part	If we want to reduce the length of the returned dendrogram, "part" states the proportion of the dendrogram that is removed. For example, "part = 4/5" means that the forth-fifth of the levels in the dendrogram (from the first level) will be removed. See this parameter in function "agnes.gas" for more information.

Value

A dendrogram as a list of lists of vectors, internal structure used in this package.

Note

Function `init.variables` most be executed before using any other function of this package. It initializes the internal variables.

See Also

`transftovector`, `dendrogram.graph`.

Examples

```
library(cluster)
data(votes.repub)
agn1 <- agnes(votes.repub, metric = "manhattan", stand = TRUE)
dendro <- transf.tree(agn1, part = 3/4)
```

---

transftovector *Converts a clustering (in the form of list) to a vector structure.*

Description

This function transforms a clustering represented as a list of vectors to a clustering as vector. This last representation is compatible with the "twins" class of the R internal structure.

Usage

```
transftovector(clustering)
```

Arguments

clustering	A clustering in the form of a list of vectors.

Value

A clustering in the form of a R vector.

Note

Function init.variables most be executed before using any other function of this package. It initializes the internal variables.

See Also

transf.tree.

Examples

```
library(cluster)
data(votes.repub)
vr.matrix <- as.matrix(daisy(votes.repub, metric = "euclidean", stand = TRUE))
dendro <- agnes.gas(vr.matrix, 5, 10, part = 3/4)
clust <- dendro[[6]] #a clustering of 7 clusters, level 6.
vector.clust <- transftovector(clust)
vector.clust
```

Index