Package ‘clValid’

February 19, 2015

Version 0.6-6

Date 2014-03-24

Title Validation of Clustering Results

Author Guy Brock <guy.brock@louisville.edu>, Vasyl Pihur <vasyl.pihur@louisville.edu>, Susmita Datta <susmita.datta@louisville.edu>, and Somnath Datta <somnath.datta@louisville.edu>

Maintainer Guy Brock <guy.brock@louisville.edu>

Depends R (>= 2.0), cluster

Imports methods, class

Suggests Biobase, annotate, GO.db, moe430a.db, RankAggreg, kohonen, mclust

Description Statistical and biological validation of clustering results.

License LGPL-3

URL http://guybrock.gpbrock.net/research

Collate clValid-Classes.R clValid-Methods.R clValid-internal.R clValid-functions.R

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-25 05:52:03

R topics documented:

annotationListToMatrix *Change annotation list to matrix*

Description

Change biological functional annotation from list to TRUE / FALSE matrix

Usage

```
annotationListToMatrix(annotation, genenames)
```

Arguments

`annotation`	functional annotation of genes, as a list
`genenames`	vector of genenames

Details

Converts biological annotation from “list” to “matrix” format. In “list” format, each item in the list is a vector giving genes belonging to a particular biological class. In “matrix” format, each column is a logical vector indicating which genes belong to the biological class. Both `BHI` and `BSI` use the TRUE / FALSE matrix format for inputting predetermined biological functional classes.

Value

A logical matrix, where each element indicates whether the gene in row i belongs to the biological functional class of column j

Note

Special thanks to Rainer Machne, who initially suggested this change.

Author(s)

Guy Brock

See Also

`BHI`, `BSI`

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
fc <- tapply(rownames(express),mouse$FC[1:25], c)
fc <- fc[-match( c("EST","Unknown"), names(fc))]
fc <- annotationListToMatrix(fc, rownames(express))

## see package vignette for example use when reading from Excel file
```

BHI *Biological Homogeneity Index*

Description

Calculates the biological homogeneity index (BHI) for a given statistical clustering partition and biological annotation.

Usage

```
BHI(statClust, annotation, names = NULL, category = "all", dropEvidence=NULL)
```

Arguments

`statClust` An integer vector indicating the statistical cluster partitioning

`annotation` Either a character string naming the Bioconductor annotation package for mapping genes to GO categories, or a matrix where each column is a logical vector indicating which genes belong to the biological functional class. See details below.

`names` A vector of labels to associate with the 'genes', to be used in conjunction with the Bioconductor annotation package. Not needed if `annotation` is a list providing the functional classes.

`category` Indicates the GO categories to use for biological validation. Can be one of "BP", "MF", "CC", or "all".

`dropEvidence` Which GO evidence codes to omit. Either NULL or a character vector, see 'Details' below.

Details

The BHI measures how homogeneous the clusters are biologically. The measure checks whether genes placed in the same statistical cluster also belong to the same functional classes. The BHI is in the range [0,1], with larger values corresponding to more biologically homogeneous clusters. For details see the package vignette.

When inputting the biological annotation and functional classes directly, the BSI function expects the input in “matrix” format, where each column is a logical vector indicating which genes belong to the biological class. For details on how to input the biological annotation from an Excel file see readAnnotationFile and for converting from list to matrix format see annotationListToMatrix.

The dropEvidence argument indicates which GO evidence codes to omit. For example, "IEA" is a relatively weak association based only on electronic information, and users may wish to omit this evidence when determining the functional annotation classes.

Value

Returns the BHI measure as a numeric value.

Note

The main function for cluster validation is clValid, and users should call this function directly if possible.

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Datta, S. and Datta, S. (2006). Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics 7:397.

See Also

For a description of the function ’clValid’ see clValid.

For a description of the class ’clValid’ and all available methods see clValidObj or clValid-class.

For additional help on the other validation measures see connectivity, dunn, stability, and BSI.

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
## hierarchical clustering
Dist <- dist(express,method="euclidean")
clusterObj <- hclust(Dist, method="average")
nc <- 4 ## number of clusters
cluster <- cutree(clusterObj,nc)

## first way - functional classes predetermined
fc <- tapply(rownames(express),mouse$FC[1:25], c)
fc <- fc[-match( c("EST","Unknown"), names(fc))]
fc <- annotationListToMatrix(fc, rownames(express))
BHI(cluster, fc)

## second way - using Bioconductor
```

```
if(require("Biobase") && require("annotate") && require("GO.db") &&
require("moe430a.db")) {
  BHI(cluster, annotation="moe430a.db", names=rownames(express), category="all")
}
```

`BSI` *Biological Stability Index*

Description

Calculates the biological stability index (BSI) for a given statistical clustering partition and biological annotation.

Usage

```
BSI(statClust, statClustDel, annotation, names = NULL, category = "all",
goTermFreq = 0.05, dropEvidence=NULL)
```

Arguments

`statClust`	An integer vector indicating the statistical cluster partitioning
`statClustDel`	An integer vector indicating the statistical cluster partitioning based on one column removed
`annotation`	Either a character string naming the Bioconductor annotation package for mapping genes to GO categories, or a matrix where each column is a logical vector indicating which genes belong to the biological functional class. See details below.
`names`	An optional vector of names for the observations
`category`	Indicates the GO categories to use for biological validation. Can be one of "BP", "MF", "CC", or "all".
`goTermFreq`	What threshold frequency of GO terms to use for functional annotation.
`dropEvidence`	Which GO evidence codes to omit. Either NULL or a character vector, see 'Details' below.

Details

The BSI inspects the consistency of clustering for genes with similar biological functionality. Each sample is removed, and the cluster membership for genes with similar functional annotation is compared with the cluster membership using all available samples. The BSI is in the range [0,1], with larger values corresponding to more stable clusters of the functionally annotated genes. For details see the package vignette.

The `dropEvidence` argument indicates which GO evidence codes to omit. For example, "IEA" is a relatively weak association based only on electronic information, and users may wish to omit this evidence when determining the functional annotation classes.

When inputting the biological annotation and functional classes directly, the BSI function expects the input in "matrix" format, where each column is a logical vector indicating which genes belong to the biological class. For details on how to input the biological annotation from an Excel file see readAnnotationFile and for converting from list to matrix format see annotationListToMatrix.

NOTE: The BSI function only calculates these measures for one particular column removed. To get the overall scores, the user must average the measures corresponding to each removed column.

Value

Returns the BSI value corresponding to the particular column that was removed.

Note

The main function for cluster validation is clValid, and users should call this function directly if possible.

To get the overall BSI value, the BSI values corresponding to each removed column should be averaged (see the examples below).

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Datta, S. and Datta, S. (2006). Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics 7:397.

See Also

For a description of the function 'clValid' see clValid.

For a description of the class 'clValid' and all available methods see clValidObj or clValid-class.

For additional help on the other validation measures see connectivity, dunn, stability, and BHI.

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
## hierarchical clustering
Dist <- dist(express,method="euclidean")
clusterObj <- hclust(Dist, method="average")
nc <- 4 ## number of clusters
cluster <- cutree(clusterObj,nc)

## first way - functional classes predetermined
fc <- tapply(rownames(express),mouse$FC[1:25], c)
fc <- fc[-match( c("EST","Unknown"), names(fc))]
fc <- annotationListToMatrix(fc, rownames(express))
```

```
bsi <- numeric(ncol(express))
## Need loop over all removed samples
for (del in 1:ncol(express)) {
  matDel <- express[,-del]
  DistDel <- dist(matDel,method="euclidean")
  clusterObjDel <- hclust(DistDel, method="average")
  clusterDel <- cutree(clusterObjDel,nc)
  bsi[del] <- BSI(cluster, clusterDel, fc)
}
mean(bsi)

## second way - using Bioconductor
if(require("Biobase") && require("annotate") && require("GO.db") &&
   require("moe430a.db")) {
  bsi <- numeric(ncol(express))
  for (del in 1:ncol(express)) {
    matDel <- express[,-del]
    DistDel <- dist(matDel,method="euclidean")
    clusterObjDel <- hclust(DistDel, method="average")
    clusterDel <- cutree(clusterObjDel,nc)
    bsi[del] <- BSI(cluster, clusterDel, annotation="moe430a.db",
                    names=rownames(express), category="all")
  }
  mean(bsi)
}
```

clValid *Validate Cluster Results*

Description

`clValid` reports validation measures for clustering results. The function returns an object of class "`clValid`", which contains the clustering results in addition to the validation measures. The validation measures fall into three general categories: "internal", "stability", and "biological".

Usage

```
clValid(obj, nClust, clMethods = "hierarchical", validation =
"stability", maxitems = 600, metric = "euclidean", method = "average",
neighbSize = 10, annotation = NULL, GOcategory = "all",
goTermFreq=0.05, dropEvidence=NULL, verbose=FALSE, ...)
```

Arguments

`obj` Either a numeric matrix, a data frame, or an `ExpressionSet` object. Data frames must contain all numeric columns. In all cases, the rows are the items to be clustered (e.g., genes), and the columns are the samples.

`nClust` A numeric vector giving the numbers of clusters to be evaluated. e.g., 4:6 would evaluate the number of clusters ranging from 4 to 6.

`clMethods` A character vector giving the clustering methods. Available options are "hierarchical", "kmeans", "diana", "fanny", "som", "model", "sota", "pam", "clara", and "agnes", with multiple choices allowed.

`validation` A character vector giving the type of validation measures to use. Available options are "internal", "stability", and "biological", with multiple choices allowed.

`maxitems` The maximum number of items (rows in matrix) which can be clustered.

`metric` The metric used to determine the distance matrix. Possible choices are "euclidean", "correlation", and "manhattan".

`method` For hierarchical clustering (`hclust` and `agnes`), the agglomeration method used. Available choices are "ward", "single", "complete", and "average".

`neighbSize` For internal validation, an integer giving the neighborhood size used for the connectivity measure.

`annotation` For biological validation, either a character string naming the Bioconductor annotation package for mapping genes to GO categories, or a list with the names of the functional classes and the observations belonging to each class.

`GOcategory` For biological validation, gives which GO categories to use for biological validation. Can be one of "BP", "MF", "CC", or "all".

`goTermFreq` For the BSI, what threshold frequency of GO terms to use for functional annotation.

`dropEvidence` Which GO evidence codes to omit. Either NULL or a character vector, see ’Details’ below.

`verbose` Logical - if TRUE will produce detailed output on the progress of cluster validation.

`...` Additional arguments to pass to the clustering functions.

Details

This function calculates validation measures for a given set of clustering algorithms and number of clusters. A variety of clustering algorithms are available, including hierarchical, self-organizing maps (SOM), K-means, self-organizing tree algorithm (SOTA), and model-based. The available validation measures fall into the three general categories of "internal", "stability", and "biological". A brief description of each measure is given below, for further details refer to the package vignette and the references.

Internal measures: The internal measures include the connectivity, and Silhouette Width, and Dunn Index. The connectivity indicates the degree of connectedness of the clusters, as determined by the k-nearest neighbors. The `neighbSize` argument specifies the number of neighbors to use. The connectivity has a value between 0 and infinity and should be minimized. Both the Silhouette Width and the Dunn Index combine measures of compactness and separation of the clusters. The Silhouette Width is the average of each observation’s Silhouette value. The Silhouette value measures the degree of confidence in a particular clustering assignment and lies in the interval [-1,1], with well-clustered observations having values near 1 and poorly clustered observations having values near -1. See the `silhouette` function in package **cluster** for more details. The Dunn Index is the ratio between the smallest distance between observations not in the same cluster to the largest intra-cluster distance. It has a value between 0 and infinity and should be maximized.

Stability measures: The stability measures are a special version of internal measures which evaluate the stability of a clustering result by comparing it with the clusters obtained by removing one column at a time. These measures include the average proportion of non-overlap (APN), the average distance (AD), the average distance between means (ADM), and the figure of merit (FOM). The APN, AD, and ADM are all based on the cross-classification table of the original clustering with the clustering based on the removal of one column. The APN measures the average proportion of observations not placed in the same cluster under both cases, while the AD measures the average distance between observations placed in the same cluster under both cases and the ADM measures the average distance between cluster centers for observations placed in the same cluster under both cases. The FOM measures the average intra-cluster variance of the deleted column, where the clustering is based on the remaining (undeleted) columns. In all cases the average is taken over all the deleted columns, and all measures should be minimized.

Biological measures: There are two biological validation measures, the biological homogeneity index (BHI) and biological stability index (BSI). The observations are typically taken to represent a 'gene' (e.g., ORF, SAGE tag, affy ID). The BHI measures the average proportion of gene pairs that are clustered together which have matching biological functional classes. The BSI is similar to the other stability measures, but inspects the consistency of clustering for genes with similar biological functionality. Each sample is removed one at a time, and the cluster membership for genes with similar functional annotation is compared with the cluster membership using all available samples.

For biological validation, the user has two options. The first option is to explicity specify the functional clustering of the genes via either a named list or logical matrix. In "list" format, each item in the list is a vector giving genes belonging to a particular biological class. In "matrix" format, each column is a logical vector indicating which genes belong to the biological class. `clValid` will convert the biological annotation to matrix format internally if initially given in list format.

The second option is to specify the appropriate annotation package from Bioconductor (`http://www.bioconductor.org`) and GO terms to determine the functional classes of the genes. To use the second option requires the **Biobase**, **annotate**, and **GO** packages from Bioconductor, in addition to the annotation package for the particular data type. If the annotation package cannot be loaded, `clValid` will attempt to automatically download the package from `www.bioconductor.org` (using the '`biocLite.R`' installation script).

The `GOcategory` options are "MF", "BP", "CC", or "all", corresponding to molecular function, biological process, cellular component, and all of the ontologies.

The `dropEvidence` argument indicates which GO evidence codes to omit. For example, "IEA" is a relatively weak association based only on electronic information, and users may wish to omit this evidence when determining the functional annotation classes.

Value

`clValid` returns an object of class "`clValid`". See the help file for the class description.

Note

The only package which is automatically attached is **cluster**. To use the clustering methods `som` and `Mclust` you will need to load the packages **kohonen** and **mclust**, respectively.

Unless the the list of genes corresponding to functional classes is prespecified, to perform biological clustering validation will require the **Biobase**, **annotate** and **GO** packages from Bioconductor, and in addition the annotation package for your particular data type. Please see http://www.bioconductor.org for installation instructions.

Further details of the validation measures and instructions in use can be found in the package vignette.

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008) clValid: An R Package for Cluster Validation Journal of Statistical Software 25(4) http://www.jstatsoft.org/v25/i04

Datta, S. and Datta, S. (2003) Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics 19(4): 459-466

Datta, S. and Datta, S. (2006) Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics 7:397 http://www.biomedcentral.com/1471-2105/7/397

Handl, J., Knowles, K., and Kell, D. (2005) Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15): 3201-3212

See Also

For a description of the class 'clValid' and all available methods see clValidObj or clValid-class.

For help on the clustering methods see hclust and kmeans in package **stats**, agnes, clara, diana, fanny, and pam in package **cluster**, som in package **kohonen**, Mclust in package **mclust**, and sota (in this package).

For additional help on the validation measures see connectivity, dunn, stability, BHI, and BSI.

Examples

```
data(mouse)

## internal validation
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
intern <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
                  validation="internal")

## view results
summary(intern)
optimalScores(intern)
plot(intern)

## stability measures
```

```
stab <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
                validation="stability")
optimalScores(stab)
plot(stab)

## biological measures
## first way - functional classes predetermined
fc <- tapply(rownames(express),mouse$FC[1:25], c)
fc <- fc[-match( c("EST","Unknown"), names(fc))]
bio <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
               validation="biological", annotation=fc)
optimalScores(bio)
plot(bio)

## second way - using Bioconductor
if(require("Biobase") && require("annotate") && require("GO.db") && require("moe430a.db")) {
  bio2 <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
                  validation="biological",annotation="moe430a.db",GOcategory="all")
  optimalScores(bio2)
  plot(bio2)
}
```

clValid-class *Class "clValid"*

Description

The class "clValid" contains the clustering results and validation measures from the accompanying call to the function clValid.

Objects from the Class

Objects can be created using the function clValid.

Slots

clusterObjs: Object of class "list". A list containing the results from the clustering methods.

measures: Object of class "array". A 3-dimensional array which contains the validation measures for the clustering results. The first dimension indicates the validation measures, the second the number of clusters, and the third the clustering methods.

measNames: Object of class "character". The names of the validation measures.

clMethods: Object of class "character". A character vector giving the clustering methods.

labels: Object of class "character". A character vector giving the item (gene) labels.

nClust: Object of class "numeric". A numeric vector giving the numbers of clusters which were evaluated.

validation: Object of class "character". A character vector giving the type of validation measures used, consisting of some combination of "internal", "stability", or "biological".

`metric`**:** Object of class `"character"`. The metric used to determine the distance matrix.

`method`**:** Object of class `"character"`. For hierarchical clustering, the agglomeration method used.

`neighbSize`**:** Object of class `"numeric"`. For internal validation, the neighborhood size used for the connectivity measure.

`annotation`**:** Object of class `"character or array or list"`. Either a character string naming the Bioconductor annotation package for mapping genes to GO categories, or a list with the names of the functional classes and the observations belonging to each class.

`GOcategory`**:** Object of class `"character"`. For biological validation, gives which GO categories to use for biological validation. Can be one of "BP", "MF", "CC", or "all"

`goTermFreq`**:** Object of class `"numeric"`. For the BSI, what threshold frequency of GO terms to use for functional annotation.

`call`**:** Object of class `"call"`. Gives the call to `clValid` used to create the `clValid` object.

Methods

clusterMethods `signature(object = "clValid")`: Returns the names of the clustering methods.

clusters `signature(object = "clValid")`: Returns the results from the clustering methods.

Additional arguments:

`method = clMethods(object)` The clustering method(s) to extract.

measNames `signature(object = "clValid")`: Returns the names of the validation measures.

measures `signature(object = "clValid")`: Returns the validation measures.

Additional arguments:

`measures = measNames(object)` The validation measure(s) to extract.

nClusters `signature(object = "clValid")`: Returns the numbers of clusters evaluated.

optimalScores `signature(object = "clValid")`: Returns the optimal value for each validation measure, along with the corresponding clustering method and number of clusters.

Additional arguments:

`measures = measNames(object)` The validation measure(s) to extract.

plot `signature(x = "clValid", y = "missing")`: Plots the validation measures.

Additional arguments:

`measures=measNames(x)` The validation measures to plot.

`legend=TRUE` If TRUE provides a legend.

`legendLoc="topright"` The location of the legend.

`main=NULL` Title of graph.

`pch=NULL` Plotting characters to use.

`type="b"` Type of plot.

`ask=prod(par("mfcol")) < length(measures) && dev.interactive()` Logical. If `TRUE` the user is prompted before each plot.

print `signature(x = "clValid")`: Print method for class `clValid`.

show `signature(object = "clValid")`: Same as print.

summary `signature(object = "clValid")`: Summary method for class `clValid`.

Additional arguments:

`digits = max(3,getOption("digits")-3)` The number of significant digits to use.

Note

See the vignette for an illustration of the class.

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008) clValid: An R Package for Cluster Validation Journal of Statistical Software 25(4) http://www.jstatsoft.org/v25/i04

Datta, S. and Datta, S. (2003) Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics 19(4): 459-466.

Datta, S. and Datta, S. (2006) Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. BMC Bioinformatics 7:397. http://www.biomedcentral.com/1471-2105/7/397

Handl, J., Knowles, K., and Kell, D. (2005) Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15): 3201-3212.

See Also

For a description of the function 'clValid' see `clValid`.

For help on the clustering methods see `hclust` and `kmeans` in package **stats**, `kmeans` in package **stats**, `agnes`, `clara`, `diana`, `fanny`, and `pam` in package **cluster**, `som` in package **kohonen**, `Mclust` in package **mclust**, and `sota`.

For additional help on the validation measures see `connectivity`, `dunn`, `stability`, `BHI`, and `BSI`.

Examples

```
## to delete
library(clValid)


data(mouse)

## internal validation
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
intern <- clValid(express, 2:6, clMethods=c("hierarchical","kmeans","pam"),
                  validation="internal")
slotNames(intern)

## view results
intern
summary(intern)
optimalScores(intern)
plot(intern)
```

```
## Extract objects from slots
measures(intern)
hierClust <- clusters(intern,"hierarchical")
plot(hierClust)
measNames(intern)
nClusters(intern)
```

connectivity *Connectivity Measure*

Description

Calculates the connectivity validation measure for a given cluster partitioning.

Usage

```
connectivity(distance = NULL, clusters, Data = NULL, neighbSize = 10,
             method = "euclidean")
```

Arguments

`distance`	The distance matrix (as a matrix object) of the clustered observations. Required if `Data` is NULL.
`clusters`	An integer vector indicating the cluster partitioning
`Data`	The data matrix of the clustered observations. Required if `distance` is NULL.
`neighbSize`	The size of the neighborhood
`method`	The metric used to determine the distance matrix. Not used if `distance` is provided.

Details

The connectivity indicates the degree of connectedness of the clusters, as determined by the k-nearest neighbors. The `neighbSize` argument specifies the number of neighbors to use. The connectivity has a value between 0 and infinity and should be minimized. For details see the package vignette.

Value

Returns the connectivity measure as a numeric value.

Note

The main function for cluster validation is `clValid`, and users should call this function directly if possible.

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Handl, J., Knowles, K., and Kell, D. (2005). Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15): 3201-3212.

See Also

For a description of the function 'clValid' see `clValid`.

For a description of the class 'clValid' and all available methods see `clValidObj` or `clValid-class`.

For additional help on the other validation measures see `dunn`, `stability`, `BHI`, and `BSI`.

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
## hierarchical clustering
Dist <- dist(express,method="euclidean")
clusterObj <- hclust(Dist, method="average")
nc <- 2 ## number of clusters
cluster <- cutree(clusterObj,nc)
connectivity(Dist, cluster)
```

dunn *Dunn Index*

Description

Calculates the Dunn Index for a given clustering partition.

Usage

```
dunn(distance = NULL, clusters, Data = NULL, method = "euclidean")
```

Arguments

`distance` The distance matrix (as a matrix object) of the clustered observations. Required if `Data` is NULL.

`clusters` An integer vector indicating the cluster partitioning

`Data` The data matrix of the clustered observations. Required if `distance` is NULL.

`method` The metric used to determine the distance matrix. Not used if `distance` is provided.

Details

The Dunn Index is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. The Dunn Index has a value between zero and infinity, and should be maximized. For details see the package vignette.

Value

Returns the Dunn Index as a numeric value.

Note

The main function for cluster validation is `clValid`, and users should call this function directly if possible.

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Dunn, J.C. (1974). Well separated clusters and fuzzy partitions. Journal on Cybernetics, 4:95-104.

Handl, J., Knowles, K., and Kell, D. (2005). Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15): 3201-3212.

See Also

For a description of the function ’clValid’ see `clValid`.

For a description of the class ’clValid’ and all available methods see `clValidObj` or `clValid-class`.

For additional help on the other validation measures see `dunn`, `stability`, `BHI`, and `BSI`.

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
## hierarchical clustering
Dist <- dist(express,method="euclidean")
clusterObj <- hclust(Dist, method="average")
nc <- 2 ## number of clusters
cluster <- cutree(clusterObj,nc)
dunn(Dist, cluster)
```

getRanksWeights *Extract ranks and weights from clValid object*

Description

Creates matrix of ranks and weights from `clValid` object, to use as input for rank aggregation using `RankAggreg` in package **RankAggreg**

Usage

```
getRanksWeights(clVObj, measures = measNames(clVObj), nClust =
                nClusters(clVObj), clAlgs = clusterMethods(clVObj))
```

Arguments

`clVObj`	a clValid object
`measures`	the cluster validation measures to use for rank aggregation
`nClust`	the number of clusters to evaluate
`clAlgs`	the clustering algorithms to evaluate

Details

This function extracts cluster validation measures from a `clValid` object, and creates a matrix of rankings where each row contains a list of clustering algorithms which are ranked according to the validation measure for that row. The function also returns the cluster validation measures as a matrix of weights, for use with weighted rank aggregation in the function `RankAggreg`. Any combination of validation measures, numbers of clusters, and clustering algorithms can be selected by the user. Number of clusters and clustering algorithms are appended into a single name.

Value

A list with components

`ranks`	Matrix with rankings for each validation measure in each row
`weights`	Matrix of weights, corresponding to the cluster validation measures, which are used for weighted rank aggregation

Author(s)

Guy Brock

References

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008) clValid: An R Package for Cluster Validation Journal of Statistical Software 25(4) `http://www.jstatsoft.org/v25/i04`

Pihur, V., Datta, S. and Datta, S. (2009) RankAggreg, an R package for weighted rank aggregation BMC Bioinformatics 10:62 `http://www.biomedcentral.com/1471-2105/10/62`

See Also

clValid, RankAggreg

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
clv <- clValid(express, 4:6, clMethods=c("hierarchical","kmeans","pam"),
                  validation=c("internal","stability"))
res <- getRanksWeights(clv)
if(require("RankAggreg")) {
  CEWS <- RankAggreg(x=res$ranks, k=5, weights=res$weights, seed=123, verbose=FALSE)
  CEWS
}
```

mouse *Mouse Mesenchymal Cells*

Description

Data from an Affymetrix microarray experiment (moe430a) comparing comparing gene expression of mesenchymal cells from two distinct lineages, neural crest and mesoderm derived. The dataset consists of 147 genes and ESTs which were determined to be significantly differentially expressed between the two cell lineages, with at least a 1.5 fold increase or decrease in expression. There are three samples for each of the neural crest and mesoderm derived cells.

Usage

```
data(mouse)
```

Format

A data frame with 147 observations on the following 8 variables.

`ID` The Affymetric GeneChip ID, from the moe430a chip

`M1` Mesoderm derived cell sample

`M2` Mesoderm derived cell sample

`M3` Mesoderm derived cell sample

`NC1` Nueral crest derived cell sample

`NC2` Nueral crest derived cell sample

`NC3` Nueral crest derived cell sample

`FC` The functional class of each gene/EST

Source

V. Bhattacherjee, P. Mukhopadhyay, S. Singh, C. Johnson, J. T. Philipose, C. P. Warner, R. M. Greene, and M. M. Pisano. Neural crest and mesoderm lineagedependent gene expression in orofacial development. Differentiation, 2007.

Examples

```
data(mouse)

## table of fuctional classifications
table(mouse$FC)

## hierarchical clustering of expression values
express <- mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID
hc <- hclust(dist(express))
plot(hc)
```

---

plot.sota *Plot Function for a SOTA Object*

Description

'plot.sota' is used to obtain a visual representation of profiles within each individual cluster. Corresponding cluster average profiles are also available. By default, plots for all clusters are displayed side by side.

Usage

```
## S3 method for class 'sota'
plot(x, cl = 0, ...)
```

Arguments

x	SOTA object, an object returned by function sota.
cl	cl specifies which cluster is to be plotted by setting it to the cluster ID. By default, cl is equal to 0 and the function plots all clusters side by side.
...	Additional arguments to pass to plot.

Author(s)

Vasyl Pihur, Guy Brock, Susmita Datta, Somnath Datta

References

Herrero, J., Valencia, A, and Dopazo, J. (2005). A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics, 17, 126-136.

See Also

sota, print.sota

Examples

```
data(mouse)
express <- mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID

sotaCl <- sota(as.matrix(express), 4)
names(sotaCl)
sotaCl
plot(sotaCl)
plot(sotaCl, cl=2)
```

print.sota *Print Function for a SOTA Object*

Description

A default print method for a SOTA object.

Usage

```
## S3 method for class 'sota'
print(x, ...)
```

Arguments

`x` a SOTA object as returned by the `sota` function

`...` Additional arguments to pass to `print`

Value

The `print` function does not return anything. It simply displays in the console window general information about the paritioning (cluster ID, Size, and Diversity Score), as well as cluster centroids (average profiles within each cluster) and the distance that has been used.

Author(s)

Vasyl Pihur, Guy Brock, Susmita Datta, Somnath Datta

References

Herrero, J., Valencia, A, and Dopazo, J. (2005). A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics, 17, 126-136.

See Also

sota, print.sota

Examples

```
data(mouse)
express <- mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID

sotaCl <- sota(as.matrix(express), 4)
names(sotaCl)
sotaCl
plot(sotaCl)
plot(sotaCl, cl=2)
```

---

readAnnotationFile *Read in biological annotation files from external file*

Description

This function reads in a biological annotation file detailing the functional classes for a given set of genes from a comma separated file.

Usage

```
readAnnotationFile(filename)
```

Arguments

filename	The name of the external file. The values must be comma separated.

Details

The required format is comma separated, with the first column indicating the biological functional category, and the remaining columns containing the gene identifiers for those genes belonging to that category.

Value

Returns a list where each item in the list is the set of genes belonging to a particular functional class. This can be converted to a TRUE/FALSE matrix using the annotationListToMatrix function, though it is not necessary to do this prior to using clValid.

Author(s)

Guy Brock, Vasyl Pihur

See Also

annotationListToMatrix, clValid

Examples

```
## For example use, see the package vignette
```

---

sota *Self-organizing Tree Algorithm (SOTA)*

Description

Computes a Self-organizing Tree Algorithm (SOTA) clustering of a dataset returning a SOTA object.

Usage

```
sota(data, maxCycles, maxEpochs = 1000, distance = "euclidean", wcell = 0.01,
     pcell = 0.005, scell = 0.001, delta = 1e-04, neighb.level = 0,
     maxDiversity = 0.9, unrest.growth = TRUE, ...)
```

Arguments

`data` data matrix or data frame. Cannot have a profile ID as the first column.

`maxCycles` integer value representing the maximum number of iterations allowed. The resulting number of clusters returned by `sota` is maxCycles+1 unless `unrest.growth` is set to FALSE and the `maxDiversity` criteria is satisfied prior to reaching the maximum number of iterations.

`maxEpochs` integer value indicating the maximum number of training epochs allowed per cycle. By default, `maxEpochs` is set to 1000.

`distance` character string used to represent the metric to be used for calculating dissimilarities between profiles. 'euclidean' is the default, with 'correlation' being another option.

`wcell` value specifying the winning cell migration weight. The default is 0.01.

`pcell` value specifying the parent cell migration weight. The default is 0.005.

`scell` value specifying the sister cell migration weight. The default is 0.001.

`delta` value specifying the minimum epoch error improvement. This value is used as a threshold for signaling the start of a new cycle. It is set to 1e-04 by default.

`neighb.level` integer value used to indicate which cells are candidates to accept new profiles. This number specifies the number of levels up the tree the algorithm moves in the search of candidate cells for the redistribution of profiles. The default is 0.

`maxDiversity` value representing a maximum variability allowed within a cluster. 0.9 is the default value.

`unrest.growth` logical flag: if TRUE then the algorithm will run `maxCycles` iterations regardless of whether the `maxDiversity` criteria is satisfied or not and `maxCycles+1` clusters will be produced; if FALSE then the algorithm can potentially stop before reaching the `maxCycles` based on the current state of cluster diversities. A smaller than usual number of clusters will be obtained. The default value is TRUE.

`...` Any other arguments.

Details

The Self-Organizing Tree Algorithm (SOTA) is an unsupervised neural network with a binary tree topology. It combines the advantages of both hierarchical clustering and Self-Organizing Maps (SOM). The algorithm picks a node with the largest Diversity and splits it into two nodes, called Cells. This process can be stopped at any level, assuring a fixed number of hard clusters. This behavior is achieved with setting the `unrest.growth` parameter to TRUE. Growth of the tree can be stopped based on other criteria, like the allowed maximum Diversity within the cluster and so on.

Further details regarding the inner workings of the algorithm can be found in the paper listed in the Reference section.

Value

`data` data matrix used for clustering

`c.tree` complete tree in a matrix format. Node ID, its Ancestor, and whether it's a terminal node (cell) are listed in the first three columns. Node profiles are shown in the remaining columns.

`tree` incomplete tree in a matrix format listing only the terminal nodes (cells). Node ID, its Ancestor, and 1's for a cell indicator are listed in the first three columns. Node profiles are shown in the remaining columns.

`clust` integer vector whose length is equal to the number of profiles in a data matrix indicating the cluster assingments for each profile in the original order.

`totals` integer vector specifying the cluster sizes.

`dist` character string indicating a distance function used in the clustering process.

`diversity` vector specifying final cluster diverisities.

Author(s)

Vasyl Pihur, Guy Brock, Susmita Datta, Somnath Datta

References

Herrero, J., Valencia, A, and Dopazo, J. (2005). A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics, 17, 126-136.

See Also

`plot.sota`, `print.sota`

Examples

```
data(mouse)
express <- mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID

sotaCl <- sota(as.matrix(express), 4)
names(sotaCl)
sotaCl
plot(sotaCl)
plot(sotaCl, cl=2)
```

`stability` *Stability Measures*

Description

Calculates the stability measures the average proportion of non-overlap (APN), the average distance (AD), the average distance between means (ADM), and the figure of merit (FOM).

Usage

```
stability(mat, Dist=NULL, del, cluster, clusterDel, method="euclidean")
```

Arguments

`mat` The data matrix of the clustered observations

`Dist` The distance matrix (as a matrix or dist object) of the clustered observations. If NULL then `method` is used with `mat` to determine the distance matrix.

`del` An integer indicating which column was removed

`cluster` An integer vector indicating the cluster partitioning based on all the data

`clusterDel` An integer vector indicating the cluster partitioning based on the data with column `del` removed.

`method` The metric used to determine the distance matrix. Not used if `distance` is provided.

Details

The stability measures evaluate the stability of a clustering result by comparing it with the clusters obtained by removing one column at a time. These measures include the average proportion of non-overlap (APN), the average distance (AD), the average distance between means (ADM), and the figure of merit (FOM). The APN, AD, and ADM are all based on the cross-classification table of the original clustering with the clustering based on the removal of one column. The APN measures the average proportion of observations not placed in the same cluster under both cases, while the AD measures the average distance between observations placed in the same cluster under both cases and the ADM measures the average distance between cluster centers for observations placed in the same cluster under both cases. The FOM measures the average intra-cluster variance of the deleted column, where the clustering is based on the remaining (undeleted) columns. In all cases the average is taken over all the deleted columns, and all measures should be minimized. For details see the package vignette.

NOTE: The `stability` function only calculates these measures for the particular column specified by `del` removed. To get the overall scores, the user must average the measures corresponding to each removed column.

Value

Returns a numeric vector with the APN, AD, ADM, and FOM measures corresponding to the particular column that was removed.

Note

The main function for cluster validation is `clValid`, and users should call this function directly if possible.

To get the overall values, the stability measures corresponding to each removed column should be averaged (see the examples below).

Author(s)

Guy Brock, Vasyl Pihur, Susmita Datta, Somnath Datta

References

Datta, S. and Datta, S. (2003). Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics 19(4): 459-466.

See Also

For a description of the function 'clValid' see `clValid`.

For a description of the class 'clValid' and all available methods see `clValidObj` or `clValid-class`.

For additional help on the other validation measures see `connectivity`, `dunn`, `BSI`, and `BHI`.

Examples

```
data(mouse)
express <- mouse[1:25,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) <- mouse$ID[1:25]
## hierarchical clustering
Dist <- dist(express,method="euclidean")
clusterObj <- hclust(Dist, method="average")
nc <- 4 ## number of clusters
cluster <- cutree(clusterObj,nc)

stab <- matrix(0,nrow=ncol(express),ncol=4)
colnames(stab) <- c("APN","AD","ADM","FOM")
```

```
## Need loop over all removed samples
for (del in 1:ncol(express)) {
  matDel <- express[,-del]
  DistDel <- dist(matDel,method="euclidean")
  clusterObjDel <- hclust(DistDel, method="average")
  clusterDel <- cutree(clusterObjDel,nc)
  stab[del,] <- stability(express, Dist, del, cluster, clusterDel)
}
colMeans(stab)
```

Index