Package 'mosclust'

May 23, 2025

Type Package

Version 1.0.2

Title Model Order Selection for Clustering

Maintainer Jessica Gliozzo <jessica.gliozzo@unimi.it>

Imports stats, graphics, cluster, clusterv

Description Stability based methods for model order selection in clustering problems (Valentini, G (2007), <doi:10.1093/bioinformatics/btl600>).
Using multiple perturbations of the data the stability of clustering solutions is assessed. Different perturbations may be used: resampling techniques, random projections and noise injection. Stability measures
for the estimate of clustering solutions and statistical tests to assess their significance are provided.

License GPL (>= 2)

URL https://valentini.di.unimi.it/SW/mosclust/

BugReports https://github.com/AnacletoLAB/mosclust/issues

Encoding UTF-8

NeedsCompilation no

Author Giorgio Valentini [aut],
Jessica Gliozzo [cre]

Contents

`mosclust-package` *Model order selection for clustering*

Description

The *mosclust* R package (that stands for **m**odel **o**rder **s**election for **clust**ering problems) implements a set of functions to discover significant structures in bio-molecular data. Using multiple perturbations of the data the stability of clustering solutions is assessed. Different perturbations may be used: resampling techniques, random projections and noise injection. Stability measures for the estimate of clustering solutions and statistical tests to assess their significance are provided.

Details

Package:	mosclust
Type:	Package
Version:	1.0.2
Date:	2006-09-08
License:	GPL (>= 2)

Recently, several methods based on the concept of stability have been proposed to estimate the "optimal" number of clusters in complex bio-molecular data. In this conceptual framework multiple clusterings are obtained by introducing perturbations into the original data, and a clustering is considered reliable if it is approximately maintained across multiple perturbations.

Several perturbation techniques have been proposed, ranging form bootstrap techniques, to random projections to lower dimensional subspaces to noise injection procedures. All these perturbation techniques are implemented in *mosclust*.

The library implements indices of stability/reliability of the clusterings based on the distribution of similarity measures between multiple instances of clusterings performed on multiple instances of data obtained through a given random perturbation of the original data.

These indices provides a "score" that can be used to compare the reliability of different clusterings. Moreover statistical tests based on χ^2 and on the classical Bernstein inequality are implemented in order to assess the statistical significance of the discovered clustering solutions. By this approach we could also find multiple structures simultaneously present in the data. For instance, it is possible that data exhibit a hierarchial structure, with subclusters inside other clusters, and using the indices and the statistical tests implemented in *mosclust* we may detect them at a given significance level.

Summarizing, this package may be used for:

- Assessment of the reliability of a given clustering solution
- Clustering model order selection: what about the "natural" number of clusters inside the data?
- Assessment of the statistical significance of a given clustering solution
- Discovery of multiple structures underlying the data: are there multiple reliable clustering solutions at a given significance level?

The statistical tests implemented in the package have been designed with the theoretical and methodological contribution of *Alberto Bertoni* (DSI, Università degli Studi di Milano).

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

Bittner, M. et al., Molecular classification of malignant melanoma by gene expression profiling, Nature, 406:536–540, 2000.

Monti, S., Tamayo P., Mesirov J. and Golub T., Consensus Clustering: A Resampling-based Method for Class Discovery and Visualization of Gene Expression Microarray Data, Machine Learning, 52:91–118, 2003.

Dudoit S. and Fridlyand J., A prediction-based resampling method for estimating the number of clusters in a dataset, Genome Biology, 3(7): 1-21, 2002.

Kerr M.K. and Curchill G.A.,Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments, PNAS, 98:8961–8965, 2001.

McShane, L.M., Radmacher, D., Freidlin, B., Yu, R., Li, M.C. and Simon, R., Method for assessing reproducibility of clustering patterns observed in analyses of microarray data, Bioinformatics, 11(8), pp. 1462-1469, 2002.

Ben-Hur, A. Ellisseeff, A. and Guyon, I., A stability based method for discovering structure in clustered data, In: "Pacific Symposium on Biocomputing", Altman, R.B. et al (eds.), pp, 6-17, 2002.

Smolkin M. and Gosh D., Cluster stability scores for microarray data in cancer studies, BMC Bioinformatics, 36(4), 2003.

W. Hoeffding, Probability inequalities for sums of independent random variables, J. Amer. Statist. Assoc. vol.58 pp. 13-30, 1963.

A.Bertoni, G. Valentini, Randomized maps for assessing the reliability of patients clusters in DNA microarray data analyses, Artificial Intelligence in Medicine 37(2):85-109 2006

A.Bertoni, G. Valentini, Model order selection for clustered bio-molecular data, In: Probabilistic Modeling and Machine Learning in Structural and Systems Biology, J. Rousu, S. Kaski and E. Ukkonen (Eds.), Tuusula, Finland, 17-18 June, 2006

A.Bertoni, G. Valentini, Discovering significant structures in clustered data through Bernstein inequality, CISI '06, Conferenza Italiana Sistemi Intelligenti, Ancona, Italia, 2006.

G. Valentini, Clusterv: a tool for assessing the reliability of clusters discovered in DNA microarray data, Bioinformatics, 22(3):369-370, 2006.

See Also

clusterv

`Bernstein.compute.pvalues`

Function to compute the stability indices and the p-values associated to a set of clusterings according to Bernstein inequality.

Description

For a given similarity matrix a list of stability indices, sorted by descending order, from the most significant clustering to the least significant is given, and the corresponding p-values, computed according to a Bernstein inequality based test are provided.

Usage

```
Bernstein.compute.pvalues(sim.matrix)

Bernstein.ind.compute.pvalues(sim.matrix)
```

Arguments

`sim.matrix`	a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings. Rows correspond to the different clusterings; columns to the n repeated clusterings for each number of clusters. Row 1 corresponds to a 2-clustering, row 2 to a 3-clustering, ... row m to a m+1 clustering.

Details

The stability index for a given clustering is computed as the mean of the similarity indices between pairs of k-clusterings obtained from the perturbed data. The similarity matrix given as input can be obtained from the functions do.similarity.resampling, do.similarity.projection, do.similarity.noise. A list of p-values, sorted by descending order, from the most significant clustering to the least significant is given according to a test based on Bernstein inequality. The test is based on the distribution of the similarity measures between pairs of clustering performed on perturbed data, but differently from the chi-square based test (see `Chi.square.compute.pvalues`), no assumptions are made about the "a priori" distribution of the similarity measures. The function `Bernstein.ind.compute.pvalues` assumes also that the the random variables represented by the means of the similarities between pairs of clusterings are independent, while, on the contrary, the function `Bernstein.compute.pvalues` no assumptions are made. Low p-value mean that there is a significant difference between the top sorted and the given clustering. Please, see the papers cited in the reference section for more technical details.

Value

a list with 4 components:

`ordered.clusterings` a vector with the number of clusters ordered from the most significant to the least significant

`p.value` a vector with the corresponding p-values computed according to Bernstein inequality and Bonferroni correction in descending order (their values correspond to the clusterings of the vector ordered.clusterings)

`means` vector with the mean similarity for each clustering

`variance` vector with the variance of the similarity for each clustering

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

W. Hoeffding, Probability inequalities for sums of independent random variables, J. Amer. Statist. Assoc. vol.58 pp. 13-30, 1963.

A.Bertoni, G. Valentini, Discovering significant structures in clustered data through Bernstein inequality, CISI '06, Conferenza Italiana Sistemi Intelligenti, Ancona, Italia, 2006.

See Also

`Chi.square.compute.pvalues`, `Hypothesis.testing`,

`do.similarity.resampling`, `do.similarity.projection`, `do.similarity.noise`

Examples

```
library("clusterv")
# Computation of the p-values according to Bernstein inequality using
# resampling techniques and a hierarchical clustering algorithm
M <- generate.sample.h2 (n=20, l=10, Delta.h=4, Delta.v=2, sd=0.15);
S.HC <- do.similarity.resampling (M, c=15, nsub=20, f=0.8, s=sFM,
                          alg.clust.sim=Hierarchical.sim.resampling);
# Bernstein test with no assumption of independence
Bernstein.compute.pvalues(S.HC)
# Bernstein test with  assumption of independence
Bernstein.ind.compute.pvalues(S.HC)
```

Bernstein.p.value *Function to compute the p-value according to Bernstein inequality.*

Description

The Bernstein inequality gives an upper bound to the probability that the means of two random variables differ by chance, considering also their variance. This function implements the Berstein inequality and it is used by the functions `Bernstein.compute.pvalues` and `Bernstein.ind.compute.pvalues`

Usage

```
Bernstein.p.value(n, Delta, v)
```

Arguments

n	number of observations of the random variable
Delta	difference between the means
v	variance of the random variable

Value

a real number that provides an upper bound to the probability that the two means differ by chance

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

W. Hoeffding, Probability inequalities for sums of independent random variables, J. Amer. Statist. Assoc. vol.58 pp. 13-30, 1963.

See Also

Bernstein.compute.pvalues, Bernstein.ind.compute.pvalues

Examples

```
# Computation of the upper bounds to the probability that the two means differ by chance
Bernstein.p.value(n=100, Delta=0.1, v=0.01)
Bernstein.p.value(n=100, Delta=0.05, v=0.01)
Bernstein.p.value(n=100, Delta=0.05, v=0.1)
Bernstein.p.value(n=1000, Delta=0.05, v=0.1)
```

Chi.square.compute.pvalues

Function to compute the stability indices and the p-values associated to a set of clusterings according to the chi-square test between multiple proportions.

Description

For a given similarity matrix a list of stability indices, sorted by descending order, from the most significant clustering to the least significant is given. Moreover the corresponding p-values, computed according to a chi-square based test are provided.

Usage

```
Chi.square.compute.pvalues(sim.matrix, s0 = 0.9)
```

Arguments

sim.matrix	a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings. Rows correspond to the different clusterings; columns to the n repeated clusterings for each number of clusters. Row 1 corresponds to a 2-clustering, row 2 to a 3-clustering, ... row m to a m+1 clustering.
s0	threshold for the similarity value (default 0.9)

Details

The stability index for a given clustering is computed as the mean of the similarity indices between pairs of k-clusterings obtained from the perturbed data. The similarity matrix given as input can be obtained from the functions do.similarity.resampling, do.similarity.projection, do.similarity.noise. For each k-clustering the proportion of pairs of perturbed clusterings having similarity indices larger than a given threshold (the parameter $s0$) is computed. The p-values are obtained according the chi-square test between multiple proportions (each proportion corresponds to a different k-clustering). A low p-value means that there is a significant difference between the top sorted and the given k-clustering.

Value

a data frame with 4 components:

ordered.clusterings	a vector with the number of clusters ordered from the most significant to the least significant
p.value	a vector with the corresponding p-values computed according to chi-square test between multiple proportions in descending order (their values correspond to the clusterings of the vector ordered.clusterings)
means	vector with the stability index (mean similarity) for each k-clustering
variance	vector with the variance of the similarity for each k-clustering

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

A.Bertoni, G. Valentini, Model order selection for clustered bio-molecular data, In: Probabilistic Modeling and Machine Learning in Structural and Systems Biology, J. Rousu, S. Kaski and E. Ukkonen (Eds.), Tuusula, Finland, 17-18 June, 2006

See Also

Bernstein.compute.pvalues, Hypothesis.testing,

do.similarity.resampling, do.similarity.projection, do.similarity.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=10, m=15, dim=800, d=3, s=0.2)
nsubsamples <- 10;  # number of pairs of clusterings to be evaluated
max.num.clust <- 6; # maximum number of cluster to be evaluated
fract.resampled <- 0.8; # fraction of samples to subsampled
# building a similarity matrix using resampling methods, considering clusterings
# from 2 to 10 clusters with the k-means algorithm
Sr.Kmeans.sample6 <- do.similarity.resampling(M, c=max.num.clust, nsub=nsubsamples,
                     f=fract.resampled, s=sFM, alg.clust.sim=Kmeans.sim.resampling);
# computing p-values according to the chi square-based test
dr.Kmeans.sample6 <- Chi.square.compute.pvalues(Sr.Kmeans.sample6);
# the same, using noise to perturbate the data and  hierarchical clustering algorithm
Sn.HC.sample6 <- do.similarity.noise(M, c=max.num.clust, nnoisy=nsubsamples, perc=0.5,
                                s=sFM, alg.clust.sim=Hierarchical.sim.noise);
dn.HC.sample6 <- Chi.square.compute.pvalues(Sn.HC.sample6);
```

Compute.Chi.sq *Function to evaluate if a set of similarity distributions significantly differ using the chi square test.*

Description

The set of similarity values for a specific value of k (number of clusters) are subdivided in two groups choosing a threshold for the similarity value (default 0.9). Then different sets are compared using the chi squared test for multiple proportions. The number of degrees of freedom are equal to the number of the different sets minus 1. This function is iteratively used by `Chi.square.compute.pvalues`.

Usage

```
Compute.Chi.sq(M, s0 = 0.9)
```

Arguments

`M`	matrix representing the similarity values for different number of clusters. Each row represents similarity values for a number of clusters. Number of rows ==> how many numbers of clusters are considered; number of columns ==> cardinality of the similarity values for a given number of clusters
`s0`	threshold for the similarity value (default 0.9)

Value

p-value (type I error) associated with the null hypothesis (no difference between the considered set of k-clusterings)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

A.Bertoni, G. Valentini, Model order selection for clustered bio-molecular data, In: Probabilistic Modeling and Machine Learning in Structural and Systems Biology, J. Rousu, S. Kaski and E. Ukkonen (Eds.), Tuusula, Finland, 17-18 June, 2006

See Also

Chi.square.compute.pvalues

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=10, m=15, dim=800, d=3, s=0.2)
# computing the similarity matrix using random projections and hierarchcial clustering
Sim <- do.similarity.projection(M, c=6, nprojections=20, dim=JL.predict.dim(60,epsilon=0.2))
# Evaluating the p-value for the group of the 5 clusterings (from 2 to 6 clusters)
Compute.Chi.sq(Sim)
# the same, considering only the clusterings wih 2 and 6 clusters:
Compute.Chi.sq(Sim[c(1,5),])
```

---

compute.cumulative.multiple

Function to compute the empirical cumulative distribution function (ECDF) of the similarity measures.

Description

The function `compute.cumulative.multiple` computes the empirical cumulative distribution function (ECDF) of the similarity measures for different number of clusters between clusterings. The function `cumulative.values` returns the values of the empirical cumulative distribution

Usage

```
compute.cumulative.multiple(sim.matrix)

cumulative.values(Fun)
```

Arguments

`sim.matrix`	a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings. Each row corresponds to a different number of clusters; number of columns equal to the number of subsamples considered for each number of clusters.
`Fun`	Function of class ecdf that stores the discrete values of the cumulative distribution

Value

Function `compute.cumulative.multiple`: a list of function of class ecdf.

Function `cumulative.values`: a list with two elements: the "x" element stores a vector with the values of the random variable for which the cumulative distribution needs to be computed; the "y" element stores a vector with the corresponding values of the cumulative distribution (i.e. y = F(x)).

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

plot_cumulative.multiple

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2);
# generation of multiple similarity measures by resampling
Sr.kmeans.sample6 <- do.similarity.resampling(M, c=10, nsub=20, f=0.8, s=sFM,
                                        alg.clust.sim=Kmeans.sim.resampling);
# computation of multiple ecdf (from 2 to 10 clusters)
list.F <- compute.cumulative.multiple(Sr.kmeans.sample6);
# values of the ecdf for 8 clusters
l <- cumulative.values(list.F[[7]])
```

---

compute.integral *Functions to compute the integral of the ecdf of the similarity values*

Description

The function compute.integral computes the integral of the ecdf form the function of class ecdf that stores the discrete values of the empirical cumulative distribution, while the function compute.integral.from.similarity computes the integral of the ecdf exploiting then empirical mean of the similarity values (see the paper cited in the reference section for details).

Usage

```
compute.integral(Fun, subdivisions = 1000)

compute.integral.from.similarity(sim.matrix)
```

Arguments

Fun	Function of class ecdf that stores the discrete values of the empirical cumulative distribution
subdivisions	maximum number of subintervals used by the integration process
sim.matrix	a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings performed on subsamples of the original data. Number or rows equal to the different numbers of clusters considered; number of columns equal to the number of subsamples considered for each number of clusters.

Value

The function compute.integral returns the value of the estimate integral.

The function compute.integral.from.similarity returns a vector of the values of the estimate integrals (one for each row of $sim.matrix$).

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

A.Bertoni, G. Valentini, Discovering significant structures in clustered data through Bernstein inequality, CISI '06, Conferenza Italiana Sistemi Intelligenti, Ancona, Italia, 2006.

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2);
# generation of multiple similarity measures by resampling
Sr.kmeans.sample6 <- do.similarity.resampling(M, c=10, nsub=20, f=0.8, s=sFM,
                                    alg.clust.sim=Kmeans.sim.resampling);
# computation of multiple ecdf (from 2 to 10 clusters)
list.F <- compute.cumulative.multiple(Sr.kmeans.sample6);
# computation of the integral of the ecdf with 2 clusters
compute.integral(list.F[[1]])
# computation of the integral of the ecdf with 8 clusters
compute.integral(list.F[[7]])
# computation of the integral of the ecdfs from 2 to 10 clusters
compute.integral.from.similarity(Sr.kmeans.sample6)
```

Do.boolean.membership.matrix

Function to compute and build up a pairwise boolean membership matrix.

Description

It computes the pairwise membership matrix for a given clustering. The number of rows is equal to the number of columns (the number of examples). The element m_{ij} is set to 1 if the examples i and j belong to the same cluster, otherwise to 0. This function may be used also with clusterings that do not define strictly a partition of the data and using diferent number of clusters for each clustering.

Usage

```
Do.boolean.membership.matrix(cl, dim.M, examplelabels)
```

Arguments

cl	a clustering (list of vectors defining a clustering)
dim.M	dimension of the similarity matrix (number of examples)
examplelabels	labels of the examples drawn from the original data

Value

the pairwise boolean membership square matrix.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

Examples

```
library("clusterv")
library("stats")
# Synthetic data set generation (3 clusters with 20 examples for each cluster)
M <- generate.sample3(n=20, m=2)
# k-means clustering with 3 clusters
r<-kmeans(t(M), c=3, iter.max = 1000);
# this function is implemented in the clusterv package:
cl <- Transform.vector.to.list(r$cluster);
# generation of boolean membership square matrix:
B <- Do.boolean.membership.matrix(cl, 60, 1:60)
```

---

do.similarity.noise *Function that computes sets of similarity indices using injection of gaussian noise.*

Description

This function may use different clustering algorithms and different similarity measures to compute similarity indices. Injection of gaussian noise is applied to perturb the data. The gaussian noise added to the data has 0 mean and the standard deviation is estimated from the data (it is set to a given percentile value of the standard deviations computed for each variable). More precisely pairs of data sets are perturbed with noise and then are clustered and the resulting clusterings are compared using similarity indices between pairs of clusterings (e.g. Rand Index, Jaccard or Fowlkes and Mallows indices). These indices are computed multiple times for different number of clusters.

Usage

```
do.similarity.noise(X, c = 2, nnoisy = 100, perc = 0.5, seed = 100, s = sFM,
alg.clust.sim = Hierarchical.sim.noise, distance = "euclidean", hmethod = "ward.D")
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	if it is a vector of length 1, number of clusters from 2 to c are considered; otherwise are considered the number of clusters stored in the vector c.
nnoisy	number of pairs of noisy data
perc	percentile of the standard deviations to be used for the added gaussian noise (default: median)
seed	numerical seed for the random generator
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)

`alg.clust.sim` method that computes the similarity indices using subsampling techniques and a specific clustering algorithm. It may be one of the following: - Hierarchical.sim.resampling (hierarchical clustering algorithm, default) - Kmeans.sim.resampling (c - mean algorithm) - PAM.sim.resampling (Prediction Around Medoid algorithm) - Fuzzy.kmeans.sim.resampling (Fuzzy c-mean)

`distance` it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)

`hmethod` the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings performed on subsamples of the original data. Number of rows equal to the length of c (number of clusters); number of columns equal to nsub, that is the number of subsamples considered for each number of clusters.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

McShane, L.M., Radmacher, D., Freidlin, B., Yu, R., Li, M.C. and Simon, R., Method for assessing reproducibility of clustering patterns observed in analyses of microarray data, Bioinformatics, 11(8), pp. 1462-1469, 2002.

See Also

`do.similarity.projection`, `do.similarity.resampling`

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing similarity indices with the fuzzy c-mean algorithm
Sn.Fuzzy.kmeans.sample6 <- do.similarity.noise(M, c=8, nnoisy=30, perc=0.5, s=sFM,
                                    alg.clust.sim=Fuzzy.kmeans.sim.noise);
# computing similarity indices using the c-mean algorithm
Sn.Fuzzy.kmeans.sample6 <- do.similarity.noise(M, c=8, nnoisy=30, perc=0.5, s=sFM,
                                    alg.clust.sim=Fuzzy.kmeans.sim.noise);
# computing similarity indices using the hierarchical clustering algorithm
Sn.HC.sample6 <- do.similarity.noise(M, c=8, nnoisy=30, perc=0.5, s=sFM,
                                    alg.clust.sim=Hierarchical.sim.noise);
```

`do.similarity.projection` *Function that computes sets of similarity indices using randomized maps.*

Description

This function may use different clustering algorithms and different similarity measures to compute similarity indices. Random projections techniques are applied to perturb the data. More precisely pairs of data sets are projected into lower dimensional subspaces using randomized maps, and then are clustered and the resulting clusterings are compared using similarity indices between pairs of clusterings (e.g. Rand Index, Jaccard or Fowlkes and Mallows indices). These indices are computed multiple times for different number of clusters.

Usage

```
do.similarity.projection(X, c = 2, nprojections = 100, dim = 2, pmethod = "PMO",
scale = TRUE, seed = 100, s = sFM, alg.clust.sim = Hierarchical.sim.projection,
distance = "euclidean", hmethod = "ward.D")
```

Arguments

`X`	matrix of data (variables are rows, examples columns)
`c`	if it is a vector of length 1, number of clusters from 2 to c are considered; otherwise are considered the number of clusters stored in the vector c.
`nprojections`	number of pairs of projected data
`dim`	dimension of the projected data
`pmethod`	pmethod : projection method. It must be one of the following: "RS" (random subspace projection) "PMO" (Plus Minus One random projection) (default) "Norm" (normal random projection) "Achlioptas" (Achlioptas random projection)
`scale`	if TRUE randomized projections are scaled (default)
`seed`	numerical seed for the random generator
`s`	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
`alg.clust.sim`	method that computes the similarity indices using subsampling techniques and a specific clustering algorithm. It may be one of the following: - Hierarchical.sim.resampling (hierarchical clustering algorithm, default) - Kmeans.sim.resampling (c - mean algorithm) - PAM.sim.resampling (Prediction Around Medoid algorithm) - Fuzzy.kmeans.sim.resampling (Fuzzy c-mean)
`distance`	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
`hmethod`	the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings performed on subsamples of the original data. Number of rows equal to the length of c (number of clusters); number of columns equal to nsub, that is the number of subsamples considered for each number of clusters.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

A.Bertoni, G. Valentini, Randomized maps for assessing the reliability of patients clusters in DNA microarray data analyses, Artificial Intelligence in Medicine 37(2):85-109 2006

See Also

do.similarity.resampling, do.similarity.noise

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing similarity indices with the fuzzy c-mean algorithm
Sp.fuzzy.kmeans.sample6 <- do.similarity.projection(M, c=8, nprojections=30,
  dim=JL.predict.dim(120,0.2), pmethod="PMO", alg.clust.sim=Fuzzy.kmeans.sim.projection);
# computing similarity indices using the c-mean algorithm
Sp.kmeans.sample6 <- do.similarity.projection(M, c=8, nprojections=30,
   dim=JL.predict.dim(120,0.2), pmethod="PMO", alg.clust.sim=Kmeans.sim.projection);
# computing similarity indices using the hierarchical clustering algorithm
Sp.HC.sample6 <- do.similarity.projection(M, c=8, nprojections=30,
  dim=JL.predict.dim(120,0.2), pmethod="PMO", alg.clust.sim=Hierarchical.sim.projection);
```

---

do.similarity.resampling

Function that computes sets of similarity indices using resampling techniques.

Description

This function may use different clustering algorithms and different similarity measures to compute similarity indices. Subsampling techniques are applied to perturb the data. More precisely pairs of data sets are sampled according to an uniform distribution without replacement and then are clustered and the resulting clusterings are compared using similarity indices between pairs of clusterings (e.g. Rand Index, Jaccard or Fowlkes and Mallows indices). These indices are computed multiple times for different number of clusters.

Usage

```
do.similarity.resampling(X, c = 2, nsub = 100, f = 0.8, s = sFM,
alg.clust.sim = Hierarchical.sim.resampling, distance = "euclidean", hmethod = "ward.D")
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	if it is a vector of length 1, number of clusters from 2 to c are considered; otherwise are considered the number of clusters stored in the vector c.
nsub	number of pairs of subsamples
f	fraction of the data resampled without replacement
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
alg.clust.sim	method that computes the similarity indices using subsampling techniques and a specific clustering algorithm. It may be one of the following: - Hierarchical.sim.resampling (hierarchical clustering algorithm, default) - Kmeans.sim.resampling (c - mean algorithm) - PAM.sim.resampling (Prediction Around Medoid algorithm) - Fuzzy.kmeans.sim.resampling (Fuzzy c-mean)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings performed on subsamples of the original data. Number of rows equal to the length of c (number of clusters); number of columns equal to nsub, that is the number of subsamples considered for each number of clusters.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

Ben-Hur, A. Ellisseeff, A. and Guyon, I., A stability based method for discovering structure in clustered data, In: "Pacific Symposium on Biocomputing", Altman, R.B. et al (eds.), pp, 6-17, 2002.

See Also

do.similarity.projection, do.similarity.noise

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing similarity indices with the fuzzy c-mean algorithm
Sr.Fuzzy.kmeans.sample6 <- do.similarity.resampling(M, c=8, nsub=30, f=0.8, s=sFM,
                           alg.clust.sim=Fuzzy.kmeans.sim.resampling);
# computing similarity indices using the c-mean algorithm
```

```
Sr.Kmeans.sample6 <- do.similarity.resampling(M, c=8, nsub=30, f=0.8, s=sFM,
                                    alg.clust.sim=Kmeans.sim.resampling)
# computing similarity indices using the hierarchical clustering algorithm
Sr.HC.sample6 <- do.similarity.resampling(M, c=8, nsub=30, f=0.8, s=sFM);
```

`Fuzzy.kmeans.sim.noise` *Function to compute similarity indices using noise injection techniques and fuzzy c-mean clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with random noise is computed for a given number of clusters. The variance of the added gaussian noise, estimated from the data as the perc percentile of the standard deviations of the input variables, the percentile itself and the similarity measure can be selected.

Usage

```
Fuzzy.kmeans.sim.noise(X, c = 2, nnoisy = 100, perc = 0.5, s = sFM,
                       distance = "euclidean", hmethod = NULL)
```

Arguments

`X`	matrix of data (variables are rows, examples columns)
`c`	number of clusters
`nnoisy`	number of pairs of noisy data
`perc`	percentile of the standard deviations to be used for the added gaussian noise (def. 0.5)
`s`	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
`distance`	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
`hmethod`	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nnoisy)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`Fuzzy.kmeans.sim.projection`, `Fuzzy.kmeans.sim.resampling`, `perturb.by.noise`

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Fuzzy.kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Fuzzy.kmeans.sim.noise(M, c = 3, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Fuzzy.kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sJaccard)
# 2 clusters using 0.95 percentile (more noise)
v095 <- Fuzzy.kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sFM, perc=0.95)
```

Fuzzy.kmeans.sim.projection

Function to compute similarity indices using random projections and fuzzy c-mean clustering.

Description

A vector of similarity measures between pairs of clusterings perturbed with random projections is computed for a given number of clusters. The dimension of the projected data, the type of randomized map and the similarity measure may be selected.

Usage

```
Fuzzy.kmeans.sim.projection(X, c = 2, nprojections = 100, dim = 2, pmethod = "PMO",
scale = TRUE, seed = 100, s = sFM, distance = "euclidean", hmethod = NULL)
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nprojections	number of pairs of projected data
dim	dimension of the projected data
pmethod	projection method. It must be one of the following: - "RS" (random subspace projection) - "PMO" (Plus Minus One random projection) - "Norm" (normal random projection) - "Achlioptas" (Achlioptas random projection)
scale	if TRUE randomized projections are scaled (default)
seed	numerical seed for the random generator
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nprojections)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Fuzzy.kmeans.sim.resampling, Fuzzy.kmeans.sim.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Fuzzy.kmeans.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                                  pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Fuzzy.kmeans.sim.projection(M, c = 3, nprojections = 20, dim = 200,
                                  pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Fuzzy.kmeans.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                                   pmethod = "PMO", s = sJaccard)
```

Fuzzy.kmeans.sim.resampling

Function to compute similarity indices using resampling techniques and fuzzy c-mean clustering.

Description

A vector of similarity measures between pairs of clusterings perturbed with resampling techniques is computed for a given number of clusters, using the fuzzy c-mean algorithm. The fraction of the resampled data (without replacement) and the similarity measure can be selected.

Usage

```
Fuzzy.kmeans.sim.resampling(X, c = 2, nsub = 100, f = 0.8, s = sFM,
                            distance = "euclidean", hmethod = NULL)
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nsub	number of subsamples
f	fraction of the data resampled without replacement
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)

distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nsub)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Fuzzy.kmeans.sim.projection, Fuzzy.kmeans.sim.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Fuzzy.kmeans.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Fuzzy.kmeans.sim.resampling(M, c = 3, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Fuzzy.kmeans.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sJaccard)
```

Hierarchical.sim.noise

Function to compute similarity indices using noise injection techniques and hierarchical clustering.

Description

A vector of similarity measures between pairs of clusterings perturbed with random noise is computed for a given number of clusters. The variance of the added gaussian noise, estimated from the data as the perc percentile of the standard deviations of the input variables, the percentile itself, the similarity measure and the type of hierarchical clustering may be selected.

Usage

```
Hierarchical.sim.noise(X, c = 2, nnoisy = 100, perc = 0.5, s = sFM,
                       distance = "euclidean", hmethod = "ward.D")
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nnoisy	number of pairs of noisy data
perc	percentile of the standard deviations to be used for the added gaussian noise (def. 0.5)

s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

vector of the computed similarity measures (length equal to nnoisy)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Hierarchical.sim.projection, Hierarchical.sim.resampling, perturb.by.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Hierarchical.sim.noise(M, c = 2, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Hierarchical.sim.noise(M, c = 3, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Hierarchical.sim.noise(M, c = 2, nnoisy = 20,  s = sJaccard)
#  2 clusters using the Jaccard index and Pearson correlation
v2JP <- Hierarchical.sim.noise(M, c = 2, nnoisy = 20, s = sJaccard, distance="pearson")
# 2 clusters using 0.95 percentile (more noise)
v095 <- Hierarchical.sim.noise(M, c = 2, nnoisy = 20,  s = sFM, perc=0.95)
```

Hierarchical.sim.projection

Function to compute similarity indices using random projections and hierarchical clustering.

Description

A vector of similarity measures between pairs of clusterings perturbed with random projections is computed for a given number of clusters. The dimension of the projected data, the type of randomized map, the similarity measure and the type of hierarchical clustering may be selected.

Usage

```
Hierarchical.sim.projection(X, c = 2, nprojections = 100, dim = 2, pmethod = "RS",
scale = TRUE, seed = 100, s = sFM, distance = "euclidean", hmethod = "ward.D")
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nprojections	number of pairs of projected data
dim	dimension of the projected data
pmethod	projection method. It must be one of the following: - "RS" (random subspace projection) - "PMO" (Plus Minus One random projection) - "Norm" (normal random projection) - "Achlioptas" (Achlioptas random projection)
scale	if TRUE randomized projections are scaled (default)
seed	numerical seed for the random generator
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

vector of the computed similarity measures (length equal to nprojections)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Hierarchical.sim.resampling, Hierarchical.sim.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Hierarchical.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                                  pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Hierarchical.sim.projection(M, c = 3, nprojections = 20, dim = 200,
                                  pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Hierarchical.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                                   pmethod = "PMO", s = sJaccard)
```

```
#  2 clusters using the Jaccard index and Pearson correlation
v2JP <- Hierarchical.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                                    pmethod = "PMO", s = sJaccard, distance="pearson")
```

`Hierarchical.sim.resampling` *Function to compute similarity indices using resampling techniques and hierarchical clustering.*

Description

Function to compute similarity indices using resampling techniques and hierarchical clustering. A vector of similarity measures between pairs of clusterings perturbed with resampling techniques is computed for a given number of clusters. The fraction of the resampled data (without replacement), the similarity measure and the type of hierarchical clustering may be selected.

Usage

```
Hierarchical.sim.resampling(X, c = 2, nsub = 100, f = 0.8, s = sFM,
                            distance = "euclidean", hmethod = "ward.D")
```

Arguments

`X`	matrix of data (variables are rows, examples columns)
`c`	number of clusters
`nsub`	number of subsamples
`f`	fraction of the data resampled without replacement
`s`	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
`distance`	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
`hmethod`	the agglomeration method to be used. This parameter is used only by the hierarchical clustering algorithm. This should be one of the following: "ward.D", "single", "complete", "average", "mcquitty", "median" or "centroid", according of the hclust method of the package stats.

Value

vector of the computed similarity measures (length equal to nsub)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`Hierarchical.sim.projection`, `Hierarchical.sim.noise`

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Hierarchical.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Hierarchical.sim.resampling(M, c = 3, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Hierarchical.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sJaccard)
#  2 clusters using the Jaccard index and Pearson correlation
v2JP <- Hierarchical.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sJaccard,
                                    distance="pearson")
```

`Hybrid.testing` *Statistical test based on stability methods for model order selection.*

Description

Statistical test to estimate if there is a significant difference between a set of clustering solutions. Given a set of clustering solutions (that is solutions for different number *k* of clusters), the statistical test using both the *Bernstein* inequality-based test and the χ^2 based test evaluates what are the significant solutions at a given significance level.

Usage

```
Hybrid.testing(sim.matrix, alpha = 0.01, s0 = 0.9)
```

Arguments

`sim.matrix`	a matrix that stores the similarity between pairs of clustering across multiple number of clusters and multiple clusterings.
`alpha`	significance level (default 0.01)
`s0`	threshold for the similarity value used in the χ^2 based test (default 0.9)

Details

The function accepts as input a similarity matrix that stores the similarity measure between multiple pairs of clusterings considering different number of clusters. Each row of the matrix corresponds to a k-clustering, each column to different repeated measures. Note that the similarities can be computed using different clustering algorithms, different perturbations methods (resampling techniques, random projections or noise-injection methods) and different similarity measures. The stability index for a given clustering is computed as the mean of the similarity indices between pairs of k-clusterings obtained from the perturbed data. The similarity matrix given as input can be obtained from the functions do.similarity.resampling, do.similarity.projection, do.similarity.noise. The clusterings are ranked according to the values of the stability indices and the Bernstein inequality-based test is iteratively performed between the top ranked and upward from the last ranked clustering until the null hypothesis (that is no significant difference between the clustering solutions) cannot be rejected. Then, to refine the solutions, the chi square-based test is performed on the remaining top ranked clusterings. The significant solutions at a given α significance level, as well as the computed p-values are returned.

Value

a list with 6 elements:

`n.Bernstein.selected`
: number of clusterings selected as significant by the Bernstein test

`n.chi.sq.selected`
: number of clusterings selected as significant by chi square test. It may be equal to 0 if Bernstein test selects only 1 clustering.

`Bernstein.res`
: data frame with the p-values obtained from Bernstein inequality

`chi.sq.res`
: data frame with the p-values obtained from chi square test. If through Bernstein inequality test only 1 clustering is significant this component is NULL

`selected.res`
: data frame with the results relative to the clusterings selected by the overall hybrid test

`F`
: a list of cumulative distribution functions (of class ecdf) (not sorted).

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

W. Hoeffding, Probability inequalities for sums of independent random variables, J. Amer. Statist. Assoc. vol.58 pp. 13-30, 1963.

A.Bertoni, G. Valentini, Model order selection for clustered bio-molecular data, In: Probabilistic Modeling and Machine Learning in Structural and Systems Biology, J. Rousu, S. Kaski and E. Ukkonen (Eds.), Tuusula, Finland, 17-18 June, 2006

A.Bertoni, G. Valentini, Discovering significant structures in clustered data through Bernstein inequality, CISI '06, Conferenza Italiana Sistemi Intelligenti, Ancona, Italia, 2006.

See Also

`Bernstein.compute.pvalues`, `Chi.square.compute.pvalues`,

`Hypothesis.testing`, `do.similarity.resampling`,

`do.similarity.projection`, `do.similarity.noise`

Examples

```
library("clusterv")
# Generation of a synthetic data set with a three-levels hierarchical structure
M1 <- generate.sample.h2 (n=20, l=20, Delta.h=6, Delta.v=3, sd=0.1)
# building a similarity matrix using resampling methods, considering clusterings
# from 2 to 15 clusters
S1.HC <- do.similarity.resampling (M1, c=15, nsub=20, f=0.8, s=sFM,
                                   alg.clust.sim=Hierarchical.sim.resampling)
# Application of the Hybrid statistical test
l1.HC <- Hybrid.testing(S1.HC, alpha=0.01, s0=0.95)
# 3 clusterings are selected, according to the hierarchical structure of the data:
l1.HC$selected.res
```

`Hypothesis.testing` *Function to select significant clusterings from a given set of p-values*

Description

For a given set of p-values returned from a given hypothesis testing, it returns the items for which there is no significant difference at alpha significance level (that is the items for which p > alpha).

Usage

```
Hypothesis.testing(d, alpha = 0.01)
```

Arguments

`d`	data frame with the p-values returned by a given test of hypothesis (e.g. Bernstein or Chi square-based tests)
`alpha`	significance level

Value

a data frame corresponding to the clusterings significant at alpha level

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`Bernstein.compute.pvalues` `Chi.square.compute.pvalues`

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2)
nsubsamples <- 10;  # number of pairs of clusterings to be evaluated
max.num.clust <- 6; # maximum number of cluster to be evaluated
fract.resampled <- 0.8; # fraction of samples to subsampled
# building a similarity matrix using resampling methods, considering clusterings
# from 2 to 10 clusters with the k-means algorithm
Sr.Kmeans.sample6 <- do.similarity.resampling(M, c=max.num.clust, nsub=nsubsamples,
                     f=fract.resampled, s=sFM, alg.clust.sim=Kmeans.sim.resampling);
# computing p-values according to the chi square-based test
dr.Kmeans.sample6 <- Chi.square.compute.pvalues(Sr.Kmeans.sample6);
# test of hypothesis based on the obtained set of p-values
hr.Kmeans.sample6 <- Hypothesis.testing(dr.Kmeans.sample6, alpha=0.01);
# at the given significance level (0.01) the clustering with 2 clusters is selected:
hr.Kmeans.sample6
```

Intersect — *Function to compute the intersection between elements of two vectors*

Description

Having as input two sets of elements represented by two vectors, the intersection between the two sets is performed and the corresponding vector is returned.

Usage

```
Intersect(sub1, sub2)
```

Arguments

sub1	first vector representing the first set
sub2	second vector representing the second set

Value

vector that stores the elements common to the two input vectors

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

Examples

```
# Intesection between two sets of elements represented by vectors
s1 <- 1:10;
s2 <- 3:12;
Intersect(s1, s2)
```

Kmeans.sim.noise — *Function to compute similarity indices using noise injection techniques and kmeans clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with random noise is computed for a given number of clusters. The variance of the added gaussian noise, estimated from the data as the perc percentile of the standard deviations of the input variables, the percentile itself and the similarity measure can be selected.

Usage

```
Kmeans.sim.noise(X, c = 2, nnoisy = 100, perc = 0.5, s = sFM,
                 distance = "euclidean", hmethod = NULL)
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nnoisy	number of pairs of noisy data
perc	percentile of the standard deviations to be used for the added gaussian noise (def. 0.5)
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	actually only the euclidean distance is available "euclidean" (default)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nnoisy)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Kmeans.sim.projection, Kmeans.sim.resampling, perturb.by.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Kmeans.sim.noise(M, c = 3, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sJaccard)
# 2 clusters using 0.95 percentile (more noise)
v095 <- Kmeans.sim.noise(M, c = 2, nnoisy = 20,  s = sFM, perc=0.95)
```

Kmeans.sim.projection *Function to compute similarity indices using random projections and kmeans clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with random projections is computed for a given number of clusters. The dimension of the projected data, the type of randomized map and the similarity measure may be selected.

Usage

```
Kmeans.sim.projection(X, c = 2, nprojections = 100, dim = 2, pmethod = "PMO",
  scale = TRUE, seed = 100, s = sFM, distance = "euclidean", hmethod = "ward.D")
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nprojections	number of pairs of projected data
dim	dimension of the projected data
pmethod	projection method. It must be one of the following: - "RS" (random subspace projection) - "PMO" (Plus Minus One random projection) - "Norm" (normal random projection) - "Achlioptas" (Achlioptas random projection)
scale	if TRUE randomized projections are scaled (default)
seed	numerical seed for the random generator
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	actually only the euclidean distance is available "euclidean" (default)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nprojections)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

Kmeans.sim.resampling, Kmeans.sim.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Kmeans.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                            pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Kmeans.sim.projection(M, c = 3, nprojections = 20, dim = 200,
                            pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Kmeans.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                             pmethod = "PMO", s = sJaccard)
```

`Kmeans.sim.resampling` *Function to compute similarity indices using resampling techniques and kmeans clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with resampling techniques is computed for a given number of clusters, using the kmeans algorithm. The fraction of the resampled data (without replacement) and the similarity measure can be selected.

Usage

```
Kmeans.sim.resampling(X, c = 2, nsub = 100, f = 0.8, s = sFM,
                      distance = "euclidean", hmethod = NULL)
```

Arguments

`X`	matrix of data (variables are rows, examples columns)
`c`	number of clusters
`nsub`	number of subsamples
`f`	fraction of the data resampled without replacement
`s`	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
`distance`	actually only the euclidean distance is available "euclidean" (default)
`hmethod`	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nsub)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`Kmeans.sim.projection`, `Kmeans.sim.noise`

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- Kmeans.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- Kmeans.sim.resampling(M, c = 3, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- Kmeans.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sJaccard)
```

PAM.sim.noise *Function to compute similarity indices using noise injection techniques and PAM clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with random noise is computed for a given number of clusters. The variance of the added gaussian noise, estimated from the data as the perc percentile of the standard deviations of the input variables, the percentile itself and the similarity measure can be selected.

Usage

```
PAM.sim.noise(X, c = 2, nnoisy = 100, perc = 0.5, s = sFM,
              distance = "euclidean", hmethod = NULL)
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nnoisy	number of pairs of noisy data
perc	percentile of the standard deviations to be used for the added gaussian noise (def. 0.5)
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nnoisy)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

PAM.sim.projection, PAM.sim.resampling, perturb.by.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- PAM.sim.noise(M, c = 2, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 3 clusters:
```

```
v3 <- PAM.sim.noise(M, c = 3, nnoisy = 20,  s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- PAM.sim.noise(M, c = 2, nnoisy = 20,  s = sJaccard)
# 2 clusters using 0.95 percentile (more noise)
v095 <- PAM.sim.noise(M, c = 2, nnoisy = 20,  s = sFM, perc=0.95)
```

`PAM.sim.projection` *Function to compute similarity indices using random projections and PAM clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with random projections is computed for a given number of clusters. The dimension of the projected data, the type of randomized map and the similarity measure may be selected.

Usage

```
PAM.sim.projection(X, c = 2, nprojections = 100, dim = 2, pmethod = "PMO",
  scale = TRUE, seed = 100, s = sFM, distance = "euclidean", hmethod = NULL)
```

Arguments

`X`	matrix of data (variables are rows, examples columns)
`c`	number of clusters
`nprojections`	number of pairs of projected data
`dim`	dimension of the projected data
`pmethod`	projection method. It must be one of the following: - "RS" (random subspace projection) - "PMO" (Plus Minus One random projection) - "Norm" (normal random projection) - "Achlioptas" (Achlioptas random projection)
`scale`	if TRUE randomized projections are scaled (default)
`seed`	numerical seed for the random generator
`s`	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
`distance`	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
`hmethod`	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nprojections)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`PAM.sim.resampling`, `PAM.sim.noise`

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- PAM.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                         pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- PAM.sim.projection(M, c = 3, nprojections = 20, dim = 200,
                         pmethod = "PMO", s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- PAM.sim.projection(M, c = 2, nprojections = 20, dim = 200,
                          pmethod = "PMO", s = sJaccard)
```

PAM.sim.resampling *Function to compute similarity indices using resampling techniques and PAM clustering.*

Description

A vector of similarity measures between pairs of clusterings perturbed with resampling techniques is computed for a given number of clusters, using the PAM algorithm. The fraction of the resampled data (without replacement) and the similarity measure can be selected.

Usage

```
PAM.sim.resampling(X, c = 2, nsub = 100, f = 0.8, s = sFM,
                   distance = "euclidean", hmethod = NULL)
```

Arguments

X	matrix of data (variables are rows, examples columns)
c	number of clusters
nsub	number of subsamples
f	fraction of the data resampled without replacement
s	similarity function to be used. It may be one of the following: - sFM (Fowlkes and Mallows) - sJaccard (Jaccard) - sM (matching coefficient) (default Fowlkes and Mallows)
distance	it must be one of the two: "euclidean" (default) or "pearson" (that is 1 - Pearson correlation)
hmethod	parameter used for internal compatibility.

Value

vector of the computed similarity measures (length equal to nsub)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

PAM.sim.projection, PAM.sim.noise

Examples

```
library("clusterv")
# Synthetic data set generation
M <- generate.sample6 (n=20, m=10, dim=600, d=3, s=0.2);
# computing a vector of similarity indices with 2 clusters:
v2 <- PAM.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 3 clusters:
v3 <- PAM.sim.resampling(M, c = 3, nsub = 20, f = 0.8, s = sFM)
# computing a vector of similarity indices with 2 clusters using the Jaccard index
v2J <- PAM.sim.resampling(M, c = 2, nsub = 20, f = 0.8, s = sJaccard)
```

---

perturb.by.noise *Function to generate a data set perturbed by noise.*

Description

This funtion adds gaussian noise to the data. The mean of the gaussian noise is 0 and the standard deviation is estimated from the data. The gaussian noise added to the data has 0 mean and the standard deviation is estimated from the data (it is set to a given percentile value of the standard deviations computed for each variable).

Usage

```
perturb.by.noise(X, perc = 0.5)
```

Arguments

X	matrix of data (variables are rows, examples columns)
perc	percentile of the standard deviation (def: 0.5)

Value

matrix of perturbed data (variables are rows, examples columns)

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

McShane, L.M., Radmacher, D., Freidlin, B., Yu, R., Li, M.C. and Simon, R., Method for assessing reproducibility of clustering patterns observed in analyses of microarray data, Bioinformatics, 11(8), pp. 1462-1469, 2002.

See Also

do.similarity.noise

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=100, d=3, s=0.2);
# generation of a data set perturbed by noise
M.perturbed <- perturb.by.noise(M);
# generation of a data set more perturbed by noise
M.more.perturbed <- perturb.by.noise(M, perc=0.95);
```

`plot_cumulative` *Function to plot the empirical cumulative distribution function of the similarity values*

Description

The function `plot_cumulative` plots the ecdf of the similarity values between pairs of clusterings for a specific number of clusters. The function `plot_cumulative.multiple` plots the graphs of the empirical cumulative distributions corresponding to different number of clusters, using different patterns and/or different colors for each graph. Up to 15 ecdf graphs can be plotted simultaneously.

Usage

```
plot_cumulative(Fun)

plot_cumulative.multiple(list.F, labels = NULL, min.x = -1, colors = TRUE)
```

Arguments

`Fun`	Function of class ecdf that stores the discrete values of the cumulative distribution
`list.F`	a list of function of class ecdf
`labels`	vector of the labels associated to the CDF. If NULL (default), then a vector of labels from 2 to lenght(list.F)+1 is used.
`min.x`	minimum value to be plotted for similarities. If -1 (default) the minimum of the similarity value is obtained from list.F
`colors`	if TRUE (default) different colors are used to plot the different ECDF, otherwise black lines are used

Value

No return value, the function is called for its side-effect of drawing a plot on the current graphics device.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

`compute.cumulative.multiple`

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2);
# generation of multiple similarity measures by resampling
Sr.kmeans.sample6 <- do.similarity.resampling(M, c=10, nsub=20, f=0.8, s=sFM,
                                    alg.clust.sim=Kmeans.sim.resampling);
# computation of multiple ecdf (from 2 to 10 clusters)
list.F <- compute.cumulative.multiple(Sr.kmeans.sample6);
# values of the ecdf for 8 clusters
l <- cumulative.values(list.F[[7]])
# plot of the ecdf for 8 clusters
plot_cumulative(list.F[[7]])
# plot of the empirical cumulative distributions from 2 to 10 clusters
plot_cumulative.multiple(list.F)
```

---

plot_histograms.similarity

Plotting histograms of similarity measures between clusterings

---

Description

These functions plot histograms of a set of similarity measures obtained through perturbation methods. In particular `plot_hist.similarity` plots a single histogram referred to a specific number of clusters, while `plot_multiple.hist.similarity` plots multiple histograms referred to different numbers of clusters (one for each number of clusters, i.e. one for each row of the matrix S of similarity values).

Usage

```
plot_hist.similarity(sim, nbins = 25)

plot_multiple.hist.similarity(S, n.col = 3, labels = NULL, nbins = 25)
```

Arguments

`sim`	vector of similarity values
`nbins`	number of the bins of the histogram
`S`	Matrix of similarity values, rows correspond to diferent number of clusters
`n.col`	number of columns in the grid of the histograms (default = 3)
`labels`	label of the histograms. If NULL (default) the number of clusters from 2 to nrow(S)+1 are used

Value

No return value, the function is called for its side-effect of drawing a plot on the current graphics device.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

plot_cumulative, plot_cumulative.multiple

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2);
# generation of multiple similarity measures by resampling
Sr.kmeans.sample6 <- do.similarity.resampling(M, c=10, nsub=20, f=0.8, s=sFM,
                                    alg.clust.sim=Kmeans.sim.resampling);
# plot of the histograms of similarity measures for clusterings from 2 to 10 clusters:
plot_multiple.hist.similarity (Sr.kmeans.sample6, n.col=3, labels=NULL, nbins=25);
# the same as postrcript file
postscript(file="histograms.eps", horizontal=FALSE, onefile = FALSE);
plot_multiple.hist.similarity (Sr.kmeans.sample6, n.col=3, labels=NULL, nbins=25);
dev.off();
unlink("histograms.eps");
# plot of a single histogram
plot_hist.similarity(Sr.kmeans.sample6[2,], nbins = 25)
```

---

plot_pvalues *Function to plot p-values for different tests of hypothesis*

Description

The p-values corresponding to different k-clusterings according to different hypothesis testing are plotted. A horizontal line corresponding to a given alpha value (significance) is also plotted. In the x axis is represented the number of clusters sorted according to the value of the stability index, and in the y axis the corresponding p-value. In this way the results of different tests of hypothesis can be compared.

Usage

```
plot_pvalues(l,alpha=1e-02,legendy=0, leg_label=NULL, colors=TRUE)
```

Arguments

l	a list of lists. Each component list represents a different test of hypothesis, and it has in turn 4 components: ordered.clusterings : a vector with the number of clusters ordered from the most significant to the least significant; p.value : a vector with the corresponding p-values computed according to chi-square test between multiple proportions in descending order (their values correspond to the clusterings of the vector ordered.clusterings); means : vector with the mean similarity (stability index) for each clustering; variance : vector with the variance of the similarity for each clustering.
alpha	alpha value for which the straight line is plotted
legendy	ordinate of the legend. If 0 (def.) no legend is plotted.
leg_label	labels of the legend. If NULL (def.) the text "test 1, test 2, ... test n" for the n tests is printed. Otherwise it is a vector of characters specifying the text to be printed
colors	if TRUE (def.) lines are printed with colors, otherwise using only different line pattern

Value

No return value, the function is called for its side-effect of drawing a plot on the current graphics device.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

See Also

plot_cumulative.multiple

Examples

```
library("clusterv")
# Data set generation
M <- generate.sample6 (n=20, m=10, dim=1000, d=3, s=0.2);
# generation of multiple similarity measures by resampling
Sr.kmeans.sample6 <- do.similarity.resampling(M, c=10, nsub=20, f=0.8, s=sFM,
                                    alg.clust.sim=Kmeans.sim.resampling);
# hypothesis testing using the chi-square based test
d.chi <- Chi.square.compute.pvalues(Sr.kmeans.sample6)
# hypothesis testing using the Bernstein based test
d.Bern <- Bernstein.compute.pvalues(Sr.kmeans.sample6)
# hypothesis testing using the Bernstein based test (with independence assumption)
d.Bern.ind <- Bernstein.ind.compute.pvalues(Sr.kmeans.sample6)
l <- list(d.chi, d.Bern, d.Bern.ind);
# plot of the corresponding computed p-values
plot_pvalues(l, alpha = 1e-05, legendy = 1e-12)
```

Similarity.measures *Similarity measures between pairs of clusterings*

Description

Classical similarity measures between pairs of clusterings are implemented. These measures use the pairwise boolean membership matrix (Do.boolean.membership.matrix) to compute the similarity between two clusterings, using the matrix as a vector and computing the result as an internal product. It may be shown that the same result may be obtained using contingency matrices and the classical definition of Fowlkes and Mallows (implemented with the function sFM), Jaccard (implemented with the function sJaccard) and Matching (Rand Index, implemented with the function sM) coefficients. Their values range from 0 to 1 (0 no similarity, 1 identity).

Usage

```
sFM(M1, M2)
sJaccard(M1, M2)
sM(M1, M2)
```

Arguments

M1	boolean membership matrix representing the first clustering
M2	boolean membership matrix representing the second clustering

Value

similarity measure between the two clusterings according to Fowlkes and Mallows (`sFM`), Jaccard (`sJaccard`) and Matching (`sM`) coefficients.

Author(s)

Giorgio Valentini <valentini@di.unimi.it>

References

Ben-Hur, A. Ellisseeff, A. and Guyon, I., A stability based method for discovering structure in clustered data, In: "Pacific Symposium on Biocomputing", Altman, R.B. et al (eds.), pp, 6-17, 2002.

See Also

`Do.boolean.membership.matrix`

Examples

```
library("clusterv")
library("stats")
library("cluster")
# Synthetic data set generation (3 clusters with 20 examples for each cluster)
M <- generate.sample3(n=20, m=2)
# k-means clustering with 3 clusters
r1<-kmeans(t(M), c=3, iter.max = 1000);
# this function is implemented in the clusterv package:
cl1 <- Transform.vector.to.list(r1$cluster);
# generation of a boolean membership square matrix:
Bkmeans <- Do.boolean.membership.matrix(cl1, 60, 1:60)
# the same as above, using PAM clustering with 3 clusters
d <- dist (t(M));
r2 <- pam (d,3,cluster.only=TRUE);
cl2 <- Transform.vector.to.list(r2);
BPAM <- Do.boolean.membership.matrix(cl2, 60, 1:60)
# computation of the Fowlkes and Mallows index between the k-means and the PAM clustering:
sFM(Bkmeans, BPAM)
# computation of the Jaccard index between the k-means and the PAM clustering:
sJaccard(Bkmeans, BPAM)
# computation of the Matching coefficient between the k-means and the PAM clustering:
sM(Bkmeans, BPAM)
```

Index