Package 'ConsensusClustering'

March 5, 2024

Type Package

Title Consensus Clustering

Version 1.2.0

Description Clustering, or cluster analysis, is a widely used technique in bioinformatics to identify groups of similar biological data points. Consensus clustering is an extension to clustering algorithms that aims to construct a robust result from those clustering features that are invariant under different sources of variation. For the reference, please cite the following paper: Yousefi, Melograna, et. al., (2023) <doi:10.3389/fmicb.2023.1170391>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports assertthat, dplyr, igraph, cluster, mvtnorm, utils, graphics, stats

NeedsCompilation no

Author Behnam Yousefi [aut, cre, cph]

Maintainer Behnam Yousefi <yousefi.bme@gmail.com>

Repository CRAN

Date/Publication 2024-03-05 11:20:03 UTC

R topics documented:

`adj_conv` *Convert adjacency function to the affinity matrix*

Description

Convert adjacency function to the affinity matrix

Usage

```
adj_conv(adj.mat, alpha = 1)
```

Arguments

`adj.mat`	Adjacency matrix. The elements must be within [-1, 1].
`alpha`	soft threshold value (see details).

Details

adj = exp(-(1-adj)^2/(2*alpha^2)) ref: Luxburg (2007), "A tutorial on spectral clustering", Stat Comput

Value

the matrix if affinity values.

Examples

```
Adj_mat = rbind(c(0.0,0.9,0.0),
                c(0.9,0.0,0.2),
                c(0.0,0.2,0.0))
adj_conv(Adj_mat)
```

`adj_mat` *Covert data matrix to adjacency matrix*

Description

Covert data matrix to adjacency matrix

Usage

```
adj_mat(X, method = "euclidian")
```

Arguments

`X`	a matrix of samples by features.
`method`	method for distance calculation: `"euclidian"`, `"cosine"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"`, `"minkowski"`,

Value

calculated adjacency matrix from the data matrix using the specified methods

Examples

```
X = gaussian_clusters()$X
Adj = adj_mat(X, method = "euclidian")
```

`CC_cluster_count` *Count the number of clusters based on stability score.*

Description

Count the number of clusters based on stability score.

Usage

```
CC_cluster_count(CM, plot.cdf = TRUE, plot.logit = FALSE)
```

Arguments

CM	list of consensus matrices each for a specific number of clusters. It can be the output of consensus_matrix() and multiview_consensus_matrix() functions.
plot.cdf	binary value to plot the cumulative distribution functions of CM (default TRUE).
plot.logit	binary value to plot the logit model of cumulative distribution functions of CM (default FALSE).

Details

Count the number of clusters given a list of consensus matrices each for a specific number of clusters. Using different methods: "LogitScore", "PAC", "deltaA", "CMavg"

Value

results as a list: "LogitScore", "PAC", "deltaA", "CMavg", "Kopt_LogitScore", "Kopt_PAC", "Kopt_deltaA", "Kopt_CMavg"

Examples

```
X = gaussian_clusters()$X
Adj = adj_mat(X, method = "euclidian")
CM = consensus_matrix(Adj, max.cluster=3, max.itter=10)
Result = CC_cluster_count(CM, plot.cdf=FALSE)
```

---

cluster_relabel *Relabeling clusters based on cluster similarities*

Description

Relabeling clusters based on cluster similarities

Usage

```
cluster_relabel(x1, x2)
```

Arguments

x1	clustering vector 1 Zero elements are are considered as unclustered samples
x2	clustering vector 2 Zero elements are are considered as unclustered samples

Details

When performing performing several clustering, the cluster labels may no match with each other. To perform maximum voting, the clustering need to be relabels based on label similarities.

Value

dataframe of relabeled clusters

Examples

```
X = gaussian_clusters()$X
x1 = kmeans(X, 5)$cluster
x2 = kmeans(X, 5)$cluster
clusters = cluster_relabel(x1, x2)
```

`coCluster_matrix` *Calculate the Co-cluster matrix for a given set of clustering results.*

Description

Calculate the Co-cluster matrix for a given set of clustering results.

Usage

```
coCluster_matrix(X, verbos = TRUE)
```

Arguments

`X`	clustering matrix of Nsamples x Nclusterings. Zero elements are are considered as unclustered samples
`verbos`	binary value for verbosity (default = `TRUE`)

Details

Co-cluster matrix or consensus matrix (CM) is a method for consensus mechanism explaned in Monti et al. (2003).

Value

The normalized matrix of Co-cluster frequency of any pairs of samples (Nsamples x Nsamples)

Examples

```
Clustering = cbind(c(1,1,1,2,2,2),
                   c(1,1,2,1,2,2))
coCluster_matrix(Clustering, verbos = FALSE)
```

connectivity_matrix *Build connectivity matrix*

Description

Build connectivity matrix

Usage

```
connectivity_matrix(clusters)
```

Arguments

clusters	a vector of clusterings. Zero elements mean that the sample was absent during clustering

Details

Connectivity matrix (M) is a binary matrix N-by-N M[i,j] = 1 if sample i and j are in the same cluster ref: Monti et al. (2003) "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data", Machine Learning

Value

Connectivity matrix

Examples

```
con_mat = connectivity_matrix(c(1,1,1,2,2,2))
```

consensus_matrix *Calculate consensus matrix for data perturbation consensus clustering*

Description

Calculate consensus matrix for data perturbation consensus clustering

Usage

```
consensus_matrix(
  X,
  max.cluster = 5,
  resample.ratio = 0.7,
  max.itter = 100,
  clustering.method = "hclust",
  adj.conv = TRUE,
  verbos = TRUE
)
```

Arguments

X	adjacency matrix a Nsample x Nsample
max.cluster	maximum number of clusters
resample.ratio	the data ratio to use at each itteration.
max.itter	maximum number of itterations at each max.cluster
clustering.method	base clustering method: c("hclust", "spectral", "pam")
adj.conv	binary value to apply soft thresholding (default=TRUE)
verbos	binary value for verbosity (default=TRUE)

Details

performs data perturbation consensus clustering and obtain consensus matrix Monti et al. (2003) consensus clustering algorithm

Value

list of consensus matrices for each k

Examples

```
X = gaussian_clusters()$X
Adj = adj_mat(X, method = "euclidian")
CM = consensus_matrix(Adj, max.cluster=3, max.itter=10, verbos = FALSE)
```

---

gaussian_clusters *Generate clusters of data points from Gaussian distribution with randomly generated parameters*

Description

Generate clusters of data points from Gaussian distribution with randomly generated parameters

Usage

```
gaussian_clusters(
  n = c(50, 50),
  dim = 2,
  sd.max = 0.1,
  sd.noise = 0.01,
  r.range = c(0.1, 1)
)
```

Arguments

n	vector of number of data points in each cluster The length of n should be equal to the number of clusters.
dim	number of dimensions
sd.max	maximum standard deviation of clusters
sd.noise	standard deviation of the added noise
r.range	the range (min, max) of distance of cluster centers from the origin

Value

a list of data points (X) and cluster labels (class)

Examples

```
data = gaussian_clusters()
X = data$X
y = data$class
```

---

gaussian_clusters_with_param

Generate clusters of data points from Gaussian distribution with given parameters

---

Description

Generate clusters of data points from Gaussian distribution with given parameters

Usage

```
gaussian_clusters_with_param(n, center, sigma)
```

Arguments

n	vector of number of data points in each cluster The length of n should be equal to the number of clusters.
center	matrix of centers Ncluster x dim
sigma	list of covariance matrices dim X dim. The length of sigma should be equal to the number of clusters.

Value

matrix of Nsamples x (dim + 1). The last column is cluster labels.

Examples

```
center = rbind(c(0,0),
               c(1,1))
sigma = list(diag(c(1,1)),
             diag(2,2))
gaussian_clusters_with_param(c(10, 10), center, sigma)
```

---

gaussian_mixture_clusters

Generate clusters of data points from Gaussian-mixture-model distributions with randomly generated parameters

---

Description

Generate clusters of data points from Gaussian-mixture-model distributions with randomly generated parameters

Usage

```
gaussian_mixture_clusters(
  n = c(50, 50),
  dim = 2,
  sd.max = 0.1,
  sd.noise = 0.01,
  r.range = c(0.1, 1),
  mixture.range = c(1, 4),
  mixture.sep = 0.5
)
```

Arguments

n	vector of number of data points in each cluster The length of n should be equal to the number of clusters.
dim	number of dimensions
sd.max	maximum standard deviation of clusters
sd.noise	standard deviation of the added noise
r.range	the range (min, max) of distance of cluster centers from the origin
mixture.range	range (min, max) of the number of Gaussian-mixtures.
mixture.sep	scaler indicating the separability between the mixtures.

Value

a list of data points (X) and cluster labels (class)

Examples

```
data = gaussian_mixture_clusters()
X = data$X
y = data$class
```

---

generate_gaussian_data

Generate a set of data points from Gaussian distribution

Description

Generate a set of data points from Gaussian distribution

Usage

```
generate_gaussian_data(n, center = 0, sigma = 1, label = NA)
```

Arguments

n	number of generated data points
center	data center of desired dimension
sigma	covariance matrix
label	cluster label

Value

Generated data points from Gaussian distribution with given parameters

Examples

```
generate_gaussian_data(10, center=c(0,0), sigma=diag(c(1,1)), label=1)
```

`hir_clust_from_adj_mat` *Hierarchical clustering from adjacency matrix*

Description

Hierarchical clustering from adjacency matrix

Usage

```
hir_clust_from_adj_mat(
  adj.mat,
  k = 2,
  alpha = 1,
  adj.conv = TRUE,
  method = "ward.D"
)
```

Arguments

`adj.mat`	adjacency matrix
`k`	number of clusters (default=2)
`alpha`	soft threshold (considered if `adj.conv = TRUE`) (default=1)
`adj.conv`	binary value to apply soft thresholding (default=TRUE)
`method`	distance method (default: `ward.D`)

Details

apply PAM (k-medoids) clustering on the adjacency matrix

Value

vector of clusters

Examples

```
Adj_mat = rbind(c(0.0,0.9,0.0),
                c(0.9,0.0,0.2),
                c(0.0,0.2,0.0))
hir_clust_from_adj_mat(Adj_mat)
```

indicator_matrix *Build indicator matrix*

Description

Build indicator matrix

Usage

```
indicator_matrix(clusters)
```

Arguments

clusters	a vector of clusterings. Zero elements mean that the sample was absent during clustering

Details

Indicator matrix (I) is a binary matrix N-by-N I[i,j] = 1 if sample i and j co-exist for clustering ref: Monti et al. (2003) "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data", Machine Learning

Value

Indicator matrix

Examples

```
ind_mat = indicator_matrix(c(1,1,1,0,0,1))
```

lebel_similarity *Similarity between different clusters*

Description

Similarity between different clusters

Usage

```
lebel_similarity(x1, x2)
```

Arguments

x1	clustering vector 1 Zero elements are are considered as unclustered samples
x2	clustering vector 2 Zero elements are are considered as unclustered samples

Details

When performing performing several clustering, the cluster labels may no match with each other. To find correspondences between clusters, the similarity between different labels need to be calculated.

Value

matrix of similarities between clustering labels

Examples

```
X = gaussian_clusters()$X
x1 = kmeans(X, 5)$cluster
x2 = kmeans(X, 5)$cluster
Sim = lebel_similarity(x1, x2)
```

Logit *Logit function*

Description

Logit function

Usage

```
Logit(x)
```

Arguments

x	numerical scaler input

Value

Logit(x) = log(1*x/(1-x))

Examples

```
y = Logit(0.5)
```

`majority_voting` *Consensus mechanism based on majority voting*

Description

Consensus mechanism based on majority voting

Usage

```
majority_voting(X)
```

Arguments

`X`	clustering matrix of Nsamples x Nclusterings. Zero elements are are considered as unclustered samples

Details

Perform majority voting as a consensus mechanism.

Value

the vector of consensus clustering result

Examples

```
X = gaussian_clusters()$X
x1 = kmeans(X, 5)$cluster
x2 = kmeans(X, 5)$cluster
x3 = kmeans(X, 5)$cluster
clusters = majority_voting(cbind(x1,x2,x3))
```

`multiview_clusters` *Generate multiview clusters from Gaussian distributions with randomly generated parameters*

Description

Generate multiview clusters from Gaussian distributions with randomly generated parameters

Usage

```
multiview_clusters(
  n = c(50, 50),
  hidden.dim = 2,
  observed.dim = c(2, 2, 3),
  sd.max = 0.1,
  sd.noise = 0.01,
  hidden.r.range = c(0.1, 1)
)
```

Arguments

n	vector of number of data points in each cluster The length of n should be equal to the number of clusters.
hidden.dim	scaler value of dimensions of the hidden state
observed.dim	vector of number of dimensions of the generate clusters. The length of observed.dim should be equal to the number of clusters.
sd.max	maximum standard deviation of clusters
sd.noise	standard deviation of the added noise
hidden.r.range	the range (min, max) of distance of cluster centers from the origin in the hidden space.

Value

a list of data points (X) and cluster labels (class)

Examples

```
data = multiview_clusters()
```

---

multiview_cluster_gen *Multiview cluster generation*

Description

Multiview cluster generation

Usage

```
multiview_cluster_gen(
  X,
  func,
  rep = 10,
  param,
  is.distance = FALSE,
  sample.set = NA
)
```

Arguments

X	List of input data matrices of Sample x feature or distance matrices. The length of X is equal to Nviews
func	custom function that accepts X and a parameter that return a vector of clusterings. cluster_func <- function(X, param)
rep	number of repeats
param	vector of parameters
is.distance	binary balue indicating if the input X[i] is distance
sample.set	vector of samples the clustering is being applied on. can be names or indices. if sample.set is NA, it considers all the datasets have the same samples with the same order

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then clustering is applied and result is returned.

Value

matrix of clusterings Nsample x (Nrepeat x Nviews)

Examples

```
data = multiview_clusters (n = c(40,40,40), hidden.dim = 2, observed.dim = c(2,2,2),
sd.max = .1, sd.noise = 0, hidden.r.range = c(.5,1))
X_observation = data[["observation"]]
cluster_func = function(X,rep,param){return(multi_kmeans_gen(X,rep=rep,range.k=param))}
Clusters = multiview_cluster_gen(X_observation, func = cluster_func, rep = 10, param = c(2,4))
```

---

multiview_consensus_matrix

Calculate consensus matrix for multi-data consensus clustering

---

Description

Calculate consensus matrix for multi-data consensus clustering

Usage

```
multiview_consensus_matrix(
  X,
  max.cluster = 5,
  sample.set = NA,
  clustering.method = "hclust",
  adj.conv = TRUE,
  verbos = TRUE
)
```

Arguments

X	list of adjacency matrices for different cohorts (or views).
max.cluster	maximum number of clusters
sample.set	vector of samples the clustering is being applied on. sample.set can be names or indices. if sample.set is NA, it considers that all the datasets have the same samples with the same order.
clustering.method	base clustering method: c("hclust", "spectral", "pam")
adj.conv	binary value to apply soft threshold (default=TRUE)
verbos	binary value for verbosity (default=TRUE)

Details

performs multi-data consensus clustering and obtain consensus matrix Monti et al. (2003) consensus clustering algorithm

Value

description list of consensus matrices for each k

Examples

```
data = multiview_clusters (n = c(40,40,40), hidden.dim = 2, observed.dim = c(2,2,2),
sd.max = .1, sd.noise = 0, hidden.r.range = c(.5,1))
X_observation = data[["observation"]]
Adj = list()
for (i in 1:length(X_observation))
  Adj[[i]] = adj_mat(X_observation[[i]], method = "euclidian")
CM = multiview_consensus_matrix(Adj, max.cluster = 4, verbos = FALSE)
```

---

multiview_kmeans_gen *Multiview K-means generation*

Description

Multiview K-means generation

Usage

```
multiview_kmeans_gen(X, rep = 10, range.k = c(2, 5), method = "random")
```

Arguments

X	List of input data matrices of Sample x feature. The length of X is equal to Nviews
rep	number of repeats
range.k	vector of minimum and maximum values for k c(min, max)
method	method for the choice of k at each repeat c("random", "silhouette")

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then k-means clustering is applied and result is returned.

Value

matrix of clusterings Nsample x (Nrepeat x Nviews)

Examples

```
data = multiview_clusters (n = c(40,40,40), hidden.dim = 2, observed.dim = c(2,2,2),
sd.max = .1, sd.noise = 0, hidden.r.range = c(.5,1))
X_observation = data[["observation"]]
Clusters = multiview_kmeans_gen(X_observation)
```

---

multiview_pam_gen *Multiview PAM (K-medoids) generation*

Description

Multiview PAM (K-medoids) generation

Usage

```
multiview_pam_gen(
  X,
  rep = 10,
  range.k = c(2, 5),
  is.distance = FALSE,
  method = "random",
  sample.set = NA
)
```

Arguments

X	List of input data matrices of Sample x feature or distance matrices. The length of X is equal to Nviews
rep	number of repeats
range.k	vector of minimum and maximum values for k c(min, max)
is.distance	binary balue indicating if the input X is distance
method	method for the choice of k at each repeat c("random", "silhouette")
sample.set	vector of samples the clustering is being applied on. can be names or indices. if sample.set is NA, it considers all the datasets have the same samples with the same order

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then PAM clustering is applied and result is returned.

Value

matrix of clusterings Nsample x (Nrepeat x Nviews)

Examples

```
data = multiview_clusters (n = c(40,40,40), hidden.dim = 2, observed.dim = c(2,2,2),
sd.max = .1, sd.noise = 0, hidden.r.range = c(.5,1))
X_observation = data[["observation"]]
Clusters = multiview_pam_gen(X_observation)
```

---

multi_cluster_gen *Multiple cluster generation*

Description

Multiple cluster generation

Usage

```
multi_cluster_gen(X, func, rep = 10, param, method = "random")
```

Arguments

X	input data Nsample x Nfeatures or a distance matrix
func	custom function that accepts X and a parameter that return a vector of clusterings. cluster_func <- function(X, param)
rep	number of repeats
param	vector of parameters
method	method for the choice of k at each repeat c("random", "silhouette")

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then clustering is applied and result is returned.

Value

matrix of clusterings Nsample x Nrepeat

Examples

```
X = gaussian_clusters()$X
cluster_func = function(X, k){return(stats::kmeans(X, k)$cluster)}
Clusters = multi_cluster_gen(X, cluster_func, param = c(2,3))
```

---

multi_kmeans_gen *Multiple K-means generation*

Description

Multiple K-means generation

Usage

```
multi_kmeans_gen(X, rep = 10, range.k = c(2, 5), method = "random")
```

Arguments

X	input data Nsample x Nfeatures
rep	number of repeats
range.k	vector of minimum and maximum values for k c(min, max)
method	method for the choice of k at each repeat c("random", "silhouette")

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then k-means clustering is applied and result is returned.

Value

matrix of clusterings Nsample x Nrepeat

Examples

```
X = gaussian_clusters()$X
Clusters = multi_kmeans_gen(X)
```

`multi_pam_gen` *Multiple PAM (K-medoids) generation*

Description

Multiple PAM (K-medoids) generation

Usage

```
multi_pam_gen(
  X,
  rep = 10,
  range.k = c(2, 5),
  is.distance = FALSE,
  method = "random"
)
```

Arguments

`X`	input data Nsample x Nfeatures or distance matrix.
`rep`	number of repeats
`range.k`	vector of minimum and maximum values for k `c(min, max)`
`is.distance`	binary balue indicating if the input `X` is distance
`method`	method for the choice of k at each repeat `c("random", "silhouette")`

Details

At each repeat, k is selected randomly or based on the best silhouette width from a discrete uniform distribution between range.k[1] and range.k[2]. Then PAM clustering is applied and result is returned.

Value

matrix of clusterings Nsample x Nrepeat

Examples

```
X = gaussian_clusters()$X
Clusters = multi_pam_gen(X)
```

pam_clust_from_adj_mat

PAM (k-medoids) clustering from adjacency matrix

Description

PAM (k-medoids) clustering from adjacency matrix

Usage

```
pam_clust_from_adj_mat(adj.mat, k = 2, alpha = 1, adj.conv = TRUE)
```

Arguments

adj.mat	adjacency matrix
k	number of clusters (default=2)
alpha	soft threshold (considered if adj.conv = TRUE) (default=1)
adj.conv	binary value to apply soft thresholding (default=TRUE)

Details

apply PAM (k-medoids) clustering on the adjacency matrix

Value

vector of clusters

Examples

```
Adj_mat = rbind(c(0.0,0.9,0.0),
                c(0.9,0.0,0.2),
                c(0.0,0.2,0.0))
pam_clust_from_adj_mat(Adj_mat)
```

spect_clust_from_adj_mat

Spectral clustering from adjacency matrix

Description

Spectral clustering from adjacency matrix

Usage

```
spect_clust_from_adj_mat(
  adj.mat,
  k = 2,
  max.eig = 10,
  alpha = 1,
  adj.conv = TRUE,
  do.plot = FALSE
)
```

Arguments

adj.mat	adjacency matrix
k	number of clusters (default=2)
max.eig	maximum number of eigenvectors in use (dafaut = 10).
alpha	soft threshold (considered if adj.conv = TRUE) (default = 1)
adj.conv	binary value to apply soft thresholding (default = TRUE)
do.plot	binary value to do plot (dafaut = FALSE)

Details

apply PAM (k-medoids) clustering on the adjacency matrix

Value

vector of clusters

Examples

```
Adj_mat = rbind(c(0.0,0.9,0.0),
                c(0.9,0.0,0.2),
                c(0.0,0.2,0.0))
hir_clust_from_adj_mat(Adj_mat)
```

Index