Package 'Evacluster'

January 20, 2025

Type Package

Title Evaluation Clustering Methods for Disease Subtypes Diagnosis

Version 0.1.0

Date 2022-03-25

Author Fahimeh Nezhadmoghadam,Jose Gerardo Tamez-Pena

Maintainer Fahimeh Nezhadmoghadam <f.nejad.moghadam@gmail.com>

Description Contains a set of clustering methods and evaluation metrics to select the best number of the clusters based on clustering stability. Two references describe the methodology: Fahimeh Nezhadmoghadam, and Jose Tamez-Pena (2021)<doi:10.1016/j.compbiomed.2021.104753>, and Fahimeh Nezhadmoghadam, et al.(2021)<doi:10.2174/1567205018666210831145825>.

License LGPL (>= 2)

Encoding UTF-8

Suggests mlbench,EMCluster,inaparc,ppclust,FRESA.CAD,MASS,stats,class,NMF,cluster,Rtsne,proxy,uwot,mclust,graphics

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-01 07:50:07 UTC

Contents

`Evacluster-package` *Evaluation Clustering Methods for Disease Subtypes Diagnosis (Evacluster)*

Description

Contains a set of clustering methods and evaluation metrics to select the best number of the clusters based on clustering stability.

Details

Package:	Evacluster
Type:	Package
Version:	0.1.0
Date:	2022-03-25
License:	LGPL (>= 2)

Purpose: The design of clustering models and evaluation metrics for finding the cluster's number via computing clustering stability. The best number of clusters is selected via consensus clustering and clustering stability.

Author(s)

Fahimeh Nezhadmoghadam, Jose Gerardo Tamez-Pena, Maintainer: <f.nejad.moghadam@gmail.com>

References

Nezhadmoghadam, Fahimeh, and Jose Tamez-Pena. "Risk profiles for negative and positive COVID-19 hospitalized patients.(2021) *Computers in biology and medicine* 136 : 104753.
Fahimeh Nezhadmoghadam, et al., Robust Discovery of Mild Cognitive impairment subtypes and their Risk of Alzheimer's Disease conversion using unsupervised machine learning and Gaussian Mixture Modeling (2021), *Current Alzheimer Research*, 18 (7), 595-606.

Examples

```
 ## Not run:
 ### Evacluster Package Examples ####
 library(datasets)
 data(iris)

# Split data to training set and testing set
rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]


## Expectation Maximization Clustering
# Perform Expectation Maximization Clustering on training set with 3 clusters
clsut <- EMCluster(trainData[,1:4],3)

# Predict the labels of the cluster for new data based on cluster labels of the training set
pre <- predict(clsut,testData[,1:4])


## Fuzzy C-means Clustering
# Perform Fuzzy C-means Clustering on training set with 3 clusters
clsut <- FuzzyCluster(trainData[,1:4],3)

# Predict the labels of the new data
pre <- predict(clsut,testData[,1:4])


## hierarchical clustering
# Perform hierarchical clustering on training set with 3 clusters
clsut <- hierarchicalCluster(trainData[,1:4],distmethod="euclidean",clusters=3)

# Predict the labels of the new data
pre <- predict(clsut,testData[,1:4])


## K-means Clustering
# Perform K-means Clustering on training set with 3 clusters
clsut <- kmeansCluster(trainData[,1:4],3)

# Predict the labels of the new data
pre <- predict(clsut,testData[,1:4])


## Partitioning Around Medoids (PAM) Clustering
# Perform pam Clustering on training set with 3 clusters
clsut <- pamCluster(trainData[,1:4],3)

# Predict the labels of the new data
pre <- predict(clsut,testData[,1:4])
```

```
## Non-negative matrix factorization (NMF)
# Perform nmf Clustering on training set with 3 clusters
clsut <- nmfCluster(trainData[,1:4],rank=3)

# Predict the labels of the new data
pre <- predict(clsut,testData[,1:4])


## t-Distributed Stochastic Neighbor Embedding (t-SNE)

library(mlbench)
data(Sonar)

rndSamples <- sample(nrow(Sonar),150)
trainData <- Sonar[rndSamples,]
testData <- Sonar[-rndSamples,]

# Perform tSNE dimensionality reduction method on training data
tsne_trainData <- tsneReductor(trainData[,1:60],dim = 3,perplexity = 10,max_iter = 1000)

# performs an embedding of new data using an existing embedding
tsne_testData <- predict(tsne_trainData,k=3,testData[,1:60])


## clustering stability function
# Compute the stability of clustering to select the best number of clusters.
library(mlbench)
data(Sonar)

Sonar$Class <- as.numeric(Sonar$Class)
Sonar$Class[Sonar$Class == 1] <- 0
Sonar$Class[Sonar$Class == 2] <- 1

# Compute the stability of clustering using kmeans clustering, UMAP as
dimensionality reduction method, and feature selection technique

ClustStab <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                              n_components = 3,featureselection="yes", outcome="Class",
                        fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=3)


# Get the labels of the subjects that share the same connectivity
clusterLabels <- getConsensusCluster(ClustStab,who="training",thr=seq(0.80,0.30,-0.1))


  # Compute the stability of clustering using PAM clustering, tSNE as
  dimensionality reduction method, and feature selection technique

ClustStab <- clusterStability(data=Sonar, clustermethod=pamCluster, dimenreducmethod="tSNE",
                              n_components = 3, perplexity=10,max_iter=100,k_neighbor=2,
                             featureselection="yes", outcome="Class",fs.pvalue = 0.05,
                               randomTests = 100,trainFraction = 0.7,k=3)
```

```
  # Get the labels of the subjects that share the same connectivity
clusterLabels <- getConsensusCluster(ClustStab,who="training",thr=seq(0.80,0.30,-0.1))


  # Compute the stability of clustering using hierarchical clustering,
  PCA as dimensionality reduction method, and without applying feature selection

 ClustStab <- clusterStability(data=Sonar, clustermethod=hierarchicalCluster,
                       dimenreducmethod="PCA", n_components = 3,featureselection="no",
                         randomTests = 100,trainFraction = 0.7,distmethod="euclidean",
                             clusters=3)

# Get the labels of the subjects that share the same connectivity
 clusterLabels <- getConsensusCluster(ClustStab,who="training",thr=seq(0.80,0.30,-0.1))


  # Show the clustering stability resuldts
  mycolors <- c("red","green","blue","yellow")

  ordermatrix <- ClustStab$dataConcensus

  heatmapsubsample <- sample(nrow(ordermatrix),70)

  orderindex <- 10*clusterLabels + ClustStab$trainJaccardpoint

  orderindex <- orderindex[heatmapsubsample]
  orderindex <- order(orderindex)
  ordermatrix <- ordermatrix[heatmapsubsample,heatmapsubsample]
  ordermatrix <- ordermatrix[orderindex,orderindex]
  rowcolors <- mycolors[1+clusterLabels[heatmapsubsample]]
  rowcolors <- rowcolors[orderindex]


  hplot <- gplots::heatmap.2(as.matrix(ordermatrix),Rowv=FALSE,Colv=FALSE,
                  RowSideColors = rowcolors,ColSideColors = rowcolors,dendrogram = "none",
                           trace="none",main="Cluster Co-Association \n (k=3)")


  # Compare the PAC values of clustering stability with different numbers of clusters

 ClustStab2 <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                             n_components = 3,featureselection="yes", outcome="Class",
                         fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=2)

 ClustStab3 <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                              n_components = 3,featureselection="yes", outcome="Class",
                          fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=3)

 ClustStab4 <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                              n_components = 3,featureselection="yes", outcome="Class",
                          fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=4)
```

```
color_range<- c(black="#FDFC74", orange="#76FF7A", skyblue="#B2EC5D")


max.temp <- c(ClustStab2$PAC,ClustStab3$PAC,ClustStab4$PAC)

barplot(max.temp,xlab = "Number of clusters",ylab = "PAC", names.arg = c( "2","3","4"),
        ylim=c(0,0.3),col= color_range[1:length(c(1,6,2,6,1))])


## End(Not run)
```

---

clusterStability *clustering stability function*

Description

This function computes the stability of clustering that helps to select the best number of clusters. Feature selection and dimensionality reduction methods can be used before clustering the data.

Usage

```
clusterStability(
  data = NULL,
  clustermethod = NULL,
  dimenreducmethod = NULL,
  n_components = 3,
  perplexity = 25,
  max_iter = 1000,
  k_neighbor = 3,
  featureselection = NULL,
  outcome = NULL,
  fs.pvalue = 0.05,
  randomTests = 20,
  trainFraction = 0.5,
  pac.thr = 0.1,
  ...
)
```

Arguments

data	A Data set
clustermethod	The clustering method. This can be one of "Mclust","pamCluster","kmeansCluster", "hierarchicalCluster",and "FuzzyCluster".
dimenreducmethod	The dimensionality reduction method. This must be one of "UMAP","tSNE", and "PCA".
n_components	The dimension of the space that data embed into. It can be set to any integer value in the range of 2 to 100.

perplexity	The Perplexity parameter that determines the optimal number of neighbors in tSNE method.(it is only used in the tSNE reduction method)
max_iter	The maximum number of iterations for performing tSNE reduction method.
k_neighbor	The k_neighbor is used for computing the means of #neighbors with min distance (#Neighbor=sqrt(#Samples/k) for performing an embedding of new data using an existing embedding in the tSNE method.
featureselection	This parameter determines whether feature selection is applied before clustering data or not. if used, it should be "yes", otherwisw "no".
outcome	The outcome feature is used for feature selection.
fs.pvalue	The threshold pvalue used for feature selection process. The default value is 0.05.
randomTests	The number of iterations of the clustering process for computing the cluster stability.
trainFraction	This parameter determines the ratio of training data. The default value is 0.5.
pac.thr	The pac.thr is the thresold to use for computing the proportion of ambiguous clustering (PAC) score. It is as the fraction of sample pairs with consensus indices falling in the interval.The default value is 0.1.
...	Additional arguments passed to clusterStability().

Value

A list with the following elements:

- randIndex - A vector of the Rand Index that computes a similarity measure between two clusterings.
- jaccIndex - A vector of jaccard Index that measures how frequently pairs of items are joined together in two clustering data sets.
- randomSamples - A vector with indexes of selected samples for training in each iteration.
- clusterLabels - A vector with clusters' labels in all iterations. jaccardpoint
- jaccardpoint - The corresponding Jaccard index for each data point of testing set
- averageNumberofClusters - The mean Number of Clusters.
- testConsesus - A vector of consensus clustering results of testing set.
- trainRandIndex - A vector of the Rand Index for training set.
- trainJaccIndex - A vector of the jaccard Index for training set.
- trainJaccardpoint - The corresponding Jaccard index for each data point of training set.
- PAC - The proportion of ambiguous clustering (PAC) score.
- dataConcensus - A vector of consensus clustering results of training set.

Examples

```
library("mlbench")
data(Sonar)

Sonar$Class <- as.numeric(Sonar$Class)
Sonar$Class[Sonar$Class == 1] <- 0
Sonar$Class[Sonar$Class == 2] <- 1

ClustStab <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                              n_components = 3,featureselection="yes", outcome="Class",
                         fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=3)


ClustStab <- clusterStability(data=Sonar, clustermethod=pamCluster, dimenreducmethod="tSNE",
                              n_components = 3, perplexity=10,max_iter=100,k_neighbor=2,
                              featureselection="yes", outcome="Class",fs.pvalue = 0.05,
                              randomTests = 100,trainFraction = 0.7,k=3)


ClustStab <- clusterStability(data=Sonar, clustermethod=hierarchicalCluster,
                           dimenreducmethod="PCA", n_components = 3,featureselection="no",
                             randomTests = 100,trainFraction = 0.7,distmethod="euclidean",
                              clusters=3)
```

`EMCluster` *Expectation Maximization Clustering*

Description

This function perform EM algorithm for model-based clustering of finite mixture multivariate Gaussian distribution.The general purpose of clustering is to detect clusters of data and to assign the data to the clusters.

Usage

```
EMCluster(data = NULL, ...)
```

Arguments

`data`	A Data set
`...`	k: The number of Clusters

Value

A list of cluster labels and a returned object from init.EM

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

clsut <- EMCluster(trainData[,1:4],3)
```

`FuzzyCluster` *Fuzzy C-means Clustering Algorithm*

Description

This function works by assigning membership to each data point corresponding to each cluster center based on the distance between the cluster center and the data point. A data object is the member of all clusters with varying degrees of fuzzy membership between 0 and 1.

Usage

```
FuzzyCluster(data = NULL, ...)
```

Arguments

`data`	A Data set
`...`	k: The number of Clusters

Value

A list of cluster labels and a R object of class "fcm ppclust"

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

cls <- FuzzyCluster(trainData[,1:4],3)
```

`getConsensusCluster` *Consensus Clustering Results*

Description

This function gets the labels of the subjects that share the same connectivity.

Usage

```
getConsensusCluster(object, who = "training", thr = seq(0.8, 0.3, -0.1))
```

Arguments

`object`	A object of "clusterStability" function result
`who`	This value shows the consensus clustering result of training and testing sets. If who="training" for training set, otherwise other sets.
`thr`	This is the seq function with three arguments that are: initial value, final value, and increment (or decrement for a declining sequence). This produces ascending or descending sequences.

Value

A list of samples' labels with same connectivity.

Examples

```
library("mlbench")
data(Sonar)

Sonar$Class <- as.numeric(Sonar$Class)
Sonar$Class[Sonar$Class == 1] <- 0
Sonar$Class[Sonar$Class == 2] <- 1

ClustStab <- clusterStability(data=Sonar, clustermethod=kmeansCluster, dimenreducmethod="UMAP",
                              n_components = 3,featureselection="yes", outcome="Class",
                          fs.pvalue = 0.05,randomTests = 100,trainFraction = 0.7,center=3)

clusterLabels <- getConsensusCluster(ClustStab,who="training",thr=seq(0.80,0.30,-0.1))
```

hierarchicalCluster *hierarchical clustering*

Description

This function seeks to build a hierarchy of clusters

Usage

```
hierarchicalCluster(data = NULL, distmethod = NULL, clusters = NULL, ...)
```

Arguments

data	A Data set
distmethod	The distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
clusters	The number of Clusters
...	Additional parameters passed to hclust function

Value

A list of cluster labels

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

cls <- hierarchicalCluster(trainData[,1:4],distmethod="euclidean",clusters=3)
```

kmeansCluster *K-means Clustering*

Description

This function classifies unlabeled data by grouping them by features, rather than pre-defined categories. It splits the data into K different clusters and describes the location of the center of each cluster. Then, a new data point can be assigned a cluster (class) based on the closed center of mass.

Usage

```
kmeansCluster(data = NULL, ...)
```

Arguments

data	A Data set
...	center: The number of centers

Value

A list of cluster labels and a R object of class "kmeans"

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

cls <- kmeansCluster(trainData[,1:4],3)
```

---

nmfCluster *Non-negative matrix factorization (NMF)*

Description

This function factorizes samples matrix into (usually) two matrices W the cluster centroids and H the cluster membership,

Usage

```
nmfCluster(data = NULL, rank = NULL)
```

Arguments

data	A Data set
rank	Specification of the factorization rank

Value

A list of cluster labels, a R object of class "nmf" and the centers of the clusters

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

cls <- nmfCluster(trainData[,1:4],rank=3)
```

pamCluster *Partitioning Around Medoids (PAM) Clustering*

Description

This function partitions (clustering) of the data into k clusters "around medoids". In contrast to the k-means algorithm, this clustering methods chooses actual data points as centers

Usage

```
pamCluster(data = NULL, ...)
```

Arguments

data	A Data set
...	k: The number of clusters

Value

A list of cluster labels and a R object of class "pam cluster"

Examples

```
library(datasets)
data(iris)

rndSamples <- sample(nrow(iris),100)
trainData <- iris[rndSamples,]
testData <- iris[-rndSamples,]

cls <- pamCluster(trainData[,1:4],3)
```

predict.EMCluster *EMCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'EMCluster'
predict(object, ...)
```

Arguments

object	A returned object of EMCluster
...	New sample set

Value

A list of cluster labels

---

predict.FuzzyCluster *FuzzyCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'FuzzyCluster'
predict(object, ...)
```

Arguments

object	A returned object of FuzzyCluster function
...	New samples set

Value

A list of cluster labels

---

predict.hierarchicalCluster *hierarchicalCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'hierarchicalCluster'
predict(object, ...)
```

Arguments

object	A returned object of hierarchicalCluster function
...	New samples set

Value

A list of cluster labels

---

predict.kmeansCluster *kmeansCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'kmeansCluster'
predict(object, ...)
```

Arguments

object	A returned object of kmeansCluster function
...	New samples set

Value

A list of cluster labels

---

predict.nmfCluster *nmfCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'nmfCluster'
predict(object, ...)
```

Arguments

object	A returned object of nmfCluster
...	New samples set

Value

A list of cluster labels

---

predict.pamCluster *pamCluster prediction function*

Description

This function predicts the labels of the cluster for new data based on cluster labels of the training set.

Usage

```
## S3 method for class 'pamCluster'
predict(object, ...)
```

Arguments

object	A returned object of pamCluster function
...	New samples set

Value

A list of cluster labels

---

predict.tsneReductor *tsneReductor prediction function*

Description

This function performs an embedding of new data using an existing embedding.

Usage

```
## S3 method for class 'tsneReductor'
predict(object, k = NULL, ...)
```

Arguments

object	A returned object of tsneReductor function
k	The number is used for computing the means of #neighbors with min distance (#Neighbor=sqrt(#Samples/k).
...	New samples set

Value

tsneY:An embedding of new data

Examples

```
library("mlbench")
data(Sonar)

rndSamples <- sample(nrow(Sonar),150)
trainData <- Sonar[rndSamples,]
testData <- Sonar[-rndSamples,]

tsne_trainData <- tsneReductor(trainData[,1:60],dim = 3,perplexity = 10,max_iter = 1000)

tsne_testData <- predict(tsne_trainData,k=3,testData[,1:60])
```

---

tsneReductor *t-Distributed Stochastic Neighbor Embedding (t-SNE)*

Description

This method is an unsupervised, non-linear technique used for data exploration and visualizing high-dimensional data.This function constructs a low-dimensional embedding of high-dimensional data, distances, or similarities.

Usage

```
tsneReductor(data = NULL, dim = 2, perplexity = 30, max_iter = 500)
```

Arguments

data	Data matrix (each row is an observation, each column is a variable)
dim	Integer number; Output dimensional (default=2)
perplexity	numeric; Perplexity parameter (should not be bigger than 3 * perplexity < nrow(X) - 1, default=30)
max_iter	Integer; Number of iterations (default: 500)

Value

tsneY: A Matrix containing the new representations for the observation with selected dimensions by user

Examples

```
library("mlbench")
data(Sonar)

rndSamples <- sample(nrow(Sonar),150)
trainData <- Sonar[rndSamples,]
testData <- Sonar[-rndSamples,]

tsne_trainData <- tsneReductor(trainData[,1:60],dim = 3,perplexity = 10,max_iter = 1000)
```

Index