Package 'mstknnclust'

January 27, 2023

Type Package

Title MST-kNN Clustering Algorithm

Version 0.3.2

Date 2023-01-23

Author Jorge Parraga-Alava [aut, cre],
Pablo Moscato [aut],
Mario Inostroza-Ponta [aut]

Maintainer Jorge Parraga-Alava <jorge.parraga@usach.cl>

Description Implements the MST-kNN clustering algorithm which was proposed by Inostroza-Ponta, M. (2008) <https://trove.nla.gov.au/work/28729389?selectedversion=NBD44634158>.

Depends R (>= 3.2.5)

License GPL-2

Encoding UTF-8

Imports igraph, stats, base

RoxygenNote 7.1.1

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2023-01-27 14:10:02 UTC

R topics documented:

dslanguages *Indo-European languages dataset*

Description

It contains the distances between 84 Indo-European languages based on the mean percent difference in cognacy, using the 200 Swadesh words.

Usage

```
data(dslanguages)
```

Format

An data frame with 84 rows and 84 columns containing a distance matrix.

Details

Once the data set is loaded, it can be accessed as an object of class dataframe called `dslanguages`.

References

Dyen, I., Kruskal, J., and Black, P. (1992). An indoeuropean classification: A lexicostatistical experiment. Transactions of the American Philosophical Society. 82, (5).

dsyeastexpression *Budding Yeast dataset*

Description

It contains the expression levels of 2467 genes on 79 samples corresponding to 8 different experiments of the budding yeast: alpha factor (18 samples), cdc15 (15 samples), cold shock (4 samples), diauxic shift (7 samples), DTT shock (4 samples), elutriation (14 samples), heat shock (6 samples) and sporulation (11 samples).

Usage

```
data(dsyeastexpression)
```

Format

An data frame with 2467 rows and 79 columns.

Details

Once the data set is loaded, it can be accessed as an object of class dataframe called `dsyeastexpression`.

Source

https://www.pnas.org/doi/10.1073/pnas.95.25.14863

References

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. (1998). Cluster analysis and display of genome-wideexpression patterns.Proceedings of the National Academy of Sciences, 95(25):14863–14868

`generate.knn` *Generates a kNN graph*

Description

This function generates the *k*-Nearest Neighbors (kNN) graph which is a subgraph contains edges between nodes if, and only if, they are one of the *k* nearest neighbors considering the edges costs (distances). Each node represents an object of the complete graph.

Usage

```
generate.knn(edges.complete.graph, suggested.k)
```

Arguments

`edges.complete.graph`
: A object of class "data.frame" with three columns (*object_i*, *object_j*, *d_ij*) representing the distance *d_ij* between *object_i* and *object_j*.

`suggested.k`
: It is an optional argument. A numeric value representing the suggested number of *k*-nearest neighbors to consider to generate the *k*NN graph.

Details

During its generation, the *k* value is automatically determined by the definition:

$$k = min\lfloor \ln(|nodes.list|)\rfloor; mink|kNNisconnected; suggested.k$$

If *suggested.k* parameter is not provided, it is not considered by the definition.

Value

A list with the elements

`edges.knn.graph`
: A object of class "data.frame" with three columns (*object_i*, *object_j*, *d_ij*) representing the *d_ij* between *object_i* and *object_j* that are part of the *k*NN graph.

`knn.graph`
: A object of class "igraph" which is the *k*-Nearest Neighbors (kNN) graph generated.

`k`
: The *k* value determined by the definition.

Author(s)

Mario Inostroza-Ponta, Jorge Parraga-Alava, Pablo Moscato

Examples

```
set.seed(1987)

##Generates a data matrix of dimension 50X13
n=50; m=13
x <- matrix(runif(n*m, min = -5, max = 10), nrow=n, ncol=m)

##Computes a distance matrix of x.

library("stats")
d <- base::as.matrix(stats::dist(x, method="euclidean"))

##Generates complete graph (CG) without suggested.k parameter

cg <- generate.complete.graph(1:nrow(x),d)

##Generates kNN graph
knn <- generate.knn(cg)

##Visualizing kNN graph
plot(knn$knn.graph,
main=paste("kNN \n k=", knn$k, sep=""))



##Generates complete graph (CG) with suggested.k parameter

cg <- generate.complete.graph(1:nrow(x),d)

##Generates kNN graph
knn <- generate.knn(cg, suggested.k=4)

##Visualizing kNN graph
plot(knn$knn.graph,
main=paste("kNN \n k=", knn$k, sep=""))
```

`generate.mst` *Generates a MST graph*

Description

This function generates the Minimal Spanning Tree (MST) graph which is a connected and acyclic subgraph contains all the nodes of the complete graph (CG) and whose edges sum (distances) has minimum costs. Each node represents an object of the complete graph.

Usage

```
generate.mst(edges.complete.graph)
```

Arguments

edges.complete.graph	A object of class "data.frame" with three columns (*object_i*, *object_j*, *d_ij*) representing the distance *d_ij* between *object i* and *object j* of the complete graph.

Details

Generation of MST graph is performed using the Prim's algorithm.

Value

A list with the elements

edges.mst.graph	A object of class "data.frame" with three columns (*object_i*, *object_j*, *d_ij*) representing the distance *d_ij* between object *i* and object *j* that are part of the MST graph.
mst.graph	A object of class "igraph" which is the Minimal Spanning Tree (MST) graph generated.

Author(s)

Mario Inostroza-Ponta, Jorge Parraga-Alava, Pablo Moscato

References

Prim, R.C. (1957). *Shortest connection networks and some generalizations*. Bell System Technical Journal, 37 1389-1401.

Ignatenkov, E. (2015). *Minimum Spanning Tree (MST) for some graph using Prim's MST algorithm*. Stanford University course on Coursera.

Examples

```
set.seed(1987)

##Generates a data matrix of dimension 50X13
n=50; m=13
x <- matrix(runif(n*m, min = -5, max = 10), nrow=n, ncol=m)

##Computes a distance matrix of x.

library("stats")
d <- base::as.matrix(stats::dist(x, method="euclidean"))

##Generates complete graph (CG)
```

```
cg <- generate.complete.graph(1:nrow(x),d)

##Generates MST graph

mstree <- generate.mst(cg)

##Visualizing MST graph
plot(mstree$mst.graph, main="MST")
```

`mst.knn` *Performs the MST-kNN clustering algorithm*

Description

Performs the MST-kNN clustering algorithm which generates a clustering solution with automatic *number of clusters* determination using two proximity graphs: Minimal Spanning Tree (MST) and k-Nearest Neighbor (*k*NN) which are recursively intersected.

To create MST, *Prim* algorithm is used. To create *k*NN, `distance.matrix` passed as input is considered.

Usage

```
mst.knn(distance.matrix, suggested.k)
```

Arguments

`distance.matrix`	A numeric matrix or data.frame with equals numbers of rows and columns representing distances between objects to group.
`suggested.k`	It is an optional argument. A numeric value representing the suggested number of k-nearest neighbors to consider during the generating the kNN graph. Note that, due to the algorithm operation, this number may be different during the algorithm execution.

Details

To see more details of how MST-kNN works refers to the quick guide.

Value

A list with the elements

`cnumber`	A numeric value representing the number of clusters of the solution.
`cluster`	A named vector of integers from `1:cnumber` representing the cluster to which each object is assigned.

partition	A partition matrix order by cluster where are shown the objects and the cluster where they are assigned.
csize	A vector with the cardinality of each cluster in the solution.
network	An object of class "igraph" as a network representing the clustering solution.

Author(s)

Mario Inostroza-Ponta, Jorge Parraga-Alava, Pablo Moscato

References

Inostroza-Ponta, M. (2008). *An Integrated and Scalable Approach Based on Combinatorial Optimization Techniques for the Analysis of Microarray Data*. Ph.D. thesis, School of Electrical Engineering and Computer Science. University of Newcastle.

Examples

```
set.seed(1987)

##load package
library("mstknnclust")

##Generates a data matrix of dimension 100X15

n=100; m=15

x <- matrix(runif(n*m, min = -5, max = 10), nrow=n, ncol=m)

##Computes a distance matrix of x.

library("stats")
d <- base::as.matrix(stats::dist(x, method="euclidean"))

##Performs MST-kNN clustering using euclidean distance.

results <- mst.knn(d)

## Visualizes the clustering solution

library("igraph")
plot(results$network, vertex.size=8,
     vertex.color=igraph::clusters(results$network)$membership,
     layout=igraph::layout.fruchterman.reingold(results$network, niter=10000),
   main=paste("MST-kNN \n Clustering solution \n Number of clusters=",results$cnumber,sep="" ))
```

Index