Package 'mstknnclust'

December 5, 2019

Type Package

Title MST-kNN Clustering Algorithm

Version 0.2.0

Date 2019-12-05

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, amap

RoxygenNote 6.1.1

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-05 20:40:02 UTC

R topics documented:

`compute.costs.proximity.graph`
Computes the edge costs sum of a proximity graph

Description

This function computes the edge costs overall sum of a proximity graph.

Usage

```
compute.costs.proximity.graph(graph.edges, distance.matrix)
```

Arguments

`graph.edges`	A object of class "matrix" with two columns (*from*, *to*) representing the edges costs of a proximity graph.
`distance.matrix`	A distance matrix between each pair of nodes in the proximity graph.

Value

`total.costs.graph`
A numeric value representing the edge costs overall sum of a proximity graph.

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("amap")
d <- base::as.matrix(amap::Dist(x, method="euclidean"))

##Generates complete graph (CG)

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

##Generates a proximity graph (MST)
mstree <- generate.mst(cg)

##Calculate the edge cost sum of proximity graph (MST)
mstree.cost=as.numeric(compute.costs.proximity.graph(as.matrix(mstree$edges.mst.graph[,1:2]), d))
mstree.cost
```

```
##Generates a proximity graph (kNN)
knneig <- generate.knn(cg)

##Calculate the edge cost sum of proximity graph (kNN)
knneig.cost=as.numeric(compute.costs.proximity.graph(as.matrix(knneig$edges.knn.graph[,1:2]), d))
knneig.cost
```

`dslanguages` *Indo-European languages dataset of class 'distance matrix'*

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 of type distance matrix with 84 rows and columns.

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 of gene expression level*

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 of type distance matrix 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

http://www.pnas.org/content/suppl/1998/12/08/95.25.14863.DC1/3917data.xls

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.complete.graph` *Generates a complete graph*

Description

This function generates a complete graph assigning costs to the edges according to `distance.matrix`.

Usage

```
generate.complete.graph(nodes.list, distance.matrix)
```

Arguments

`nodes.list`	A vector with a subset of objects (nodes) of the data matrix for which the complete graph must be generated.
`distance.matrix`	A distance matrix between each pair of elements in `nodes.list`. It is used as the edges costs to generate the complete graph.

Value

`edges.complete.graph`

A object of class "data.frame" with three columns (*from*, *to*, *costs*) representing the edges costs of complete graph. For instance:

from	*to*	*costs*
1	2	1.60
1	3	0.08
1	4	1.21
...	...	...
n-1	n	...

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("amap")
d <- base::as.matrix(amap::Dist(x, method="euclidean"))

##Generates complete graph (CG)

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

head(cg)

##Visualizing CG graph
library("igraph")
cg.network=igraph::graph.adjacency(d, mode="undirected", weighted=TRUE)
plot(cg.network, edge.label=round(E(cg.network)$weight, 2), main="Complete Graph")
```

---

generate.intersections.mst.knn

Performs the intersections between MST y kNN graphs

Description

This function performs a graph partition based on the intersection of the edges of two proximity graphs: MST and kNN.

Usage

```
generate.intersections.mst.knn(nodes.list, distance.matrix, k.user)
```

Arguments

nodes.list	A vector with a subset of objects (nodes) of the data matrix for which the MST y kNN graphs must be generated.

`distance.matrix`
: A distance matrix between each pair of elements in `nodes.list`. It is used as the edges costs to generate MST y kNN graphs.

`k.user`
: A numeric value representing the number of nearest neighbor to consider to generate the *k*NN graph.

Value

A list with the elements

`cc`
: A numeric value representing the number of connected components (cc) generated after graphs intersection.

`subgraphs`
: A list where each item contains the nodes of the connected components (cc) generated.

`ccgraph`
: A object of class "igraph" which is a network with each connected components (cc) generated.

Author(s)

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

`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.

Usage

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

Arguments

`edges.complete.graph`
: A object of class "data.frame" with three columns (*from*, *to*, *costs*) representing the edges costs of a complete graph.

`k.user`
: A numeric value representing the number of nearest neighbor to consider to generate the *k*NN graph.

Details

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

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

If *k.user* 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 (*from*, *to*, *costs*) representing the edges costs forming the kNN 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("amap")
d <- base::as.matrix(amap::Dist(x, method="euclidean"))

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

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

##Generates kNN graph
knn <- generate.knn(cg, k.user=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 has minimum costs.

Usage

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

Arguments

`edges.complete.graph`
: A object of class "data.frame" with three columns (*from*, *to*, *costs*) representing the edges costs of a 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 (*from*, *to*, *costs*) representing the edges costs forming 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("amap")
d <- base::as.matrix(amap::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 generate a clustering solution with automatic *k* 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, k.user)
```

Arguments

`distance.matrix`	A numeric matrix or data.frame with equals numbers of rows and columns representing objects to group.
`k.user`	A numeric value representing the number of cluster to yield. Note that, due to the algorithm operation, this number may be different at the end of the algorithm execution.

Details

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

Value

A list with the elements

`k`	Number of cluster of the solution.
`cluster`	A named vector of integers from `1:k` 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("amap")
d <- base::as.matrix(amap::Dist(x, method="euclidean"))

##Performs MST-kNN clustering using euclidean distance and automatic k determination.

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 k=",results$k,sep="" ))
```

Index