Package ‘easyDist’

October 31, 2023

Title Creating, Manipulating, and Expanding ``dist'' Objects

Version 0.0.1

Author Minh Long Nguyen <edelweiss611428@gmail.com>

Maintainer Minh Long Nguyen <edelweiss611428@gmail.com>

Description The package provides functions for creating, manipulating, and expanding ``dist'' objects, which are commonly used in cluster analysis in R.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo Rcpp

Imports cluster,
microbenchmark,
Rcpp

R topics documented:

`extractDist` *Extract pair-wises distances of units in two groups from a "dist" object*

Description

This function allows us to extract pair-wise distances of units in two groups, specified by their indexes, from a "dist" object

Usage

```
extractDist(dist, idxGroup1 = NULL, idxGroup2 = NULL)
```

Arguments

`dist`	A "dist" object, which can be obtained with the "dist" function.
`idxGroup1`	An integer vector specifying the indexes of units the FIRST group. If idxGroup1 is not NULL, indexes can't be smaller than 1 or larger the dataset size N. By default, idxGroup1 = NULL.
`idxGroup2`	An integer vector specifying the indexes of units the SECOND group. If idxGroup2 is not NULL, indexes can't be smaller than 1 or larger the dataset size N. By default, idxGroup2 = NULL.

Details

Extracting pair-wises distances between units in two groups from a "dist" object may be of interest. However, we can't use the bracket operator directly to extract rows and columns from the "dist" object as we do with numeric matrices. A simple way to do that involves converting the "dist" object to a symmetric numeric matrix using the as.matrix function. However, it is extremely inefficient and slow as we only partially extract the "dist" object. The function allows us to extract pair-wise distances without the need of conversion.

When either idxGroup1 or idxGroup2 is NULL, the function extracts the entire "columns" specified by the not-null vector. Since the distance matrix is symmetric. It does not matter mathematically if we extract the rows or the columns. However, our implementation is more efficient for extracting "columns" from a "dist" object. If idxGroup1 and idxGroup2 are not specified (NULL), the "dist" object is fully converted to a numeric matrix.

Value

A numeric matrix storing pair-wise distances between the units in each subset.

Author(s)

Minh Long Nguyen <`edelweiss611428@gmail.com`>

Examples

```
x = rnorm(50)
dx = dist(x) #Euclidean distance matrix of class "dist"
#Extract the pairwise distances between the first unit and the other units.
extractDist(dx, idxGroup1 = 1)

library("microbenchmark")
x = rnorm(100)
dx = dist(x) #Euclidean distance matrix of class "dist"
microbenchmark(extractDist(dx, idxGroup1 = 1), as.matrix(dx)[,1])
```

`subDist` *Extracting a sub-distance matrix of class "dist" from a "dist" object*

Description

This function allows us to efficiently extract a sub-distance matrix of class "dist" from a "dist" object.

Usage

```
subDist(dist, idx, diag = F, upper = F)
```

Arguments

`dist`	A "dist" object, which can be obtained with the "dist" function.
`idx`	An integer vector specifying the indexes of units in the subsets. Indexes can't be smaller than 1 or larger than the dataset size N.
`diag`	A boolean value controls whether or not the diagonal elements (0) are displayed. By default, diag = F.
`upper`	A boolean value controls whether or not the upper-triangular elements (0) are displayed. By default, diag = F.

Details

Extracting a sub-distance matrix of class "dist" from a "dist" object can be done by back and forth conversion between a "dist" object and a numeric matrix using as.dist and as.matrix functions. However, it is extremely inefficient and slow as we only partially extract the "dist" object. This function allows us to directly extract the relevant values directly without the need of conversion.

Value

a sub-distance matrix of class "dist"

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("cluster")
#Generate four clusters of size 50 from 2d Gaussian distributions.
sdev = 0.1
x1 = cbind(rnorm(50, 0, sdev), rnorm(50, 0, sdev))
x2 = cbind(rnorm(50, 1, sdev), rnorm(50, 1, sdev))
x3 = cbind(rnorm(50, 1, sdev), rnorm(50, 0, sdev))
x4 = cbind(rnorm(50, 0, sdev), rnorm(50, 1, sdev))
X = rbind(x1, x2, x3, x4)
dx = dist(X)
C = pam(dx, 4)$clustering #Apply PAM for clustering X.
X2 = X[c(1:10, 51:60, 101:110, 151:160),]
dx2 = subDist(dx, c(1:10, 51:60, 101:110, 151:160))
C2 = pam(dx2, 4)$clustering #Apply PAM for clustering X2.
par(mfrow = c(1,2))
plot(X, col = C, pch = as.character(C), xlab = "X1", ylab = "X2")
plot(X2, col = C2, pch = as.character(C2), xlab = "X1", ylab = "X2")

library("microbenchmark")
x = rnorm(1:1000)
dx = dist(x)
#Extract a sub-distance matrix of class "dist" corresponding to the first 10 units
microbenchmark(as.dist(as.matrix(dx)[1:10, 1:10]),
               subDist(dx, 1:10))
```

Index