Package 'dissimilarities'

May 17, 2025

Type Package

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

Version 0.2.0

Maintainer Minh Long Nguyen <edelweiss611428@gmail.com>

Description Efficiently creates, manipulates, and subsets ``dist'' objects, commonly used in cluster analysis. Designed to minimise unnecessary conversions and computational overhead while enabling seamless interaction with distance matrices.

License CC BY 4.0

Encoding UTF-8

LazyData true

URL https://github.com/edelweiss611428/dissimilarities

BugReports https://github.com/edelweiss611428/dissimilarities/issues

Imports Rcpp,
microbenchmark,
proxy,
stats

LinkingTo Rcpp

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Contents

Dist2Mat *Dist2Mat conversion*

Description

Efficiently converts a "dist" object into a symmetric distance "matrix".

Usage

```
Dist2Mat(dist)
```

Arguments

dist	A "dist" object, which can be computed via the stats::dist function, representing pairwise distances between observations.

Details

Converts a "dist" object, typically created using the stats::dist function, into a symmetric matrix form. This implementation is optimised for speed and performs significantly faster than base::as.matrix or proxy::as.matrix when applied to "dist" objects.

Value

A distance "matrix".

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
x = matrix(rnorm(200), nrow = 50)
dx = dist(x)
#Dist2Mat conversion
microbenchmark(base::as.matrix(dx),
               proxy::as.matrix(dx),
               Dist2Mat(dx))
#Check if equal
v1 = as.vector(base::as.matrix(dx))
v2 = as.vector(Dist2Mat(dx))
all.equal(v1, v2)
```

expandDist *Expanding a distance matrix given new data*

Description

Efficiently appends new "rows" to an existing "dist" object without recomputing a full pairwise distance matrix.

Usage

```
expandDist(distA, A, B, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

Arguments

distA	A "dist" object, representing the pairwise distance matrix between observations in matrix A, ideally computed via the distance metric specified in this function. This requires manual check.
A	A numeric matrix.
B	A numeric matrix.
method	A character string specifying the distance metric to use. Supported methods include "euclidean", "manhattan", "maximum", "minkowski", "cosine", and "canberra".
diag	A boolean value, indicating whether to display the diagonal entries.
upper	A boolean value, indicating whether to display the upper triangular entries.
p	A positive integer, required for computing Minkowski distance; by default p = 2 (i.e., Euclidean).

Details

Expands an existing distance matrix of class "dist" for matrix A, given new data B, without explicitly computing the distance matrix of rbind(A,B). This supports multiple commonly used distance measures and is optimised for speed.

Value

A distance matrix of class "dist" for rbind(A,B).

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
A = matrix(rnorm(100), nrow = 20)
B = matrix(rnorm(250), nrow = 50)
AB = rbind(A,B)
distA = fastDist(A)
v1 = as.vector(expandDist(distA, A, B))
v2 = as.vector(fastDist(AB))
all.equal(v1, v2)
```

`fastDist` *"dist" object computation*

Description

Efficiently computes a "dist" object from a numeric matrix using various distance metrics.

Usage

```
fastDist(X, method = "euclidean", diag = FALSE, upper = FALSE, p = 2L)
```

Arguments

`X`	A numeric matrix.
`method`	A character string specifying the distance metric to use. Supported methods include `"euclidean"`, `"manhattan"`, `"maximum"`, `"minkowski"`, `"cosine"`, and `"canberra"`.
`diag`	A boolean value, indicating whether to display the diagonal entries.
`upper`	A boolean value, indicating whether to display the upper triangular entries.
`p`	A positive integer, required for computing Minkowski distance; by default p = 2 (i.e., Euclidean).

Details

Calculates pairwise distances between rows of a numeric matrix and returns the result as a compact "dist" object, which stores the lower-triangular entries of a complete distance matrix. Supports multiple distance measures, including "euclidean", "manhattan", "maximum", "minkowski", "cosine", and "canberra".

This implementation is optimised for speed, especially on large matrices.

Value

A distance matrix of class "dist".

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
x = matrix(rnorm(200), nrow = 50)
microbenchmark(stats::dist(x, "minkowski", p = 5),
               fastDist(x, "minkowski", p = 5))
v1 = as.vector(stats::dist(x, "minkowski", p = 5))
v2 = as.vector(fastDist(x, "minkowski", p = 5))
all.equal(v1, v2)
```

`fastDistAB` *Computing pairwise distances between rows of two matrices*

Description

Efficiently computes pairwise distances between the rows of two numeric matrices using various distance metrics.

Usage

```
fastDistAB(A, B, method = "euclidean", p = 2L)
```

Arguments

`A`	A numeric matrix.
`B`	A numeric matrix.
`method`	A character string specifying the distance metric to use. Supported methods include `"euclidean"`, `"manhattan"`, `"maximum"`, `"minkowski"`, `"cosine"`, and `"canberra"`.
`p`	A positive integer, required for computing Minkowski distance; by default p = 2 (i.e., Euclidean).

Details

This function computes the full pairwise distance matrix between the rows of matrices `A` and `B`, without forming a concatenated matrix or performing unnecessary intermediate conversions. It supports multiple commonly used distance measures and is optimised for speed.

Value

A numeric matrix of dimensions `nrow(A)` by `nrow(B)`, where each entry represents the distance between a row in `A` and a row in `B`.

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
X = matrix(rnorm(200), nrow = 50)
A = X[1:25,]
B = X[26:50,]
microbenchmark(proxy::dist(A,B, "minkowski", p = 5),
               fastDistAB(A,B, "minkowski", p = 5L))
#Check if equal
v1 = as.vector(proxy::dist(A,B, "minkowski", p = 5))
v2 = as.vector(fastDistAB(A,B, "minkowski", p = 5L))
all.equal(v1, v2)
```

`get1dFrom2d` *2D-indexing to 1D-indexing*

Description

Efficiently computes 1D-indexing from 2D-indexing

Usage

```
get1dFrom2d(i,j, N)
```

Arguments

`i`	An integer specifying the row index
`j`	An integer specifying the column index - must be different from i as "dist" object does not store the diagonal entries.
`N`	The number of observations in the original data matrix

Details

Converts 2D indexing (a row-column pair) into 1D indexing (as used in R's "dist" objects), given the number of observations N.

Value

An integer specifying the 1d index

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
N = 5
for(i in 1:4){
  for(j in (i+1):5){
    print(get1dFrom2d(i,j,N))
  }
}
```

get2dFrom1d *1D-indexing to 2D-indexing*

Description

Efficiently computes 2D-indexing from 1D-indexing

Usage

```
get2dFrom1d(idx1d, N)
```

Arguments

idx1d	An integer vector of 1D indexes
N	The number of observations in the original data matrix

Details

Converts 1D indexing (as used in R's "dist" objects) into 2D indexing (row-column pairs) for a distance matrix of size $N \times N$.

Value

An integer matrix storing the corresponding 2D indexes.

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
get2dFrom1d(1:10, 5)
```

subCols *Subsetting a "dist" object by columns*

Description

Efficiently subsets a "dist" object by selecting specified columns, returning the corresponding section of the distance matrix.

Usage

```
subCols(dist, idx)
```

Arguments

dist	A "dist" object, which can be computed via the stats::dist function, representing pairwise distances between observations.
idx	An integer vector, specifying the column indices of the subsetted matrix.

Details

This function extracts specified columns from a "dist" object without explicit conversion to a dense distance "matrix", resulting in better performance and reduced memory overhead. Particularly useful when only a subset of distances is needed for downstream tasks.

Value

A numeric "matrix" containing the pairwise distances between all rows and the specified columns.

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
x = matrix(rnorm(200), nrow = 50)
dx = dist(x)
#Randomly subsetting a 50x10 matrix
idx = sample(1:50, 10)
microbenchmark(base::as.matrix(dx)[1:50,idx],
               proxy::as.matrix(dx)[1:50,idx],
               subCols(dx, idx))
#Check if equal
v1 = as.vector(base::as.matrix(dx)[1:50,idx])
v2 = as.vector(subCols(dx, idx))
all.equal(v1, v2)
```

---

subDist2Dist *Dist2Dist subsetting*

Description

Efficiently extracts a subset of observations from a "dist" object and returns a new "dist" object representing only the selected distances.

Usage

```
subDist2Dist(dist, idx, diag = FALSE, upper = FALSE)
```

Arguments

dist	A "dist" object, which can be computed via the stats::dist function, representing the full pairwise distance matrix between observations.
idx	An integer vector, specifying the indices of the observations to retain.
diag	A boolean value, indicating whether to display the diagonal entries.
upper	A boolean value, indicating whether to display the upper triangular entries.

Details

This function subsets a "dist" object directly without explicit conversion to a dense distance "matrix". It extracts only the relevant distances corresponding to the selected indices, improving both performance and memory efficiency. The result is returned as a subsetted "dist" object, preserving compatibility with downstream functions that accept this class.

Value

A numeric "matrix" storing pairwise distances between the selected observations.

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
x = matrix(rnorm(200), nrow = 50)
dx = dist(x)
#Subsetting the first 10 units
microbenchmark(as.dist(base::as.matrix(dx)[1:10,1:10]),
               as.dist(proxy::as.matrix(dx)[1:10,1:10]),
               subDist2Dist(dx, 1:10))
#Check if equal
v1 = as.vector(as.dist(base::as.matrix(dx)[1:10,1:10]))
v2 = as.vector(subDist2Dist(dx, 1:10))
all.equal(v1, v2)
```

---

subDist2Mat *Dist2Mat subsetting*

Description

Efficiently extracts a 2d submatrix of pairwise distances from a "dist" object.

Usage

```
subDist2Mat(dist, idx1, idx2)
```

Arguments

dist	A "dist" object, which can be computed via the stats::dist function, representing the full pairwise distance matrix between observations.
idx1	An integer vector, specifying the row indices of the subsetted matrix.
idx2	An integer vector, specifying the column indices of the subsetted matrix.

Details

This function efficiently subsets a "dist" object by row and column indices, returning the corresponding rectangular section as a numeric matrix. It avoids explicit conversion from the "dist" object to a dense "matrix", improving memory efficiency and computational speed, especially with large datasets.

Value

A numeric matrix storing pairwise distances between observations column-indexed by `idx1` and row-indexed by `idx2`.

Author(s)

Minh Long Nguyen <edelweiss611428@gmail.com>

Examples

```
library("microbenchmark")
x = matrix(rnorm(200), nrow = 50)
dx = dist(x)
#Randomly subsetting a 10x10 matrix
idx1 = sample(1:50, 10)
idx2 = sample(1:50, 10)
microbenchmark(base::as.matrix(dx)[idx1,idx2],
               proxy::as.matrix(dx)[idx1,idx2],
               subDist2Mat(dx, idx1, idx2))
#Check if equal
v1 = as.vector(base::as.matrix(dx)[idx1,idx2])
v2 = as.vector(subDist2Mat(dx, idx1, idx2))
all.equal(v1, v2)
```