Package: disto (via r-universe)

June 17, 2024

Type Package

Title Unified Interface to Distance, Dissimilarity, Similarity Matrices

Version 0.2.6

Description Provides a high level API to interface over sources storing distance, dissimilarity, similarity matrices with matrix style extraction, replacement and other utilities. Currently, in-memory dist object backend is supported.

URL https://github.com/talegari/disto

BugReports https://github.com/talegari/disto/issues

Imports proxy (>= 0.4.19), assertthat (>= 0.2.0), fastmatch(>= 1.1.0), pbmcapply (>= 1.2.5), data.table (>= 1.11.4), magrittr (>= 1.5), bigdist (>= 0.1.0), Rfast (>= 1.9.2), ggplot2 (>= 3.1.0), methods, stats, graphics

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

RoxygenNote 6.1.0

Suggests knitr (>= 1.15.1), rmarkdown (>= 1.4),

VignetteBuilder knitr

Repository https://talegari.r-universe.dev

RemoteUrl https://github.com/talegari/disto

RemoteRef HEAD

RemoteSha f05c83bee3d8d3cbcf5d138368cd1fa64a65ee80

Contents

`dapply` *lapply like function for disto object*

Description

Apply function for data underlying disto object

Usage

```
dapply(x, margin = 1, fun, subset, nproc = 1, progress)
```

Arguments

`x`	disto object
`margin`	(one among 1 or 2) dimension to apply function along
`fun`	Function to apply over the margin
`subset`	(integer vector) Row/Column numbers along the margin. If this is missing, all rows or columns are considered
`nproc`	Number of parallel processes (unix only)
`progress`	(flag) Whether to show the progress bar or not. If missing and if the length of the subset is at least 1e4, progress is considered to be TRUE.

Details

fun should accept two arguments(in the same order): First, a vector of distances (row or column of a disto depending on the margin). Second, the row or the column number. See the example below

Value

A list

Examples

```
temp <- dist(iris[,1:4])
dio  <- disto(objectname = "temp")
# function to pick unsorted indexes of 5 nearest neighbors (excepting itself)
udf_nn <- function(distances, index){

  nnPlusOne <- which(
                data.table::frankv(distances, ties.method = "dense") <= 6)
  setdiff(nnPlusOne, index)
}
hi <- dapply(dio, 1, udf_nn)
head(hi)
max(sapply(hi, length))
```

---

disto *Constructor for class 'disto'*

Description

Create mapping to data sources storing distances(symmetric), dissimilarities(non-symmetric), similarities and so on

Provides a high level API to interface over backends storing distance, dissimilarity, similarity matrices with matrix style extraction, replacement and other utilities. Currently, in-memory dist object backend is supported.

Usage

```
disto(..., backend = "dist")
```

Arguments

...	Arguments for a backend. See details
backend	(string) Specify a backend. Currently supported: 'dist', 'bigdist'

Details

This is a wrapper to create a 'disto' handle over different backends storing distances, dissimilarities, similarities etc with minimal data overhead like a database connection. The following named arguments are required to set-up the backend:

- **dist**:
  - objectname: Object of the class 'dist' or the name of the object as a 'string'.
  - env: Environment where the object exists. When this is missing, its assumed to be parent environment.#'
- **bigdist**:
  - object: Object of the class 'bigdist'

Value

Object of class 'disto' which is a thin wrapper on a list

Author(s)

Srikanth KS

See Also

Useful links:

- https://github.com/talegari/disto
- Report bugs at https://github.com/talegari/disto/issues

Examples

```
temp <- stats::dist(iris[,1:4])
dio  <- disto(objectname = "temp")
dio
unclass(dio)
```

`disto_bigdist` *Constructior of disto with bigdist backend*

Description

Constructior of disto with bigdist backend

Usage

```
disto_bigdist(arguments)
```

Arguments

arguments	to construct disto object

Details

to be used by disto constructor function

Value

returns a list

---

disto_dist *Constructior of disto with dist backend*

Description

Constructior of disto with dist backend

Usage

```
disto_dist(arguments)
```

Arguments

arguments	to construct disto object

Details

to be used by disto constructor function

Value

returns a list

dist_extract *Matrix style extraction from dist object*

Description

Matrix style extraction supports 'inner' and 'outer'(default) products

Usage

```
dist_extract(object, i, j, k, product = "outer")
```

Arguments

object	dist object
i	(integer vector) row positions
j	(integer vector) column positions
k	(integer vector) positions
product	(string) One among: 'inner', 'outer'(default)

Details

In k-mode, both i and j should be missing and k should not be missing. In ij-mode, k should be missing and both i and j are optional. If i or j are missing, they are interpreted as all values of i or j (similar to matrix or dataframe subsetting). If i and j are of unequal length, the smaller one is recycled.

Value

A matrix or vector of distances when product is 'outer' and 'inner' respectively

Examples

```
# examples for dist_extract

# create a dist object
temp <- dist(iris[,1:4])
attr(temp, "Labels") <- outer(letters, letters, paste0)[1:150]
head(temp)
max(temp)
as.matrix(temp)[1:5, 1:5]


dist_extract(temp, 1, 1)
dist_extract(temp, 1, 2)
dist_extract(temp, 2, 1)
dist_extract(temp, "aa", "ba")

dist_extract(temp, 1:10, 11:20)
```

```
dim(dist_extract(temp, 1:10, ))
dim(dist_extract(temp, , 1:10))
dist_extract(temp, 1:10, 11:20, product = "inner")
length(dist_extract(temp, 1:10, , product = "inner"))
length(dist_extract(temp, , 1:10, product = "inner"))

dist_extract(temp, c("aa", "ba", "ca"), c("ca", "da", "fa"))
dist_extract(temp, c("aa", "ba", "ca"), c("ca", "da", "fa"), product = "inner")

dist_extract(temp, k = 1:3) # product is always inner when k is specified
```

`dist_ij_k` *Vectorized version of dist_ij_k_*

Description

Convert ij indexes to k indexes for a dist object

Usage

```
dist_ij_k(i, j, size)
```

Arguments

`i`	row indexes
`j`	column indexes
`size`	value of size attribute of the dist object

Value

k indexes

`dist_ij_k_` *Convert ij index to k index*

Description

Convert ij index to k index for a dist object

Usage

```
dist_ij_k_(i, j, size)
```

Arguments

i	row index
j	column index
size	value of size attribute of the dist object

Value

k index

---

dist_k_ij *Vectorized version of dist_k_ij_*

Description

Convert kth indexes to ij indexes of a dist object

Usage

```
dist_k_ij(k, size)
```

Arguments

k	kth indexes
size	value of size attribute of the dist object

Value

ij indexes as 2*n matrix where n is length of k vector

---

dist_k_ij_ *Convert kth index to ij index*

Description

Convert kth index to ij index of a dist object

Usage

```
dist_k_ij_(k, size)
```

Arguments

k	kth index
size	value of size attribute of the dist object

Value

ij index as a length two integer vector

`dist_replace` *Replacement values in dist*

Description

Replacement values of a dist object with either ij or position indexing

Usage

```
dist_replace(object, i, j, value, k)
```

Arguments

`object`	dist object
`i`	(integer vector) row positions
`j`	(integer vector) column positions
`value`	(integer/numeric vector) Values to replace
`k`	(integer vector) positions

Details

There are two modes to specify the positions:

- ij-mode where i and j are specified and k is missing. If i or j are missing, they are interpreted as all values of i or j (similar to matrix or dataframe subsetting). Lengths of i, j are required to be same. If 'value' is singleton, then it is extended to the length of i or j. Else, 'value' should have same length as i or j.
- k-mode where k is present and both i and k are missing. k is the positions in the dist object. If 'value' is singleton, then it is extended to the length of k. Else, 'value' should have same length as k.

Value

dist object

Examples

```
# create a dist object
d <- dist(iris[,1:4])
attr(d, "Labels") <- outer(letters, letters, paste0)[1:150]
head(d)
max(d)
as.matrix(d)[1:5, 1:5]

# replacement in ij-mode
d <- dist_replace(d, 1, 2, 100)
dist_extract(d, 1, 2, product = "inner")
```

```
d <- dist_replace(d, "ca", "ba", 102)
dist_extract(d, "ca", "ba", product = "inner")

d <- dist_replace(d, 1:5, 6:10, 11:15)
dist_extract(d, 1:5, 6:10, product = "inner")
d <- dist_replace(d, c("ca", "da"), c("aa", "ba"), 102)
dist_extract(d, c("ca", "da"), c("aa", "ba"), product = "inner")

# replacement in k-mode
d <- dist_replace(d, k = 2, value = 101)
dist_extract(d, k = 2)
dist_extract(d, 3, 1, product = "inner") # extracting k=2 in ij-mode
```

`dist_subset` *dist_subset*

Description

Compute subset faster than regular '[[' on a dist object. This is from **proxy** package (not exported by proxy).

Usage

```
dist_subset(x, subset, ...)
```

Arguments

`x`	dist object
`subset`	index of the subset. This has to be unique.
`...`	additional arguments

Value

returns a dist subset

`eps_k` *Distance corresponding to kth neighbor*

Description

Distance corresponding to kth neighbor

Usage

```
eps_k(x, k, ...)
```

Arguments

x	Disto object
k	A positive integer
...	Arguments to 'dapply'. Should be among: subset, nproc, progress

Value

A vector of distances

---

names.disto *Get names/labels*

Description

Get names/labels of the underlying distance storing backend

Usage

```
## S3 method for class 'disto'
names(x)
```

Arguments

x	disto object

Value

A character vector

Examples

```
temp <- stats::dist(iris[,1:4])
dio <- disto(objectname = "temp")
dio
names(dio) <- paste0("a", 1:150)
```

nn *Nearest neighbors and distances*

Description

Obtain nearest neighbors and distances from a matrix or disto handle. k nearest or fixed radius neighbors are supported

Usage

```
nn(x, k, r, method = "euclidean", ...)
```

Arguments

x	Object of class 'disto' or a numeric matrix
k	Number of nearest neighbors
r	Radius for nearest neighbors
method	(string or function) distance metric when x is a matrix. Passed to 'proxy::dist'. Ignored when x is not a matrix.
...	Additional arguments for dapply when x is 'disto' object. Else additional arguments are sent to pbmclapply

Details

Exactly one among k or r has to be provided

Value

Object of class nn. A list with these elements:

- *triplet*: Matrix with three columns: row, col and distance. For a fixed observation(value in 'row'), all corresponding values in 'col' are the indexes of the nearest neighbors. All corresponding values in 'distance' are the distances to those nearest neighbors
- *size*: Size of the distance matrix or number of rows of the matrix
- *k* or *r* : Depending on the input

Examples

```
## Not run:
# create a matrix
set.seed(100)
mat <- cbind(rnorm(3e3), rpois(3e3, 1))

# compute a distance matrix and get a disto handle
do <- stats::dist(mat)
dio <- disto(objectname = "do")
```

```
# nearest neighbors: k nearest and fixed radius
nn(dio, k = 1)
nn(mat, k = 1) # distance method defaults to 'euclidean'
str(nn(mat, k = 1)) # observe the structure of the output

nn(dio, r = 0.1)
nn(mat, r = 0.1)

# nearest neighbors parallelized: k nearest and fixed radius
# fast computation, higher memory usage
nn(dio, k = 1, nproc = 2)
nn(mat, k = 1, mc.cores = 2)

nn(dio, r = 0.1, nproc = 2)
nn(mat, r = 0.1, mc.cores = 2)

# different distance method
do <- stats::dist(mat, method = "manhattan")

nn(dio, k = 1, nproc = 2)
nn(mat, k = 1, method = "manhattan", mc.cores = 2)

nn(dio, r = 0.1, nproc = 2)
nn(mat, r = 0.1, method = "manhattan", mc.cores = 2)

## End(Not run)
```

---

nn2 *Extension of nn method for two matrices*

Description

Find k or fixed radius nearest neighbors of each observation(row) matrix y in matrix x

Usage

```
nn2(x, y, k, r, method = "euclidean", ...)
```

Arguments

x	Numeric matrix
y	Numneric matrix
k	Number of nearest neighbors
r	Radius for nearest neighbors
method	(string or function) Distance metric passed to ‘proxy::dist‘
...	Additional arguments are sent to pbmclapply

Details

Exactly one among k or r has to be provided

Value

Object of class 'nn'. A list with these elements:

- *triplet*: Matrix with three columns: row, col and distance. For a fixed observation of matrix y (value in 'row'), all corresponding values in 'col' are the indexes of the nearest neighbors in matrix x. All corresponding values in 'distance' are the distances to those nearest neighbors
- *size*: Number of rows of matrix y
- *sizeX*: Number of rows of matrix x
- *k* or *r* : Depending on the input

Examples

```
temp <- nn2(x = matrix(rnorm(1e4), ncol = 10)
            , y = matrix(runif(1e3), ncol = 10)
            , r = 2
            )
temp
```

---

nn_k *k Nearest neighbors*

Description

k Nearest neighbors from a vector of distances

Usage

```
nn_k(vec, index, k)
```

Arguments

vec	Vector of distances
index	dummy to facilitate dapply
k	Number of nearest neighbors

nn_r *Fixed radius Nearest neighbors*

Description

Fixed radius Nearest neighbors from a vector of distances

Usage

```
nn_r(vec, index, r)
```

Arguments

vec	Vector of distances
index	dummy to facilitate dapply
r	Radius for nearest neighbors

plot.disto *Plot a disto object*

Description

Density plot a disto object

Usage

```
## S3 method for class 'disto'
plot(x, ...)
```

Arguments

x	object of class disto
...	Currently unused

Value

Plot output as side effect

Examples

```
temp <- stats::dist(iris[,1:4])
dio  <- disto(objectname = "temp")
plot(dio)
```

print.disto *Print method for dist class*

Description

Print method for dist class

Usage

```
## S3 method for class 'disto'
print(x, ...)
```

Arguments

x	object of class disto
...	currently not in use

Value

invisible NULL. Function writes backend type and size to terminal as a message.

Examples

```
temp <- stats::dist(iris[,1:4])
dio   <- disto(objectname = "temp")
print(dio)
```

print.nn *Print method for class 'nn'*

Description

Print method for class 'nn'

Usage

```
## S3 method for class 'nn'
print(x, ...)
```

Arguments

x	Object of class 'nn'
...	stub

Value

Returns the input invisibly besides printing on the screen

size *Obtain size of the disto object*

Description

Obtain size of the disto object

Usage

```
size(x, ...)
```

Arguments

x	object of class disto
...	currently not in use

Value

Integer vector of length 1

Examples

```
temp <- stats::dist(iris[,1:4])
dio   <- disto(objectname = "temp")
size(dio)
```

summary.disto *Summary method for dist class*

Description

Summary method for dist class

Usage

```
## S3 method for class 'disto'
summary(object, ...)
```

Arguments

object	object of class disto
...	currently not in use

Value

Result of summary function

Examples

```
temp <- stats::dist(iris[,1:4])
dio   <- disto(objectname = "temp")
dio
summary(dio)
```

'names<-.disto' *Set names/labels*

Description

Set names/labels of the underlying distance storing backend

Usage

```
## S3 replacement method for class 'disto'
names(x) <- value
```

Arguments

x	disto object
value	A character vector

Value

invisible disto object

Examples

```
temp <- stats::dist(iris[,1:4])
dio <- disto(objectname = "temp")
dio
names(dio) <- paste0("a", 1:150)
```

'[.disto' *Extract from a disto object in matrix style extraction*

Description

Extract a disto object in matrix style extraction and via direct indexing. 'product' specification allows both outer (matrix output, default option) and inner (vector) product type extraction. For dist backend see: dist_extract.

Usage

```
## S3 method for class 'disto'
x[i, j, k, product = "outer"]
```

Arguments

x	object of class 'disto'
i	(integer vector) row indexes
j	(integer vector) column indexes
k	(integer vector) direct indexes
product	(string) One among: "inner", "outer"

Value

When product is 'outer', returns a matrix. Else, a vector.

Examples

```
temp <- stats::dist(iris[,1:4])
dio <- disto(objectname = "temp")
dio
names(dio) <- paste0("a", 1:150)

dio[1, 2]
dio[2, 1]
dio[c("a1", "a10"), c("a5", "a72")]
dio[c("a1", "a10"), c("a5", "a72"), product = "inner"]
dio[k = c(1,3,5)]
```

'[<-.disto' *In-place replacement of values*

Description

For dist backend see: dist_replace.

Usage

```
## S3 replacement method for class 'disto'
x[i, j, k] <- value
```

Arguments

x	object of class 'disto'
i	(integer vector) row index
j	(integer vector) column index
k	(integer vector) direct index
value	(integer/numeric vector) Values to replace

Value

Invisible disto object. Note that this function is called for its side effect.

Examples

```
temp       <- stats::dist(iris[,1:4])
dio        <- disto(objectname = "temp")
names(dio) <- paste0("a", 1:150)
dio

dio[1, 2] <- 10
dio[1,2]

dio[1:10, 2:11] <- 100
dio[1:10, 2:11, product = "inner"]

dio[paste0("a", 1:5), paste0("a", 6:10)] <- 101
dio[paste0("a", 1:5), paste0("a", 6:10), product = "inner"]
```

‘[[.disto‘ *Extract a single value from disto object*

Description

Extract a single value from disto object in matrix style extraction and via direct indexing. This does not support using names. This is faster than `link{extract}`. For dist backend see: `dist_extract`.

Usage

```
## S3 method for class 'disto'
x[[i, j, k]]
```

Arguments

`x`	object of class 'disto'
`i`	(integer vector) row index
`j`	(integer vector) column index
`k`	(integer vector) direct index

Value

(A real number) Distance value

Examples

```
temp <- stats::dist(iris[,1:4])
dio  <- disto(objectname = "temp")
dio

dio[[1, 2]]
dio[[2, 1]]
dio[[k = 3]]
```

Index