

Package ‘proxyC’

April 25, 2025

Type Package

Title Computes Proximity in Large Sparse Matrices

Version 0.5.2

Description Computes proximity between rows or columns of large matrices efficiently in C++. Functions are optimised for large sparse matrices using the Armadillo and Intel TBB libraries. Among various built-in similarity/distance measures, computation of correlation, cosine similarity, Dice coefficient and Euclidean distance is particularly fast.

URL <https://github.com/koheiw/proxyC>, <https://koheiw.github.io/proxyC/>

BugReports <https://github.com/koheiw/proxyC/issues>

License GPL-3

Depends R (>= 3.1.0), methods

Imports Matrix (>= 1.2), Rcpp (>= 0.12.12)

Suggests testthat, entropy, proxy, knitr, rmarkdown

LinkingTo Rcpp, RcppArmadillo (>= 0.7.600.1.0)

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/Needs/website coop, Rfast, parallelDist, distances, proxy, microbenchmark, ggplot2

Author Kohei Watanabe [cre, aut, cph]
(<https://orcid.org/0000-0001-6519-5265>),
Robrecht Cannoodt [aut] (<https://orcid.org/0000-0003-3641-729X>)

Maintainer Kohei Watanabe <watanabe.kohei@gmail.com>

Repository CRAN

Date/Publication 2025-04-25 12:40:06 UTC

Contents

colSds	2
colZeros	2
crossprod	3
mask	3
simil	4

Index	7
--------------	----------

colSds	<i>Standard deviation of columns and rows of large matrices</i>
--------	---

Description

Produces the same result as `apply(x, 1, sd)` or `apply(x, 2, sd)` without coercing matrix to dense matrix. Values are not identical to `sd()` because of the floating point precision issue in C++.

Usage

`colSds(x)`

`rowSds(x)`

Arguments

`x` a `base::matrix` or `Matrix::Matrix` object.

Examples

```
mt <- Matrix::rsparsematrix(100, 100, 0.01)
colSds(mt)
apply(mt, 2, sd) # the same
```

colZeros	<i>Number of zeros in columns and rows of large matrices</i>
----------	--

Description

Produces the same result as applying `sum(x == 0)` to each row or column.

Usage

`colZeros(x)`

`rowZeros(x)`

Arguments

x a [base::matrix](#) or [Matrix::Matrix](#) object.

Examples

```
mt <- Matrix::rsparsematrix(100, 100, 0.01)
colZeros(mt)
apply(mt, 2, function(x) sum(x == 0)) # the same
```

crossprod	<i>Cross-product of large sparse matrices</i>
-----------	---

Description

Compute the (transposed) cross-product of large sparse matrices using the same infrastructure as [simil\(\)](#) and [dist\(\)](#).

Usage

```
crossprod(x, y = NULL, min_prod = NULL, digits = 14)
tcrossprod(x, y = NULL, min_prod = NULL, digits = 14)
```

Arguments

x a [base::matrix](#) or [Matrix::Matrix](#) object. Dense matrices are covered to the [Matrix::CsparseMatrix](#) internally.

y if a [base::matrix](#) or [Matrix::Matrix](#) object is provided, proximity between documents or features in x and y is computed.

min_prod the minimum product to be recorded.

digits determines rounding of small values towards zero. Use primarily to correct floating point errors. Rounding is performed in C++ in a similar way as [base::zapsmall](#).

mask	<i>Create a pattern matrix for masking</i>
------	--

Description

Create a pattern matrix for [simil\(\)](#) or [dist\(\)](#) to enable masked similarity computation. If the matrix is passed to the function, it computes similarity scores only for cells with TRUE.

Usage

```
mask(x, y = NULL)
```

Arguments

x a numeric or character vector matched against each other.
 y a numeric or character vector matched against x if provided.

Value

a sparse logical matrix with TRUE for matched pairs.

Examples

```
mt1 <- Matrix::rsparsematrix(100, 6, 1.0)
colnames(mt1) <- c("a", "a", "d", "d", "e", "e")
mt2 <- Matrix::rsparsematrix(100, 5, 1.0)
colnames(mt2) <- c("a", "b", "c", "d", "e")

(msk <- mask(colnames(mt1), colnames(mt2)))
simil(mt1, mt2, margin = 2, mask = msk, drop0 = TRUE)
```

simil	<i>Compute similarity/distance between rows or columns of large matrices</i>
-------	--

Description

Fast similarity/distance computation function for large sparse matrices. You can floor small similarity value to save computation time and storage space by an arbitrary threshold (`min_simil`) or rank (`rank`). You can specify the number of threads for parallel computing via `options(proxyC.threads)`.

Usage

```
simil(
  x,
  y = NULL,
  margin = 1,
  method = c("cosine", "correlation", "dice", "edice", "jaccard", "ejaccard", "fjaccard",
    "hamann", "faith", "simple matching"),
  mask = NULL,
  min_simil = NULL,
  rank = NULL,
  drop0 = FALSE,
  diag = FALSE,
  use_nan = NULL,
  sparse = TRUE,
  digits = 14
)

dist(
```

```

x,
y = NULL,
margin = 1,
method = c("euclidean", "chisquared", "kullback", "jeffreys", "jensen", "manhattan",
           "maximum", "canberra", "minkowski", "hamming"),
mask = NULL,
p = 2,
smooth = 0,
drop0 = FALSE,
diag = FALSE,
use_nan = NULL,
sparse = TRUE,
digits = 14
)

```

Arguments

<code>x</code>	a <code>base::matrix</code> or <code>Matrix::Matrix</code> object. Dense matrices are covered to the <code>Matrix::CsparseMatrix</code> internally.
<code>y</code>	if a <code>base::matrix</code> or <code>Matrix::Matrix</code> object is provided, proximity between documents or features in <code>x</code> and <code>y</code> is computed.
<code>margin</code>	integer indicating margin of similarity/distance computation. 1 indicates rows or 2 indicates columns.
<code>method</code>	method to compute similarity or distance
<code>mask</code>	a pattern matrix created using <code>mask()</code> for masked similarity/distance computation. The shape of the matrix must be the same as the resulting matrix.
<code>min_simil</code>	the minimum similarity value to be recorded.
<code>rank</code>	an integer value specifying top-n most similarity values to be recorded.
<code>drop0</code>	if TRUE, removes zero values to make the similarity/distance matrix sparse. It has no effect when <code>dense = TRUE</code> .
<code>diag</code>	if TRUE, only compute diagonal elements of the similarity/distance matrix; useful when comparing corresponding rows or columns of <code>x</code> and <code>y</code> .
<code>use_nan</code>	if TRUE, returns NaN if the standard deviation of a vector is zero when method is "correlation"; if all the values are zero in a vector when method is "cosine", "chisquared", "kullback", "jeffreys" or "jensen". Note that use of NaN makes the similarity/distance matrix denser and therefore larger in RAM. If FALSE, return zero in same use situations as above. If NULL, will also return zero but also generate a warning (default).
<code>sparse</code>	if TRUE, returns <code>Matrix::sparseMatrix</code> object. When neither <code>min_simil</code> nor <code>rank</code> is used, dense matrices require less space in RAM.
<code>digits</code>	determines rounding of small values towards zero. Use primarily to correct floating point errors. Rounding is performed in C++ in a similar way as <code>base::zapsmall</code> .
<code>p</code>	weight for Minkowski distance.
<code>smooth</code>	adds a fixed value to all the cells to avoid division by zero. Only used when method is "chisquared", "kullback", "jeffreys" or "jensen".

Details

Available Methods:

Similarity:

- cosine: cosine similarity
- correlation: Pearson's correlation
- jaccard: Jaccard coefficient
- ejaccard: the real value version of jaccard
- fjaccard: Fuzzy Jaccard coefficient
- dice: Dice coefficient
- edice: the real value version of dice
- hamann: Hamann similarity
- faith: Faith similarity
- simple matching: the percentage of common elements

Distance:

- euclidean: Euclidean distance
- chisquared: chi-squared distance
- kullback: Kullback–Leibler divergence
- jeffreys: Jeffreys divergence
- jensen: Jensen–Shannon divergence
- manhattan: Manhattan distance
- maximum: the largest difference between values
- canberra: Canberra distance
- minkowski: Minkowski distance
- hamming: Hamming distance

See the vignette for how the similarity and distance are computed: `vignette("measures", package = "proxyC")`

Parallel Computing:

It performs parallel computing using Intel oneAPI Threads Building Blocks. The number of threads for parallel computing should be specified via `options(proxyC.threads)` before calling the functions. If the value is -1, all the available threads will be used. Unless the option is used, the number of threads will be limited by the environmental variables (`OMP_THREAD_LIMIT` or `RCPP_PARALLEL_NUM_THREADS`) to comply with CRAN policy and offer backward compatibility.

See Also

`zapsmall`

Examples

```
mt <- Matrix::rsparsematrix(100, 100, 0.01)
simil(mt, method = "cosine")[1:5, 1:5]
mt <- Matrix::rsparsematrix(100, 100, 0.01)
dist(mt, method = "euclidean")[1:5, 1:5]
```

Index

`base::matrix`, 2, 3, 5
`base::zapsmall`, 3, 5

`colSds`, 2
`colZeros`, 2
`crossprod`, 3

`dist(simil)`, 4
`dist()`, 3

`mask`, 3
`mask()`, 5
`Matrix::CsparseMatrix`, 3, 5
`Matrix::Matrix`, 2, 3, 5
`Matrix::sparseMatrix`, 5

`rowSds(colSds)`, 2
`rowZeros(colZeros)`, 2

`simil`, 4
`simil()`, 3

`tcrossprod(crossprod)`, 3