# Package 'proxyC'

April 7, 2024

**Type** Package

**Title** Computes Proximity in Large Sparse Matrices

**Version** 0.4.1

**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 several built-in similarity/distance measures, computation of correlation, cosine similarity and Euclidean distance is particularly fast.

**URL** https://github.com/koheiw/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** 7.3.1

**VignetteBuilder** knitr

**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** 2024-04-07 08:10:02 UTC

## R topics documented:

---

`colSds` *Standard deviation of columns and rows of large matrices*

---

## 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` | matrix or Matrix object |

## Examples

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

---

`colZeros` *Count number of zeros in columns and rows of large matrices*

---

## Description

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

## Usage

```
colZeros(x)

rowZeros(x)
```

## Arguments

| | |
|---|---|
| `x` | matrix or Matrix object |

## Examples

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

simil *Compute similarity/distance between rows or columns of large matrices*

## Description

Fast similarity/distance computation function for large sparse matrices. You can floor small similarity value to 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", "jaccard", "ejaccard", "fjaccard", "dice", "edice",
    "hamann", "faith", "simple matching"),
  min_simil = NULL,
  rank = NULL,
  drop0 = FALSE,
  diag = FALSE,
  use_nan = NULL,
  digits = 14
)

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

## Arguments

| | |
|---|---|
| x | matrix or Matrix object. Dense matrices are covered to the CsparseMatrix-class internally. |
| y | if a matrix or Matrix object is provided, proximity between documents or features in x and y is computed. |

| | |
|---|---|
| margin | integer indicating margin of similarity/distance computation. 1 indicates rows or 2 indicates columns. |
| method | method to compute similarity or distance |
| min_simil | the minimum similarity value to be recorded. |
| rank | an integer value specifying top-n most similarity values to be recorded. |
| drop0 | if TRUE, zero values are removed regardless of min_simil or rank. |
| diag | if TRUE, only compute diagonal elements of the similarity/distance matrix; useful when comparing corresponding rows or columns of x and y. |
| use_nan | if TRUE, return 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). |
| digits | determines rounding of small values towards zero. Use primarily to correct rounding errors in C++. See zapsmall. |
| p | weight for Minkowski distance |
| smooth | 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