Package ‘proxyC’

July 21, 2019

Type Package

Title Computes Proximity in Large Sparse Matrices

Version 0.1.5

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.

Encoding UTF-8

LazyData true

LinkingTo Rcpp, RcppParallel, RcppArmadillo (>= 0.7.600.1.0)

BugReports `https://github.com/koheiw/proxyC/issues`

SystemRequirements C++11

License GPL-3

Depends R (>= 3.1.0), methods

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

Suggests testthat, proxy

RoxygenNote 6.1.1

NeedsCompilation yes

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 2019-07-21 07:00:02 UTC

R topics documented:

colSds *Standard deviasion of columns and rows in sparse matrix*

Description

Produces the same result as `apply(x,1,sd)` or `apply(x,2,sd)` as without coecirng 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 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 in sparse matrix*

Description

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

Usage

```
colZeros(x)

rowZeros(x)
```

Arguments

`x`	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 threashold (`min_simil`) or rank (`rank`). Please increase the numbner of threads for better perfromance using `setThreadOptions`.

Usage

```
simil(x, y = NULL, margin = 1, method = c("cosine", "correlation",
  "jaccard", "ejaccard", "dice", "edice", "hamman", "simple matching",
  "faith"), min_simil = NULL, rank = NULL, drop0 = FALSE,
  digits = 14)

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

Arguments

`x`	Matrix object
`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 recoded.
`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`.
`digits`	determines rounding of small values towards zero. Use primarily to correct rounding errors in C++. See zapsmall.
`p`	weight for minkowski distance

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]
```