Package ‘DelayedMatrixStats’

March 29, 2024

Type Package

Title Functions that Apply to Rows and Columns of 'DelayedMatrix' Objects

Version 1.24.0

Date 2023-10-12

Description A port of the 'matrixStats' API for use with DelayedMatrix objects from the 'DelayedArray' package. High-performing functions operating on rows and columns of DelayedMatrix objects, e.g. col / rowMedians(), col / rowRanks(), and col / rowSds(). Functions optimized per data type and for subsetted calculations such that both memory usage and processing time is minimized.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends MatrixGenerics (>= 1.13.1), DelayedArray (>= 0.27.10)

Imports methods, sparseMatrixStats (>= 1.13.2), Matrix (>= 1.5-0), S4Vectors (>= 0.17.5), IRanges (>= 2.25.10)

Suggests testthat, knitr, rmarkdown, BiocStyle, microbenchmark, profmem, HDF5Array, matrixStats (>= 1.0.0)

VignetteBuilder knitr

URL https://github.com/PeteHaitch/DelayedMatrixStats

BugReports https://github.com/PeteHaitch/DelayedMatrixStats/issues

biocViews Infrastructure, DataRepresentation, Software

git_url https://git.bioconductor.org/packages/DelayedMatrixStats

git_branch RELEASE_3_18

git_last_commit 8fb9c6e

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18
R topics documented:

colAlls,DelayedMatrix-method .............................................. 3
colAnyNAs,DelayedMatrix-method ........................................... 5
colAvgsPerRowSet,DelayedMatrix-method ................................. 7
colCollapse,DelayedMatrix-method ........................................ 9
colCounts,DelayedMatrix-method .......................................... 11
colCummaxs,DelayedMatrix-method ....................................... 13
colDiffs,DelayedMatrix-method ............................................ 16
colIQRDiffs,DelayedMatrix-method ....................................... 18
colIQRs,DelayedMatrix-method ............................................ 22
colLogSumExps,DelayedMatrix-method .................................... 24
colMads,DelayedMatrix-method ............................................. 25
colMeans2,DelayedMatrix-method .......................................... 28
colMedians,DelayedMatrix-method ....................................... 30
colOrderStats,DelayedMatrix-method ..................................... 32
colProds,DelayedMatrix-method .......................................... 33
colQuantiles,DelayedMatrix-method ...................................... 36
colRanks,DelayedMatrix-method ........................................... 38
colSums2,DelayedMatrix-method .......................................... 40
colTabulates,DelayedMatrix-method ...................................... 42
colVars,DelayedMatrix-method ............................................. 44
colWeightedMads,DelayedMatrix-method .................................. 46
colWeightedMeans,DelayedMatrix-method ................................. 48
colWeightedMedians,DelayedMatrix-method ............................... 49
colWeightedSds,DelayedMatrix-method .................................... 51
DelayedMatrixStats .............................................................. 54
DelayedMatrixStats-deprecated ............................................. 54
from_DelayedArray_to_simple_seed_class ................................. 56
reexports ............................................................................. 57
subset_by_Nindex ................................................................ 57

Index .................................................................................. 58
colAlls,DelayedMatrix-method

Check if all elements in a row (column) of a matrix-like object are equal to a value

Description

Check if all elements in a row (column) of a matrix-like object are equal to a value.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colAlls(
x, 
rows = NULL,
cols = NULL,
value = TRUE,
na.rm = FALSE,
force_block_processing = FALSE,
..., 
useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
colAnys(
x, 
rows = NULL,
cols = NULL,
value = TRUE,
na.rm = FALSE,
force_block_processing = FALSE,
..., 
useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowAlls(
x, 
rows = NULL,
cols = NULL,
value = TRUE,
na.rm = FALSE,
force_block_processing = FALSE,
..., 
useNames = TRUE
)
```
## S4 method for signature 'DelayedMatrix'
rowAnys(
  x,
  rows = NULL,
  cols = NULL,
  value = TRUE,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ...
}

### Arguments

- **x**: A NxK DelayedMatrix.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **value**: The value to search for.
- **na.rm**: If TRUE, missing values (NA or NaN) are omitted from the calculations.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- **...**: Additional arguments passed to specific methods.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

### Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowAlls / matrixStats::colAlls.

### Value

Returns a logical vector of length N (K).

### Author(s)

Peter Hickey

### See Also

- matrixStats::rowAlls() and matrixStats::colAlls() which are used when the input is a matrix or numeric vector.
- For checks if any element is equal to a value, see rowAnys().
- base::all().
Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
                                  as.integer((0:4) ^ 2),
                                  seq(-5L, -1L, 1L)),
                                  ncol = 3))

# A DelayedMatrix with a 'SolidRleArraySeed' seed
dm_Rle <- RleArray(Rle(c(rep(1L, 5),
                        as.integer((0:4) ^ 2),
                        seq(-5L, -1L, 1L))),
                   dim = c(5, 3))

colAlls(dm_matrix, value = 1)
colAnys(dm_matrix, value = 2)
rowAlls(dm_Rle, value = 1)
rowAnys(dm_Rle, value = 2)

Description

Check if any elements in a row (column) of a matrix-like object is missing.

Usage

## S4 method for signature 'DelayedMatrix'
colAnyNAs(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowAnyNAs(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
Arguments

- **x** A NxK `DelayedMatrix`.
- **rows, cols** A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
- **force_block_processing** FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to `TRUE` (typically not advised). The block-processing strategy loads one or more (depending on `getAutoBlockSize()`) columns (`colFoo()`) or rows (`rowFoo()`) into memory as an ordinary `base::array`.
- **useNames** If `NA`, the default behavior of the function about naming support is remained. If `FALSE`, no naming support is done. Else if `TRUE`, names attributes of result are set.

Details

The S4 methods for `x` of type `matrix`, `array`, `table`, or `numeric` call `matrixStats::rowAnyNAs / matrixStats::colAnyNAs`.

Value

Returns a logical vector of length N (K).

Author(s)

Peter Hickey

See Also

- `matrixStats::rowAnyNAs()` and `matrixStats::colAnyNAs()` which are used when the input is a matrix or numeric vector.
- For checks if any element is equal to a value, see `rowAnys()`.
- `base::is.na()` and `base::any()`.

Examples

```r
# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
      as.integer((0:4) ^ 2),
      seq(-5L, -1L, 1L)),
    ncol = 3))

# A DelayedMatrix with a 'HDF5ArraySeed' seed
# NOTE: Requires that the HDF5Array package is installed
library(HDF5Array)
dm_HDF5 <- writeHDF5Array(matrix(c(rep(1L, 5),
      as.integer((0:4) ^ 2),
      seq(-5L, -1L, 1L)),
```
Calculates for each row (column) a summary statistic for equally sized subsets of columns (rows).

Description

Calculates for each row (column) a summary statistic for equally sized subsets of columns (rows).

Usage

```r
## S4 method for signature 'DelayedMatrix'
colAvgsPerRowSet(
  X,
  W = NULL,
  cols = NULL,
  S,
  FUN = colMeans,
  ..., 
  force_block_processing = FALSE,
  na.rm = NA,
  tFUN = FALSE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowAvgsPerColSet(
  X,
  W = NULL,
  rows = NULL,
  S,
  FUN = rowMeans,
  ..., 
  force_block_processing = FALSE,
  na.rm = NA,
  tFUN = FALSE
)
```

Arguments

- `X` A NxM `DelayedMatrix`.

Arguments

- `W` A NxM `DelayedMatrix`.
- `cols` A vector of column indices.
- `S` A vector of subset sizes.
- `FUN` A function to compute the summary statistic.
- `tFUN` A logical value indicating whether to transpose the result.
- `na.rm` A logical value indicating whether to remove missing values.
- `force_block_processing` A logical value indicating whether to force block processing.

Examples

```r
dm_matrix[dm_matrix > 3] <- NA
colAnyNAs(dm_matrix)
dm_HDF5[dm_HDF5 > 3] <- NA
rowAnyNAs(dm_HDF5)
```
colAvgsPerRowSet, DelayedMatrix-method

W
An optional numeric N x M matrix of weights.

S
An integer K x J matrix that specifying the J subsets. Each column hold K column (row) indices for the corresponding subset. The range of values is [1, M] ([1, N]).

FUN
A row-by-row (column-by-column) summary statistic function. It is applied to to each column (row) subset of X that is specified by S.

... Additional arguments passed to specific methods.

force_block_processing
FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize()}() columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

na.rm
(logical) Argument passed to FUN() as na.rm = na.rm. If NA (default), then na.rm = TRUE is used if X or S holds missing values, otherwise na.rm = FALSE.

tFUN
If TRUE, X is transposed before it is passed to FUN.

rows, cols
A vector indicating the subset (and/or columns) to operate over. If NULL, no subsetting is done.

Details
The S4 methods for X of type matrix, array, table, or numeric call matrixStats::rowAvgsPerColSet / matrixStats::colAvgsPerRowSet.

Value
Returns a numeric J x N (M x J) matrix.

Author(s)
Peter Hickey

See Also
- matrixStats::rowAvgsPerColSet() and matrixStats::colAvgsPerRowSet() which are used when the input is a matrix or numeric vector.

Examples
# A DelayedMatrix with a 'DataFrame' seed
dm_DF <- DelayedArray(S4Vectors::DataFrame(C1 = rep(1L, 5),
                                        C2 = as.integer((0:4) ^ 2),
                                        C3 = seq(-5L, -1L, 1L)))
colAvgsPerRowSet(dm_DF, S = matrix(1:2, ncol = 2))
rowAvgsPerColSet(dm_DF, S = matrix(1:2, ncol = 1))
Description

Extract one cell from each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colCollapse(
  x,
  idxs,
  cols = NULL,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowCollapse(
  x,
  idxs,
  rows = NULL,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)
```

Arguments

- **x**: A NxK `DelayedMatrix`
- **idxs**: An index vector with the position to extract. It is recycled to match the number of rows (column)
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on `DelayedArray::getAutoBlockSize()`) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- **...**: Additional arguments passed to specific methods.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
Details

The S4 methods for \texttt{x} of type \texttt{matrix}, \texttt{array}, \texttt{table}, or \texttt{numeric} call \texttt{matrixStats::rowCollapse} / \texttt{matrixStats::colCollapse}.

Value

Returns a \texttt{numeric vector} of length N (K).

Author(s)

Peter Hickey

See Also

- \texttt{matrixStats::rowCollapse()} and \texttt{matrixStats::colCollapse()} which are used when the input is a \texttt{matrix} or \texttt{numeric vector}.

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
                             as.integer((0:4) ^ 2),
                             seq(-5L, -1L, 1L)),
                             ncol = 3))

# A DelayedMatrix with a 'HDF5ArraySeed' seed
# NOTE: Requires that the HDF5Array package is installed
library(HDF5Array)
dm_HDF5 <- writeHDF5Array(matrix(c(rep(1L, 5),
                                 as.integer((0:4) ^ 2),
                                 seq(-5L, -1L, 1L)),
                                 ncol = 3))

# Extract the 4th row as a vector
# NOTE: An ordinary vector is returned regardless of the backend of
#       theDelayedMatrix object
colCollapse(dm_matrix, 4)
colCollapse(dm_HDF5, 4)

# Extract the 2nd column as a vector
# NOTE: An ordinary vector is returned regardless of the backend of
#       the DelayedMatrix object
rowCollapse(dm_matrix, 2)
rowCollapse(dm_HDF5, 2)
Count how often an element in a row (column) of a matrix-like object is equal to a value.

**Description**

Count how often an element in a row (column) of a matrix-like object is equal to a value.

**Usage**

```r
## S4 method for signature 'DelayedMatrix'
colCounts(
  x,
  rows = NULL,
  cols = NULL,
  value = TRUE,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowCounts(
  x,
  rows = NULL,
  cols = NULL,
  value = TRUE,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)
```

**Arguments**

- **x** A NxK `DelayedMatrix`.
- **rows, cols** A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
- **value** The value to search for.
- **na.rm** If `TRUE`, missing values (`NA` or `NaN`) are omitted from the calculations.
- **force_block_processing** FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to `TRUE` (typically not advised). The block-processing strategy loads
one or more (depending on \linkDelayedArray{getAutoBlockSize()}) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary \code{base::array}.

... Additional arguments passed to specific methods.

**useNames** If \code{NA}, the default behavior of the function about naming support is remained. If \code{FALSE}, no naming support is done. Else if \code{TRUE}, names attributes of result are set.

### Details

The S4 methods for \code{X} of type \code{matrix}, \code{array}, \code{table}, or \code{numeric} call \code{matrixStats::rowCounts} / \code{matrixStats::colCounts}.

### Value

Returns a \code{integer vector} of length \code{N (K)}.

### Author(s)

Peter Hickey

### See Also

- \code{matrixStats::rowCounts()} and \code{matrixStats::colCounts()} which are used when the input is a matrix or numeric vector.
- For checks if any element is equal to a value, see \code{rowAnys()}. To check if all elements are equal, see \code{rowAlls()}.

### Examples

```r
# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
as.integer((0:4) ^ 2),
       seq(-5L, -1L, 1L)),
      ncol = 3))
# A DelayedMatrix with a 'DataFrame' seed
dm_DF <- DelayedArray(S4Vectors::DataFrame(C1 = rep(1L, 5),
                                  C2 = as.integer((0:4) ^ 2),
                                  C3 = seq(-5L, -1L, 1L)))

colCounts(dm_matrix, value = 1)
# Only count those in the first 4 rows
colCounts(dm_matrix, rows = 1:4, value = 1)

rowCounts(dm_DF, value = 5)
# Only count those in the odd-numbered rows of the 2nd column
rowCounts(dm_DF, rows = seq(1, nrow(dm_DF), 2), cols = 2, value = 5)
```
Calculates the cumulative maxima for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colCummaxs(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
colCummins(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
colCumprods(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
colCumsums(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
```
## S4 method for signature 'DelayedMatrix'
rowCummaxs(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE 
)

## S4 method for signature 'DelayedMatrix'
rowCummins(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE 
)

## S4 method for signature 'DelayedMatrix'
rowCumprods(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE 
)

## S4 method for signature 'DelayedMatrix'
rowCumsums(
  x,
  rows = NULL,
  cols = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE 
)

**Arguments**

- **x** A NxK `DelayedMatrix`.
- **rows, cols** A vector indicating the subset of rows (and/or columns) to operate over. If

colCummaxs, DelayedMatrix-method

NULL, no subsetting is done.

force_block_processing
FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (\code{colFoo()}) or rows (\code{rowFoo()}) into memory as an ordinary \code{base::array}.

... Additional arguments passed to specific methods.

useNames If \code{NA}, the default behavior of the function about naming support is remained. If \code{FALSE}, no naming support is done. Else if \code{TRUE}, names attributes of result are set.

Details
The S4 methods for \code{x} of type \code{matrix}, \code{array}, \code{table}, or \code{numeric} call \code{matrixStats::rowCummaxs} / \code{matrixStats::colCummaxs}.

Value
Returns a \code{numeric matrix} with the same dimensions as \code{x}.

Author(s)
Peter Hickey

See Also

• \code{matrixStats::rowCummaxs()} and \code{matrixStats::colCummaxs()} which are used when the input is a matrix or numeric vector.

• For single maximum estimates, see \code{rowMaxs()}.

• \code{base::cummax()}.

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
as.integer((0:4) ^ 2),
seq(-5L, -1L, 1L)),
ncol = 3))

# A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
as.integer((0:4) ^ 2),
seq(-5L, -1L, 1L)),
ncol = 3))

colCummaxs(dm_matrix)

colCummins(dm_matrix)
colCumprods(dm_matrix)

colCumsums(dm_matrix)

# Only use rows 2-4
rowCummaxs(dm_Matrix, rows = 2:4)

# Only use rows 2-4
rowCummins(dm_Matrix, rows = 2:4)

# Only use rows 2-4
rowCumprods(dm_Matrix, rows = 2:4)

# Only use rows 2-4
rowCumsums(dm_Matrix, rows = 2:4)

---

**Usage**

```r
# S4 method for signature 'DelayedMatrix'
colDiffs(
  x,
  rows = NULL,
  cols = NULL,
  lag = 1L,
  differences = 1L,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)

# S4 method for signature 'DelayedMatrix'
rowDiffs(
  x,
  rows = NULL,
  cols = NULL,
  lag = 1L,
  differences = 1L,
  force_block_processing = FALSE,
  ...,
```
Arguments

- **x**: A NxK DelayedMatrix.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **lag**: An integer specifying the lag.
- **differences**: An integer specifying the order of difference.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on `getAutoBlockSize()`) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- **...**: Additional arguments passed to specific methods.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowDiffs / matrixStats::colDiffs.

Value

Returns a numeric matrix with one column (row) less than x: $N(K - 1)$ or $(N - 1)K$.

Author(s)

Peter Hickey

See Also

- matrixStats::rowDiffs() and matrixStats::colDiffs() which are used when the input is a matrix or numeric vector.
- base::diff().

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
as.integer((0:4) ^ 2),
seq(-5L, -1L, 1L)),
ncol = 3))
# A DelayedMatrix with a 'HDF5ArraySeed' seed
# NOTE: Requires that the HDF5Array package is installed
library(HDF5Array)
dm_HDF5 <- writeHDF5Array(matrix(c(rep(1L, 5),
    as.integer((0:4) ^ 2),
    seq(-5L, -1L, 1L)),
    ncol = 3))

colDiffs(dm_matrix)
rowDiffs(dm_HDF5)
# In reverse column order
rowDiffs(dm_HDF5, cols = seq(ncol(dm_HDF5), 1, -1))

colIQRDiffs,DelayedMatrix-method

Calculates the interquartile range of the difference between each element of a row (column) of a matrix-like object

Description

Calculates the interquartile range of the difference between each element of a row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'
colIQRDiffs(
    x,
    rows = NULL,
    cols = NULL,
    na.rm = FALSE,
    diff = 1L,
    trim = 0,
    force_block_processing = FALSE,
    ...,
    useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
colMadDiffs(
    x,
    rows = NULL,
    cols = NULL,
    na.rm = FALSE,
    diff = 1L,
    trim = 0,
    force_block_processing = FALSE,
    ...,
## S4 method for signature 'DelayedMatrix'
\texttt{colSdDiffs(}
  \texttt{x,}
  \texttt{rows = NULL,}
  \texttt{cols = NULL,}
  \texttt{na.rm = FALSE,}
  \texttt{diff = 1L,}
  \texttt{trim = 0,}
  \texttt{force\_block\_processing = FALSE,}
  \texttt{...,}
  \texttt{useNames = TRUE}
\texttt{)}

## S4 method for signature 'DelayedMatrix'
\texttt{colVarDiffs(}
  \texttt{x,}
  \texttt{rows = NULL,}
  \texttt{cols = NULL,}
  \texttt{na.rm = FALSE,}
  \texttt{diff = 1L,}
  \texttt{trim = 0,}
  \texttt{force\_block\_processing = FALSE,}
  \texttt{...,}
  \texttt{useNames = TRUE}
\texttt{)}

## S4 method for signature 'DelayedMatrix'
\texttt{rowIQRDiffs(}
  \texttt{x,}
  \texttt{rows = NULL,}
  \texttt{cols = NULL,}
  \texttt{na.rm = FALSE,}
  \texttt{diff = 1L,}
  \texttt{trim = 0,}
  \texttt{force\_block\_processing = FALSE,}
  \texttt{...,}
  \texttt{useNames = TRUE}
\texttt{)}

## S4 method for signature 'DelayedMatrix'
\texttt{rowMadDiffs(}
  \texttt{x,}
  \texttt{rows = NULL,}
  \texttt{cols = NULL,}
  \texttt{na.rm = FALSE,}
Arguments

\textbf{x}  
A N\times K \texttt{DelayedMatrix}.

\textbf{rows, cols}  
A \texttt{vector} indicating the subset of rows (and/or columns) to operate over. If \texttt{NULL}, no subsetting is done.

\textbf{na.rm}  
If \texttt{TRUE}, missing values (\texttt{NA} or \texttt{NaN}) are omitted from the calculations.

\textbf{diff}  
An integer specifying the order of difference.

\textbf{trim}  
A double in [0,1/2] specifying the fraction of observations to be trimmed from each end of (sorted) \texttt{x} before estimation.

\textbf{force_block_processing}  
\texttt{FALSE} (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to \texttt{TRUE} (typically not advised). The block-processing strategy loads one or more (depending on \texttt{\link[DelayedArray]{getAutoBlockSize}()}) columns (\texttt{colFoo()}) or rows (\texttt{rowFoo()}) into memory as an ordinary \texttt{base::array}. 
Additional arguments passed to specific methods.

useNames

If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowIQRDiffs / matrixStats::colIQRDiffs.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

• matrixStats::rowIQRDiffs() and matrixStats::colIQRDiffs() which are used when the input is a matrix or numeric vector.

• For the direct interquartile range see also rowIQRs.

Examples

# A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
                   as.integer((0:4) ^ 2),
                   seq(-5L, -1L, 1L)),
                   ncol = 3))

# A DelayedMatrix with a 'SolidRleArraySeed' seed
dm_Rle <- RleArray(Rle(c(rep(1L, 5),
                        as.integer((0:4) ^ 2),
                        seq(-5L, -1L, 1L))),
                   dim = c(5, 3))

colIQRDiffs(dm_Matrix)
colMadDiffs(dm_Matrix)
colSdDiffs(dm_Matrix)
colVarDiffs(dm_Matrix)

# Only using rows 2-4
rowIQRDiffs(dm_Rle, rows = 2:4)

# Only using rows 2-4
rowMadDiffs(dm_Rle, rows = 2:4)
# Only using rows 2-4
rowSdDiffs(dm_Rle, rows = 2:4)

# Only using rows 2-4
rowVarDiffs(dm_Rle, rows = 2:4)

colIQRs,DelayedMatrix-method

Calculates the interquartile range for each row (column) of a matrix-like object

Description

Calculates the interquartile range for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colIQRs(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowIQRs(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., useNames = TRUE
)
```

Arguments

- **x**
  - A NxK `DelayedMatrix`.
- **rows, cols**
  - A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **na.rm**
  - If TRUE, missing values (NA or NaN) are omitted from the calculations.
force_block_processing
   FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on `\link[DelayedArray]{getAutoBlockSize}()`) columns (`colFoo()`) or rows (`rowFoo()`) into memory as an ordinary `base::array`.

useNames
   If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details
   The S4 methods for x of type `matrix`, `array`, `table`, or `numeric` call `matrixStats::rowIQRs` / `matrixStats::colIQRs`.

Value
   Returns a numeric vector of length N (K).

Author(s)
   Peter Hickey

See Also
   • `matrixStats::rowIQRs()` and `matrixStats::colIQRs()` which are used when the input is a matrix or numeric vector.
   • For a non-robust analog, see `rowSds()`. For a more robust version see `rowMads()`
   • `stats::IQR()`.

Examples
   # A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
              as.integer((0:4) ^ 2),
              seq(-5L, -1L, 1L)),
              ncol = 3))

   # A DelayedMatrix with a 'Matrix' seed
dM_matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
              as.integer((0:4) ^ 2),
              seq(-5L, -1L, 1L)),
              ncol = 3))

   colIQRs(dm_matrix)

   # Only using rows 2-4
   rowIQRs(dm_matrix, rows = 2:4)
colLogSumExps(DelayedMatrix-method

Accurately calculates the logarithm of the sum of exponentials for each row (column) of a matrix-like object

**Description**

Accurately calculates the logarithm of the sum of exponentials for each row (column) of a matrix-like object.

**Usage**

```r
## S4 method for signature 'DelayedMatrix'
colLogSumExps(
lx,
rows = NULL,
cols = NULL,
na.rm = FALSE,
force_block_processing = FALSE,
..., 
useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowLogSumExps(
lx,
rows = NULL,
cols = NULL,
na.rm = FALSE,
force_block_processing = FALSE,
..., 
useNames = TRUE
)
```

**Arguments**

- `lx`: A NxK `DelayedMatrix`. Typically, `lx` are \( \log(x) \) values.
- `rows`, `cols`: A vector indicating the subset (and/or columns) to operate over. If `NULL`, no subsetting is done.
- `na.rm`: If `TRUE`, missing values (`NA` or `NaN`) are omitted from the calculations.
- `force_block_processing`: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to `TRUE` (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (`colFoo()`) or rows (`rowFoo()`) into memory as an ordinary `base::array`. 

- `useNames`: TRUE
... Additional arguments passed to specific methods.

useNames If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowLogSumExps / matrixStats::colLogSumExps.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

- matrixStats::rowLogSumExps() and matrixStats::colLogSumExps() which are used when the input is a matrix or numeric vector.
- rowSums2()

Examples

```r
x <- DelayedArray(matrix(runif(10), ncol = 2))
colLogSumExps(log(x))
rowLogSumExps(log(x))
```

Description

Calculates the median absolute deviation for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colMads(  
  x,  
  rows = NULL,  
  cols = NULL,  
  center = NULL,  
  constant = 1.4826,
)```
Arguments

x
A NxK DelayedMatrix.

rows, cols
A vector indicating the subset of rows (and/or columns) to operate over. If
NULL, no subsetting is done.
colMads, DelayedMatrix-method

center (optional) the center, defaults to the row means
constant A scale factor. See stats::mad() for details.
na.rm If TRUE, missing values (NA or NaN) are omitted from the calculations.
force_block_processing FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

... Additional arguments passed to specific methods.

useNames If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowMads / matrixStats::colMads.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

• matrixStats::rowMads() and matrixStats::colMads() which are used when the input is a matrix or numeric vector.
• For mean estimates, see rowMeans2() and rowMeans().
• For non-robust standard deviation estimates, see rowSds().

Examples

# A DelayedMatrix with a 'data.frame' seed
dm_df <- DelayedArray(data.frame(C1 = rep(1L, 5),
C2 = as.integer((0:4) ^ 2),
C3 = seq(-5L, -1L, 1L)))

# A DelayedMatrix with a 'DataFrame' seed
dm_DF <- DelayedArray(S4Vectors::DataFrame(C1 = rep(1L, 5),
C2 = as.integer((0:4) ^ 2),
C3 = seq(-5L, -1L, 1L)))

colMads(dm_df)
colSds(dm_df)
rowMads(dm_DF)
rowSds(dm_DF)

## S4 method for signature 'DelayedMatrix'
colMeans2(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'Matrix'
colMeans2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)

colMeans2,DelayedMatrix-method

Calculates the mean for each row (column) of a matrix-like object

Description

Calculates the mean for each row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'
colMeans2(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'Matrix'
colMeans2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)

## S4 method for signature 'SolidRleArraySeed'
colMeans2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)
Arguments

- **x**: A NxK `DelayedMatrix`.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
- **na.rm**: If `TRUE`, missing values (NA or NaN) are omitted from the calculations.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to `TRUE` (typically not advised). The block-processing strategy loads one or more (depending on `\link[DelayedArray]{getAutoBlockSize}`) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary `base::array`.
- **useNames**: If `NA`, the default behavior of the function about naming support is remained. If `FALSE`, no naming support is done. Else if `TRUE`, names attributes of result are set.

Details

The S4 methods for `x` of type `matrix`, `array`, `table`, or `numeric` call `matrixStats::rowMeans2` / `matrixStats::colMeans2`.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

- `matrixStats::rowMeans2()` and `matrixStats::colMeans2()` which are used when the input is a matrix or numeric vector.
- See also `rowMeans()` for the corresponding function in base R.
- For variance estimates, see `rowVars()`.
- See also the base R version `base::rowMeans()`.

Examples

```r
# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
     as.integer((0:4) ^ 2),
     seq(-5L, -1L, 1L)),
     ncol = 3))
# A DelayedMatrix with a 'SolidRleArraySeed' seed
dm_Rle <- RleArray(Rle(c(rep(1L, 5),
     as.integer((0:4) ^ 2)),
```
colMedians(DelayedMatrix-method)

Calculates the median for each row (column) of a matrix-like object

Description

Calculates the median for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colMedians(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowMedians(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

Arguments

- **x**: A NxK DelayedMatrix.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **na.rm**: If TRUE, missing values (NA or NaN) are omitted from the calculations.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- **...**: Additional arguments passed to specific methods.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowMedians / matrixStats::colMedians.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

- matrixStats::rowMedians() and matrixStats::colMedians() which are used when the input is a matrix or numeric vector.
- For mean estimates, see rowMeans2() and rowMeans().

Examples

# A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
                                          as.integer(c(0:4) ^ 2),
                                          seq(-5L, -1L, 1L)),
                                          ncol = 3))

colMedians(dm_Matrix)

rowMedians(dm_Matrix)
colOrderStats,DelayedMatrix-method

Calculates an order statistic for each row (column) of a matrix-like object

Description

Calculates an order statistic for each row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'

```
colOrderStats(
  x, 
  rows = NULL, 
  cols = NULL, 
  which, 
  force_block_processing = FALSE, 
  ..., 
  useNames = TRUE
)
```

## S4 method for signature 'DelayedMatrix'

```
rowOrderStats(
  x, 
  rows = NULL, 
  cols = NULL, 
  which, 
  force_block_processing = FALSE, 
  ..., 
  useNames = TRUE
)
```

Arguments

- `x` A NxK DelayedMatrix.
- `rows, cols` A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- `which` An integer index in [1,K] ([1,N]) indicating which order statistic to be returned
- `force_block_processing` FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- `...` Additional arguments passed to specific methods.
useNames  
If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details  
The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowOrderStats / matrixStats::colOrderStats.

Value  
Returns a numeric vector of length N (K).

Author(s)  
Peter Hickey

See Also  
• matrixStats::rowOrderStats() and matrixStats::colOrderStats() which are used when the input is a matrix or numeric vector.

Examples  
# A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
   as.integer((0:4) ^ 2),
   seq(-5L, -1L, 1L)),
   ncol = 3))
# Only using columns 2-3
colOrderStats(dm_Matrix, cols = 2:3, which = 1)

# Different algorithms, specified by "which", may give different results
rowOrderStats(dm_Matrix, which = 1)
rowOrderStats(dm_Matrix, which = 2)
Usage

```r
### S4 method for signature 'DelayedMatrix'
colProds(
x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  method = c("direct", "expSumLog"),
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)

### S4 method for signature 'SolidRleArraySeed'
colProds(
x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  method = c("direct", "expSumLog"),
  ...,
  useNames = TRUE
)

### S4 method for signature 'DelayedMatrix'
rowProds(
x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  method = c("direct", "expSumLog"),
  force_block_processing = FALSE,
  ...,
  useNames = TRUE
)
```

Arguments

- **x**
  - A NxK `DelayedMatrix`.

- **rows, cols**
  - A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.

- **na.rm**
  - If `TRUE`, missing values (NA or NaN) are omitted from the calculations.

- **method**
  - A character vector of length one that specifies the how the product is calculated. Note, that this is not a generic argument and not all implementation have to provide it.

- **force_block_processing**
  - FALSE (the default) means that a seed-aware, optimised method is used (if avail-
able). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

Additional arguments passed to specific methods.

\dontrun{

useNames

If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowProds / matrixStats::colProds.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

* matrixStats::rowProds() and matrixStats::colProds() which are used when the input is a matrix or numeric vector.
* For sums across rows (columns), see rowSums2() (colSums2())
* base::prod().

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5),
    as.integer((0:4) ^ 2),
    seq(-5L, -1L, 1L)),
    ncol = 3))

# A DelayedMatrix with a 'HDF5ArraySeed' seed
# NOTE: Requires that the HDF5Array package is installed
library(HDF5Array)
dm_HDF5 <- writeHDF5Array(matrix(c(rep(1L, 5),
    as.integer((0:4) ^ 2),
    seq(-5L, -1L, 1L)),
    ncol = 3))

colProds(dm_matrix)

colProds(dm_HDF5)

rowProds(dm_matrix)

rowProds(dm_HDF5)
colQuantiles, DelayedMatrix-method

Calculates quantiles for each row (column) of a matrix-like object

Description

Calculates quantiles for each row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'
colQuantiles(
  x,
  rows = NULL,
  cols = NULL,
  probs = seq(from = 0, to = 1, by = 0.25),
  na.rm = FALSE,
  type = 7L,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE,
  drop = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowQuantiles(
  x,
  rows = NULL,
  cols = NULL,
  probs = seq(from = 0, to = 1, by = 0.25),
  na.rm = FALSE,
  type = 7L,
  force_block_processing = FALSE,
  ...,
  useNames = TRUE,
  drop = TRUE
)

Arguments

- **x**: A N x K DelayedMatrix.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **probs**: A numeric vector of J probabilities in \([0, 1]\).
- **na.rm**: If TRUE, missing values (NA or NaN) are omitted from the calculations.
colQuantiles, DelayedMatrix-method

- **type**: An integer specifying the type of estimator. See `stats::quantile()` for more details.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on `DelayedArray::getAutoBlockSize()`) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary `base::array`.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.
- **drop**: If TRUE a vector is returned if J == 1.

### Details

The S4 methods for `x` of type `matrix`, `array`, `table`, or `numeric` call `matrixStats::rowQuantiles` / `matrixStats::colQuantiles`.

### Value

A numeric N×J (K×J) matrix, where N (K) is the number of rows (columns) for which the J values are calculated.

### Author(s)

Peter Hickey

### See Also

- `matrixStats::rowQuantiles()` and `matrixStats::colQuantiles()` which are used when the input is a matrix or numeric vector.
- `stats::quantile`

### Examples

```r
# A DelayedMatrix with a 'data.frame' seed
dm_df <- DelayedArray(data.frame(C1 = rep(1L, 5),
                             C2 = as.integer((0:4) ^ 2),
                             C3 = seq(-5L, -1L, 1L)))

# colnames, if present, are preserved as rownames on output
colQuantiles(dm_df)

# Input has no rownames so output has no rownames
rowQuantiles(dm_df)
```
colRanks,DelayedMatrix-method

Calculates the rank of the elements for each row (column) of a matrix-like object

Description

Calculates the rank of the elements for each row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'
colRanks(
  x,
  rows = NULL,
  cols = NULL,
  ties.method = c("max", "average", "first", "last", "random", "max", "min", "dense"),
  preserveShape = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowRanks(
  x,
  rows = NULL,
  cols = NULL,
  ties.method = c("max", "average", "first", "last", "random", "max", "min", "dense"),
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

Arguments

- **x** A N \times K DelayedMatrix.
- **rows, cols** A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **ties.method** A character string specifying how ties are treated. Note that the default specifies fewer options than the original matrixStats package.
- **preserveShape** If TRUE the output matrix has the same shape as the input x. Note, that this is not a generic argument and not all implementation of this function have to provide it.
force_block_processing
    FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

... Additional arguments passed to specific methods.

useNames
    If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details
The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowRanks / matrixStats::colRanks.

The matrixStats::rowRanks() function can handle a lot of different values for the ties.method argument. Users of the generic function should however only rely on max and average because the other ones are not guaranteed to be implemented:

max  for values with identical values the maximum rank is returned
average for values with identical values the average of the ranks they cover is returned. Note, that in this case the return value is of type numeric.

Value
    a matrix of type integer is returned unless ties.method = "average". It has dimensions' NxJ (KxJ) matrix, where N (K) is the number of rows (columns) of the input x.

Author(s)
    Peter Hickey

See Also
    • matrixStats::rowRanks() and matrixStats::colRanks() which are used when the input is a matrix or numeric vector.
    • base::rank

Examples
    # A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
                                          as.integer((0:4) ^ 2),
                                          seq(-5L, -1L, 1L)),
                                          ncol = 3))

colRanks(dm_Matrix)

rowRanks(dm_Matrix)
colSums2,DelayedMatrix-method

Calculates the sum for each row (column) of a matrix-like object

Description

Calculates the sum for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colSums2(x, rows = NULL, cols = NULL, na.rm = FALSE,
        force_block_processing = FALSE,
        ..., useNames = TRUE)

## S4 method for signature 'Matrix'
colSums2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)

## S4 method for signature 'SolidRleArraySeed'
colSums2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)

## S4 method for signature 'DelayedMatrix'
rowSums2(x, rows = NULL, cols = NULL, na.rm = FALSE,
         force_block_processing = FALSE,
         ..., useNames = TRUE)

## S4 method for signature 'Matrix'
rowSums2(x, rows = NULL, cols = NULL, na.rm = FALSE, ..., useNames = TRUE)
```

Arguments

- `x` A NxK `DelayedMatrix`.
- `rows, cols` A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
colSums2, DelayedMatrix-method

Arguments

na.rm If TRUE, missing values (NA or NaN) are omitted from the calculations.

force_block_processing FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary \code{base::array}.

useNames If \code{NA}, the default behavior of the function about naming support is remained. If \code{FALSE}, no naming support is done. Else if \code{TRUE}, names attributes of result are set.

Details

The S4 methods for \code{x} of type \code{matrix}, \code{array}, \code{table}, or \code{numeric} call \code{matrixStats::rowSums2} / \code{matrixStats::colSums2}.

Value

Returns a \code{numeric vector} of length \(N \times K\).

Author(s)

Peter Hickey

See Also

- \code{matrixStats::rowSums2()} and \code{matrixStats::colSums2()} which are used when the input is a matrix or numeric vector.
- For mean estimates, see \code{rowMeans2()} and \code{rowMeans()}.
- \code{base::sum()}.

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5), as.integer((0:4) ^ 2), seq(-5L, -1L, 1L)), ncol = 3))

# A DelayedMatrix with a 'Matrix' seed
dm_Matric <- DelayedArray(Matrix::Matrix(c(rep(1L, 5), as.integer((0:4) ^ 2), seq(-5L, -1L, 1L)), ncol = 3))

colSums2(dm_matrix)

# NOTE: Temporarily use verbose output to demonstrate which method is
# which method is being used
colTabulates,DelayedMatrix-method

Tabulates the values in a matrix-like object by row (column)

Description
Tabulates the values in a matrix-like object by row (column).

Usage

```r
## S4 method for signature 'DelayedMatrix'
colTabulates(
  x,
  rows = NULL,
  cols = NULL,
  values = NULL,
  force_block_processing = FALSE,
  ..., useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowTabulates(
  x,
  rows = NULL,
  cols = NULL,
  values = NULL,
  force_block_processing = FALSE,
  ..., useNames = TRUE
)
```

Arguments

- **x**: A NxK DelayedMatrix.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **values**: the values to search for.
force_block_processing
FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

... Additional arguments passed to specific methods.

useNames If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call\code{matrixStats::rowTabulates} /\code{matrixStats::colTabulates}.

Value

a numeric N\times J (K\times J) matrix, where N (K) is the number of rows (columns) for which the J values are calculated.

Author(s)

Peter Hickey

See Also

- \code{matrixStats::rowTabulates} and \code{matrixStats::colTabulates} which are used when the input is a matrix or numeric vector.
- \code{base::table}()

Examples

# A DelayedMatrix with a 'DataFrame' seed
dm_DF <- DelayedArray(S4Vectors::DataFrame(C1 = rep(1L, 5),
                               C2 = as.integer((0:4) ^ 2),
                               C3 = seq(-5L, -1L, 1L)))

colTabulates(dm_DF)
rowTabulates(dm_DF)
Description

Calculates the variance for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colVars(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  center = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowVars(
  x,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  center = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

Arguments

- `x`: A NxK `DelayedMatrix`.
- `rows, cols`: A vector indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
- `na.rm`: If `TRUE`, missing values (NA or NaN) are omitted from the calculations.
- `center`: (optional) the center, defaults to the row means.
- `force_block_processing`: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to `TRUE` (typically not advised). The block-processing strategy loads one or more (depending on `DelayedArray::getAutoBlockSize()`) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary `base::array`. 

... Additional arguments passed to specific methods.

useNames If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowVars / matrixStats::colVars.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

• matrixStats::rowVars() and matrixStats::colVars() which are used when the input is a matrix or numeric vector.

• For mean estimates, see rowMeans2() and rowMeans().

• For standard deviation estimates, see rowSds().

• stats::var().

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5), as.integer((0:4)^2), seq(-5L, -1L, 1L)), ncol = 3))

# A DelayedMatrix with a 'HDF5ArraySeed' seed
# NOTE: Requires that the HDF5Array package is installed
library(HDF5Array)
dm_HDF5 <- writeHDF5Array(matrix(c(rep(1L, 5), as.integer((0:4)^2), seq(-5L, -1L, 1L)), ncol = 3))

colVars(dm_matrix)

rowVars(dm_matrix)
CollectedMads, DelayedMatrix-method

Calculates the weighted median absolute deviation for each row (column) of a matrix-like object

Description

Calculates the weighted median absolute deviation for each row (column) of a matrix-like object.

Usage

## S4 method for signature 'DelayedMatrix'
colWeightedMads(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  constant = 1.4826,
  center = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowWeightedMads(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  constant = 1.4826,
  center = NULL,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

Arguments

x

A NxK DelayedMatrix.

w

A numeric vector of length K (N) that specifies by how much each element is weighted.

rows, cols

A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
colWeightedMads,DelayedMatrix-method

na.rm If TRUE, missing values (NA or NaN) are omitted from the calculations.
constant A scale factor. See stats::mad() for details.
center (optional) the center, defaults to the row means
force_block_processing FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

useNames If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowWeightedMads / matrixStats::colWeightedMads.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

• matrixStats::rowWeightedMads() and matrixStats::colWeightedMads() which are used when the input is a matrix or numeric vector.
• See also rowMads for the corresponding unweighted function.

Examples

# A DelayedMatrix with a 'matrix' seed
dm_matrix <- DelayedArray(matrix(c(rep(1L, 5), as.integer((0:4) ^ 2), seq(-5L, -1L, 1L)), ncol = 3))

colWeightedMads(dm_matrix, w = 1:5)
rowWeightedMads(dm_matrix, w = 3:1)
colWeightedMeans,DelayedMatrix-method

Calculates the weighted mean for each row (column) of a matrix-like object.

Description

Calculates the weighted mean for each row (column) of a matrix-like object.

Usage

```r
## S4 method for signature 'DelayedMatrix'
colWeightedMeans(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowWeightedMeans(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

Arguments

- **x**: A NxK DelayedMatrix.
- **w**: A numeric vector of length K (N) that specifies by how much each element is weighted.
- **rows, cols**: A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.
- **na.rm**: If TRUE, missing values (NA or NaN) are omitted from the calculations.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by
setting this to \code{TRUE} (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (\code{colFoo()}) or rows (\code{rowFoo()}) into memory as an ordinary \code{base::array}.

... Additional arguments passed to specific methods.

\code{useNames} If \code{NA}, the default behavior of the function about naming support is remained. If \code{FALSE}, no naming support is done. Else if \code{TRUE}, names attributes of result are set.

\section*{Details}

The S4 methods for \code{x} of type \code{matrix, array, table, or numeric} call \code{matrixStats::rowWeightedMeans} / \code{matrixStats::colWeightedMeans}.

\section*{Value}

Returns a \code{numeric vector} of length \(N\) (\(K\)).

\section*{Author(s)}

Peter Hickey

\section*{See Also}

\begin{itemize}
  \item \code{matrixStats::rowWeightedMeans()} and \code{matrixStats::colWeightedMeans()} which are used when the input is a matrix or numeric vector.
  \item See also \code{rowMeans2} for the corresponding unweighted function.
\end{itemize}

\section*{Examples}

\begin{verbatim}
# A DelayedMatrix with a 'Matrix' seed
dm_Matrix <- DelayedArray(Matrix::Matrix(c(rep(1L, 5),
                                          as.integer((0:4) ^ 2),
                                          seq(-5L, -1L, 1L)),
                                          ncol = 3))

colWeightedMeans(dm_Matrix)
# Specifying weights inversely proportional to rowwise variances
colWeightedMeans(dm_Matrix, w = 1 / rowVars(dm_Matrix))
rowWeightedMeans(dm_Matrix, w = 1:3)
\end{verbatim}

\section*{Description}

Calculates the weighted median for each row (column) of a matrix-like object.
Usage

```r
## S4 method for signature 'DelayedMatrix'
colWeightedMedians(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

```r
## S4 method for signature 'DelayedMatrix'
rowWeightedMedians(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)
```

Arguments

- **x**: A NxK `DelayedMatrix`.
- **w**: A `numeric` vector of length K (N) that specifies by how much each element is weighted.
- **rows, cols**: A `vector` indicating the subset of rows (and/or columns) to operate over. If `NULL`, no subsetting is done.
- **na.rm**: If `TRUE`, missing values (NA or NaN) are omitted from the calculations.
- **force_block_processing**: FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on `delayedarray::getAutoBlockSize()` columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.
- **...**: Additional arguments passed to specific methods.
- **useNames**: If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.
Details
The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowWeightedMedians / matrixStats::colWeightedMedians.

Value
Returns a numeric vector of length N (K).

Author(s)
Peter Hickey

See Also
• matrixStats::rowWeightedMedians() and matrixStats::colWeightedMedians() which are used when the input is a matrix or numeric vector.
• See also rowMedians for the corresponding unweighted function.

Examples
# A DelayedMatrix with a 'SolidRleArraySeed' seed
dm_Rle <- RleArray(Rle(c(rep(1L, 5),
as.integer((0:4) ^ 2),
seq(-5L, -1L, 1L))),
dim = c(5, 3))

# Specifying weights inversely proportional to rowwise MADs
colWeightedMedians(dm_Rle, w = 1 / rowMads(dm_Rle))

Description
Calculates the weighted standard deviation for each row (column) of a matrix-like object.

Usage
## S4 method for signature 'DelayedMatrix'
colWeightedSds(
x,
w = NULL,
rows = NULL,
cols = NULL,
na.rm = FALSE,
force_block_processing = FALSE,
colWeightedSds, DelayedMatrix-method

## S4 method for signature 'DelayedMatrix'
colWeightedVars(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

## S4 method for signature 'DelayedMatrix'
rowWeightedSds(
  x,
  w = NULL,
  rows = NULL,
  cols = NULL,
  na.rm = FALSE,
  force_block_processing = FALSE,
  ..., 
  useNames = TRUE
)

Arguments

**x**
A NxK DelayedMatrix.

**w**
A numeric vector of length K (N) that specifies by how much each element is weighted.

**rows, cols**
A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.

**na.rm**
If TRUE, missing values (NA or NaN) are omitted from the calculations.
force_block_processing

FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

useNames

If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

Details

The S4 methods for x of type matrix, array, table, or numeric call matrixStats::rowWeightedSds / matrixStats::colWeightedSds.

Value

Returns a numeric vector of length N (K).

Author(s)

Peter Hickey

See Also

• matrixStats::rowWeightedSds() and matrixStats::colWeightedSds() which are used when the input is a matrix or numeric vector.
• See also rowSds for the corresponding unweighted function.

Examples

# A DelayedMatrix with a 'SolidRleArraySeed' seed
dm_Rle <- RleArray(Rle(c(rep(1L, 5),
    as.integer((0:4) ^ 2),
    seq(-5L, -1L, 1L))),(dim = c(5, 3))

colWeightedSds(dm_Rle, w = 1 / rowMeans2(dm_Rle))

# Specifying weights inversely proportional to rowwise means
colWeightedVars(dm_Rle, w = 1 / rowMeans2(dm_Rle))

# Specifying weights inversely proportional to columnwise means
rowWeightedSds(dm_Rle, w = 1 / colMeans2(dm_Rle))

# Specifying weights inversely proportional to columnwise means
rowWeightedVars(dm_Rle, w = 1 / colMeans2(dm_Rle))
Description

DelayedMatrixStats is a port of the matrixStats API to work with DelayedMatrix objects from the DelayedArray package. High-performing functions operating on rows and columns of DelayedMatrix objects, e.g. colMedians() / rowMedians(), colRanks() / rowRanks(), and colSds() / rowSds(). Functions optimized per data type and for subsetted calculations such that both memory usage and processing time is minimized.

Description

Deprecated functions in package ‘DelayedMatrixStats’

These functions are provided for compatibility with older versions of ‘DelayedMatrixStats’ only, and will be defunct at the next release.

Usage

colAnyMissings(x, rows = NULL, cols = NULL, ..., useNames = NA)

rowAnyMissings(x, rows = NULL, cols = NULL, ..., useNames = NA)

## S4 method for signature 'DelayedMatrix'

colAnyMissings(  
x,  
rows = NULL,  
cols = NULL,  
force_block_processing = FALSE,  
...,  
useNames = TRUE
)

## S4 method for signature 'matrix'

colAnyMissings(x, rows = NULL, cols = NULL, ..., useNames = TRUE)

## S4 method for signature 'DelayedMatrix'

rowAnyMissings(  
x,  
rows = NULL,  
cols = NULL,
DelayedMatrixStats-deprecated

```r
force_block_processing = FALSE,
...,  
useNames = TRUE  
)
```

```r
## S4 method for signature 'matrix'
rowAnyMissings(x, rows = NULL, cols = NULL, ..., useNames = TRUE)
```

### Arguments

- **x**
  - A NxK DelayedMatrix.

- **rows, cols**
  - A vector indicating the subset of rows (and/or columns) to operate over. If NULL, no subsetting is done.

- **...**
  - Additional arguments passed to specific methods.

- **useNames**
  - If NA, the default behavior of the function about naming support is remained. If FALSE, no naming support is done. Else if TRUE, names attributes of result are set.

- **force_block Processing**
  - FALSE (the default) means that a seed-aware, optimised method is used (if available). This can be overridden to use the general block-processing strategy by setting this to TRUE (typically not advised). The block-processing strategy loads one or more (depending on \link[DelayedArray]{getAutoBlockSize}()) columns (colFoo()) or rows (rowFoo()) into memory as an ordinary base::array.

### Details

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- colAnyMissings: colAnyNAs
- rowAnyMissings: rowAnyNAs

### Value

Returns a logical vector of length N (K).

### See Also

- matrixStats::rowAnyNAs() and matrixStats::colAnyNAs() which are used when the input is a matrix or numeric vector.
- For checks if any element is equal to a value, see rowAnys().
- base::is.na() and base::any().

### Examples

```r
mat <- matrix(rnorm(15), nrow = 5, ncol = 3)
mat[2, 1] <- NA
mat[3, 3] <- Inf
```
from_DelayedArray_to_simple_seed_class

Coerce DelayedArray to its 'simple seed' form

Description

Coerce DelayedArray to its 'simple seed' form

Usage

from_DelayedArray_to_simple_seed_class(x, drop = FALSE, do_transpose = TRUE)

Arguments

x A DelayedArray
drop If TRUE the result is coerced to the lowest possible dimension
do_transpose Should transposed input be physically transposed?

Details

Like DelayedArray:::.from_DelayedArray_to_array but returning an object of the same class as seedClass(x) instead of an array. In doing so, all delayed operations are realised (including subsetting).

Value

An object of the same class as seedClass(x).

Note

Can be more efficient to leave the transpose implicit (do_transpose = FALSE) and switch from a row*() method to a col*() method (or vice versa).

Only works on DelayedArray objects with 'simple seeds'
objects exported from other packages

Description

These objects are imported from other packages. Follow the links below to see their documentation.

DelayedArray colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

subset_by_Nindex subset_by_Nindex

Description

subset_by_Nindex() is an internal generic function not aimed to be used directly by the user. It is basically an S4 generic for DelayedArray:::subset_by_Nindex.

Usage

subset_by_Nindex(x, Nindex)

Arguments

x An array-like object.
Nindex An unnamed list of subscripts as positive integer vectors, one vector per dimension in x. Empty and missing subscripts (represented by integer(0) and NULL list elements, respectively) are allowed. The subscripts can contain duplicated indices. They cannot contain NAs or non-positive values.

Details

subset_by_Nindex(x, Nindex) conceptually performs the operation x[Nindex[1], ..., Nindex[length(Nindex)]].

subset_by_Nindex() methods need to support empty and missing subscripts, e.g., subset_by_Nindex(x, list(NULL, integer(0))) must return an M x 0 object of class class(x) and subset_by_Nindex(x, list(integer(0), integer(0))) a 0 x 0 object of class class(x).

Also, subscripts are allowed to contain duplicate indices so things like subset_by_Nindex(x, list(c(1:3, 3:1), 2L)) need to be supported.

Value

A object of class class(x) of the appropriate type (e.g., integer, double, etc.). For example, if x is a data.frame representing an M x N matrix of integers, subset_by_Nindex(x, list(NULL, 2L)) must return its 2nd column as a data.frame with M rows and 1 column of type integer.
Index

* internal
  from_DelayedArray_to_simple_seed_class, 56
  reexports, 57

all, 4
any, 6, 55
array, 4, 6, 8, 10, 12, 15, 17, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53
base::array, 4, 6, 8, 9, 12, 15, 17, 20, 23, 24, 27, 29, 31, 32, 35, 37, 39, 41, 43, 44, 47, 49, 50, 53, 55
base::rank, 39
colAlls, 4
colAlls, DelayedMatrix-method, 3
colAnyMissings
  (DelayedMatrixStats-deprecated), 54
colAnyMissings, DelayedMatrix-method
  (DelayedMatrixStats-deprecated), 54
colAnyMissings, matrix-method
  (DelayedMatrixStats-deprecated), 54
colAnyNAs, 6, 55
colAnyNAs, DelayedMatrix-method, 5
colAnys, DelayedMatrix-method
  (colAlls, DelayedMatrix-method), 3
colAvgsPerRowSet, 8
colAvgsPerRowSet, DelayedMatrix-method, 7
colCollapse, 10
colCollapse, DelayedMatrix-method, 9
colCounts, 12
colCounts, DelayedMatrix-method, 11
colCummaxs, 13
colCummins, DelayedMatrix-method
  (colCummaxs, DelayedMatrix-method), 13
colCumprods, DelayedMatrix-method
  (colCummaxs, DelayedMatrix-method), 13
colCumsums, DelayedMatrix-method
  (colCummaxs, DelayedMatrix-method), 13
colDiffs, 17
colDiffs, DelayedMatrix-method, 16
colIQRDiffs, 21
colIQRDiffs, DelayedMatrix-method, 18
colIQRs, 23
colIQRs, DelayedMatrix-method, 22
colLogSumExps, 25
colLogSumExps, DelayedMatrix-method, 24
colMadDiffs, DelayedMatrix-method
  (colIQRDiffs, DelayedMatrix-method), 18
colMads, 27
colMads, DelayedMatrix-method, 25
colMaxs, 57
colMaxs (reexports), 57
colMeans2, 29
colMeans2, DelayedMatrix-method, 28
colMeans2, Matrix-method
  (colMeans2, DelayedMatrix-method), 28
colMeans2, SolidRleArraySeed-method
  (colMeans2, DelayedMatrix-method), 28
colMedians, 31
colMedians, DelayedMatrix-method, 30
colMins, 57
colMins (reexports), 57
colOrderStats, 33
colOrderStats, DelayedMatrix-method, 32

58
INDEX

colProds, 35
colProds,DelayedMatrix-method, 33
colProds, SolidRleArraySeed-method
(colProds,DelayedMatrix-method), 33
colQuantiles, 37
colQuantiles,DelayedMatrix-method, 36
colRanges, 57
colRanges (reexports), 57
colRanks, 39
colRanks,DelayedMatrix-method, 38
colSdDiffs,DelayedMatrix-method
(colIQRDiffs,DelayedMatrix-method), 18
colSds,DelayedMatrix-method
(colMads,DelayedMatrix-method), 25
colSums2, 41
colSums2(), 35
colSums2,DelayedMatrix-method, 40
colSums2, Matrix-method
(colSums2,DelayedMatrix-method), 40
colSums2, SolidRleArraySeed-method
(colSums2,DelayedMatrix-method), 40
colTabulates, 43
colTabulates,DelayedMatrix-method, 42
colVarDiffs,DelayedMatrix-method
(colIQRDiffs,DelayedMatrix-method), 18
colVars, 45
colVars,DelayedMatrix-method, 44
colWeightedMads, 47
colWeightedMads,DelayedMatrix-method, 46
colWeightedMeans, 49
colWeightedMeans,DelayedMatrix-method, 48
colWeightedMedians, 51
colWeightedMedians,DelayedMatrix-method, 49
colWeightedSds, 53
colWeightedSds,DelayedMatrix-method, 51
colWeightedVars,DelayedMatrix-method
(colWeightedSds,DelayedMatrix-method), 51
cummax, 15
data.frame, 57
DelayedArray, 56
DelayedMatrix, 4, 6, 7, 9, 11, 14, 17, 20, 22,
24, 26, 29, 31, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 55
DelayedMatrixStats, 54
DelayedMatrixStats-deprecated, 54
diff, 17
FALSE, 4, 6, 9, 12, 15, 17, 21, 23, 25, 27, 29,
31, 33, 35, 37, 39, 41, 43, 45, 47, 49,
50, 53, 55
from_DelayedArray_to_simple_seed_class,
56
integer, 8, 12, 39
IQR, 23
is.na, 6, 55
logical, 4, 6, 55
mad, 27, 47
matrix, 4, 6, 8, 10, 12, 15, 17, 21, 23, 25, 27,
29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
49, 51, 53
NA, 4, 6, 9, 11, 12, 15, 17, 20–25, 27, 29, 31,
33–37, 39, 41, 43–45, 47–50, 52, 53,
55
NaN, 4, 11, 20, 22, 24, 27, 29, 31, 34, 36, 41,
44, 47, 48, 50, 52
NULL, 4, 6, 8, 9, 11, 15, 17, 20, 22, 24, 26, 29,
31, 32, 34, 36, 38, 40, 42, 44, 46, 48,
50, 52, 55
numeric, 4, 6, 8, 10, 12, 15, 17, 21, 23, 25, 27,
29, 31, 33, 35, 37, 39, 41, 43, 45–53
prod, 35
quantile, 37
reexports, 57
rowAlls, 4, 12
rowAlls,DelayedMatrix-method
(colAlls,DelayedMatrix-method), 3
rowAnyMissings
(DelayedMatrixStats-deprecated), 54
rowAnyMissings,DelayedMatrix-method
(DelayedMatrixStats-deprecated), 54
rowAnyMissings,matrix-method
(DelayedMatrixStats-deprecated), 54
rowAnyNAs,6,55
rowAnyNAs,DelayedMatrix-method
(colAnyNAs,DelayedMatrix-method), 5
rowAnys,4,6,12,55
rowAnys,DelayedMatrix-method
(colAlls,DelayedMatrix-method), 3
rowAvgsPerColSet,8
rowAvgsPerColSet,DelayedMatrix-method
(colAvgsPerRowSet,DelayedMatrix-method), 7
rowCollapse,10
rowCollapse,DelayedMatrix-method
(colCollapse,DelayedMatrix-method), 9
rowCounts,12
rowCounts,DelayedMatrix-method
(colCounts,DelayedMatrix-method), 11
rowCummaxs,15
rowCummaxs,DelayedMatrix-method
(colCummaxs,DelayedMatrix-method), 13
rowCummins,DelayedMatrix-method
(colCummaxs,DelayedMatrix-method), 13
rowCumprods,DelayedMatrix-method
(colCummaxs,DelayedMatrix-method), 13
rowCumsums,DelayedMatrix-method
(colCummaxs,DelayedMatrix-method), 13
rowDiffs,17
rowDiffs,DelayedMatrix-method
(colDiffs,DelayedMatrix-method), 16
rowIQRDiffs,21
rowIQRDiffs,DelayedMatrix-method
(colIQRDiffs,DelayedMatrix-method), 18
rowIQRs,21,23
rowLogSumExps,25
rowLogSumExps,DelayedMatrix-method
(colLogSumExps,DelayedMatrix-method), 24
rowMadDiffs,DelayedMatrix-method
(colIQRDiffs,DelayedMatrix-method), 18
rowMads,27,47
rowMads(),23
rowMads,DelayedMatrix-method
(colMads,DelayedMatrix-method), 25
rowMaxs,15,57
rowMaxs,DelayedMatrix-method
(colMaxs,DelayedMatrix-method), 28
rowMeans,27,29,31,41,45
rowMeans,DelayedMatrix-method
(colMeans,DelayedMatrix-method), 28
rowMeans2,27,29,31,41,45,49
rowMeans2,DelayedMatrix-method
(colMeans2,DelayedMatrix-method), 28
rowMeans2,Matrix-method
(colMeans2,DelayedMatrix-method), 28
rowMedians,31,51
rowMedians,DelayedMatrix-method
(colMedians,DelayedMatrix-method), 30
rowMins,57
rowMins(reexports),57
rowOrderStats,33
rowOrderStats,DelayedMatrix-method
(colOrderStats,DelayedMatrix-method), 32
rowProds,35
rowProds,DelayedMatrix-method
(colProds,DelayedMatrix-method), 33
rowQuantiles,37
rowQuantiles,DelayedMatrix-method
(colQuantiles,DelayedMatrix-method), 36
rowRanges,57
rowRanges(reexports),57
rowRanks,39
rowRanks,DelayedMatrix-method
(colRanks,DelayedMatrix-method), 45
INDEX

rowSdDiffs,DelayedMatrix-method
  (colIQRDiffs,DelayedMatrix-method), 18
rowSds, 23, 27, 45, 53
rowSds,DelayedMatrix-method
  (colMads,DelayedMatrix-method), 25
rowSums2, 35, 41
rowSums2(), 25
rowSums2,DelayedMatrix-method
  (colSums2,DelayedMatrix-method), 40
rowSums2,Matrix-method
  (colSums2,DelayedMatrix-method), 40
rowTabulates, 43
rowTabulates,DelayedMatrix-method
  (colTabulates,DelayedMatrix-method), 42
rowVarDiffs,DelayedMatrix-method
  (colIQRDiffs,DelayedMatrix-method), 18
rowVars, 29, 45
rowVars,DelayedMatrix-method
  (colVars,DelayedMatrix-method), 44
rowWeightedMads, 47
rowWeightedMads,DelayedMatrix-method
  (colWeightedMads,DelayedMatrix-method), 46
rowWeightedMeans, 49
rowWeightedMeans,DelayedMatrix-method
  (colWeightedMeans,DelayedMatrix-method), 48
rowWeightedMedians, 51
rowWeightedMedians,DelayedMatrix-method
  (colWeightedMedians,DelayedMatrix-method), 49
rowWeightedSds, 53
rowWeightedSds,DelayedMatrix-method
  (colWeightedSds,DelayedMatrix-method), 51
rowWeightedVars,DelayedMatrix-method
  (colWeightedSds,DelayedMatrix-method), 51

stats::quantile, 37
subset_by_Nindex, 57