

# Package ‘matrixStats’

January 2, 2012

**Version** 0.4.3

**Date** 2011-12-11

**Title** Methods that apply to rows and columns of a matrix

**Author** Henrik Bengtsson, Hector Corrada Bravo, Robert Gentleman, Harris Jaffee

**Maintainer** Henrik Bengtsson <henrikb@braju.com>

**Depends** R (>= 2.9.0), methods, R.methodsS3 (>= 1.2.1)

**Suggests** aroma.light (>= 1.22.0)

**Description** This packages provides methods operating on rows and columns of matrices. The objective is to have all methods being optimized for speed and memory consumption.

**License** Artistic-2.0

**LazyLoad** TRUE

**biocViews** Infrastructure, Statistics

**Repository** CRAN

**Date/Publication** 2011-12-13 08:36:35

## R topics documented:

matrixStats-package . . . . .	2
anyMissing . . . . .	4
indexByRow . . . . .	4
rowCollapse . . . . .	5
rowCounts . . . . .	6
rowDiffs . . . . .	7
rowIQRs . . . . .	8
rowMedians . . . . .	9
rowOrderStats . . . . .	10
rowProds . . . . .	11

rowQuantiles . . . . .	11
rowRanges . . . . .	12
rowRanks . . . . .	13
rowSds . . . . .	14
rowTabulates . . . . .	15
rowVars . . . . .	16
rowWeightedMeans.matrix . . . . .	17
rowWeightedMedians.matrix . . . . .	19
varDiff . . . . .	20

<b>Index</b>	<b>22</b>
--------------	-----------

---

matrixStats-package    *Package matrixStats*

---

## Description

This packages provides methods operating on rows and columns of matrices. The objective is to have all methods being optimized for speed and memory consumption.. This package went public (on CRAN) in June 2009. It is currently in a beta version where new methods are added.

## Installation

To install this package, please do:

```
install.packages("matrixStats")
```

## Dependancies and other requirements

See DESCRIPTION file for now.

## To get started

### Counts and logicals:

- rowCounts()
- rowAlls()
- rowAnys()

### Sums and products:

- rowSums()
- rowProds()

### Estimates of the mean:

- rowMeans()
- rowMedians()

- rowWeightedMeans()
- rowWeightedMedians()

**Estimates of the standard deviation, variance and more:**

- rowSds()
- rowMads()
- rowVars()
- rowIQRs()
- rowQuantiles()
- rowOrderStats()
- rowRanks()
- rowRanges()
- rowMins()
- rowMaxs()

**Miscellaneous:**

- rowDiffs()
- anyMissing()

**How to contribute**

This is an open-source project which embraces collaborations. If you have improvements on code and/or documentation, or new function, please consider contributing them to this package.

**For developers**

It is currently not decided whether the methods should be S4 or S3 methods. This is the reason why some methods are based on S4 and some on S3. The ones using S3 rely on the **R.methodsS3** package to define the methods. There are also dependancies on other packages. The plan is to remove all such dependancies as soon as the API settles, but until then, we keep the dependancies for conveniency and in order to avoid reduncancy of available implementations of identical methods.

**How to cite this package**

Henrik Bengtsson, Hector Corrada Bravo, Robert Gentleman and Harris Jaffee (2011). matrixStats: Methods that apply to rows and columns of a matrix. R package version 0.4.2.

**Author(s)**

Henrik Bengtsson, Hector Corrada Bravo, Robert Gentleman, Harris Jaffee.

---

anyMissing	<i>Checks if there are any missing values in an object or not</i>
------------	---

---

**Description**

Checks if there are any missing values in an object or not.

**Usage**

```
anyMissing(x, ...)
```

**Arguments**

x	A <a href="#">vector</a> , a <a href="#">list</a> , a <a href="#">matrix</a> , a <a href="#">data.frame</a> , or <a href="#">NULL</a> .
...	Not use.

**Details**

The implementation of this method is optimized for both speed and memory. The method will return [TRUE](#) at the first detected missing value.

**Value**

Returns [TRUE](#) if a missing value was detected, otherwise [FALSE](#).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```
x <- rnorm(n=1000)
x[seq(300,length(x),by=100)] <- NA
stopifnot(anyMissing(x) == any(is.na(x)))
```

---

indexByRow	<i>Translates matrix indices by rows into indices by columns</i>
------------	--

---

**Description**

Translates matrix indices by rows into indices by columns.

**Usage**

```
indexByRow(x, idxs=NULL, ...)
```

**Arguments**

`x` A [matrix](#).  
`idxs` A [vector](#) of indices. If `NULL`, all indices are returned.  
... Not use.

**Value**

Returns an [integer vector](#) of indices.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```
x <- matrix(NA, nrow=5, ncol=4)
y <- t(x)
idxs <- seq(along=x)

# Assign by columns
x[idxs] <- idxs
print(x)

# Assign by rows
y[indexByRow(y, idxs)] <- idxs
print(y)

stopifnot(x == t(y))
```

---

`rowCollapse`*Extracts one cell per row (column) from a matrix*

---

**Description**

Extracts one cell per row (column) from a matrix. The implementation is optimized for memory and speed.

**Usage**

```
rowCollapse(x, idxs, ...)
colCollapse(x, idxs, ...)
```

**Arguments**

`x` An `NxK` [matrix](#).  
`idxs` An index [vector](#) of (maximum) length `N` (`K`) specifying the columns (rows) to be extracted.  
... Not used.

**Value**

Returns a [vector](#) of length N (K).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

*Matrix indexing* to index elements in matrices and arrays, cf. [\[\]](#).

**Examples**

```
x <- matrix(1:27, ncol=3)

y <- rowCollapse(x, 1)
stopifnot(identical(y, x[,1]))

y <- rowCollapse(x, 2)
stopifnot(identical(y, x[,2]))

y <- rowCollapse(x, c(1,1,1,1,1,3,3,3,3))
stopifnot(identical(y, c(x[1:5,1], x[6:9,3])))

y <- rowCollapse(x, 1:3)
print(y)
yT <- c(x[1,1],x[2,2],x[3,3],x[4,1],x[5,2],x[6,3],x[7,1],x[8,2],x[9,3])
stopifnot(identical(y, yT))
```

---

rowCounts

*Counts the number of TRUE values in each row (column) of a matrix*

---

**Description**

Counts the number of TRUE values in each row (column) of a matrix.

**Usage**

```
rowCounts(x, na.rm=FALSE, ...)
colCounts(x, na.rm=FALSE, ...)
rowAlls(x, na.rm=FALSE, ...)
colAlls(x, na.rm=FALSE, ...)
rowAnys(x, na.rm=FALSE, ...)
colAnys(x, na.rm=FALSE, ...)
```

**Arguments**

`x` A logical NxK matrix.  
`na.rm` If TRUE, NAs are excluded first, otherwise not.  
`...` Not used.

**Value**

`rowCounts()` (`colCounts()`) returns an integer vector of length N (K). The other methods returns a logical vector of length N (K).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```
x <- matrix(FALSE, nrow=10, ncol=5)
x[3:7,c(2,4)] <- TRUE
x[2:4,] <- TRUE
x[,1] <- TRUE
x[5,] <- FALSE
x[,5] <- FALSE

print(x)

print(rowCounts(x))      # 1 4 4 4 0 3 3 1 1 1
print(colCounts(x))     # 9 5 3 5 0

print(rowAnys(x))
print(which(rowAnys(x))) # 1 2 3 4 6 7 8 9 10
print(colAnys(x))
print(which(colAnys(x))) # 1 2 3 4
```

---

rowDiffs

*Calculates difference for each row (column) in a matrix*


---

**Description**

Calculates difference for each row (column) in a matrix.

**Usage**

```
rowDiffs(x, ...)
colDiffs(x, ...)
```

**Arguments**

`x` A numeric NxK matrix.  
`...` Not used.

**Value**

Returns a `numeric`  $N \times (K-1)$  or  $(N-1) \times K$  `matrix`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

Internally `diff()` is used.

---

`rowIQRs`*Estimates of the interquartile range for each row (column) in a matrix*

---

**Description**

Estimates of the interquartile range for each row (column) in a matrix.

**Usage**

```
rowIQRs(x, ...)  
colIQRs(x, ...)
```

**Arguments**

`x`                    A `numeric`  $N \times K$  `matrix`.  
`...`                Additional arguments passed to `rowQuantiles()` (`colQuantiles()`).

**Value**

Returns a `numeric vector` of length  $N$  ( $K$ ).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See `IQR`. See `rowSds()`.

## Examples

```
set.seed(1)

x <- matrix(rnorm(50*40), nrow=50, ncol=40)
str(x)

# Row IQRs
q <- rowIQRs(x)
print(q)
q0 <- apply(x, MARGIN=1, FUN=IQR)
stopifnot(all.equal(q0, q))

# Column IQRs
q <- colIQRs(x)
print(q)
q0 <- apply(x, MARGIN=2, FUN=IQR)
stopifnot(all.equal(q0, q))
```

---

rowMedians

*Calculates the median for each row (column) in a matrix*

---

## Description

Calculates the median for each row (column) in a matrix.

## Usage

```
rowMedians(x, na.rm=FALSE, ...)
colMedians(x, na.rm=FALSE, ...)
```

## Arguments

x	A numeric N x K matrix.
na.rm	If TRUE, NAs are excluded first, otherwise not.
...	Not used.

## Details

The implementation of rowMedians() and colMedians() is optimized for both speed and memory. To avoid coercing to doubles (and hence memory allocation), there is a special implementation for integer matrices. That is, if x is an integer matrix, then rowMedians(as.double(x)) (rowMedians(as.double(x))) would require three times the memory of rowMedians(x) (colMedians(x)), but all this is avoided.

## Value

Returns a numeric vector of length N (K).

**Author(s)**

Henrik Bengtsson and Harris Jaffee.

**See Also**

See `rowMedians()` and `colMedians()` for weighted medians. For mean estimates, see `rowMeans()` in `colSums()`.

---

rowOrderStats	<i>Gets an order statistic for each row (column) in a matrix</i>
---------------	--

---

**Description**

Gets an order statistic for each row (column) in a matrix.

**Usage**

```
rowOrderStats(x, which, ...)
colOrderStats(x, which, ...)
```

**Arguments**

<code>x</code>	A <b>numeric</b> $N \times K$ matrix.
<code>which</code>	An <b>integer</b> index in $[1, K]$ ( $[1, N]$ ) indicating which order statistic to be returned.
<code>...</code>	Not used.

**Details**

The implementation of `rowOrderStats()` is optimized for both speed and memory. To avoid coercing to **doubles** (and hence memory allocation), there is a unique implementation for **integer** matrices. Currently, `colOrderStats(x)` is calling `rowOrderStats(t(x))`.

**Value**

Returns a **numeric vector** of length  $N$  ( $K$ ).

**Missing values**

This method does *not* handle missing values, that is, the result corresponds to having `na.rm=FALSE` (if such an argument would be available).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>) The native implementation of `rowOrderStats()` was adopted from Robert Gentleman's `rowQ` in the **Biobase** package.

**See Also**

See `rowMeans()` in `colSums()`.

---

rowProds	<i>Calculates the product for each row (column) in a matrix</i>
----------	---

---

**Description**

Calculates the product for each row (column) in a matrix.

**Usage**

```
rowProds(x, ...)  
colProds(x, ...)
```

**Arguments**

x	A <a href="#">numeric NxK matrix</a> .
...	Arguments passed to <code>rowSums()</code> .

**Details**

Internally the product is calculated via the logarithmic transform, treating zeros and negative values specially.

**Value**

Returns a [numeric vector](#) of length N (K).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

---

rowQuantiles	<i>Estimates quantiles for each row (column) in a matrix</i>
--------------	--

---

**Description**

Estimates quantiles for each row (column) in a matrix.

**Usage**

```
rowQuantiles(x, probs=seq(from=0, to=1, by=0.25), ..., drop=TRUE)  
colQuantiles(x, ...)
```

**Arguments**

`x` A numeric  $N \times K$  matrix with  $N \geq 0$ .  
`probs` A numeric vector of  $J$  probabilities in  $[0,1]$ .  
`...` Additional arguments passed to `quantile`.  
`drop` If TRUE, singleton dimensions in the result are dropped, otherwise not.

**Value**

Returns a numeric  $J \times N$  ( $J \times K$ ) matrix.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

[quantile](#).

**Examples**

```
set.seed(1)

x <- matrix(rnorm(50*40), nrow=50, ncol=40)
str(x)

probs <- c(0.25,0.5,0.75)

# Row quantiles
q <- rowQuantiles(x, probs=probs)
print(q)
q0 <- apply(x, MARGIN=1, FUN=quantile, probs=probs)
stopifnot(all.equal(q0, t(q)))

# Column IQRs
q <- colQuantiles(x, probs=probs)
print(q)
q0 <- apply(x, MARGIN=2, FUN=quantile, probs=probs)
stopifnot(all.equal(q0, t(q)))
```

---

rowRanges

*Gets the range of values in each row (column) of a matrix*

---

**Description**

Gets the range of values in each row (column) of a matrix.

**Usage**

```

rowRanges(x, na.rm=FALSE, ...)
colRanges(x, na.rm=FALSE, ...)
rowMins(x, na.rm=FALSE, ...)
colMins(x, na.rm=FALSE, ...)
rowMaxs(x, na.rm=FALSE, ...)
colMaxs(x, na.rm=FALSE, ...)

```

**Arguments**

x	A <b>numeric</b> NxK <b>matrix</b> .
na.rm	If <b>TRUE</b> , <b>NA</b> s are excluded first, otherwise not.
...	Not used.

**Details**

The rowRanges() function uses the much faster rowOrderStats() if there are no missing values.

**Value**

rowRanges() (colRanges()) returns a **numeric** Nx2 (Kx2) **matrix**. rowMins()/rowMaxs() (colMins()/colMaxs()) returns a **numeric vector** of length N (K).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

[rowOrderStats\(\)](#) and [rowRanges\(\)](#).

---

rowRanks

*Gets the rank of each row (column) of a matrix*


---

**Description**

Gets the rank of each row (column) of a matrix.

**Usage**

```

rowRanks(x, ...)
colRanks(x, ...)

```

**Arguments**

x	A <b>numeric</b> or <b>integer</b> NxK <b>matrix</b> .
...	Not used.

**Details**

The row- (column-) ranks of  $x$  are collected as *rows* of the result matrix. The implementation is optimized for both speed and memory. To avoid coercing to `doubles` (and hence memory allocation), there is a unique implementation for `integer` matrices. Currently, `colRanks(x)` is just `rowRanks(t(x))`. Any `names` of  $x$  are ignored and absent in the result.

**Value**

`rowRanks()` (`colRanks()`) returns an `integer`  $N \times K$  ( $K \times N$ ) *matrix*.

**Missing and non- values**

These are ranked as NA, as with `na.last="keep"` in the `rank()` function.

**Ties**

Ties are ranked equally, as with setting `ties.method="max"` in the `rank()` function.

**Author(s)**

Hector Corrada Bravo and Harris Jaffee. The native implementation of `rowRanks()` was adapted from Robert Gentleman's `rowQ` in the **Biobase** package.

**See Also**

`rank()`. For developers, see also Section 'Utility functions' in 'Writing R Extensions manual', particularly the native functions `R_qsort_I()` and `R_qsort_int_I()`.

---

rowSds

*Standard deviation estimates for each row (column) in a matrix*


---

**Description**

Standard deviation estimates for each row (column) in a matrix.

**Usage**

```
rowSds(x, ...)
colSds(x, ...)
rowMads(x, centers=NULL, constant=1.4826, ...)
colMads(x, centers=NULL, constant=1.4826, ...)
```

**Arguments**

<code>x</code>	A <code>numeric</code> $N \times K$ <i>matrix</i> .
<code>centers</code>	A optional <code>numeric vector</code> of length $N$ ( $K$ ) with centers. If <code>NULL</code> , they are calculated using <code>rowMedians()</code> .
<code>constant</code>	A scale factor. See <code>mad</code> for details.
<code>...</code>	Additional arguments passed to <code>rowVars()</code> and <code>rowMedians()</code> , respectively.

**Value**

Returns a [numeric vector](#) of length N (K).

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

[sd](#), [mad](#) and [var](#). [rowIQRs\(\)](#).

---

rowTabulates	<i>Tabulates the values in a matrix by row (column)</i>
--------------	---

---

**Description**

Tabulates the values in a matrix by row (column).

**Usage**

```
rowTabulates(x, values=NULL, ...)  
colTabulates(x, values=NULL, ...)
```

**Arguments**

x	An <a href="#">integer</a> or <a href="#">raw</a> NxK <a href="#">matrix</a> .
values	An <a href="#">vector</a> of values of count. If <a href="#">NULL</a> , all (unique) values are counted.
...	Not used.

**Value**

Returns a NxJ (KxJ) [matrix](#) where J is the number of values counted.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```
x <- matrix(1:5, nrow=10, ncol=5)  
print(x)  
print(rowTabulates(x))  
print(colTabulates(x))  
# Count only certain values  
print(rowTabulates(x, values=1:3))
```

```
y <- as.raw(x)
dim(y) <- dim(x)
print(y)
print(rowTabulates(y))
print(colTabulates(y))
```

---

rowVars

*Variance estimates for each row (column) in a matrix*

---

### Description

Variance estimates for each row (column) in a matrix.

### Usage

```
rowVars(x, center=NULL, ...)
colVars(x, ...)
```

### Arguments

`x` A numeric NxK matrix.  
`center` (optional) The center, defaults to the row means.  
`...` Additional arguments passed to `rowMeans()` and `rowSums()`.

### Value

Returns a numeric vector of length N (K).

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### See Also

See `rowMeans()` and `rowSums()` in `colSums()`.

### Examples

```
set.seed(1)

x <- matrix(rnorm(20), nrow=5, ncol=4)
print(x)

# Row averages
print(rowMeans(x))
print(rowMedians(x))

# Column averages
print(colMeans(x))
```

```
print(colMedians(x))

# Row variabilities
print(rowVars(x))
print(rowSds(x))
print(rowMads(x))
print(rowIQRs(x))

# Column variabilities
print(rowVars(x))
print(colSds(x))
print(colMads(x))
print(colIQRs(x))

# Row ranges
print(rowRanges(x))
print(cbind(rowMins(x), rowMaxs(x)))
print(cbind(rowOrderStats(x, 1), rowOrderStats(x, ncol(x))))

# Column ranges
print(colRanges(x))
print(cbind(colMins(x), colMaxs(x)))
print(cbind(colOrderStats(x, 1), colOrderStats(x, nrow(x))))

x <- matrix(rnorm(2400), nrow=50, ncol=40)

# Row standard deviations
d <- rowDiffs(x)
s1 <- rowSds(d)/sqrt(2)
s2 <- rowSds(x)
print(summary(s1-s2))

# Column standard deviations
d <- colDiffs(x)
s1 <- colSds(d)/sqrt(2)
s2 <- colSds(x)
print(summary(s1-s2))
```

---

rowWeightedMeans.matrix

*Calculates the weighted means for each row (column) in a matrix*

---

## Description

Calculates the weighted means for each row (column) in a matrix.

**Usage**

```
## S3 method for class 'matrix'
rowWeightedMeans(x, w=NULL, na.rm=FALSE, ...)
## S3 method for class 'matrix'
colWeightedMeans(x, w=NULL, na.rm=FALSE, ...)
```

**Arguments**

x	A <b>numeric NxK matrix</b> .
w	A <b>numeric vector</b> of length K (N).
na.rm	If <b>TRUE</b> , missing values are excluded from the calculation, otherwise not.
...	Not used.

**Details**

The implementations of these methods are optimized for both speed and memory. If no weights are given, the corresponding `rowMeans()`/`colMeans()` is used.

**Value**

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

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See `rowMeans()` and `colMeans()` in `colSums()` for non-weighted means. See also [weighted.mean](#).

**Examples**

```
x <- matrix(rnorm(20), nrow=5, ncol=4)
print(x)

# Non-weighted row averages
xM0 <- rowMeans(x)
xM <- rowWeightedMeans(x)
stopifnot(all.equal(xM, xM0))

# Weighted row averages (uniform weights)
w <- rep(2.5, ncol(x))
xM <- rowWeightedMeans(x, w=w)
stopifnot(all.equal(xM, xM0))

# Weighted row averages (excluding some columns)
w <- c(1,1,0,1)
xM0 <- rowMeans(x[, (w == 1), drop=FALSE]);
xM <- rowWeightedMeans(x, w=w)
stopifnot(all.equal(xM, xM0))
```

```
# Weighted row averages (excluding some columns)
w <- c(0,1,0,0)
xM0 <- rowMeans(x[, (w == 1), drop=FALSE]);
xM <- rowWeightedMeans(x, w=w)
stopifnot(all.equal(xM, xM0))

# Weighted averages by rows and columns
w <- 1:4
xM1 <- rowWeightedMeans(x, w=w)
xM2 <- colWeightedMeans(t(x), w=w)
stopifnot(all.equal(xM2, xM1))
```

---

rowWeightedMedians.matrix

*Calculates the weighted medians for each row (column) in a matrix*

---

## Description

Calculates the weighted medians for each row (column) in a matrix.

## Usage

```
## S3 method for class 'matrix'
rowWeightedMedians(x, w=NULL, na.rm=FALSE, ...)
## S3 method for class 'matrix'
colWeightedMedians(x, w=NULL, na.rm=FALSE, ...)
```

## Arguments

x	A <a href="#">numeric NxK matrix</a> .
w	A <a href="#">numeric vector</a> of length K (N).
na.rm	If <b>TRUE</b> , missing values are excluded from the calculation, otherwise not.
...	Additional arguments passed to <a href="#">weightedMedian</a> .

## Details

The implementations of these methods are optimized for both speed and memory. If no weights are given, the corresponding [rowMedians\(\)](#)/[colMedians\(\)](#) is used.

## Value

Returns a [numeric vector](#) of length N (K).

## Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See `rowMedians()` and `colMedians()` for non-weighted medians. Internally, `weightedMedian` of **aroma.light** is used.

**Examples**

```

if (require("aroma.light")) {
x <- matrix(rnorm(20), nrow=5, ncol=4)
print(x)

# Non-weighted row averages
xM0 <- rowMedians(x)
xM <- rowWeightedMedians(x)
stopifnot(all.equal(xM, xM0))

# Weighted row averages (uniform weights)
w <- rep(2.5, ncol(x))
xM <- rowWeightedMedians(x, w=w)
stopifnot(all.equal(xM, xM0))

# Weighted row averages (excluding some columns)
w <- c(1,1,0,1)
xM0 <- rowMedians(x[, (w == 1), drop=FALSE]);
xM <- rowWeightedMedians(x, w=w)
stopifnot(all.equal(xM, xM0))

# Weighted row averages (excluding some columns)
w <- c(0,1,0,0)
xM0 <- rowMedians(x[, (w == 1), drop=FALSE]);
xM <- rowWeightedMedians(x, w=w)
stopifnot(all.equal(xM, xM0))

# Weighted averages by rows and columns
w <- 1:4
xM1 <- rowWeightedMedians(x, w=w)
xM2 <- colWeightedMedians(t(x), w=w)
stopifnot(all.equal(xM2, xM1))

}

```

---

varDiff

*Estimation of discrepancies based on sequential order differences in a vector*


---

**Description**

Estimation of discrepancies based on sequential order differences in a vector.

**Usage**

```
varDiff(x, na.rm=FALSE, diff=1, ...)  
sdDiff(x, na.rm=FALSE, diff=1, ...)  
madDiff(x, na.rm=FALSE, diff=1, ...)
```

**Arguments**

x	A <a href="#">numeric vector</a> of length N.
na.rm	If <a href="#">TRUE</a> , <a href="#">NAs</a> are excluded, otherwise not.
diff	The positional distance of elements for which the difference should be calculated.
...	Not used.

**Value**

Returns a [numeric](#) scalar.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See [diff\(\)](#).

# Index

- \*Topic **array**
  - rowCounts, 6
  - rowDiffs, 7
  - rowIQRs, 8
  - rowMedians, 9
  - rowOrderStats, 10
  - rowProds, 11
  - rowQuantiles, 11
  - rowRanges, 12
  - rowRanks, 13
  - rowSds, 14
  - rowVars, 16
  - rowWeightedMeans.matrix, 17
  - rowWeightedMedians.matrix, 19
- \*Topic **iteration**
  - anyMissing, 4
  - indexByRow, 4
  - rowCounts, 6
  - rowDiffs, 7
  - rowIQRs, 8
  - rowMedians, 9
  - rowOrderStats, 10
  - rowProds, 11
  - rowQuantiles, 11
  - rowRanges, 12
  - rowRanks, 13
  - rowSds, 14
  - rowVars, 16
  - rowWeightedMeans.matrix, 17
  - rowWeightedMedians.matrix, 19
  - varDiff, 20
- \*Topic **logic**
  - anyMissing, 4
  - indexByRow, 4
  - rowCounts, 6
- \*Topic **methods**
  - rowWeightedMeans.matrix, 17
  - rowWeightedMedians.matrix, 19
- \*Topic **package**
  - matrixStats-package, 2
- \*Topic **robust**
  - rowDiffs, 7
  - rowIQRs, 8
  - rowMedians, 9
  - rowOrderStats, 10
  - rowProds, 11
  - rowQuantiles, 11
  - rowRanges, 12
  - rowRanks, 13
  - rowSds, 14
  - rowVars, 16
  - rowWeightedMeans.matrix, 17
  - rowWeightedMedians.matrix, 19
  - varDiff, 20
- \*Topic **univar**
  - rowCounts, 6
  - rowDiffs, 7
  - rowIQRs, 8
  - rowMedians, 9
  - rowOrderStats, 10
  - rowProds, 11
  - rowQuantiles, 11
  - rowRanges, 12
  - rowRanks, 13
  - rowSds, 14
  - rowVars, 16
  - rowWeightedMeans.matrix, 17
  - rowWeightedMedians.matrix, 19
  - varDiff, 20
- \*Topic **utilities**
  - rowCollapse, 5
  - rowTabulates, 15
- [, 6
- anyMissing, 4
- anyMissing, character-method (anyMissing), 4
- anyMissing, complex-method (anyMissing), 4

- anyMissing, data.frame-method
  - (anyMissing), 4
- anyMissing, list-method (anyMissing), 4
- anyMissing, logical-method (anyMissing), 4
- anyMissing, matrix-method (anyMissing), 4
- anyMissing, NULL-method (anyMissing), 4
- anyMissing, numeric-method (anyMissing), 4
  
- colAlls (rowCounts), 6
- colAlls, matrix-method (rowCounts), 6
- colAnys (rowCounts), 6
- colAnys, matrix-method (rowCounts), 6
- colCollapse (rowCollapse), 5
- colCollapse, matrix-method (rowCollapse), 5
- colCounts (rowCounts), 6
- colCounts, matrix-method (rowCounts), 6
- colDiffs (rowDiffs), 7
- colIQRs (rowIQRs), 8
- colMads (rowSds), 14
- colMaxs (rowRanges), 12
- colMaxs, matrix-method (rowRanges), 12
- colMedians (rowMedians), 9
- colMedians, matrix-method (rowMedians), 9
- colMins (rowRanges), 12
- colMins, matrix-method (rowRanges), 12
- colOrderStats (rowOrderStats), 10
- colOrderStats, matrix-method (rowOrderStats), 10
- colProds (rowProds), 11
- colQuantiles (rowQuantiles), 11
- colRanges (rowRanges), 12
- colRanges, matrix-method (rowRanges), 12
- colRanks (rowRanks), 13
- colRanks, matrix-method (rowRanks), 13
- colSds (rowSds), 14
- colSds, matrix-method (rowSds), 14
- colSums, 10, 11, 16, 18
- colTabulates (rowTabulates), 15
- colTabulates, matrix-method (rowTabulates), 15
- colVars (rowVars), 16
- colVars, matrix-method (rowVars), 16
- colWeightedMeans
  - (rowWeightedMeans.matrix), 17
- colWeightedMedians
  - (rowWeightedMedians.matrix), 19
  
- data.frame, 4
- diff, 8, 21
- double, 9, 10, 14
  
- FALSE, 4
  
- indexByRow, 4
- indexByRow, matrix-method (indexByRow), 4
- integer, 5, 7, 9, 10, 13–15
- IQR, 8
  
- list, 4
- logical, 7
  
- mad, 14, 15
- madDiff (varDiff), 20
- madDiff, numeric-method (varDiff), 20
- matrix, 4, 5, 7–16, 18, 19
- matrixStats (matrixStats-package), 2
- matrixStats-package, 2
  
- NA, 7, 9, 13, 21
- names, 14
- NULL, 4, 5, 14, 15
- numeric, 7–16, 18, 19, 21
  
- quantile, 12
  
- rank, 14
- raw, 15
- rowAlls (rowCounts), 6
- rowAlls, matrix-method (rowCounts), 6
- rowAnys (rowCounts), 6
- rowAnys, matrix-method (rowCounts), 6
- rowCollapse, 5
- rowCollapse, matrix-method (rowCollapse), 5
- rowCounts, 6
- rowCounts, matrix-method (rowCounts), 6
- rowDiffs, 7
- rowDTabulates (rowTabulates), 15
- rowIQRs, 8, 15
- rowMads (rowSds), 14
- rowMaxs (rowRanges), 12
- rowMaxs, matrix-method (rowRanges), 12
- rowMedians, 9, 10, 14, 19, 20
- rowMedians, matrix-method (rowMedians), 9
- rowMins (rowRanges), 12
- rowMins, matrix-method (rowRanges), 12
- rowOrderStats, 10, 13

rowOrderStats, matrix-method  
    (rowOrderStats), 10

rowProds, 11

rowQ, 10, 14

rowQuantiles, 8, 11

rowRanges, 12, 13

rowRanges, matrix-method (rowRanges), 12

rowRanks, 13

rowRanks, matrix-method (rowRanks), 13

rowSds, 8, 14

rowSds, matrix-method (rowSds), 14

rowSums, 11

rowTabulates, 15

rowTabulates, matrix-method  
    (rowTabulates), 15

rowVars, 14, 16

rowVars, matrix-method (rowVars), 16

rowWeightedMeans  
    (rowWeightedMeans.matrix), 17

rowWeightedMeans.matrix, 17

rowWeightedMedians  
    (rowWeightedMedians.matrix), 19

rowWeightedMedians.matrix, 19

sd, 15

sdDiff (varDiff), 20

sdDiff, numeric-method (varDiff), 20

TRUE, 4, 7, 9, 13, 18, 19, 21

var, 15

varDiff, 20

varDiff, numeric-method (varDiff), 20

vector, 4–16, 18, 19, 21

weighted.mean, 18

weightedMedian, 19, 20