Package ‘matrixStats’

February 20, 2015

Version 0.14.0
Depends R (>= 2.9.0)
Imports methods
Suggests R.rsp, microbenchmark, R.devices, base64enc, ggplot2, knitr
VignetteBuilder R.rsp
Date 2015-02-13
Title Methods that Apply to Rows and Columns of a Matrix
Author Henrik Bengtsson [aut, cre, cph], Hector Corrada Bravo [ctb], Robert Gentleman [ctb], Ola Hossjer [ctb], Harris Jaffee [ctb], Peter Langfelder [ctb]
Maintainer Henrik Bengtsson <henrikb@braju.com>
Description Methods operating on rows and columns of matrices, e.g. col/rowMedians(), col/rowRanks(), and col/rowSds(). There are also some vector-based methods, e.g. binMeans(), madDiff() and weightedMedians(). All methods have been optimized for speed and memory usage.
License Artistic-2.0
LazyLoad TRUE
NeedsCompilation yes
ByteCompile TRUE
biocViews Infrastructure, Statistics
URL https://github.com/HenrikBengtsson/matrixStats
BugReports https://github.com/HenrikBengtsson/matrixStats/issues
Repository CRAN
Date/Publication 2015-02-14 01:18:55

R topics documented:

matrixStats-package .................................................. 2
anyMissing ................................................................. 3
binCounts ................................................................. 4
Description

Methods operating on rows and columns of matrices, e.g. col / rowMedians(), col / rowRanks(), and col / rowSds(). There are also some vector-based methods, e.g. binMeans(), madDiff() and weightedMedians(). All methods have been optimized for speed and memory usage.

Installation

To install this package, please do:

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

Vignettes

For an overview of the package, see the `vignettes`:

1. Summary of functions.
anyMissing

How to cite this package

Author(s)
Henrik Bengtsson, Hector Corrada Bravo, Robert Gentleman, Ola Hossjer, Harris Jaffee, Peter Langfelder

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

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

Usage
anyMissing(x, ...)  
colAnyMissings(x, ...)  
rowAnyMissings(x, ...)

Arguments
x A vector, a list, a matrix, a data.frame, or NULL.
... Not used.

Details
The implementation of this method is optimized for both speed and memory. The method will return TRUE as soon as a missing value is detected.

Value
Returns TRUE if a missing value was detected, otherwise FALSE.

Author(s)
Henrik Bengtsson

See Also
Starting with R v3.1.0, there is anyNA() in the base, which provides the same functionality as this function.
Examples

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

---

**binCounts**

*Fast element counting in non-overlapping bins*

**Description**

Counts the number of elements in non-overlapping bins

**Usage**

```r
binCounts(x, bx, right=FALSE, ...)
```

**Arguments**

- `x`: A numeric vector of K positions for to be binned and counted.
- `bx`: A numeric vector of B+1 ordered positions specifying the B bins \([bx[1], bx[2]), [bx[2], bx[3]), ..., [bx[B], bx[B+1]])\.
- `right`: If `TRUE`, the bins are right-closed (left open), otherwise left-closed (right open).
- `...`: Not used.

**Details**

`binCounts(x, bx, right=TRUE)` gives equivalent results as `rev(binCounts(-x, bx=rev(-bx), right=FALSE))`, but is faster and more memory efficient.

**Value**

Returns an integer vector of length B with non-negative integers.

**Missing and non-finite values**

Missing values in `x` are ignored/dropped. Missing values in `bx` are not allowed and gives an error.

**Author(s)**

Henrik Bengtsson

**See Also**

An alternative for counting occurrences within bins is `hist`, e.g. `hist(x, breaks=bx, plot=FALSE)$counts`. That approach is ~30-60% slower than `binCounts(..., right=TRUE)`.

To count occurrences of indices `x` (positive integers) in \([1, B])`, use `tabulate(x, nbins=B)`. where `x` does not have to be sorted first. For details, see `tabulate()`.

To average values within bins, see `binMeans()`.
binMeans

Fast mean calculations in non-overlapping bins

Description
Computes the sample means in non-overlapping bins

Usage
binMeans(y, x, bx, na.rm=TRUE, count=TRUE, right=FALSE, ...)

Arguments
y
A numeric vector of K values to calculate means on.

x
A numeric vector of K positions for to be binned.

bx
A numeric vector of B+1 ordered positions specifying the B bins \([bx[1], bx[2]], [bx[2], bx[3]], \ldots, [bx[B], bx[B+1]]\).

na.rm
If TRUE, missing values in y are dropped before calculating the mean, otherwise not.

count
If TRUE, the number of data points in each bins is returned as attribute count, which is an integer vector of length B.

right
If TRUE, the bins are right-closed (left open), otherwise left-closed (right open).

Details
binMeans(x, bx, right=TRUE) gives equivalent results as rev(binMeans(-x, bx=sort(-bx), right=FALSE)), but is faster.

Value
Returns a numeric vector of length B.

Missing and non-finite values
Data points where either of y and x is missing are dropped (and therefore are also not counted). Non-finite values in y are not allowed and gives an error. Missing values in bx are not allowed and gives an error.

Empty bins
Empty bins will get value NaN.

Author(s)
Henrik Bengtsson with initial code contributions by Martin Morgan [1].
References


See Also

`binCounts()`, `aggregate` and `mean()`.

Examples

```r
x <- 1:200
mu <- double(length(x))
mu[1:50] <- 5
mu[101:150] <- -5
y <- mu + rnorm(length(x))

# Binning
bx <- c(0,50,100,150,200)+0.5
yS <- binMeans(y, x=x, bx=bx)

plot(x,y)
for (kk in seq(along=yS)) {
  lines(bx[c(kk,kk+1)], yS[c(kk,kk)], col="blue", lwd=2)
}
```

indexByRow

*Translates matrix indices by rows into indices by columns*

Description

Translates matrix indices by rows into indices by columns.

Usage

```r
indexByRow(dim, idxs=NULL, ...)
```

Arguments

- `dim` A *numeric vector* of length two specifying the length of the "template" 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
logSumExp

Examples

dim <- c(5, 4)
X <- matrix(NA_integer_, nrow=dim[1], ncol=dim[2])
Y <- t(X)
idxs <- seq(along=X)

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

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

stopifnot(X == t(Y))

logSumExp

Accurately computes the logarithm of the sum of exponentials

Description

Accurately computes the logarithm of the sum of exponentials, that is, \( \log(\text{sum}(\exp(lx))) \). If \( lx = \log(x) \), then this is equivalently to calculating \( \log(\text{sum}(x)) \).

This function, which avoid numerical underflow, is often used when computing the logarithm of the sum of small numbers (\( |x| << 1 \)) such as probabilities.

Usage

logSumExp(lx, na.rm=FALSE, ...)

Arguments

lx

A numeric vector. Typically \( lx \) are \( \log(x) \) values.

na.rm

If TRUE, any missing values are ignored, otherwise not.

...

Not used.

Details

This is function is more accurate than \( \log(\text{sum}(\exp(lx))) \) when the values of \( x = \exp(lx) \) are \( |x| << 1 \). The implementation of this function is based on the observation that

\[
\log(a + b) = [la = \log(a)], lb = \log(b)] = \log(\exp(la) + \exp(lb)) = la + \log(1 + \exp(lb - la))
\]

Assuming \( la > lb \), then \( |lb - la| < |lb| \), and it is less likely that the computation of \( 1 + \exp(lb - la) \) will not underflow/overflow numerically. Because of this, the overall result from this function should be more accurate. Analogously to this, the implementation of this function finds the maximum value of \( lx \) and subtracts it from the remaining values in \( lx \).
Value

Returns a numeric scalar.

Benchmarking

This method is optimized for correctness, that avoiding underflowing. It is implemented in native code that is optimized for speed and memory.

Author(s)

Henrik Bengtsson

References


See Also

To compute this function on rows or columns of a matrix, see rowLogSumExps().

For adding two double values in native code, R provides the C function logspace_add() [1]. For properties of the log-sum-exponential function, see [2].

Examples

```r
## EXAMPLE #1
lx <- c(1000.01, 1000.02)
y0 <- log(sum(exp(lx)))
print(y0) ## Inf

y1 <- logSumExp(lx)
print(y1) ## 1000.708

## EXAMPLE #2
lx <- c(-1000.01, -1000.02)
y0 <- log(sum(exp(lx)))
print(y0) ## -Inf

y1 <- logSumExp(lx)
print(y1) ## -999.3218

## EXAMPLE #3
## R-help thread 'Beyond double-precision?' on May 9, 2009.
```
set.seed(1)
x <- runif(50)

# The logarithm of the harmonic mean
y0 <- log(1/mean(1/x))
print(y0)  ## -1.600885

lx <- log(x)
y1 <- log(length(x)) - logSumExp(-lx)
print(y1)  ## [1] -1.600885

# Sanity check
stopifnot(all.equal(y1, y0))

---

**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**

```r
rowCollapse(x, idxs, dim.=dim(x), ...)
colCollapse(x, idxs, dim.=dim(x), ...)
```

**Arguments**

- **x**  
  An NxK matrix.

- **idxs**  
  An index vector of (maximum) length N (K) specifying the columns (rows) to be extracted.

- **dim.**  
  An integer vector of length two specifying the dimension of x, also when not a matrix.

- **...**  
  Not used.

**Value**

Returns a vector of length N (K).

**Author(s)**

Henrik Bengtsson

**See Also**

*Matrix indexing* to index elements in matrices and arrays, cf. `[]`. 
Examples

```r
x <- matrix(1:27, ncol=3)
y <- rowCounts(x, 1)
stopifnot(identical(y, x[,1]))

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

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

y <- rowCounts(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

```r
count(x, value=TRUE, na.rm=FALSE, ...)  
rowCounts(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)  
colCounts(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)  
rowAlls(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)  
colAlls(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)  
rowAnys(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)  
colAnys(x, value=TRUE, na.rm=FALSE, dim=dim(x), ...)
```

### Arguments

- **x** An NxK **matrix** or an N*K **vector**.
- **value** A value to search for.
- **na.rm** If TRUE, NAs are excluded first, otherwise not.
- **dim** An integer vector of length two specifying the dimension of x, also when not a **matrix**.
- **...** Not used.

### Details

These functions take either a @matrix or a @vector as input. If a @vector, then argument dim must be specified and fulfill prod(dim) == length(x). The result will be identical to the results obtained when passing matrix(x, nrow=dim[1L], ncol=dim[2L]), but avoids having to temporarily create/allocate a @matrix, if only such is needed only for these calculations.
Value

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

Author(s)

Henrik Bengtsson

Examples

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

print(x)

print(rowCounts(x)) # 1 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
```

Description

Cumulative sums, products, minima and maxima for each row (column) in a matrix.

Usage

```r
rowCumsums(x, dim.=dim(x), ...)
colCumsums(x, dim.=dim(x), ...)
rowCumprods(x, dim.=dim(x), ...)
colCumprods(x, dim.=dim(x), ...)
rowCummins(x, dim.=dim(x), ...)
colCummins(x, dim.=dim(x), ...)
rowCummaxs(x, dim.=dim(x), ...)
colCummaxs(x, dim.=dim(x), ...)
```
Arguments

- **x**: A numeric NxK matrix.
- **dim**: An integer vector of length two specifying the dimension of x, also when not a matrix.
- **...**: Not used.

Value

Returns a numeric NxK matrix of the same mode as x.

Author(s)

Henrik Bengtsson

See Also

See `cumsum()`, `cumprod()`, `cummin()`, and `cummax()`.

Examples

```r
x <- matrix(1:12, nrow=4, ncol=3)
print(x)

yr <- rowCumsums(x)
print(yr)

yc <- colCumsums(x)
print(yc)

yr <- rowCumprods(x)
print(yr)

yc <- colCumprods(x)
print(yc)

yr <- rowCummaxs(x)
print(yr)

yc <- colCummaxs(x)
print(yc)

yr <- rowCummins(x)
print(yr)

yc <- colCummins(x)
print(yc)
```
Description

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

Usage

```r
rowDiffs(x, lag=1L, differences=1L, ...)
coldiffs(x, lag=1L, differences=1L, ...)
```

Arguments

- `x`: A numeric NxK matrix.
- `lag`: An integer specifying the lag.
- `differences`: An integer specifying the order of difference.
- `...`: Not used.

Value

Returns a numeric N×(K-1) or (N-1)×K matrix.

Author(s)

Henrik Bengtsson

See Also

See also `diff2()`.

Examples

```r
x <- matrix(1:27, ncol=3)
d1 <- rowDiffs(x)
print(d1)

d2 <- t(coldiffs(t(x))
stopifnot(all.equal(d2, d1))
```
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, na.rm=FALSE, ...)  
colIQRs(x, na.rm=FALSE, ...)  
iqr(x, na.rm=FALSE, ...)

Arguments

x A numeric NxK matrix.
na.rm If TRUE, missing values are dropped first, otherwise not.
... Additional arguments passed to rowQuantiles() (colQuantiles()).

Value

Returns a numeric vector of length N (K).

Missing values

Contrary to IQR, which gives an error if there are missing values and na.rm=FALSE, iqr() and its corresponding row and column-specific functions return NA_real_.

Author(s)

Henrik Bengtsson

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))
### Description

Accurately computes the logarithm of the sum of exponentials across rows or columns.

### Usage

```r
rowLogSumExps(lx, na.rm=FALSE, dim.=dim(lx), ...) 
colLogSumExps(lx, na.rm=FALSE, dim.=dim(lx), ...) 
```

### Arguments

- `lx`: A numeric N\times K matrix. Typically `lx` are `log(x)` values.
- `na.rm`: If `TRUE`, any missing values are ignored, otherwise not.
- `dim.`: An integer vector of length two specifying the dimension of `x`, also when not a matrix.
- `...`: Not used.

### Value

A numeric vector of length N (K).

### Benchmarking

These methods are implemented in native code and have been optimized for speed and memory.

### Author(s)

Native implementation by Henrik Bengtsson. Original R code by Nakayama ??? (Japan).

### See Also

To calculate the same on vectors, `logSumExp()`.
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, dim.=dim(x), ...)
colMedians(x, na.rm=FALSE, dim.=dim(x), ...)
```

**Arguments**

- `x`: A numeric NxK matrix.
- `na.rm`: If TRUE, NAs are excluded first, otherwise not.
- `dim.`: An integer vector of length two specifying the dimension of `x`, also when not a matrix.
- `...`: 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))` (or `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, Harris Jaffee

**See Also**

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

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

---

**Description**

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

**Usage**

```r
rowOrderStats(x, which, dim.=dim(x), ...)  
colOrderStats(x, which, dim.=dim(x), ...)
```

**Arguments**

- **x**: A *numeric* NxK matrix.
- **which**: An *integer* index in [1.K] ([1.N]) indicating which order statistic to be returned.
- **dim.**: An *integer vector* of length two specifying the dimension of x, also when not a matrix.
- **...**: 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.

**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)**

The native implementation of `rowOrderStats()` was adopted by Henrik Bengtsson from Robert Gentleman’s `rowQ()` in the **Biobase** package.

**See Also**

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

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

Description

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

Usage

rowProds(x, na.rm=FALSE, method=c("direct", "expSumLog"), ...)
colProds(x, na.rm=FALSE, method=c("direct", "expSumLog"), ...)
product(x, na.rm=FALSE, ...)

Arguments

x
A numeric NxK matrix.

na.rm
If TRUE, missing values are ignored, otherwise not.

method
A character string specifying how each product is calculated.

... Not used.

Details

If method="expSumLog", then then product() function is used, which calculates the produce via the logarithmic transform (treating negative values specially). This improves the precision and lowers the risk for numeric overflow. If method="direct", the direct product is calculated via the prod() function.

Value

Returns a numeric vector of length N (K).

Missing values

Note, if method="expSumLog", na.rm=FALSE, and x contains missing values (NA or NaN), then the calculated value is also missing value. Note that it depends on platform whether NaN or NA is returned when an NaN exists, cf. is.nan().

Author(s)

Henrik Bengtsson
rowQuantiles

Estimates quantiles for each row (column) in a matrix

Description

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

Usage

rowQuantiles(x, probs=seq(from = 0, to = 1, by = 0.25), na.rm=FALSE, type=7L, ..., drop=TRUE)
colQuantiles(x, probs=seq(from = 0, to = 1, by = 0.25), na.rm=FALSE, type=7L, ..., drop=TRUE)

Arguments

x A numeric NxK matrix with N >= 0.
probs A numeric vector of J probabilities in [0,1].
na.rm If TRUE, NAs are excluded first, otherwise not.
type An integer specify the type of estimator. See quantile for more details.
... Additional arguments passed to quantile.
drop If TRUE, singleton dimensions in the result are dropped, otherwise not.

Value

Returns a numeric NxJ (KxJ) matrix, where N (K) is the number of rows (columns) for which the J quantiles are calculated.

Author(s)

Henrik Bengtsson

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)
rowRanges

*Description*

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

*Usage*

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

*Arguments*

- `x` A numeric NxK matrix.
- `na.rm` If TRUE, NAs are excluded first, otherwise not.
- `dim.` An integer vector of length two specifying the dimension of x, also when not a matrix.
- `...` Not used.

*Value*

- `rowRanges()` (colRanges()) returns a numeric Nx2 (Kx2) matrix, where N (K) is the number of rows (columns) for which the ranges are calculated.
- `rowMins()`/rowMaxs() (colMins()/colMaxs()) returns a numeric vector of length N (K).

*Author(s)*

Henrik Bengtsson

*See Also*

rowOrderStats() and pmin.int().
**Description**

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

**Usage**

```r
colRanks(x, ties.method=c("max", "average", "min"), dim.=dim(x), ...) rowRanks(x, ties.method=c("max", "average", "min"), dim.=dim(x), preserveShape=FALSE, ...)
```

**Arguments**

- `x` A numeric or integer NxK matrix.
- `ties.method` A character string specifying how ties are treated. For details, see below.
- `dim.` An integer vector of length two specifying the dimension of `x`, also when not a matrix.
- `preserveShape` A logical specifying whether the matrix returned should preserve the input shape of `x`, or not.
- `...` Not used.

**Details**

The row ranks of `x` are collected as rows of the result matrix.

The column ranks of `x` are collected as rows if `preserveShape = FALSE`, otherwise as columns.

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. It is more memory efficient to do `colRanks(x, preserveShape=TRUE)` than `t(colRanks(x, preserveShape=FALSE))`. Any names of `x` are ignored and absent in the result.

**Value**

An integer matrix is returned. The `rowRanks()` function always returns an NxK matrix, where `N` (`K`) is the number of rows (columns) whose ranks are calculated.

The `colRanks()` function returns an NxK matrix, if `preserveShape = TRUE`, otherwise a KxN matrix.

**Missing and non-values**

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

When some values are equal ("ties"), argument ties.method specifies what their ranks should be. If ties.method is "max", ties are ranked as the maximum value. If ties.method is "average", ties are ranked by their average. If ties.method is "max" ("min"), ties are ranked as the maximum (minimum) value. If ties.method is "average", ties are ranked by their average. For further details, see rank().

Author(s)

Hector Corrada Bravo and Harris Jaffee. Peter Langfelder for adding 'ties.method' support. Henrik Bengtsson adapted the original native implementation of rowRanks() 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**

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

**Usage**

rowSds(x, ...)
colsds(x, ...)
rowMads(x, center=NULL, constant=1.4826, na.rm=FALSE, dim.=dim(x), centers=NULL, ...)
colMads(x, center=NULL, constant=1.4826, na.rm=FALSE, dim.=dim(x), centers=NULL, ...)

**Arguments**

- **x** A numeric NxK matrix.
- **center** A optional numeric vector of length N (K) with centers. By default, they are calculated using rowMedians().
- **constant** A scale factor. See mad for details.
- **na.rm** If TRUE, missing values are removed first, otherwise not.
- **dim.** An integer vector of length two specifying the dimension of x, also when not a matrix.
- **...** Additional arguments passed to rowVars() and rowMedians(), respectively.
- **centers** (deprectated) use center instead.
Value
Returns a numeric vector of length N (K).

Author(s)
Henrik Bengtsson

See Also
sd, mad and var.rowIQRs().

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

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

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

Arguments
x An integer or raw NxK matrix.
values An vector of J values of count. If NULL, all (unique) values are counted.
... Not used.

Value
Returns a NxJ (KxJ) matrix where N (K) is the number of row (column) vectors tabulated and J is the number of values counted.

Author(s)
Henrik Bengtsson

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))
rowVars

Variance estimates for each row (column) in a matrix

Description

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

Usage

rowVars(x, na.rm=FALSE, center=NULL, dim.=dim(x), ...)
colVars(x, na.rm=FALSE, center=NULL, dim.=dim(x), ...)

Arguments

x A numeric N x K matrix.
center (optional) The center, defaults to the row means.
na.rm If TRUE, NAs are excluded first, otherwise not.
dim. An integer vector of length two specifying the dimension of x, also when not a matrix.
... Additional arguments passed to rowMeans() and rowSums().

Value

Returns a numeric vector of length N (K).

Author(s)

Henrik Bengtsson

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

Description

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

Usage

rowWeightedMeans(x, w=NULL, na.rm=FALSE, ...)
colWeightedMeans(x, w=NULL, na.rm=FALSE, ...)

Arguments

x A numeric NxK matrix.
w A numeric vector of length K (N).
na.rm If TRUE, 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

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)
rowWeightedMedians

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

Description

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

Usage

rowWeightedMedians(x, w=NULL, na.rm=FALSE, ...)  
colWeightedMedians(x, w=NULL, na.rm=FALSE, ...)

Arguments

x  A numeric N×K matrix.
w  A numeric vector of length K (N).
na.rm  If TRUE, missing values are excluded from the calculation, otherwise not.
...  Additional arguments passed to weightedMedian().

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

See Also

See rowMedians() and colMedians() for non-weighted medians. Internally, weightedMedian() is used.
Examples

```r
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 scale based on sequential-order differences*

**Description**

Estimation of scale based on sequential-order differences, corresponding to the scale estimates provided by `var`, `sd`, `mad` and `IQR`.

**Usage**

```r
varDiff(x, na.rm=FALSE, diff=1L, trim=0, ...)
colVarDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
rowVarDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)

sdDiff(x, na.rm=FALSE, diff=1L, trim=0, ...)
colSdDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
```
varDiff

rowSdDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
madDiff(x, na.rm=FALSE, diff=1L, trim=0, constant=1.4826, ...)
colMadDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
rowMadDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
iqrDiff(x, na.rm=FALSE, diff=1L, trim=0, ...)
colIQRDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)
rowIQRDiffs(x, na.rm=FALSE, diff=1L, trim=0, ...)

Arguments

x A numeric vector of length N or a numeric NxK matrix.
na.rm If TRUE, NAs are excluded, otherwise not.
diff The positional distance of elements for which the difference should be calculated.
trim A double in [0,1/2] specifying the fraction of observations to be trimmed from each end of (sorted) x before estimation.
constant A scale factor adjusting for asymptotically normal consistency.
... Not used.

Details

Note that n-order difference MAD estimates, just like the ordinary MAD estimate by mad, apply a correction factor such that the estimates are consistent with the standard deviation under Gaussian distributions.

The interquartile range (IQR) estimates does not apply such a correction factor. If asymptotically normal consistency is wanted, the correction factor for IQR estimate is 1 / (2 * qnorm(3/4)), which is half of that used for MAD estimates, which is 1 / qnorm(3/4). This correction factor needs to be applied manually, i.e. there is no constant argument for the IQR functions.

Value

Returns a numeric vector of length 1, length N, or length K.

Author(s)

Henrik Bengtsson

References

See Also

For the corresponding non-differentiated estimates, see \texttt{var}, \texttt{sd}, \texttt{mad} and \texttt{IQR}. Internally, \texttt{diff2()} is used which is a faster version of \texttt{diff()}.

---

### weightedMad

#### Weighted Median Absolute Deviation (MAD)

**Description**

Computes a weighted MAD of a numeric vector.

**Usage**

\[
\text{weightedMad}(x, w, \text{na.rm}=\text{FALSE}, \text{constant}=1.4826, \text{center}=\text{NULL}, \ldots) \\
\text{colWeightedMads}(x, w=\text{NULL}, \text{na.rm}=\text{FALSE}, \ldots) \\
\text{rowWeightedMads}(x, w=\text{NULL}, \text{na.rm}=\text{FALSE}, \ldots)
\]

**Arguments**

- **x**: a numeric vector containing the values whose weighted MAD is to be computed.
- **w**: a vector of weights the same length as \(x\) giving the weights to use for each element of \(x\). Negative weights are treated as zero weights. Default value is equal weight to all values.
- **na.rm**: a logical value indicating whether NA values in \(x\) should be stripped before the computation proceeds, or not. If \(\text{NA}\), no check at all for NAs is done. Default value is \(\text{NA}\) (for efficiency).
- **constant**: A numeric scale factor, cf. \texttt{mad}.
- **center**: Optional numeric scalar specifying the center location of the data. If \texttt{NULL}, it is estimated from data.
- **...**: Not used.

**Value**

Returns a numeric scalar.

**Missing values**

Missing values are dropped at the very beginning, if argument \texttt{na.rm} is \texttt{TRUE}, otherwise not.

**Author(s)**

Henrik Bengtsson
See Also

For the non-weighted MAD, see mad. Internally weightedMedian() is used to calculate the weighted median.

Examples

```r
x <- 1:10
n <- length(x)

m1 <- mad(x)
m2 <- weightedMad(x)
stopifnot(identical(m1, m2))

w <- rep(1, times=n)
m1 <- weightedMad(x, w)
stopifnot(identical(m1,m2))

# All weight on the first value
w[1] <- Inf
m <- weightedMad(x, w)
stopifnot(m == 0)

# All weight on the first two values
w[1:2] <- Inf
m1 <- mad(x[1:2])
m2 <- weightedMad(x, w)
stopifnot(identical(m1,m2))

# All weights set to zero
w <- rep(0, times=n)
m <- weightedMad(x, w)
stopifnot(is.na(m))
```

 weightedMean  Weighted Arithmetic Mean

Description

Computes the weighted sample mean of a numeric vector.

Usage

weightedMean(x, w, na.rm=FALSE, refine=FALSE, ...)

Arguments

- **x**: a numeric vector containing the values whose weighted mean is to be computed.
- **w**: a vector of weights the same length as x giving the weights to use for each element of x. Negative weights are treated as zero weights. Default value is equal weight to all values.
- **na.rm**: a logical value indicating whether NA values in x should be stripped before the computation proceeds, or not. If NA, no check at all for NAs is done. Default value is NA (for efficiency).
- **refine**: If TRUE and x is numeric, then extra effort is used to calculate the average with greater numerical precision, otherwise not.
- **...**: Not used.

Value

Returns a numeric scalar.

Author(s)

Henrik Bengtsson

See Also

mean() and weighted.mean.

Examples

```r
x <- 1:10
n <- length(x)

w <- rep(1, times=n)
m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

# Pull the mean towards zero
w[1] <- 5
m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

# Put even more weight on the zero
w[1] <- 8.5
m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

# All weight on the first value
w[1] <- Inf
```
weightedMedian

m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

# All weight on the last value
w[1] <- 1
w[n] <- Inf
m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

# All weights set to zero
w <- rep(0, n)
m0 <- weighted.mean(x, w)
m1 <- weightedMean(x, w)
stopifnot(identical(m1,m0))

weightedMedian  Weighted Median Value

Description
Computes a weighted median of a numeric vector.

Usage
weightedMedian(x, w=rep(1, times = length(x)), na.rm=FALSE, interpolate=is.null(ties), ties=NULL, ...)

Arguments
x  a numeric vector containing the values whose weighted median is to be computed.
w  a vector of weights the same length as x giving the weights to use for each element of x. Negative weights are treated as zero weights. Default value is equal weight to all values.
na.rm  a logical value indicating whether NA values in x should be stripped before the computation proceeds, or not. If NA, no check at all for NAs is done. Default value is NA (for efficiency).
interpolate  If TRUE, linear interpolation is used to get a consistent estimate of the weighted median.
ties  If interpolate == FALSE, a character string specifying how to solve ties between two x’s that are satisfying the weighted median criteria. Note that at most two values can satisfy the criteria. When ties is "min", the smaller value of the two is returned and when it is "max", the larger value is returned. If ties is "mean", the mean of the two values is returned. Finally, if ties is "weighted" (or NULL) a weighted average of the two are returned, where the weights are weights of all values x[i] <= x[k] and x[i] >= x[k], respectively.
...  Not used.
Details

For the \( n \) elements \( x = (x[1], x[2], \ldots, x[n]) \) with positive weights \( w = (w[1], w[2], \ldots, w[n]) \) such that \( \sum w = S \), the \textit{weighted median} is defined as the element \( x[k] \) for which the total weight of all elements \( x[i] < x[k] \) is less or equal to \( S/2 \) and for which the total weight of all elements \( x[i] > x[k] \) is less or equal to \( S/2 \) (c.f. [1]).

If \( w \) is missing then all elements of \( x \) are given the same positive weight. If all weights are zero, \texttt{NA\_real} is returned.

If one or more weights are \( \text{Inf} \), it is the same as these weights have the same weight and the others has zero. This makes things easier for cases where the weights are result of a division with zero.

The weighted median solves the following optimization problem:

\[
\alpha^* = \arg \min_{\alpha} \sum_{k=1}^{K} w_k |x_k - \alpha|
\]

where \( x = (x_1, x_2, \ldots, x_K) \) are scalars and \( w = (w_1, w_2, \ldots, w_K) \) are the corresponding "weights" for each individual \( x \) value.

Value

Returns a \texttt{numeric} scalar.

Author(s)

Henrik Bengtsson and Ola Hössjer, Centre for Mathematical Sciences, Lund University. Thanks to Roger Koenker, Econometrics, University of Illinois, for the initial ideas.

References


See Also

\texttt{median}, \texttt{mean()} and \texttt{weightedMean()}. 

Examples

\begin{verbatim}
x <- 1:10
n <- length(x)

m1 <- median(x) # 5.5
m2 <- weightedMedian(x) # 5.5
stopifnot(identical(m1, m2))

w <- rep(1, n)
m1 <- weightedMedian(x, w) # 5.5 (default)
m2 <- weightedMedian(x, ties="weighted") # 5.5 (default)
m3 <- weightedMedian(x, ties="min") # 5
m4 <- weightedMedian(x, ties="max") # 6
\end{verbatim}
weightedVar

Weighted variance and weighted standard deviation

stopifnot(identical(m1,m2))

# Pull the median towards zero
w[1] <- 5
m1 <- weightedMedian(x, w)  # 3.5
y <- c(rep(0,w[1]), x[-1])   # Only possible for integer weights
m2 <- median(y)              # 3.5
stopifnot(identical(m1,m2))

# Put even more weight on the zero
w[1] <- 8.5
weightedMedian(x, w)         # 2

# All weight on the first value
w[1] <- Inf
weightedMedian(x, w)         # 1

# All weight on the last value
w[1] <- 1
w[n] <- Inf
weightedMedian(x, w)         # 10

# All weights set to zero
w <- rep(0, n)
weightedMedian(x, w)         # NA

# Simple benchmarking
bench <- function(N=1e5, K=10) {
  x <- rnorm(N)
gc()
t <- c()
t[1] <- system.time(for (k in 1:K) median(x))[3]
t[2] <- system.time(for (k in 1:K) weightedMedian(x))[3]
t <- t / t[1]
names(t) <- c("median", "weightedMedian")
t
}

d----------
print(bench(N= 5, K=100))
d----------
print(bench(N= 50, K=100))
d----------
print(bench(N= 200, K=100))
d----------
print(bench(N=1000, K=100))
d----------
print(bench(N=10e3, K= 20))
d----------
print(bench(N=100e3, K= 20))

d----------

Description

Computes a weighted variance / standard deviation of a numeric vector or across rows or columns of a matrix.

Usage

weightedVar(x, w, na.rm=FALSE, center=NULL, ...)
colWeightedVars(x, w=NULL, na.rm=FALSE, ...)
rowWeightedVars(x, w=NULL, na.rm=FALSE, ...)

weightedSd(...)
colWeightedSds(x, w=NULL, na.rm=FALSE, ...)
rowWeightedSds(x, w=NULL, na.rm=FALSE, ...)

Arguments

x a numeric vector containing the values whose weighted variance is to be computed.

w a vector of weights the same length as x giving the weights to use for each element of x. Negative weights are treated as zero weights. Default value is equal weight to all values.

na.rm a logical value indicating whether NA values in x should be stripped before the computation proceeds, or not. If NA, no check at all for NAs is done. Default value is NA (for efficiency).

center Optional numeric scalar specifying the center location of the data. If NULL, it is estimated from data.

... Not used.

Value

Returns a numeric scalar.

Missing values

Missing values are dropped at the very beginning, if argument na.rm is TRUE, otherwise not.

Author(s)

Henrik Bengtsson

See Also

For the non-weighted variance, see var.
Index

*Topic **array**
  rowCounts, 10
  rowCumsums, 11
  rowDiffs, 13
  rowIQRs, 14
  rowLogSumExp, 15
  rowMedians, 16
  rowOrderStats, 17
  rowProds, 18
  rowQuantile, 19
  rowRanges, 20
  rowRanks, 21
  rowSds, 22
  rowVars, 24
  rowWeightedMeans, 25
  rowWeightedMedians, 27

*Topic **iteration**
  anyMissing, 3
  indexByRow, 6
  rowCounts, 10
  rowCumsums, 11
  rowDiffs, 13
  rowIQRs, 14
  rowMedians, 16
  rowOrderStats, 17
  rowProds, 18
  rowQuantiles, 19
  rowRanges, 20
  rowRanks, 21
  rowSds, 22
  rowVars, 24
  rowWeightedMeans, 25
  rowWeightedMedians, 27
  varDiff, 28

*Topic **logic**
  anyMissing, 3
  indexByRow, 6
  rowCounts, 10

*Topic **package**
  matrixStats-package, 2

*Topic **robust**
  rowDiffs, 13
  rowIQRs, 14
  rowMedians, 16
  rowOrderStats, 17
  rowProds, 18
  rowQuantiles, 19
  rowRanges, 20
  rowRanks, 21
  rowSds, 22
  rowVars, 24
  rowWeightedMeans, 25
  rowWeightedMedians, 27
  varDiff, 28
  weightedMad, 30
  weightedMean, 31
  weightedMedian, 33
  weightedVar, 35

*Topic **univar**
  binCounts, 4
  binMeans, 5
  rowCounts, 10
  rowCumsums, 11
  rowDiffs, 13
  rowIQRs, 14
  rowMedians, 16
  rowOrderStats, 17
  rowProds, 18
  rowQuantiles, 19
  rowRanges, 20
  rowRanks, 21
  rowSds, 22
  rowVars, 24
  rowWeightedMeans, 25
  rowWeightedMedians, 27
  varDiff, 28
  weightedMad, 30
  weightedMean, 31
weightedMedian, 33
weightedVar, 35

*Topic utilities*
rowCollapse, 9
rowTabulates, 23
[
,
9
aggregate, 6
allValue (rowCounts), 10
anyMissing, 3
anyValue (rowCounts), 10
binCounts, 4, 6
binMeans, 4, 5
character, 18, 21
colAlls (rowCounts), 10
colAnyMissings (anyMissing), 3
colAnyS (rowCounts), 10
colCollapse (rowCollapse), 9
colCounts (rowCounts), 10
colCummaxs (rowCumsums), 11
colCummins (rowCumsums), 11
colCumprods (rowCumsums), 11
colCumsums (rowCumsums), 11
colDiffs (rowDiffs), 13
colIQRs (rowIQRs), 14
colLogSumExps (rowLogSumExps), 15
colLogSumExps, matrix-method (rowLogSumExps), 15
colMads (varDiffs), 28
colMads (rowSds), 22
colMaxs (rowRanges), 20
colMedians (rowMedians), 16
colMedians, matrix-method (rowMedians), 16
colMins (rowRanges), 20
colOrderStats (rowOrderStats), 17
colProds (rowProds), 18
colQuantiles (rowQuantiles), 19
colRanges (rowRanges), 20
colRanks (rowRanks), 21
colSds (rowSds), 22
colSds (varDiffs), 28
colSds, matrix-method (rowSds), 22
colSums, 16, 17, 24, 26
colTabulates (rowTabulates), 23
colVarDiffs (varDiffs), 28

colVars (rowVars), 24
colVars, matrix-method (rowVars), 24
colWeightedMads (weightedMad), 30
colWeightedMeans (rowWeightedMeans), 25
colWeightedMedians (rowWeightedMedians), 27
colWeightedSds (weightedVar), 35
colWeightedVars (weightedVar), 35
count (rowCounts), 10
cummax, 12
cummin, 12
cumprod, 12
cumsum, 12
data.frame, 3
diff, 30
diff2, 13, 30
double, 16, 17, 21, 29
FALSE, 3
hist, 4
indexByRow, 6
integer, 4–6, 9–13, 15–17, 19–24
IQR, 14, 28, 30
iqr (rowIQRs), 14
iqrDiffs (varDiffs), 28
is.nan, 18
list, 3
logical, 11, 21
logSumExp, 7, 15
mad, 22, 23, 28–31
madDiffs (varDiffs), 28
matrix, 3, 9, 10, 12–24, 26, 27, 29
matrixStats (matrixStats-package), 2
matrixStats-package, 2
mean, 6, 32, 34
median, 34
NA, 10, 14, 16, 18–20, 24, 29, 30, 32–34, 36
names, 21
NaN, 5, 18
NULL, 3, 6, 23, 30, 33, 36
numeric, 4–8, 12–24, 26, 27, 29, 30, 32–34, 36
pmin.int, 20
prod, 18
INDEX

product, 18
product (rowProds), 18
quantile, 19
rank, 21, 22
raw, 23
rowAlls (rowCounts), 10
rowAnyMissings (anyMissings), 3
rowAny (rowCounts), 10
rowCollapse, 9
rowCounts, 10
rowCummmaxs (rowCumsums), 11
rowCummins (rowCumsums), 11
rowCumprods (rowCumsums), 11
rowCumsums, 11
rowDiffs, 13
rowIQRDiffs (varDiff), 28
rowIQRs, 14, 27
rowLogSumExps, 8, 15
rowLogSumExps, matrix-method
 (rowLogSumExps), 15
rowMadDiffs (varDiff), 28
rowMads (rowSds), 22
rowMaxs (rowRanges), 20
rowMedians, 16, 16, 22, 27
rowMedians, matrix-method (rowMedians), 16
rowMins (rowRanges), 20
rowOrderStats, 17, 20
rowProds, 18
rowQuantiles, 14, 19
rowRanges, 20
rowRanks, 21
rowSdDiffs (varDiff), 28
rowSds, 14, 22
rowSds, matrix-method (rowSds), 22
rowTabulates, 23
rowVarDiffs (varDiff), 28
rowVars, 22, 24
rowVars, matrix-method (rowVars), 24
rowWeightedMads (weightedMad), 30
rowWeightedMeans, 25
rowWeightedMedians, 27
rowWeightedSds (weightedVar), 35
rowWeightedVars (weightedVar), 35
sd, 23, 28, 30
sdDiff (varDiff), 28
tabulate, 4
TRUE, 3–5, 7, 10, 14–16, 18–20, 22, 24, 26, 27,
29, 30, 32, 33, 36
var, 23, 28, 30, 36
varDiff, 28
vector, 3–7, 9–12, 14–24, 26, 27, 29, 30, 32,
33, 36
weighted.mean, 26, 32
weightedMads, 30
weightedMean, 31, 34
weightedMedian, 27, 31, 33
weightedSds (weightedVar), 35
weightedVar, 35

sd, 23, 28, 30
sdDiff (varDiff), 28