Package ‘spatstat.sparse’

March 16, 2021

Version 2.0-0
Date 2021-03-16
Title Sparse Three-Dimensional Arrays and Linear Algebra Utilities
Maintainer Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>
Depends R (>= 3.5.0), stats, utils, methods, Matrix, abind, tensor
Imports spatstat.utils (>= 2.1-0)
Description Defines sparse three-dimensional arrays and supports standard operations on them. The package also includes utility functions for matrix calculations that are common in statistics, such as quadratic forms.
License GPL (>= 2)
URL http://spatstat.org/
LazyData no
NeedsCompilation yes
ByteCompile true
BugReports https://github.com/spatstat/spatstat.sparse/issues
Author Adrian Baddeley [aut, cre, cph]
Rolf Turner [aut, cph] (<https://orcid.org/0000-0001-9499-8382>),
Ege Rubak [aut, cph] (<https://orcid.org/0000-0002-6675-533X>)
Repository CRAN
Date/Publication 2021-03-16 06:50:03 UTC

R topics documented:

  spatstat.sparse-package .................................................. 2
  aperm.sparse3Darray ...................................................... 4
  as.array.sparse3Darray .................................................. 5
  as.sparse3Darray ......................................................... 6
The `spatstat.sparse` package defines three-dimensional sparse arrays, and supports standard operations on them. It also provides some utility functions for matrix calculations such as quadratic forms.

**Sparse 3D Arrays**

The main purpose of `spatstat.sparse` is to define a class of sparse three-dimensional arrays. An array $A$ is three-dimensional if it is indexed by three integer indices, so that $A[i, j, k]$ specifies an element of the array. The array is called sparse if only a small fraction of the entries are non-zero. A sparse array can be represented economically by listing only the entries which are non-zero.

The `spatstat.sparse` package defines the class `sparse3Darray` of sparse three-dimensional arrays. These arrays can have numeric, integer, logical, or complex entries.

The package supports:

- creation of sparse arrays from raw data
- conversion to/from other data types
The `spatstat.sparse` package uses the `Matrix` package to handle slices of three-dimensional arrays which are two-dimensional (sparse matrices) or one-dimensional (sparse vectors).

The main functions are:

- `sparse3Darray` Create a sparse 3D array
- `as.sparse3Darray` Convert other data to a sparse 3D array
- `[.sparse3Darray` Subset operator
- `aperm.sparse3Darray` Permute a sparse array
- `Ops.sparse3Darray` arithmetic and logical operators
- `Complex.sparse3Darray` complex operators
- `Math.sparse3Darray` standard mathematical functions
- `Summary.sparse3Darray` mean, maximum etc
- `tensorSparse` Tensor product
- `as.array.sparse3Darray` Convert sparse array to full array

The class "sparse3Darray" has methods for `anyNA`, `dim`, `dim<-`, `dimnames`, `dimnames<-` and `print`, documented in `methods.sparse3Darray`.

For other undocumented functions, see `spatstat.sparse-internal`.

**Matrix Utilities**

The package also includes some utilities for matrix calculations:

- `sumouter` sum of outer products of rows of a matrix
- `quadform` quadratic form involving rows of a matrix
- `bilinearform` bilinear form involving rows of a matrix
- `matrixsqrt` square root of a matrix
- `matrixpower` powers of a matrix

**Licence**

This library and its documentation are usable under the terms of the “GNU General Public License”, a copy of which is distributed with R.

**Author(s)**

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>. 
Transposition of Sparse Array

Description

Transpose a sparse three-dimensional array by permuting its dimensions.

Usage

```r
## S3 method for class 'sparse3Darray'
aperm(a, perm = NULL, resize = TRUE, ...)
```

Arguments

- `a`: A sparse three-dimensional array (object of class "sparse3Darray").
- `perm`: The subscript permutation vector, a permutation of the integers 1:3.
- `resize`: Logical value specifying whether the dimensions and dimnames of the array should also be adjusted, by permuting them according to the permutation.
- `...`: Ignored.

Details

The function `aperm` is generic. This is the method for the class "sparse3Darray" of sparse three-dimensional arrays.

Value

Another sparse three-dimensional array (object of class "sparse3Darray").

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

`sparse3Darray`, `tensorSparse`.

Examples

```r
M <- sparse3Darray(i=1:4, j=sample(1:4, replace=TRUE),
                  k=c(1,2,1,2), x=1:4, dims=c(5,7,2))
dim(M)
P <- aperm(M, c(3,1,2))
dim(P)
```
as.array.sparse3Darray

Convert Sparse Array to Full Array

Description
Convert a sparse three-dimensional array to a full three-dimensional array.

Usage
```r
## S3 method for class 'sparse3Darray'
as.array(x, ...)
```

Arguments
- `x`  Sparse three-dimensional array (object of class "sparse3Darray").
- `...` Ignored.

Details
This is a method for the generic `as.array` for sparse three-dimensional arrays (class "sparse3Darray"). It converts the sparse three-dimensional array `x` into an `array` representing the same data.

Value
An array (class "array") with the same dimensions as `x` and the same type of entries as `x`.

Author(s)
Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also
- `sparse3Darray`, `as.sparse3Darray`

Examples
```r
M <- sparse3Darray(i=1:3, j=c(3,1,2), k=4:2,
                      x=runif(3), dims=rep(4,3))
V <- as.array(M)
```
as.sparse3Darray

**Convert Data to a Sparse Three-Dimensional Array**

**Description**

Convert other kinds of data to a sparse three-dimensional array.

**Usage**

```
as.sparse3Darray(x, ...)
```

**Arguments**

- `x` Data in another format (see Details).
- `...` Ignored.

**Details**

This function converts data in various formats into a sparse three-dimensional array (object of class "sparse3Darray").

The argument `x` can be

- a sparse three-dimensional array (class "sparse3Darray")
- an array
- a matrix, which will be interpreted as an array with dimension `c(dim(x),1)`
- a sparse matrix (inheriting class "sparseMatrix" in the **Matrix** package) which will be interpreted as an array with dimension `c(dim(x),1)`
- a vector of atomic values, which will be interpreted as an array of dimension `c(length(x),1,1)`
- a sparse vector (inheriting class "sparseVector" in the **Matrix** package) which will be interpreted as an array of dimension `c(x@length,1,1)`
- a list of matrices with the same dimensions, which will be interpreted as slices `A[,,k]` of an array `A`
- a list of sparse matrices (each inheriting class "sparseMatrix" in the **Matrix** package) with the same dimensions, which will be interpreted as slices `A[,,k]` of an array `A`.

**Value**

Sparse three-dimensional array (object of class "sparse3Darray").

**Author(s)**

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.
See Also

sparse3Darray

Examples

A <- array(c(1,3,0,0,0,0,4,2,5,
            0,0,1,0,0,1,0,0,1,0,
            0,0,1,0,0,0,1,0,0,1,0),
           dim=c(3,4,2))
# array to sparse array
B <- as.sparse3Darray(A) # positive extent
# list of matrices to sparse array
B <- as.sparse3Darray(list(A[,1], A[,2]))
# matrix to sparse array
B1 <- as.sparse3Darray(A[,1])
# vector to sparse array
B11 <- as.sparse3Darray(A[,1,1])

Extract.sparse3Darray  Extract or Replace Entries in a Sparse Array

Description

Extract or replace entries in a sparse three-dimensional array.

Usage

## S3 method for class "sparse3Darray"

x[i, j, k, drop=TRUE, ...]

## S3 replacement method for class "sparse3Darray"

x[i, j, k, ...] <- value

Arguments

x     Sparse three-dimensional array (object of class "sparse3Darray").

i, j, k Subset indices for each dimension of the array. See Details.

value Replacement value for the subset.

drop Logical value indicating whether to return a lower-dimensional object (matrix
       or vector) when appropriate.

...    Ignored. This argument is required for compatibility with the generic function.

Details

These functions are defined for a sparse three-dimensional array x. They extract a designated subset of the array, or replace the values in the designated subset.

The function [.sparse3Darray is a method for the generic subset extraction operator [. The function [<- sparse3Darray is a method for the generic subset replacement operator [<-.

These methods use the same indexing rules as the subset operator for full arrays:
• If i, j and k are integer vectors, the subset is the Cartesian product (i.e. all cells in the array identified by an entry of i, an entry of j and an entry of k).

• Some or all of the arguments i, j and k may be missing from the call; a missing index argument is interpreted as meaning that all possible values of that index are allowed.

• Arguments i, j and k may be logical vectors (with the value TRUE assigned to entries that should be included).

• Arguments i, j and k may be character vectors with entries matching the corresponding dimnames.

• Argument i may be an integer matrix with 3 columns (and the arguments j,k should be absent). Each row of the matrix contains the indices of one cell in the array.

If the designated subset lies within the array bounds, then the result of \([\) will be a sparse three-dimensional array, sparse matrix or sparse vector. If drop=FALSE the result will always be three-dimensional; if drop=TRUE (the default) the result will be reduced to two or one dimensions when appropriate.

If the designated subset does not lie within the array bounds, then the result of \([\) will be a full three-dimensional array, matrix or vector containing NA values at the positions that were outside the array bounds.

The result of \(<-\) is always a sparse three-dimensional array. If the designated subset did not lie within the array bounds of \(x\), then the array bounds will be extended (with a warning message).

Value

\([\).sparse3Darray returns either a sparse three-dimensional array (class "sparse3Darray"), a sparse matrix (class sparseMatrix in the Matrix package), a sparse vector (class sparseVector in the Matrix package), or in some cases a full array, matrix or vector.

\(<-\).sparse3Darray returns another sparse three-dimensional array.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

sparse3Darray, methods.sparse3Darray.

Examples

\(M \leftarrow \text{sparse3Darray}(i=1:4, j=\text{sample}(1:4, \text{replace=TRUE}), k=c(1,2,1,2), x=1:4, \text{dims}=c(5,5,2)))\)
\(\text{dimnames}(M) \leftarrow \text{list(letters}[1:5], \text{LETTERS}[1:5], c("yes", "no"))\)
\(M[, 3, ]\)
\(M[, 3:4, ]\)
\(M[, 3:4, 2:4, ]\)
\(M[, 4:3, 4:2, 1:2]\)
gridadjacencymatrix

Create Adjacency Matrix for Spatial Grid

Description

Given the dimensions of a rectangular grid of points, this command creates the adjacency matrix for the corresponding neighbourhood graph, whose vertices are the grid points, and whose edges are the joins between neighbouring grid points.

Usage

gridadjacencymatrix(dims, across = TRUE, down = TRUE, diagonal=TRUE)

Arguments

dims: Grid dimensions. An integer, or a vector of two integers. First entry specifies the number of points in the y direction.
across: Logical value equal to TRUE if horizontal neighbours should be joined.
down: Logical value equal to TRUE if vertical neighbours should be joined.
diagonal: Logical value equal to TRUE if diagonal neighbours should be joined.

Details

If \(N = \text{prod}(\text{dims})\) is the total number of grid points, then the result is an \(N \times N\) sparse matrix with logical entries equal to TRUE if the corresponding grid points are joined.

Value

A sparse matrix.

Author(s)

Adrian Baddeley.

Examples

gridadjacencymatrix(c(2,3))
Description
For a sparse matrix or sparse array, compute the sum of array entries for a specified margin or margins.

Usage
marginSumsSparse(X, MARGIN)

Arguments
X
A matrix, an array, a sparse matrix (of class "sparseMatrix" from the Matrix package) or a sparse three-dimensional array (of class "sparse3Darray" from the spatstat.sparse package).

MARGIN
Integer or integer vector specifying the margin or margins.

Details
This function computes the equivalent of apply(X,MARGIN,sum) for sparse matrices and arrays X. The argument x may be
- a matrix
- an array of any number of dimensions
- a sparse matrix (object inheriting class "sparseMatrix" in the Matrix package)
- a sparse three-dimensional array (of class "sparse3Darray" from the spatstat.sparse package).

In the first two cases, the computation is performed by calling apply(X,MARGIN,sum) and the result is a vector, matrix or array. In the last two cases, the result is a single value, a sparse vector, a sparse matrix, or a sparse three-dimensional array.

Value
A single value, vector, matrix, array, sparse vector (class "sparseVector" in the Matrix package), sparse matrix (class "sparseMatrix" in the Matrix package), or sparse three-dimensional array (class "sparse3Darray" from the spatstat.sparse package).

Author(s)
Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also
apply
**Examples**

```r
M <- sparse3Darray(i=1:3, j=c(3,1,2), k=4:2,
                   x=round(runif(3), 2), dims=rep(4, 3))
marginSumsSparse(M, 1:2)
marginSumsSparse(M, 1)
marginSumsSparse(M, integer(0))  # equivalent to sum(M)
```

**Description**

Group generic methods which make it possible to apply the familiar mathematical operators and functions to sparse three-dimensional arrays (objects of class "sparse3Darray"). See Details for a list of implemented functions.

**Usage**

```r
## S3 methods for group generics have prototypes:
Math(x, ...)
Ops(e1, e2)
Complex(z)
Summary(..., na.rm=FALSE)
```

**Arguments**

- `x, z, e1, e2` Sparse three-dimensional arrays (objects of class "sparse3Darray"). Alternatively `e1` or `e2` can be a single scalar, vector, sparse vector, matrix or sparse matrix.
- `...` further arguments passed to methods.
- `na.rm` Logical value specifying whether missing values should be removed.

**Details**

These group generics make it possible to perform element-wise arithmetic and logical operations with sparse three-dimensional arrays, or apply mathematical functions element-wise, or compute standard summaries such as the mean and maximum.

Below is a list of mathematical functions and operators which are defined for sparse 3D arrays.

1. Group "Math":
   - `abs, sign, sqrt, floor, ceiling, trunc, round, signif`
Math.sparse3Darray

1. Group "Math":
   - `exp`, `log`, `expm1`, `log1p`, `cos`, `sin`, `tan`, `cospi`, `sinpi`, `tanpi`, `acos`, `asin`, `atan`, `cosh`, `sinh`, `tanh`, `acosh`, `asinh`, `atanh`
   - `lgamma`, `gamma`, `digamma`, `trigamma`

2. Group "Ops":
   - `+`, `-`, `*`, `/`, `^`, `%%`, `%/%`
   - `&`, `|`, `!`
   - `==`, `!=`, `<`, `<=`, `>`, `>=`

3. Group "Summary":
   - `all`, `any`
   - `sum`, `prod`
   - `min`, `max`
   - `range`

4. Group "Complex":
   - `Arg`, `Conj`, `Im`, `Mod`, `Re`

Value

The result of group "Math" functions is another three-dimensional array of the same dimensions as `x`, which is sparse if the function maps 0 to 0, and otherwise is a full three-dimensional array.

The result of group "Ops" operators is another three-dimensional array of the same dimensions as `e1` and `e2`, which is sparse if both `e1` and `e2` are sparse.

The result of group "Complex" functions is another sparse three-dimensional array of the same dimensions as `z`.

The result of group "Summary" functions is a logical value or a numeric value or a numeric vector of length 2.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

sparse3DArray, tensorSparse

Examples

```r
M <- sparse3Darray(i=1:4, j=sample(1:4, replace=TRUE), k=c(1,2,1,2), x=1:4, dims=c(5,5,2))
negM <- -M
twoM <- M + M
```
matrixpower

Mplus <- M + 1 ## not sparse!
posM <- (M > 0)
range(M)

sinM <- sin(M)
cosM <- cos(M) ## not sparse!
expM1 <- expm1(M)

matrixpower

Power of a Matrix

Description

Evaluate a specified power of a matrix.

Usage

matrixpower(x, power, complexOK = TRUE)
matrixsqrt(x, complexOK = TRUE)
matrixinvsqrt(x, complexOK = TRUE)

Arguments

x A square matrix containing numeric or complex values.
power A numeric value giving the power (exponent) to which x should be raised.
complexOK Logical value indicating whether the result is allowed to be complex.

Details

These functions raise the matrix x to the desired power: matrixsqrt takes the square root, matrixinvsqrt takes the inverse square root, and matrixpower takes the specified power of x.

Up to numerical error, matrixpower(x, 2) should be equivalent to x ** 2, and matrixpower(x, -1) should be equivalent to solve(x), the inverse of x.

The square root y <- matrixsqrt(x) should satisfy y ** 2 = x. The inverse square root z <- matrixinvsqrt(x) should satisfy z ** 2 = solve(x).

Computations are performed using the eigen decomposition (eigen).

Value

A matrix of the same size as x containing numeric or complex values.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

See Also

eigen, svd
Examples

```r
x <- matrix(c(10,2,2,1), 2, 2)
y <- matrixsqrt(x)
y
y ** y
z <- matrixinvsqrt(x)
z ** y
matrixpower(x, 0.1)
```

Description

Methods for the class "sparse3Darray" of sparse three-dimensional arrays.

Usage

```r
## S3 method for class 'sparse3Darray'
anyNA(x, recursive = FALSE)
## S3 method for class 'sparse3Darray'
dim(x)
## S3 replacement method for class 'sparse3Darray'
dim(x) <- value
## S3 method for class 'sparse3Darray'
dimnames(x)
## S3 replacement method for class 'sparse3Darray'
dimnames(x) <- value
## S3 method for class 'sparse3Darray'
print(x, ...)
```

Arguments

- `x` A sparse three-dimensional array (object of class "sparse3Darray").
- `value` Replacement value (see Details).
- `recursive,...` Ignored.

Details

These are methods for the generics `anyNA`, `dim`, `dim<-`, `dimnames`, `dimnames<-` and `print` for the class "sparse3Darray" of sparse three-dimensional arrays. For `dimnames(x) <- value`, the value should either be `NULL`, or a list of length 3 containing character vectors giving the names of the margins. For `dim(x) <- value`, the value should be an integer vector of length 3 giving the new dimensions of the array. Note that this operation does not change the array positions of the non-zero entries (unlike `dim(x) <- value` for a full array). An error occurs if some of the non-zero entries would lie outside the new extent of the array.
sparse3Darray

Value

- `anyNA` returns a single logical value.
- `dim` returns an integer vector of length 3.
- `dimnames` returns `NULL`, or a list of length 3 whose entries are character vectors.
- `dim<-` and `dimnames<-` return a sparse 3D array.
- `print` returns `NULL`, invisibly.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

`sparse3Darray`

Examples

```r
M <- sparse3Darray(i=1:4, j=sample(1:4, replace=TRUE),
                    k=c(1,2,1,2), x=1:4, dims=c(5,5,2))

anyNA(M)
dim(M)
dimnames(M)
dimnames(M) <- list(letters[1:5], LETTERS[1:5], c("Yes", "No"))
print(M)
```

---

`sparse3Darray`  
*Create a Sparse Three-Dimensional Array*

Description

Create a sparse representation of a three-dimensional array.

Usage

```r
sparse3Darray(i = integer(0), j = integer(0), k = integer(0),
              x = numeric(0),
              dims = c(max(i), max(j), max(k)), dimnames = NULL,
              strict = FALSE, nonzero = FALSE)
```
Arguments

\(i,j,k\)  
Integer vectors of equal length (or length 1), specifying the cells in the array which have non-zero entries.

\(x\) 
Vector (numeric, integer, logical or complex) of the same length as \(i, j\) and \(k\), giving the values of the array entries that are not zero.

dims 
Dimension of the array. An integer vector of length 3.

dimnames 
Names for the three margins of the array. Either \texttt{NULL} or a list of three character vectors.

strict 
Logical value specifying whether to enforce the rule that each entry in \(i,j,k,x\) refers to a different cell. If \(strict=\text{TRUE}\), entries which refer to the same cell in the array will be reduced to a single entry by summing the \(x\) values. Default is \(strict=\text{FALSE}\).

nonzero 
Logical value specifying whether to remove any entries of \(x\) which equal zero.

Details

An array \(A\) is three-dimensional if it is indexed by three integer indices, so that \(A[i,j,k]\) specifies an element of the array. The array is called sparse if only a small fraction of the entries are non-zero. A sparse array can be represented economically by listing only the entries which are non-zero.

The \texttt{spatstat.sparse} package defines the class \texttt{sparse3Darray} of sparse three-dimensional arrays. These arrays can have numeric, integer, logical, or complex entries.

The function \texttt{sparse3Darray} creates an object of class "\texttt{sparse3Darray}". This object is essentially a list containing the vectors \(i,j,k,x\) and the arguments \texttt{dims, dimnames}.

The arguments \(i,j,k,x\) should be vectors of equal length identifying the cells in the array which have non-zero entries (indexed by \(i,j,k\)) and giving the values in these cells (given by \(x\)).

The default behaviour of \texttt{sparse3Darray} is to accept the arguments \(i,j,k,x\) without modifying them. This would allow some entries of \(x\) to be equal to zero, and would allow a cell in the array to be referenced more than once in the indices \(i,j,k\).

If \(nonzero=\text{TRUE}\), entries will be removed if the \(x\) value equals zero.

If \(strict=\text{TRUE}\), entries which refer to the same cell in the array will be combined into a single entry by summing the \(x\) values.

Value

An object of class "\texttt{sparse3Dvector}".

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

\texttt{as.sparse3Darray}
Examples

```r
## creation by specifying nonzero elements
M <- sparse3Darray(i=1:3, j=c(3,1,2), k=4:2,
x=runif(3), dims=rep(4, 3))
M
## duplicate entries
Mn <- sparse3Darray(i=c(1,1,2), j=c(2,2,1), k=c(3,3,2),
x=runif(3), dims=rep(3, 3))
## cumulate entries in duplicate positions
Ms <- sparse3Darray(i=c(1,1,2), j=c(2,2,1), k=c(3,3,2),
x=runif(3), dims=rep(3, 3), strict=TRUE)
```

### sumouter

**Compute Quadratic Forms**

#### Description

Calculates certain quadratic forms of matrices.

#### Usage

```r
sumouter(x, w=NULL, y=x)
quadform(x, v)
bilinearform(x, v, y)
```

#### Arguments

- **x, y**: A matrix, whose rows are the vectors in the quadratic form.
- **w**: Optional vector of weights
- **v**: Matrix determining the quadratic form

#### Details

The matrices x and y will be interpreted as collections of row vectors. They must have the same number of rows. The entries of x and y may be numeric, integer, logical or complex values.

The command `sumouter` computes the sum of the outer products of corresponding row vectors, weighted by the entries of w:

\[ M = \sum_i w_i x_i^T y_i \]

where \( x_i \) is the i-th row of x and \( y_i \) is the i-th row of y (after removing any rows containing NA or other non-finite values). If w is missing, the weights will be taken as 1. The result is a \( p \times q \) matrix where \( p = \text{ncol}(x) \) and \( q = \text{ncol}(y) \).

The command `quadform` evaluates the quadratic form, defined by the matrix v, for each of the row vectors of x:

\[ y_i = x_i^T V x_i \]
The result \( y \) is a numeric vector of length \( n \) where \( n = \text{nrow}(x) \). If \( x[i,] \) contains \( \text{NA} \) or other non-finite values, then \( y[i] = \text{NA} \).

The command `bilinearform` evaluates the more general bilinear form defined by the matrix \( v \). Here \( x \) and \( y \) must be matrices of the same dimensions. For each row vector of \( x \) and corresponding row vector of \( y \), the bilinear form is

\[
z_i = x_i' V y_i
\]

The result \( z \) is a numeric vector of length \( n \) where \( n = \text{nrow}(x) \). If \( x[i,] \) or \( y[i,] \) contains \( \text{NA} \) or other non-finite values, then \( z[i] = \text{NA} \).

**Value**

A vector or matrix.

**Author(s)**

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au> and Rolf Turner <r.turner@auckland.ac.nz>

**Examples**

```r
x <- matrix(1:12, 4, 3)
dimnames(x) <- list(c("Wilma", "Fred", "Barney", "Betty"), letters[1:3])
x
sumouter(x)

w <- 4:1
sumouter(x, w)

v <- matrix(1, 3, 3)
quadform(x, v)

# should be the same as quadform(x, v)
bilinearform(x, v, x)

# See what happens with NA's
x[3,2] <- NA
sumouter(x, w)
quadform(x, v)
```

---

**tensorSparse**

Tensor Product of Sparse Vectors, Matrices or Arrays

**Description**

Compute the tensor product of two vectors, matrices or arrays which may be sparse or non-sparse.

**Usage**

```r
tensorSparse(A, B, alongA = integer(0), alongB = integer(0))
```
Arguments

A, B

Vectors, matrices, three-dimensional arrays, or objects of class sparseVector, sparseMatrix or sparse3Darray.

alongA

Integer vector specifying the dimensions of A to be collapsed.

alongB

Integer vector specifying the dimensions of B to be collapsed.

Details

This function is a generalisation, to sparse arrays, of the function tensor in the tensor package.
tensorSparse has the same syntax and interpretation as tensor. For example, if A and B are matrices, then tensor(A, B, 2, 1) is the matrix product A %*% B while tensor(A, B, 2, 2) is A %*% t(B).

This function tensorSparse handles sparse vectors (class "sparseVector" in the Matrix package), sparse matrices (class "sparseMatrix" in the Matrix package) and sparse three-dimensional arrays (class "sparse3Darray" in the spatstat.sparse package) in addition to the usual vectors, matrices and arrays.

The result is a sparse object if at least one of A and B is sparse. Otherwise, if neither A nor B is sparse, then the result is computed using tensor.

The main limitation is that the result cannot have more than 3 dimensions (because sparse arrays with more than 3 dimensions are not yet supported).

Value

Either a scalar, a vector, a matrix, an array, a sparse vector (class "sparseVector" in the Matrix package), a sparse matrix (class "sparseMatrix" in the Matrix package) or a sparse three-dimensional array (class "sparse3Darray" in the spatstat.sparse package).

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

See Also

sparse3Darray, aperm.sparse3Darray

Examples

M <- sparse3Darray(i=1:4, j=sample(1:4, replace=TRUE),
                 k=c(1,2,1,2), x=1:4, dims=c(5,5,2))
A <- tensorSparse(M, M, 1:2, 2:1)
Index

* algebra
  marginSumsSparse, 10
  matrixpower, 13
  spatstat.sparse-package, 2
  tensorSparse, 18

* array
  aperm.sparse3Darray, 4
  as.array.sparse3Darray, 5
  as.sparse3Darray, 6
  Extract.sparse3Darray, 7
  marginSumsSparse, 10
  matrixpower, 13
  methods.sparse3Darray, 14
  sparse3Darray, 15
  spatstat.sparse-package, 2
  sumouter, 17
  tensorSparse, 18

* datagen
  gridadjacencymatrix, 9

* manip
  aperm.sparse3Darray, 4
  as.array.sparse3Darray, 5
  as.sparse3Darray, 6
  Extract.sparse3Darray, 7
  marginSumsSparse, 10
  methods.sparse3Darray, 14

* methods
  Math.sparse3Darray, 11

* package
  spatstat.sparse-package, 2

* sparse
  aperm.sparse3Darray, 4
  as.array.sparse3Darray, 5
  as.sparse3Darray, 6
  Extract.sparse3Darray, 7
  marginSumsSparse, 10
  methods.sparse3Darray, 14
  sparse3Darray, 15
  tensorSparse, 18

* spatial
  Math.sparse3Darray, 11
  [. 7
  [.sparse3Darray, 3
  [.sparse3Darray
    (Extract.sparse3Darray), 7
  [<-.sparse3Darray
    (Extract.sparse3Darray), 7
  anyNA, 14
  anyNA.sparse3Darray
    (methods.sparse3Darray), 14
  aperm, 4
  aperm.sparse3Darray, 3, 4, 19
  apply, 10
  array, 5
  as.array, 5
  as.array.sparse3Darray, 3, 5
  as.sparse3Darray, 3, 5, 6, 16
  bilinearform, 3
  bilinearform(sumouter), 17
  Complex.sparse3Darray, 3
  Complex.sparse3Darray
    (Math.sparse3Darray), 11
  dim, 14
  dim.sparse3Darray
    (methods.sparse3Darray), 14
  dim<-.sparse3Darray
    (methods.sparse3Darray), 14
  dimnames, 14
  dimnames.sparse3Darray
    (methods.sparse3Darray), 14
  dimnames<-.sparse3Darray
    (methods.sparse3Darray), 14
  eigen, 13
  Extract.sparse3Darray, 7
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridadjacencymatrix</td>
<td>9</td>
</tr>
<tr>
<td>marginSumsSparse</td>
<td>10</td>
</tr>
<tr>
<td>Math.sparse3Darray</td>
<td>3, 11</td>
</tr>
<tr>
<td>matrixinvsqrt(matrixpower)</td>
<td>13</td>
</tr>
<tr>
<td>matrixpower</td>
<td>3, 13</td>
</tr>
<tr>
<td>matrixsqrt</td>
<td>3</td>
</tr>
<tr>
<td>matrixsqrt(matrixpower)</td>
<td>13</td>
</tr>
<tr>
<td>methods.sparse3Darray</td>
<td>3, 8, 14</td>
</tr>
<tr>
<td>Ops.sparse3Darray</td>
<td>3</td>
</tr>
<tr>
<td>Ops.sparse3Darray(Math.sparse3Darray)</td>
<td>11</td>
</tr>
<tr>
<td>print</td>
<td>14</td>
</tr>
<tr>
<td>print.sparse3Darray</td>
<td></td>
</tr>
<tr>
<td>quadform</td>
<td>3</td>
</tr>
<tr>
<td>quadform(sumouter)</td>
<td>17</td>
</tr>
<tr>
<td>sparse3Darray</td>
<td>3–5, 7, 8, 12, 15, 19</td>
</tr>
<tr>
<td>spatstat.sparse</td>
<td></td>
</tr>
<tr>
<td>(spatstat.sparse-package)</td>
<td>2</td>
</tr>
<tr>
<td>spatstat.sparse-package</td>
<td>2</td>
</tr>
<tr>
<td>Summary.sparse3Darray</td>
<td>3</td>
</tr>
<tr>
<td>Summary.sparse3Darray(Math.sparse3Darray)</td>
<td>11</td>
</tr>
<tr>
<td>sumouter</td>
<td>3, 17</td>
</tr>
<tr>
<td>svd</td>
<td>13</td>
</tr>
<tr>
<td>tensor</td>
<td>19</td>
</tr>
<tr>
<td>tensorSparse</td>
<td>3, 4, 12, 18</td>
</tr>
</tbody>
</table>