Package 'tableMatrix'

October 14, 2022

Title Combines 'data.table' and 'matrix' Classes

Version 0.82.0

Description Provides two classes extending 'data.table' class. Simple 'tableList' class wraps 'data.table' and any additional structures together. More complex 'tableMatrix' class combines 'data.table' and 'matrix'. See <http://github.com/InferenceTechnologies/tableMatrix> for more information and examples.

Depends R (>= 3.3.1)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports data.table

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Petr Lenhard [aut, cre], Inference Technologies [cph], Petra Hudeckova [ctb]

Maintainer Petr Lenhard <petr.lenhard@inferencetech.com>

Repository CRAN

Date/Publication 2018-06-26 20:24:01 UTC

R topics documented:

`aid` *S3 tableMatrix generic to get or set aid attribute*

Description

S3 tableMatrix generic to get or set aid attribute

Usage

```
aid(obj, ...)

aid(obj) <- value
```

Arguments

obj	Object.
...	Passed arguments.
value	User defined

Value

User aid data.

---

aid.tableList *Get or set aid attribute*

Description

tableList method to get or set aid attribute.

Usage

```
## S3 method for class 'tableList'
aid(obj, ...)

## S3 replacement method for class 'tableList'
aid(obj) <- value
```

Arguments

obj	tableList object.
...	Passed arguments.
value	User defined.

Value

Aid part.

cbind.tableList *Column binding*

Description

Column binding

Column binding of tab part of `tableMatrix` or `tableList` object.

Usage

```
## S3 method for class 'tableList'
cbind(x, y, ...)

## S3 method for class 'tableMatrix'
cbind(x, y, ...)
```

Arguments

`x`	`tableList` or `tableMatrix` object.
`y`	`tableList` or `tableMatrix` object.
`...`	Passed arguments.

Value

Matrix object.

copy.tableList *Copy*

Description

Copy of a `tableList` object.

Usage

```
## S3 method for class 'tableList'
copy(obj)
```

Arguments

`obj`	`tableList` object.

Value

`tableList` object.

copy.tableMatrix *Copy*

Description

Copy of a tableMatrix object.

Usage

```
## S3 method for class 'tableMatrix'
copy(obj)
```

Arguments

obj	tableMatrix object.

Value

tableMatrix object.

dataType *S3 tableMatrix generic to get or set dataType attribute*

Description

S3 tableMatrix generic to get or set dataType attribute

Usage

```
dataType(obj, ...)

dataType(obj) <- value
```

Arguments

obj	Object.
...	Passed arguments.
value	Named list.

Value

List.

dataType.tableList *Get or set dataType attribute*

Description

tableList and tableMarix method to get or set dataType attribute.

Usage

```
## S3 method for class 'tableList'
dataType(obj, ...)

## S3 replacement method for class 'tableList'
dataType(obj) <- value

## S3 method for class 'tableMatrix'
dataType(obj, ...)

## S3 replacement method for class 'tableMatrix'
dataType(obj) <- value
```

Arguments

obj	tableMatrix or tableList object.
...	Passed arguments.
value	List of datatypes.

Value

List of datatypes.

dim.tableList *Dimensions*

Description

tableList method, returns dimensions of the table attribute.

Usage

```
## S3 method for class 'tableList'
dim(x)
```

Arguments

x	tableList object.

Value

Dimensions of the table part.

`dimnames.tableList` *Dimension names*

Description

`tableList` method, returns dimension names of the table attribute.

Usage

```
## S3 method for class 'tableList'
dimnames(x)
```

Arguments

`x`	`tableList` object.

Value

Column and row names of the table part.

`getRow` *S3 tableMatrix generic to get or set row from matrix attribute*

Description

S3 tableMatrix generic to get or set row from matrix attribute

Usage

```
getRow(...)

setRow(...)
```

Arguments

`...`	Passed arguments.

Value

numeric vector.

getRow.tableMatrix *Get or set row from the matrix attribute*

Description

tableMatrix method to get or set a row from the matrix attribute.

Usage

```
## S3 method for class 'tableMatrix'
getRow(obj, i = NULL, repo = NULL, ...)

## S3 method for class 'tableMatrix'
setRow(obj, value, i = NULL, repo = NULL, ...)
```

Arguments

obj	tableMatrix object.
i	Integer. Row index in tab.
repo	Numeric vector. Vector with 2 elements - matN and matRow.
...	Passed arguments.
value	Vector for setting value.

Value

Row from the matrix part.

Examples

```
data(images8By8)
data(images10By10)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10), list(1:3, 1:3),
list(4:ncol(images8By8),4:ncol(images10By10)))

# Row 91 in tab(TM) corresponds to second matrix, first row
row <- getRow(TM, 91)

# Row 91 in tab(TM) corresponds to second matrix, first row
getRow(TM, repo=c(2,1))

# Change matrix row corresponding to row 91 in tab(TM)
TM <- setRow(TM, rep(2,length(row)), 91)
getRow(TM, 91)
```

getRowDim *S3 tableMatrix generic to get row dim*

Description

S3 tableMatrix generic to get row dim

Usage

```
getRowDim(...)
```

Arguments

...	Passed arguments.

Value

integer vector.

getRowDim.tableMatrix *Get row dimensions*

Description

tableMatrix method to get row dimensions from matDim attribute.

Usage

```
## S3 method for class 'tableMatrix'
getRowDim(obj, i = NULL, repo = NULL, ...)
```

Arguments

obj	tableMatrix object.
i	Integer. Row index in tab.
repo	Numeric vector. Vector with 2 elements - matN and matRow.
...	Passed arguments.

Value

Dimensions corresponding to the row.

Examples

```
data(images8By8)
data(images10By10)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10),
list(r=c(1,3), r=c(1,3)),
list(r=c(4,ncol(images8By8)),r=c(4,ncol(images10By10))),list(c(8,8),c(10,10)),
dimNames =c("dimX", "dimY"))
matDim(TM)
tab(TM)

# Dimensions corresponding to row 1 in tab(TM)
getRowDim(TM, 1)

# Dimensions corresponding to row 91 in tab(TM)
getRowDim(TM, 91)

# Dimensions corresponding to row 1 in second matrix in mat(TM)
getRowDim(TM, repo=c(2,1))
```

---

getRowRepo *S3 tableMatrix generic to get row repo for matrix attribute*

Description

S3 tableMatrix generic to get row repo for matrix attribute

Usage

```
getRowRepo(...)
```

Arguments

...	Passed arguments.

Value

numeric vector.

getRowRepo.tableMatrix *Get row repo*

Description

tableMatrix method to get row repo (vector of matN and matRow) for the matrix attribute

Usage

```
## S3 method for class 'tableMatrix'
getRowRepo(obj, i = NULL, repo = NULL, ...)
```

Arguments

obj	tableMatrix object.
i	Integer. Row index in tab.
repo	Numeric vector. Vector with 2 elements - matN and matRow.
...	Passed arguments.

Value

Vector of matN and matRow.

Examples

```
data(images8By8)
data(images10By10)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10), list(1:3, 1:3),
list(c(4:ncol(images8By8)),c(4:ncol(images10By10))))
tab(TM)

# Row 2 in tab(TM) corresponds to first matrix, second row
getRowRepo(TM, 2)

# Row 91 in tab(TM) corresponds to second matrix, first row
getRowRepo(TM, 91)
```

images10By10 *Images of distributions*

Description

Dataset containing samples of multivariate normal distributions. Each row in the dataset represents a generated matrix. Dimensions are stored in dimX and dimY columns. Directions "up" and "down" correspond to different covariance matrices. Direction "both" was created as a combination of "up" and "down".

Usage

```
images10By10
```

Format

Data frame with 90 rows and 103 variables:

- direction direction of diagonal, factor "up", "down", "both"
- dimX dimension x of image, in pixels
- dimY dimension y of image, in pixels
- pixel1:pixel100 pixel of image, in number

images8By8 *Images of distributions*

Description

Dataset containing samples of multivariate normal distributions. Each row in the dataset represents a generated matrix. Dimensions are stored in dimX and dimY columns. Directions "up" and "down" correspond to different covariance matrices. Direction "both" was created as a combination of "up" and "down".

Usage

```
images8By8
```

Format

Data frame with 90 rows and 67 variables:

- direction direction of diagonal, factor "up", "down", "both"
- dimX dimension x of image, in pixels
- dimY dimension y of image, in pixels
- pixel1:pixel64 pixel of image, in number

is.tableList *tableList test*

Description

Tests if passed object is of class `tableList`

Usage

```
is.tableList(obj)
```

Arguments

`obj`	`tableList` object

Value

logical vector.

is.tableMatrix *tableMatrix test*

Description

Tests if passed object is of class `tableMatrix`

Usage

```
is.tableMatrix(obj)
```

Arguments

`obj`	`tableTable` object

Value

logical vector.

mat *S3 tableMatrix generic to get or set matrix attribute*

Description

S3 tableMatrix generic to get or set matrix attribute

Usage

```
mat(obj, ...)

mat(obj) <- value
```

Arguments

obj	Object.
...	Passed arguments.
value	List of matrices.

Value

List of matrices

mat.tableMatrix *Get or set matrix attribute*

Description

tableMatrix method to get or set matrix part attribute. Matrix part is a list of matrices.

Usage

```
## S3 method for class 'tableMatrix'
mat(obj, matN = NULL, ...)

## S3 replacement method for class 'tableMatrix'
mat(obj) <- value
```

Arguments

obj	tableMatrix object.
matN	Integer. Matrix number in the matrix part list.
...	Passed arguments.
value	List of matrices.

Value

Full matrix part of `tableMatrix` or a matrix.

Examples

```
data(images8By8)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10), list(1:3, 1:3),
list(4:ncol(images8By8),4:ncol(images10By10)))

# Full matrix part of TM
mat(TM)

# Matrix part of TM corresponding to matrix type 2
mat(TM, 2)
```

`matDim` *S3 tableMatrix generic to get or set matDim attribute*

Description

S3 tableMatrix generic to get or set matDim attribute

Usage

```
matDim(obj, ...)

matDim(obj) <- value
```

Arguments

`obj`	Object.
`...`	Passed arguments.
`value`	data.table

Value

data.table

`matDim.tableMatrix` *Get or set matDim attribute*

Description

`tableMatrix` method to get or set matDim attribute.

Usage

```
## S3 method for class 'tableMatrix'
matDim(obj, matN = NULL, resetN = FALSE, ...)

## S3 replacement method for class 'tableMatrix'
matDim(obj) <- value
```

Arguments

`obj`	`tableMatrix` object.
`matN`	Integer. Matrix number in the matrix part list.
`resetN`	Logical. When FALSE `tm.matN` of returned `matDim` won't be reset to 1.
`...`	Passed arguments.
`value`	data.table

Value

Full dimensions part of `tableMatrix` or a subset of dimensions part.

Examples

```
data(images8By8)
data(images10By10)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10), list(1:3, 1:3),
list(r=c(4,ncol(images8By8)),r=c(4,ncol(images10By10))))

# Dimensions part of TM
matDim(TM)

# Dimensions part of TM corresponding to matrix type 2
matDim(TM, 2)
```

`merge.tableList` *Merging tableList*

Description

Merging tableList

`tableMatrix` method, merges `tableMatrix` or `tableList` objects with data.table, data.frame, `tableList` or `tableMatrix`.

Usage

```
## S3 method for class 'tableList'
merge(x, y, key, ...)

## S3 method for class 'tableMatrix'
merge(x, y, key, ...)
```

Arguments

`x`	`tableMatrix` object
`y`	data.table, data.frame, `tableList`, `tableMatrix` object.
`key`	Shared columns as merging key.
`...`	Passed arguments.

Value

`tableMatrix` object.

Examples

```
data(images8By8)

# Create tableMatrix from images8By8
TM <- tableMatrix(images8By8, c("direction","dimX","dimY"), 4:ncol(images8By8), c(8,8))

# Merge tableMatrix object with a data.frame
merge(TM, data.frame(direction="down", flag=TRUE), key="direction")
```

print.tableList *Print*

Description

tableList method, prints table attribute.

Usage

```
## S3 method for class 'tableList'
print(x, ...)
```

Arguments

x	tableList object.
...	Passed arguments.

print.tableMatrix *Print*

Description

tableMatrix method, prints table attribute without tmName columns.

Usage

```
## S3 method for class 'tableMatrix'
print(x, ...)
```

Arguments

x	tableMatrix object.
...	Passed arguments.

rbind.tableList *Combine by rows*

Description

tableList method, binds rows of tableList objects.

Usage

```
## S3 method for class 'tableList'
rbind(..., use.names = TRUE, fill = FALSE)
```

Arguments

...	tableList objects.
use.names	Passed to rbind.data.table.
fill	Passed to rbind.data.table.

Value

tableList object.

Examples

```
data(images8By8)
data(images10By10)

# Create tableList objects from images8By8[,1:3] and images10By10[,1:3]
TL1 <- tableList(images8By8[,1:3])
TL2 <- tableList(images10By10[,1:3])

# Bindind rows of two tableList objects
rbind(TL1, TL2)
```

rbind.tableMatrix *Combine by rows*

Description

tableMatrix method, binds rows of tableMatrix objects. If matrix parts have the same dimensions, it combines rows in table and matrix parts using data.table and matrix rbind methods respectively. In case when dimensions differ it combines rows in table part and adds new matrix to the matrix part.

Usage

```
## S3 method for class 'tableMatrix'
rbind(..., use.names = TRUE, fill = FALSE)
```

Arguments

...	tableMatrix objects.
use.names	Passed to rbind.data.table.
fill	Passed to rbind.data.table.

Value

tableMatrix object.

Examples

```
data(images8By8)
dim(images8By8)

data(images10By10)
dim(images10By10)

# Create tableMatrix objects from images8By8 and images10By10
TM1 <- tableMatrix(images8By8, 1:3, 4:ncol(images8By8))
TM2 <- tableMatrix(images10By10, 1:3, 4:ncol(images10By10))

# Combining tableMatrix objects with same dimensions
TM <- rbind(TM1, TM1)
# Combined table part
tab(TM)
# One matrix in the matrix part
length(mat(TM))
# One dimension type
matDim(TM)

# Combining tableMatrix objects with different dimensions
TM <- rbind(TM1, TM2)
# Combined table part
tab(TM)
# Two matrices in the matrix part
length(mat(TM))
# Two dimension types
matDim(TM)
```

tab *S3 tableMatrix generic to get or set table attribute*

Description

S3 tableMatrix generic to get or set table attribute

Usage

```
tab(obj, ...)

tab(obj) <- value
```

Arguments

obj	Object.
...	Passed arguments.
value	data.table

Value

data.table

tab.tableList *Get or set table attribute*

Description

tableList method to get or set table attribute.

Usage

```
## S3 method for class 'tableList'
tab(obj, ...)

## S3 replacement method for class 'tableList'
tab(obj) <- value
```

Arguments

obj	tableList object.
...	Passed arguments.
value	data.table

Value

Table part.

`tab.tableMatrix` *Get or set table attribute*

Description

`tableMatrix` method to get or set table attribute of `tableMatrix` object.

Usage

```
## S3 method for class 'tableMatrix'
tab(obj, matN = NULL, addRow = FALSE, resetN = TRUE,
  ...)
```

Arguments

`obj`	`tableMatrix` object.
`matN`	Integer. Matrix number in `mat` list.
`addRow`	Logical. If TRUE column `tm.allRow` with row indexes before subsetting is added.
`resetN`	Logical. Used when matN is specified. When FALSE `tm.matN` of returned tab won't be reset to 1. Default TRUE.
`...`	Passed arguments.

Value

Full table part of `tableMatrix` or a subset of table part.

Examples

```
data(images8By8)
data(images10By10)

# Create tableMatrix from images8By8 and images10By10
TM <- tableMatrix(list(images8By8, images10By10), list(1:3, 1:3),
list(4:ncol(images8By8),4:ncol(images10By10)))
matDim(TM)

# Table part of TM
tab(TM)

# Table part of TM corresponding to matrix type 2
tab(TM, 2)

# Add row indexes
tab(TM, 2, TRUE)
```

tableList *tableList constructor*

Description

`tableList` constructor, creates tableList object from a data.frame or a data.table and from aid data. The goal is to wrap data.table and any additional data structures together. `tableList` behaves like a data.table object. Groups of attributes can be stored together via `dataType`.

Usage

```
tableList(tabData, aidData = list(), dataType = NULL)
```

Arguments

`tabData`	A data.frame or a data.table.
`aidData`	Aid structures, generally in the form of a list.
`dataType`	Named list. Contains names of groups of attributes, elements are attributes which belong to the group.

Value

A `tableList` object

See Also

`dim.tableList`, `dimnames.tableList` `rbind.tableList`

Examples

```
data(chickwts)

# Bundle chickwts data.frame together with a linear model
TL <- tableList(chickwts, lm(weight~feed, chickwts))

# tableList behaves like a data.table
mean(TL[feed=="casein", weight])

# Aid part of the tableList object carries the linear model
aid(TL)

# dataType support
TL <- tableList(chickwts, lm(weight~feed, chickwts), dataType=list(group1="weight", group2="feed"))
```

`tableMatrix` *tableMatrix constructor*

Description

`tableMatrix` constructor, creates tableMatrix object from a list of data.frames or data.tables. It is useful for datasets with the following structure: first set of columns of varying types is intended as meta data, second set of columns of the same type is intended as main data. `tableMatrix` combines strengths of data.table (access via bracket to the meta data part) and matrix (main data). It also stores dimensions of main data, thus allowing to combine rows of varying lengths into one object. As in `tableList`, `tableMatrix` can carry any additional aid data and data types groups.

Usage

```
tableMatrix(dataList, tabCol, matCol, dims = NULL, dimNames = NULL,
  aidData = list(), dataType = NULL)
```

Arguments

`dataList`	Dataset(s) in the form of data.frame or data.table or list of data.frames or data.tables. All datasets must have the same meta data columns, matrix parts can be different.
`tabCol`	Integer or character vector or list of these vectors. Specifies column names or indices of meta data part. For list input, names `j` or `r` can be used. List name `j` indicates that column names or indices are specified, name `r` indicates range of column names or indices. By default or when a vector is provided, `j` is used.
`matCol`	Integer or character vector or list of these vectors. Specifies column names or indices of main data part. For list input, names `j` or `r` can be used as in `tabCol`.
`dims`	Numeric vector or list of vectors. Specifies dimensions for the `matDim` part. By default with no dimension details given by user, main data is considered to be vectors with one dimension.
`dimNames`	Character vector. Specifies dimension names in `matDim` for each element of `dims` parameter. If not specified these names are generated automatically.
`aidData`	Aid structures generally in the form of a list.
`dataType`	Named list. Contains names of groups of attributes, elements are attributes which belong to the group.

Details

`tableMatrix` is a S3 class that consists of 4 parts. `tab` - table part - is used for storing meta data, `mat` - matrix part - for storing main data and `matDim` - dimensions part - for dimensions of main data. `mat` is a list of matrices. `tab` is a data.table. In `tab` first column `tm.matN` is the matrix number in `mat`, second column `tm.matRow` is the row in the matrix. `matDim` is `data.table`. In `matDim` for each matrix number `tm.matN` dimensions can be specified with user defined dimensions. `dataType` is a list which contains names of groups of attributes and vectors of attributes belonging to it. Default print of `tableMatrix` is the print of the `tab` part without `tm.matN` and `tm.matRow` columns.

Value

A tableMatrix object

See Also

getRowRepo.tableMatrix, getRowDim.tableMatrix, merge.tableMatrix, rbind.tableMatrix

Examples

```
data(images8By8)
dim(images8By8)

data(images10By10)
dim(images10By10)
images10By10DT <- data.table::as.data.table(images10By10)

# Generate tableMatrix from data.frame images8By8: use columns 1:3 as meta data and
# columns 4:ncol(images8By8) as main data
TM <- tableMatrix(images8By8, 1:3, 4:ncol(images8By8))
matDim(TM) # show matDim
tab(TM) # show meta data part of tableMatrix
head(mat(TM)[[1]]) # show head of main data part

# Generate tableMatrix from data.frame images8By8: use columns "direction" and "dimY"
# as meta data and columns 4:ncol(images8By8) as main data
tableMatrix(images8By8, c("direction","dimY"), 4:ncol(images8By8))

# User defined dimensions with default names
TM <- tableMatrix(images8By8, c("direction","dimX","dimY"), 4:ncol(images8By8), c(8,8))
matDim(TM)

# User defined dimensions with custom names
dims <- c(8,8)
names(dims) <- c("dimX", "dimY")

TM <- tableMatrix(images8By8, 1:3, 4:ncol(images8By8), dims)
matDim(TM)

# tabCol and matCol list input with "j" option
# Column indices: first 3 columns in tab, rest in mat
tableMatrix(images8By8, list(j=1:3), list(j=4:ncol(images8By8)))
# Column names: columns "direction" and "dimY" in tab,
# columns "pixel1" and "pixel2" in mat
tableMatrix(images8By8, list(j=c("direction","dimY")), list(j=c("pixel1","pixel2")))

# tabCol and matCol list input with "r" option
# Column indices: first 3 columns in tab, rest in mat
tableMatrix(images8By8, list(r=c(1,3)), list(r=c(4,ncol(images8By8))))
# Same with column names
tableMatrix(images8By8, list(r=c("direction","dimY")), list(r=c("pixel1","pixel100")))
```

```
# data.table as the start dataset
tableMatrix(images10By10DT, 1:3, 4:ncol(images10By10DT))

# data.frame and data.table with different main data parts -> two matrices in mat.
# Elements in tabCol and matCol lists correspond to images8By8 and images10By10DT
# respectively
TM <- tableMatrix(list(images8By8, images10By10DT),
list(r=c("direction","dimY"), j=c("direction","dimX","dimY")),
list(4:ncol(images8By8),4:ncol(images10By10DT)))
matDim(TM)
length(mat(TM)) # 2 matrices in mat

# User defined named dimensions
TM <- tableMatrix(list(images8By8, images10By10),
list(r=c("direction","dimY"), j=c("direction","dimX","dimY")),
list(c(4:ncol(images8By8)),c(4:ncol(images10By10))),list(c(8,8),c(10,10)),
dimNames =c("dimX", "dimY"))
matDim(TM)

# Same main data parts -> only one matrix in mat
TM <- tableMatrix(list(images8By8, images8By8),
list(r=c("direction","dimY"), j=c("direction","dimX","dimY")),
list(j=4:ncol(images8By8),4:ncol(images8By8)))
matDim(TM)
length(mat(TM)) # 1 matrix in mat

# dataType support
TM <- tableMatrix(images10By10, 1:3, 4:ncol(images10By10),
dataType=list("group1"="direction", "group2"=c("dimX","dimY")))
dataType(TM)
```

`[.tableList` *Bracket*

Description

`tableList` method, passes data.table bracket functionality to the table attribute. Usage is the same as in data.table[] and data.table[] <-. Assigning works only for `tab` part and only `i` and `j` from `data.table` is used.

Bracket

Usage

```
## S3 method for class 'tableList'
x[...]

## S3 replacement method for class 'tableList'
x[i, j] <- value
```

Arguments

x	tableList object.
...	Passed arguments.
i	Same as i in data.table
j	Same as j in data.table
value	Value to be set.

Value

tableList or vector.

Examples

```
data(images8By8)

# Create tableList from images8By8[,1:3]
TL <- tableList(images8By8[,1:3])

# Apply data.table bracket on a tableList object
TL[direction=="both"]


TL[2,1] <- "aaa"

# setting row
TL[2,] <- list("aaa", 1000, 1000)

# setting column
TL[,2] <- 1
```

[.tableMatrix *Bracket*

Description

tableMatrix method, passes data.table bracket functionality to the table attribute. Assigning works only for tab part and only i and j from data.table is used. If j is numeric, it corresponds to first attribute after tmName names.

Bracket

Usage

```
## S3 method for class 'tableMatrix'
x[...]

## S3 replacement method for class 'tableMatrix'
x[i, j] <- value
```

Arguments

x	tableMatrix object.
...	Passed arguments.
i	Same as i in data.table
j	Same as j in data.table
value	Value to be set.

Value

tableMatrix or vector.

Examples

```
data(images8By8)

# Create tableMatrix from images8By8
TM <- tableMatrix(images8By8, 1:3, 4:ncol(images8By8))


TM[2,1] <- "aaa"

TM[1,"dimX"] <- 1000

# setting row
TM[2,] <- list("aaa", 1000, 1000)

# setting column
TM[,2] <- 1
```

[[.tableList *Double bracket*

Description

tableList method, passes double bracket functionality to the table attribute.

Usage

```
## S3 method for class 'tableList'
x[[...]]
```

Arguments

x	tableList object.
...	Passed arguments.

Value

vector

Index