Package 'vectools'

January 9, 2020

Title Supplementary Vector-Related Tools

Version 0.1.1

Date 2020-01-08

License GPL (>= 2)

Maintainer Abby Spurdle <spurdle.a@gmail.com>

Author Abby Spurdle

URL https://sites.google.com/site/spurdlea/r

Description Supports formatted nested/partitioned matrices, formatted object arrays and similar formatted data.frame(s), via coercion. These objects can be printed with plain text mark up, including their partitions and submatrices. Also, includes an SQL-like select function, grouped head functions and combined head and tail functions.

Imports methods

Suggests intoo, Matrix, barsurf

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-09 10:10:02 UTC

R topics documented:

1_subsetting_operators *Subsetting Operators*

Description

Subsetting operators for vector-like objects.

Usage

```
## S3 method for class 'ObjectArray'
x[[...]]
## S3 replacement method for class 'ObjectArray'
x[[...]] <- value

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

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

Arguments

`x`	A vector-like object.
`i, j, ...`	The indices.
`value`	The value to assign. For SectMatrix, the dimensions of the value need to match the dimensions set by setmap.

Details

ObjectArray and NestMatrix, are similar to lists, so:
Double bracket subsetting gets or sets a single element.

For SectMatrix (including PartMatrix):
Single bracket subsetting indexes submatrices and double bracket subsetting indexes the combined matrix.

Note that this may be changed in the future.

Examples

```
s = matrix (1:16, 4, 4)

#2x2 nested matrix, with 4 2x2 submatrices
x = as.NestMatrix (s, 2, 2)
x

x [[1, 1]]       #2x2 matrix
x [[1, 1]][2, 2] #6

#4x4 nested matrix, with 16 scalar values
x = as.NestMatrix.2 (s)
x

x [[2, 2]]       #6

#4x4 partitioned matrix, partitioned into 4 2x2 submatrices
x = as.PartMatrix (s, 2, 2)
x

x [1, 1]        #2x2 matrix
x [[2, 2]]      #6
x [1, 1][2, 2]  #6
```

`2_object_arrays` *Object Arrays*

Description

One, two and higher-dimensional object arrays.

Usage

```
ObjectArray (dim, names, default.value=NA)
as.ObjectArray (x, dim, names)
```

Arguments

`dim`	Dimensions, same as dim, in standard array objects.
`names`	List of names, which should match dim, above.
`default.value`	Default value, for each element.
`x`	A list with the same length as prod (dims).

Details

ObjectArray(s) are similar to list matrices and list arrays, however, there are some differences, especially, in the way that they're formatted.

Examples

```
x = ObjectArray (c (2, 2) )
x [[1, 1]] = x [[2, 2]] = x
x
```

`3_matrix_like_objects` *Matrix-Like Objects*

Description

Nested, sectioned and partitioned matrices.

Usage

```
NestMatrix (nr, nc, rnames, cnames, default.value=NA)
SectMatrix (nsect, nr, nc, rnames, cnames, default.value=NA)
PartMatrix (Rb, Cb, nr, nc, rnames, cnames, default.value=NA)

as.NestMatrix (x, Rb, Cb, rnames, cnames)
as.NestMatrix.2 (x, rnames, cnames)

as.SectMatrix (x, nsect, rnames, cnames)
as.PartMatrix (x, Rb, Cb, rnames, cnames)

setmap (x, ...) <- value
getmap (x, ...)

rnames (x)
cnames (x)
rnames (x, ...) <- value
cnames (x, ...) <- value
```

Arguments

`nsect`	The number (possibly in two or more dimensions) of submatrices.
`Rb, Cb`	The inter-row and inter-column indices.
`nr, nc`	The dimensions.
`rnames, cnames`	Character vectors giving the row and column names.
`default.value`	The default value of entries in the matrix.
`value`	For setmap, a two by two integer-valued matrix, or a length-4 vector, giving top-left row index, top-left column index, bottom-right row index and bottom-right column index. For the rnames/cnames functions, a character vector.
`x`	For the as functions, a standard matrix or data.frame. For the getmap/setmap functions, a SectMatrix object. For the rnames/cnames functions, any matrix-like object from above.
`...`	For getmap/setmap, the indices of the submatrix.

Details

Refer to the vignette for more information.

Examples

```
s = matrix (1:100, 10, 10)

x = as.SectMatrix (s, 2)
setmap (x, 1) = c (2, 2, 8, 8)
setmap (x, 2) = c (3, 3, 9, 9)
x

x = as.PartMatrix (s, 5, c (2, 4, 6, 8) )
x
```

4_most_methods *Most Methods*

Description

Dim, print, format, head and tail methods for vector-like objects.

Usage

```
## S3 method for class 'ObjectArray'
dim(x)
## S3 method for class 'NestMatrix'
dim(x)
## S3 method for class 'SectMatrix'
dim(x)

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

## S3 method for class 'ObjectArray'
format(x, na.string="", ...)
## S3 method for class 'NestMatrix'
format(x, na.string="", ...)
## S3 method for class 'SectMatrix'
format(x, na.string="", ...)

## S3 method for class 'SectMatrix'
as.matrix(x, ...)

## S3 method for class 'VectorLike'
head(x, n=6, ...)
## S3 method for class 'VectorLike'
tail(x, n=6, ...)
```

Arguments

x	A VectorLike object.
na.string	What to format NAs as.
n	Number of items.
...	Other arguments.

Details

Refer to vignette for more information.

Note that the dimensions of object arrays and nested matrices are the dimensions of the top level (or collapsed) object. However, the dimensions of sectioned matrices (including partitioned matrices) are the dimensions or the combined (or expanded) matrix.

Note that the format method for ObjectArray, calls the objtag function, for each of its elements.

Value

The format, head and tail functions return formatted character matrices.

Examples

```
s = matrix (1:100, 10, 10)
x = as.PartMatrix (s, 5, c (2, 4, 6, 8) )

dim (x)
as.matrix (x)
head (x, 2)
```

5_element_level_formatting

Object Array Formatting

Description

Element-level formatting functions for object arrays.

Usage

```
objtag (...)

## Default S3 method:
objtag(x, ...)
## S3 method for class 'ObjectArray'
objtag(x, ...)
## S3 method for class 'NestMatrix'
objtag(x, ...)
## S3 method for class 'SectMatrix'
```

```
objtag(x, ...)
## S3 method for class 'PartMatrix'
objtag(x, ...)
## S3 method for class 'function'
objtag(x, ...)
## S3 method for class 'list'
objtag(x, ...)
## S3 method for class 'matrix'
objtag(x, ...)
## S3 method for class 'data.frame'
objtag(x, ...)
```

Arguments

x	An object.
...	.

Details

These functions map an object to a single compact string, regardless of the length of the object.

The format method for ObjectArray calls the objtag function for each of its elements.

To format an object of a different class, you need to write an (S3) objtag method for that class.

Value

Each method returns a single compact string.
(i.e. A length-one character vector, that's relatively short).

If you write a new objtag method, it should do the same.

Examples

```
#near-trivial classes
alphabet.1 = function ()
    structure (LETTERS, class="alphabet.1")
alphabet.2 = function ()
    structure (sample (LETTERS), class="alphabet.2")

#near-trivial 2x2 object array
x = ObjectArray (c (2, 2) )
x [[1, 1]] = alphabet.1 ()
x [[2, 1]] = alphabet.1 ()
x [[1, 2]] = alphabet.2 ()
x [[2, 2]] = alphabet.2 ()

#printed with default formatting
x

#objtag methods
objtag.alphabet.1 = function (x)
    paste ("<A1 ", x [1], ":", x [26], ">", sep="")
```

```
objtag.alphabet.2 = function (x)
    paste ("<A2 ", x [1], ":", x [26], ">", sep="")

#reprinted with custom formatting
x
```

`6_select_function` *Select Function*

Description

SQL-like select function, for grouping, sorting and grouped aggregation functions, with limited support for automatic formatting.

Usage

```
select (...)

#select constructs:
#   from (...)
#   group.by (...)
#   partition.by (...)
#   sort.by (...)
#   where (...)
```

Arguments

...	Refer to details section.

Details

This function is a R (only) function with standard R syntax, but nonstandard evaluation.

Currently, it supports a subset of SQL select functionality, and is designed for convenience only and not for high performance or large datasets.

The function needs to be called with one or more unique variable names and a from construct, and optional group.by, partition by, sort.by and where constructs, along with optional new variables (currently, for use with group.by only). These constructs need to be included inside the select function call, and not called separately. Results (i.e. table rows) may not be unique. So, if you need unique results, then you should use the unique function after calling the select function.

Alternatively, a dot (.) may be used rather than specifying each variable name. Currently, dot is not allowed with other variable names. And if there's a group.by construct, any variables not included in group.by (or for use with group.by) are discarded. However, this may be changed in the future.

Currently, the from construct needs to include the name of a single data.frame.

The group.by construct should include comma-separated variable names only, giving variable names from the source data. The partition.by construct needs a single variable name from variables present after possible grouping. The sort.by construct should include comma-separated variables names

from the variables present after possible grouping, optionally prefixed with a plus or a minus, for ascending and descending order, respectively. The where statement, should include simple comma-separated (in)equalities, such as x == 1 or y >= 1000, with variable names from either the grouped data or source data.

Re-iterating, new variables are for use with group.by only. If you need to create arbitrary variables, then you need to do it before or after calling select. Currently, new variables need to be defined using the "<-" operator and not "=". The operand to the left gives the new variable name, and the operand to the right can be any call that maps variables from the grouped data to a scalar value. Currently, new variables are listed after old variables, regardless of their input order.

Note that the partition.by construct is primarily for situations where you only want to print the results, without further operations. Also, note that subsetting operations corresponding to the where construct, may be applied at two different points in the function's execution, before or after grouping, depending on which variables are involved.

Expanding on the previous point, the execution order is:
(1) Apply the where construct, for variables in the source data.
(2) Apply the group.by construct and select variables.
(3) Apply the where construct, for variables in the grouped data.
(4) Apply the sort.by construct.
(5) Renumber the rows.
(6) Apply the partition.by construct.

Value

A data.frame unless it includes a partition.by construct.

Partitioning returns a SectMatrix object, with row separators between each change in the partitioning variable, and column separators around it. Also, repetitions in the partitioning variable are replaced with spaces.

Examples

```
#all variables
select (., from (mtcars) )

#some variables
select (am, cyl, mpg, from (mtcars) )

#grouped by am and cyl
#with mean of mpg, by group
select (am, cyl,
    from (mtcars),
    group.by (am, cyl),
        count <- length (mpg),
        mean.mpg <- mean (mpg) )

#same as above
#but partitioned and sorted
select (am, cyl,
    from (mtcars),
    group.by (am, cyl), partition.by (am), sort.by (-am, -mean.mpg),
```

```
        count <- length (mpg),
        mean.mpg <- mean (mpg) )

#earlier example but with a where construct
select (am, cyl, mpg, from (mtcars), where (mpg >= 20) )
```

`7_head_and_tail_generalizations` *Head and Tail Generalizations*

Description

Combined head and tail functions, and a grouped head function for data.frames.

Usage

```
headg (...)
headt (...)

## Default S3 method:
headt(x, nh=3, nt=nh, ...)
## S3 method for class 'ObjectArray'
headt(x, nh=3, nt=nh, ...)
## S3 method for class 'MatrixLike'
headt(x, nh=3, nt=nh, ...)
## S3 method for class 'matrix'
headt(x, nh=3, nt=nh, ...)

## S3 method for class 'data.frame'
headg(x, group.by, n=3, ...)
## S3 method for class 'data.frame'
headt(x, nh=3, nt=nh, ...)
```

Arguments

`x`	An object.
`group.by`	A string, giving the name of the grouping variable.
`n, nh, nt`	Similar to head and tail.
`...`	Ignored.

Details

Refer to vignette for more information.

Value

These functions return formatted character matrices.

Examples

```
s = matrix (1:100, 10, 10)
x = as.PartMatrix (s, 5, c (2, 4, 6, 8) )
headt (x, c (2, 6) , 2)

headg (iris, "Species")

headt (trees)
```

Index