Package: listArray (via r-universe)

September 5, 2024

Title Incomplete Array with Arbitrary R Objects as Indices

Version 0.1.1

Description The aim of the package is to create data objects which allow for accesses like x[``test"] and x[``test",``test"].

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Sigbert Klinke [aut, cre]

Maintainer Sigbert Klinke <sigbert@hu-berlin.de>

Date/Publication 2020-09-10 14:40:03 UTC

Repository https://sigbertklinke.r-universe.dev

RemoteUrl https://github.com/cran/listArray

RemoteRef HEAD

RemoteSha 8f81a6e9111562a3901d377e24b320c705942236

Contents

hasKey *hasKey*

Description

Checks if a specific index exists

Usage

```
hasKey(x, ...)
```

Arguments

x	listArray object
...	index to check

Value

logical: TRUE index exists, FALSE index exists not

Examples

```
l <- listArray()
l[1] <- 1
hasKey(l, 1)
l[2,3] <- "test"
hasKey(l, 2, 3)
l[2:3] <- "vector"
hasKey(l, 2:3)
l['iris'] <- iris
hasKey(l, iris) # FALSE
l[mean] <- mean
hasKey(l, mean)
# if you have not stored NULL objects in your listArray
is.null(l[mean])
is.null(l[iris])
```

key *key*

Description

Creates a character key from arbitray R objects. For more details see vignette("listArray").

Usage

```
key(...)
```

Arguments

...	R objects

Value

a unique character key

Examples

```
key(1)
key(2,3)
key(1:3)
key(mean)
key('test')
key(letters[1:5])
key(list(1))
key(iris)
```

`keys` *Returns a list of indices as string, list or unique values (like* `dimnames`*).*

Description

Returns a list of indices as string, list or unique values (like `dimnames`).

Usage

```
keys(x, type = "character")
```

Arguments

`x`	listArray object
`type`	character: return the indices as string, list or unique values (default: `character`)

Value

Returns the indices as string, list or unique indices. If `type` is

`type="character"` a character vector with the retranslated indices

`type="list"` as list of lists with the retranslated indices

`type="names"` as list of lists with the retranslated unique(!) indices like `dimnames`

Examples

```
l <- listArray(matrix(1:9, 3, 3))
k <- keys(l)
k
# access object in listArray
pos <- which(k=='3, 2')
l[[pos]]
#
l["test"] <- "test"
keys(l, 'c') # as keys(l)
keys(l, 'l')
keys(l, 'n')
# Note that l['test'][3] will deliver NULL since the entry does not exist
```

`listArray` *listArray*

Description

Creates either an empty `listArray` object or a `listArray` object from a vector, array or list. See also `vignette("listArray")`.

Usage

```
listArray(x, ...)

## Default S3 method:
listArray(x, use.names = TRUE, ignore = NULL, env = FALSE, ...)
```

Arguments

`x`	vector, array or list
`...`	further arguments given to `new.env` if an environment is used
`use.names`	logical: if the names from `x` or indices should be used (default: `TRUE`)
`ignore`	values to ignore for the listArray object
`env`	logical: if the listArray creates a list or an environment (default: `FALSE`)

Value

a `listArray` object

Examples

```
# empty listArray
l <- listArray()
# listArray from a numerical vector
v <- 1:5
l <- listArray(v)
# listArray from a text vector
v <- letters[1:5]
l <- listArray(v)
#' # listArray from a matrix
m <- matrix(1:9, 3, 3)
l <- listArray(m)
#' # listArray from a list
v <- as.list(1:5)
l <- listArray(v)
```

[.listArray *Extract or Replace one Element of a* listArray

Description

Operators acting on one element of a `listArray` to extract or replace it.

Usage

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

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

Arguments

`x`	object from which to extract a element or in which to replace a element.
`...`	indices specifying the element to extract or replace. Indices can consist of any R Object.
`value`	value which replaces a `listArray` element

Value

Returns or sets the selected element.

Examples

```
l <- listArray()
l[1] <- 1
l[1]
#
l[2,3] <- "test"
l[2,3]
#
l[2:3] <- "vector"
l[2:3]
l[2,3]
#
l['iris'] <- iris
head(l['iris'])
#
l[letters[1:5]] <- letters[1:5]
l[letters[1:5]]
#
l[mean] <- mean
l[mean](0:10)
```

Index