

# Package ‘crochet’

May 21, 2020

**Version** 2.3.0

**License** MIT + file LICENSE

**Title** Implementation Helper for '[' and '[<-' of Custom Matrix-Like Types

**Description** Functions to help implement the extraction / subsetting / indexing function '[' and replacement function '[<-' of custom matrix-like types (based on S3, S4, etc.), modeled as closely to the base matrix class as possible (with tests to prove it).

**URL** <https://github.com/agrueneberg/crochet>

**BugReports** <https://github.com/agrueneberg/crochet/issues>

**Depends** R (>= 3.0.0)

**Imports** methods

**Suggests** tinytest, knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Alexander Grueneberg [aut, cre]

**Maintainer** Alexander Grueneberg <cran@agrueneberg.info>

**Repository** CRAN

**Date/Publication** 2020-05-20 22:40:02 UTC

## R topics documented:

crochet-package . . . . .	2
convertIndex . . . . .	2
extract . . . . .	3
ijtok . . . . .	5
ktoj . . . . .	6
replace . . . . .	6

<b>Index</b>	<b>9</b>
--------------	----------

crochet-package

*Implementation Helper for '[' and '['<-' of Custom Matrix-Like Types*

---

**Description**

Functions to help implement the extraction / subsetting / indexing function '[' and replacement function '['<-' of custom matrix-like types (based on S3, S4, etc.), modeled as closely to the base matrix class as possible (with tests to prove it).

**Important Functions**

`extract`: Create an implementation of '[' for custom matrix-like types.

`replace`: Create an implementation of '['<-' for custom matrix-like types.

`convertIndex`: Convert non-numeric index types to positive integers.

**Discussion**

R used to export functions for index conversion such as `makeSubscript`, `vectorSubscript`, and `arraySubscript` in `src/main/subscript.c` to package developers until R 2.3.1. These exports were removed in R 2.4.0 as part of a **cleanup**. `arraySubscript` was later **re-added** as some packages such as `arules` and `cba` still rely on it. I still need to investigate whether `arraySubscript` would be useful for this package.

**See Also**

`vignette("StringMatrix", package = "crochet")` for a vignette containing a full example for a custom type.

---

convertIndex*Convert Non-Numeric Index Types to Positive Integers*

---

**Description**

Converts different index types such as negative integer vectors, character vectors, or logical vectors into positive integer vectors.

**Usage**

```
convertIndex(x, i, type, allowDoubles = FALSE)
```

**Arguments**

<code>x</code>	A matrix-like object.
<code>i</code>	The index to convert: may be a one-dimensional or two-dimensional logical, character, integer, or double vector.
<code>type</code>	The type of index to convert to: <code>k</code> is a one-dimensional index, <code>i</code> is the part of a two-dimensional index that determines the rows, and <code>j</code> is the part of a two-dimensional index that determines the columns.
<code>allowDoubles</code>	If set, indices of type double are not converted to integers if the operation would overflow to support matrices with <code>nrow()</code> , <code>ncol()</code> , or <code>length()</code> greater than the largest integer that can be represented ( <code>.Machine\$integer.max</code> ).

**Value**

The converted index.

**See Also**

[extract](#) and [replace](#) to generate implementations for `[` and `[<-` for custom types that use `convertIndex` under the hood.

**Examples**

```
x <- matrix(data = rnorm(25), nrow = 5, ncol = 5)
dimnames(x) <- list(letters[1:5], letters[1:5])

convertIndex(x, c(1, 2, 3), "k")
convertIndex(x, -25, "k")
convertIndex(x, c(TRUE, FALSE), "k")

convertIndex(x, c(1, 2, 3), "i")
convertIndex(x, -5, "i")
convertIndex(x, c(TRUE, FALSE), "i")
convertIndex(x, c("a", "b", "c"), "i")

convertIndex(x, c(1, 2, 3), "j")
convertIndex(x, -5, "j")
convertIndex(x, c(TRUE, FALSE), "j")
convertIndex(x, c("a", "b", "c"), "j")
```

---

 extract

---

*Create an Implementation of [ For Custom Matrix-Like Types*


---

**Description**

`extract` is a function that converts different index types such as negative integer vectors, character vectors, or logical vectors passed to the `[` function as `i` (e.g. `X[i]`) or `i` and `j` (e.g. `X[i, j]`) into positive integer vectors. The converted indices are provided as the `i` parameter of `extract_vector` or `i` and `j` parameters of `extract_matrix` to facilitate implementing the extraction mechanism for custom matrix-like types.

**Usage**

```
extract(extract_vector, extract_matrix, allowDoubles = FALSE)
```

**Arguments**

- `extract_vector` A function in the form of `function(x, i, ...)` that takes a subset of `x` based on a single index `i` and returns a vector.
- `extract_matrix` A function in the form of `function(x, i, j, ...)` that takes a subset of `x` based on two indices `i` and `j` and returns a matrix.
- `allowDoubles` If set, indices of type `double` are not converted to integers if the operation would overflow to support matrices with `nrow()`, `ncol()`, or `length()` greater than the largest integer that can be represented (`.Machine$integer.max`).

**Details**

The custom type must implement methods for `length`, `dim` and `dimnames` for this function to work. Implementing methods for `nrow`, `ncol`, `rownames`, and `colnames` is not necessary as the default method of those generics calls `dim` or `dimnames` internally.

Optional arguments are supported and will be passed to `extract_vector` and `extract_matrix` as long as they are named.

**Value**

A function in the form of `function(x, i, j, ..., drop = TRUE)` that is meant to be used as a method for `[` for a custom type.

**See Also**

`vignette("StringMatrix", package = "crochet")` for a vignette containing a complete example on how to use `extract` to implement `[` for a custom type.

**Examples**

```
b <- matrix(data = rnorm(25), nrow = 5, ncol = 5)
dimnames(b) <- list(letters[1:5], letters[1:5])

a <- structure(list(), class = "TestMatrix")

dim.TestMatrix <- function(x) {
  dim(b)
}

dimnames.TestMatrix <- function(x) {
  dimnames(b)
}

extract_vector <- function(x, i) {
  # Dispatch to b instead to x for this demo
  b[i, drop = FALSE]
```

```
}  
extract_matrix <- function(x, i, j) {  
  # Dispatch to b instead to x for this demo  
  b[i, j, drop = FALSE]  
}  
  
`[.TestMatrix` <- extract(extract_vector = extract_vector, extract_matrix = extract_matrix)
```

---

ijtok

*Convert Two-Dimensional Indices  $i$  and  $j$  to One-Dimensional Index  $k$*

---

## Description

ijtok is a helper function that converts two-dimensional indices  $i$  and  $j$  to a one-dimensional index  $k$ . This can be useful if, for example, one-dimensional indexing is easier to implement than two-dimensional indexing.

## Usage

```
ijtok(x, i, j)
```

## Arguments

$x$	A matrix-like object.
$i$	The first component of a two-dimensional index.
$j$	The second component of a two-dimensional index.

## Details

It is assumed that all indices are one-based.

## Value

A one-dimensional index.

---

ktoij	<i>Convert One-Dimensional Index k to Two-Dimensional Indices i and j</i>
-------	---

---

### Description

ktoij is a helper function that converts a one-dimensional index k to two-dimensional indices i and j. This can be useful if, for example, two-dimensional indexing is easier to implement than one-dimensional indexing.

### Usage

```
ktoij(x, k)
```

### Arguments

x	A matrix-like object.
k	A one-dimensional index.

### Details

It is assumed that all indices are one-based.

### Value

A list containing indices i and j.

---

replace	<i>Create an Implementation of [<math>\leftarrow</math> For Custom Matrix-Like Types</i>
---------	--

---

### Description

replace is a function that converts different index types such as negative integer vectors, character vectors, or logical vectors passed to the [ $\leftarrow$  function as i (e.g. X[i]) or i and j (e.g. X[i, j]) into positive integer vectors. The converted indices are provided as the i parameter of replace\_vector or i and j parameters of replace\_matrix to facilitate implementing the replacement mechanism for custom matrix-like types. Values are recycled to match the replacement length.

### Usage

```
replace(replace_vector, replace_matrix, allowDoubles = FALSE)
```

**Arguments**

- `replace_vector` A function in the form of `function(x, i, ..., value)` that replaces a vector subset of `x` based on a single index `i` with the values in `value` and returns `x`.
- `replace_matrix` A function in the form of `function(x, i, j, ..., value)` that replaces a matrix subset of `x` based on two indices `i` and `j` with the values in `value` and returns `x`.
- `allowDoubles` If set, indices of type `double` are not converted to integers if the operation would overflow to support matrices with `nrow()`, `ncol()`, or `length()` greater than the largest integer that can be represented (`.Machine$integer.max`).

**Details**

The custom type must implement methods for `length`, `dim` and `dimnames` for this function to work. Implementing methods for `nrow`, `ncol`, `rownames`, and `colnames` is not necessary as the default method of those generics calls `dim` or `dimnames` internally.

**Value**

A function in the form of `function(x, i, j, ..., value)` that is meant to be used as a method for `[<-` for a custom type.

**See Also**

`vignette("StringMatrix", package = "crochet")` for a vignette containing a complete example on how to use `replace` to implement `[<-` for a custom type.

**Examples**

```
b <- matrix(data = rnorm(25), nrow = 5, ncol = 5)
dimnames(b) <- list(letters[1:5], letters[1:5])

a <- structure(list(), class = "TestMatrix")

dim.TestMatrix <- function(x) {
  dim(b)
}

dimnames.TestMatrix <- function(x) {
  dimnames(b)
}

extract_vector <- function(x, i) {
  # Dispatch to b instead to x for this demo
  b[i, drop = FALSE]
}

extract_matrix <- function(x, i, j) {
  # Dispatch to b instead to x for this demo
  b[i, j, drop = FALSE]
}
```

```
`[.TestMatrix` <- extract(extract_vector = extract_vector, extract_matrix = extract_matrix)

replace_vector <- function(x, i, value) {
  .GlobalEnv$i <- i
  .GlobalEnv$value <- value
  # Dispatch to b instead to x for this demo
  with(.GlobalEnv, b[i] <- value)
  # Don't forget to return x
  return(x)
}

replace_matrix <- function(x, i, j, value) {
  .GlobalEnv$i <- i
  .GlobalEnv$j <- j
  .GlobalEnv$value <- value
  # Dispatch to b instead to x for this demo
  with(.GlobalEnv, b[i, j] <- value)
  # Don't forget to return x
  return(x)
}

`[<-.TestMatrix` <- replace(replace_vector = replace_vector, replace_matrix = replace_matrix)
```



# Index

\*Topic **package**

crochet-package, 2  
[, 4

colnames, 4, 7  
convertIndex, 2  
crochet-package, 2

dim, 4, 7  
dimnames, 4, 7

extract, 3, 3

ijtok, 5

ktoij, 6

length, 4, 7

ncol, 4, 7  
nrow, 4, 7

replace, 3, 6  
rownames, 4, 7