Package 'maskr'

July 22, 2025

Type Package

Title Visual Class for Vectors with Non-Publishing Requirements

Version 0.1.0

Description Create vectors with sticky flags for elements that should not be displayed. Numeric vectors have basic subset and arithmetic methods implemented.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, pillar, rlang, vctrs (>= 0.6.5)

RoxygenNote 7.3.2

Suggests dplyr, fansi, testthat (>= 3.0.0), tidyr, withr

Config/testthat/edition 3

URL https://github.com/inpowell/maskr

BugReports https://github.com/inpowell/maskr/issues

NeedsCompilation no

Author Ian Powell [cre, aut, cph]

Maintainer Ian Powell <powell.ian.n@gmail.com>

Repository CRAN

Date/Publication 2024-08-23 08:50:07 UTC

Contents

`format.maskr_masked` *Display and write masked vectors*

Description

Masked vectors generally cannot be converted to other types without unmasking them. However, they can be written as character vectors (losing the underlying masked data) for printing to the console, and saving to files.

Usage

```
## S3 method for class 'maskr_masked'
format(x, ..., rep = getOption("maskr.replacement", "n.p."))

## S3 method for class 'maskr_masked'
as.character(x, ..., rep = getOption("maskr.replacement", "n.p."))
```

Arguments

`x`	A masked vector.
`...`	For `format()`, passed to other format methods. For `as.character()`, ignored.
`rep`	A single character value that defines a string to show instead of the underlying value for data that is masked.

Value

A character vector with masked values replaced by `rep`.

Examples

```
abc <- masked(letters, letters %in% c('a', 'e', 'i', 'o', 'u'))
# Prints with default n.p. label - uses format() under the hood
print(abc)

# Format as string with * instead of n.p.
format(abc, rep = '*')
print(abc, rep = '*') # Also works with print()

as.character(abc) # Similar to format(), but without alignment

# Dispatches format to underlying data
nums <- masked(
  c(1:12, NA, NA, 15:26),
  rep_len(c(FALSE, FALSE, FALSE, TRUE, FALSE), 26L)
)
print(nums)
print(nums, rep = '*') # Automatically right-aligned for numeric types
```

```
# as.character() useful for saving tables without revealing data
alphanum <- data.frame(alpha = abc, num = nums)
write.csv(head(alphanum))
```

`masked` *Create a masked atomic vector*

Description

Masked vectors contain a base R data type that can be used in calculations, but is not revealed by default on the console or when converted to a character. This can be useful for preventing publication of small cells, such as in official statistics.

Usage

```
masked(data = numeric(), mask = logical())

unmask(masked)

unmask(masked) <- value

mask(masked)

mask(masked) <- value
```

Arguments

`data`	An atomic vector to mask values from. Lists and data frames are not supported.
`mask`	A logical vector that indicates which values of `data` to mask.
`masked`	A masked vector to extract fields from.
`value`	A replacement vector for unmasked data, or the mask flags.

Value

A masked vector.

Pretty printing and conversion to character

Masked vectors have pretty printing that replaces masked values of a vector with `n.p.` in the console (customisable with `options(maskr.replacement = ...)`).

Converting a masked vector to character results in masked vectors being replaced with `n.p.`, or its alternative in `getOption('maskr.replacement')`.

Masked vectors cannot be converted to their raw types, to prevent accidental release of data. Instead, use `unmask()` to explicitly unmask a vector.

Arithmetic and unary mathematical functions

Elementwise arithmetic operators have been implemented in masked vectors. The resulting data will be as if performing the operation on the unmasked vectors. Mask flags are sticky; the result of any operation involving a masked value will also be masked.

Most Math and Summary group generics have been implemented. These first force the underlying data to double type (or logical for `any()` and `all()`). Results from summary functions will be masked if any input is masked, while elementwise operations will preserve the mask from their input. Cumulative functions (`cumsum()`, `cummean()` and friends) are not implemented.

Examples

```
# Mask vowels in the alphabet
masked(letters, letters %in% c('a', 'e', 'i', 'o', 'u'))

# Mask doubles...
sepals <- head(masked(iris$Sepal.Length, iris$Sepal.Length > 5))
sepals
# ...and get back the underlying values
unmask(sepals)

# Use a different mask character
op <- options(maskr.replacement = '*')
sepals
options(op)

# Mask integers in a vector strictly between 0 and 5
x <- 0:8
xm <- masked(x, 0L < x & x < 5L)

# Arithmetic with unmasked values does not change mask:
xm / xm[8] * 100
2 ^ xm
# But arithmetic with a masked value will mask outputs:
xm + xm[3]
xm - rev(xm)

# The mean will be masked, because at least one if its inputs is masked
mean(xm)
unmask(mean(xm))

# Other mathematical functions will keep the same mask
log1p(xm)
unmask(log1p(xm))
```

Index