Package 'ox'

December 15, 2021

Type Package

Title Shorthand if-Else

Version 0.1.0

Maintainer Dawid Kałędkowski <dawid.kaledkowski@gmail.com>

Description Short hand if-else function to easily switch the values depending on a logical condition.

License GPL (>= 2)

BugReports https://github.com/gogonzo/ox/issues

Encoding UTF-8

Language en-US

Depends R (>= 3.0)

Suggests checkmate, dplyr, magrittr, knitr, rmarkdown, testthat, utils

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Dawid Kałędkowski [aut, cre] (<https://orcid.org/0000-0001-9533-457X>), Paweł Rucki [aut]

Repository CRAN

Date/Publication 2021-12-15 09:00:02 UTC

R topics documented:

ox *Short hand if-else*

Description

Short hand if-else functions for simple values switching.

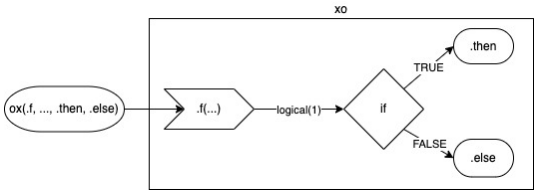
Usage

```
ox(.f, ..., .then = list(...)[[1]], .else = rev(list(...))[[1]])

xo(.f, ..., .then = list(...)[[1]], .else = rev(list(...))[[1]])
```

Arguments

`.f`	(`function`) to be applied on `...`. Must return single logical value.
`...`	arguments passed to the `.f`.
`.then`	A positive-replacement. NOTE, that if `.then` is not specified directly by named argument then the first argument from `...` will be taken.
`.else`	A negative-replacement. NOTE, that if `.else` is not specified directly by named argument then the last argument from `...` will be considered as a replacement.

Details



`ox` evaluates function `.f` which returns a single logical value and depending on the result `ox` returns:

- on `TRUE` returns `.then`.
- on `FALSE` returns `.else`.

It's important to note is that `.then` and `.else` have a default values set as the first and the last element of the `...`. This means that they don't have to be specified if both are arguments of `.f`.
If any of them are not arguments of `.f` then this argument should be specified directly as `.then` or `.else`, otherwise it will be passed to the function.

As, far as `.then` and `.else` won't be specified order of arguments in `...` matters for `ox` but it's up to you if they are passed in the correct order to the `.f`. Then one, might consider name the argument so `.f` will be executed as expected. Consider following example

```
greater_than_by <- function(x, y, by) x > (y + by)
ox(greater_than_by, x = 5, by = 3, y = 3)
```

In above, one needs to move y to the end so that it will be considered as .else.

To invert the switch one can use xo which is equivalent of ox(Negate(.f),...,.then,.else).

Value

object identical to .then or .else depending on the condition result.

Examples

```
# if (is.null(NULL)) NULL else 1
ox(NULL, .f = is.null, .else = 1)

# if (is("text", "character")) "text" else "not a character"
ox("text", .f = is, "character", .else = "not a character")

# if (1 > 2) 1 else 2
ox(`>`, 1, 2)
# if (!is.null(NULL)) NULL else 1
xo(NULL, .f = is.null, .else = 1)

# if (!is("text", "character")) "text" else "not a character"
xo("text", .f = is, "character", .else = "not a character")

# if (!1 > 2) 1 else 2
xo(`>`, 1, 2)
```

---

vectorized-ox *Vectorized* ox

Description

Switch the values of the vector by another on specific indices.

Usage

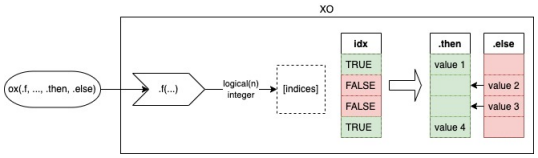
```
OX(.f, ..., .then = list(...)[[1]], .else = rev(list(...))[[1]])

XO(.f, ..., .then = list(...)[[1]], .else = rev(list(...))[[1]])
```

Arguments

.f	(function) evaluated as .f(...). Must return a vector of indices (logical or integer), which defines which values will be replaced by .else.
...	arguments passed to the .f.
.then	(list, atomic, NULL) A positive-replacement. NOTE, that if .then is not specified directly by named argument then the first argument from ... will be taken.

`.else` (`list`, `atomic`, `NULL`)
A negative-replacement. Should be of length equal to length of `.then`, or single value or `NULL`. NOTE, that if `.else` is not specified directly by named argument then the last argument from `...` will be considered as a replacement.

Details



#' `OX` evaluates function `.f` which returns a vector of indices which are then decide which values of `.then` are replaced by else.

```
.then[!idx] <- .else[!idx]
```

Consequence of above is that `idx = .f(...)` should be a `logical` vector or `integer` vector which would be valid indices for `.then` and `.else`. This means that `.then` and `.else` should be of the same length, but there are two exceptions:

- when `.else` is a single value, than this value will replace `.then` at returned indices `.then[!idx] <-.else`
- when `.else` is `NULL`

To invert the switch one can use `XO` which is equivalent of `OX(Negate(.f),...,.then,.else)`.

Value

`atomic` or `list`. Returned object is a `.then` object with elements replaced by `.else` depending on a result of the logical condition.

Examples

```
# switch values of the vector when condition is true
OX(is.na, c(1, NA, 3), .else = c(2, 2, 2))

# use OX to invert negate the condition
XO(is.na, c(1, NA, 3), .else = c(2, 2, 2))
```

Index