Package 'intrval'

May 8, 2026

Type Package

Title Relational Operators for Intervals

Version 1.0-0

Date 2025-07-12

Maintainer Peter Solymos <psolymos@gmail.com>

Description Evaluating if values
of vectors are within different open/closed intervals
(`x %[]% c(a, b)`), or if two closed
intervals overlap (`c(a1, b1) %[]o[]% c(a2, b2)`).
Operators for negation and directional relations also implemented.

License GPL-2

Imports fpCompare

URL https://github.com/psolymos/intrval

BugReports https://github.com/psolymos/intrval/issues

LazyLoad yes

NeedsCompilation no

Author Peter Solymos [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7337-1740>)

Repository CRAN

Date/Publication 2025-07-14 00:10:02 UTC

Contents

`intrval-package` *Relational Operators for Intervals*

Description

Evaluating if values of vectors are within different open/closed intervals (‘x %[]% c(a, b)‘), or if two closed intervals overlap (‘c(a1, b1) %[]o[]% c(a2, b2)‘). Operators for negation and directional relations also implemented.

Details

The DESCRIPTION file:

Package:	intrval
Type:	Package
Title:	Relational Operators for Intervals
Version:	1.0-0
Date:	2025-07-12
Authors@R:	c(person(given = "Peter", family = "Solymos", comment = c(ORCID = "0000-0001-7337-1740"), role = c("aut
Maintainer:	Peter Solymos <psolymos@gmail.com>
Description:	Evaluating if values of vectors are within different open/closed intervals (‘x %[]% c(a, b)‘), or if two closed int
License:	GPL-2
Imports:	fpCompare
URL:	https://github.com/psolymos/intrval
BugReports:	https://github.com/psolymos/intrval/issues
LazyLoad:	yes
LazyData:	true
Author:	Peter Solymos [aut, cre] (<https://orcid.org/0000-0001-7337-1740>)

Index of help topics:

Relational operators for value-to-interval comparisons: `%[]%` and alikes.

Relational operators for interval-to-interval comparisons: `%[o]%` and alikes.

Negated value matching: `%ni%`.

Author(s)

Peter Solymos [aut, cre] (<https://orcid.org/0000-0001-7337-1740>)

Maintainer: Peter Solymos <psolymos@gmail.com>

`intrval` *Relational Operators Comparing Values to Intervals*

Description

Functions for evaluating if values of vectors are within intervals.

Usage

```
x %[]% interval
x %)(% interval
x %[<]% interval
x %[>]% interval

x %[)% interval
x %)[% interval
x %[<)% interval
x %[>)% interval

x %(]% interval
x %](% interval
x %(<]% interval
x %(>]% interval

x %()% interval
x %][% interval
x %(<)% interval
x %(>)% interval

intrval_types(type = NULL, plot = FALSE)
```

Arguments

`x`	vector or `NULL`: the values to be compared to interval endpoints.
`interval`	vector, 2-column matrix, list, or `NULL`: the interval end points.
`type`	character, type of operator for subsetting the results. The default `NULL` means that all types will be displayed.
`plot`	logical, whether to plot the results, or print a table to the console instead.

Details

Values of x are compared to interval endpoints a and b (a <= b). Endpoints can be defined as a vector with two values (c(a, b)): these values will be compared as a single interval with each value in x. If endpoints are stored in a matrix-like object or a list, comparisons are made element-wise. If lengths do not match, shorter objects are recycled. These value-to-interval operators work for numeric (integer, real) and ordered vectors, and object types which are measured at least on ordinal scale (e.g. dates), see Examples. Note: interval endpoints are sorted internally thus ensuring the condition a <= b is not necessary.

The type argument or the specification of the special function determines the open ((and)) or closed ([and]) endpoints and relations.

There are four types of intervals ([], [), (], ()), their negation](,)[,][,](, respectively), less than ([<], [<), (<], (<)), and greater than ([>], [>), (>], (>)) relations.

Note that some operators return identical results but are syntactically different: %[<]% and %[<)% both evaluate x < a; %[>]% and %(>]% both evaluate x > b; %(<]% and %(<)% evaluate x <= a; %[>)% and %(>)% both evaluate x >= b. This is so because we evaluate only one end of the interval but still conceptually referring to the relationship defined by the right-hand-side interval object and given that a <= b. This implies 2 conditional logical evaluations instead of treating it as a single 3-level ordered factor.

Value

A logical vector, indicating if x is in the specified interval. Values are TRUE, FALSE, or NA (when any of the 3 values (x or endpoints in interval) are NA).

The helper function intrval_types can be used to understand and visualize the operators' effects. It returns a matrix explaining the properties of the operators.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

See help page for relational operators: Comparison.

See %[o]% for relational operators for interval-to-interval comparisons.

See factor for the behavior with factor arguments. See also %in% for value matching and %ni% for negated value matching for factors.

See Syntax for operator precedence.

Examples

```
## motivating example from example(lm)

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
```

```
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
## compare 95% confidence intervals with 0
(CI.D9 <- confint(lm.D9))
0 %[]% CI.D9

## comparing dates

DATE <- as.Date(c("2000-01-01","2000-02-01", "2000-03-31"))
DATE %[<]% as.Date(c("2000-01-151", "2000-03-15"))
DATE %[]% as.Date(c("2000-01-151", "2000-03-15"))
DATE %[>]% as.Date(c("2000-01-151", "2000-03-15"))

## interval formats

x <- rep(4, 5)
a <- 1:5
b <- 3:7
cbind(x=x, a=a, b=b)
x %[]% cbind(a, b) # matrix
x %[]% data.frame(a=a, b=b) # data.frame
x %[]% list(a, b) # list

## helper functions

intrval_types() # print
intrval_types(plot = TRUE) # plot

## graphical examples

## bounding box
set.seed(1)
n <- 10^4
x <- runif(n, -2, 2)
y <- runif(n, -2, 2)
iv1 <- x %[]% c(-1, 1) & y %[]% c(-1, 1)
plot(x, y, pch = 19, cex = 0.25, col = iv1 + 1, main = "Bounding box")

## time series filtering
x <- seq(0, 4*24*60*60, 60*60)
dt <- as.POSIXct(x, origin="2000-01-01 00:00:00")
f <- as.POSIXlt(dt)$hour %[]% c(0, 11)
plot(sin(x) ~ dt, type="l", col="grey",
    main = "Filtering date/time objects")
points(sin(x) ~ dt, pch = 19, col = f + 1)

## watch precedence
(2 * 1:5) %[]% (c(2, 3) * 2)
2 * 1:5 %[]% (c(2, 3) * 2)
(2 * 1:5) %[]% c(2, 3) * 2
2 * 1:5 %[]% c(2, 3) * 2
```

`intrval-options` *Global options for the intrval package*

Description

Options store and allow to set global values for the intrval functions.

Usage

```
intrval_options(...)
```

Arguments

`...`	Options to set.

Value

When parameters are set by `intrval_options`, their former values are returned in an invisible named list. Such a list can be passed as an argument to `intrval_options` to restore the parameter values. Tags are the following:

- `use_fpCompare`: use the fpCompare package for the reliable comparison of floating point numbers.

Examples

```
str(intrval_options())

x1 <- 0.5 - 0.3
x2 <- 0.3 - 0.1

# save old values and set the new one
op <- intrval_options(use_fpCompare = FALSE)

# this is the base R behavior
x1
x2

# reset defaults
intrval_options(op)

# using fpCompare
x1
x2
```

ovrlap *Relational Operators Comparing Two Intervals*

Description

Functions for evaluating if two intervals overlap or not.

Usage

```
interval1 %[o]% interval2
interval1 %)o(% interval2
interval1 %[<o]% interval2
interval1 %[o>]% interval2

interval1 %(o)% interval2
interval1 %]o[% interval2
interval1 %(<o)% interval2
interval1 %(o>)% interval2

interval1 %[]o[]% interval2
interval1 %[]o[)% interval2
interval1 %[]o(]% interval2
interval1 %[]o()% interval2
interval1 %[)o[]% interval2
interval1 %[)o[)% interval2
interval1 %[)o(]% interval2
interval1 %[)o()% interval2
interval1 %(]o[]% interval2
interval1 %(]o[)% interval2
interval1 %(]o(]% interval2
interval1 %(]o()% interval2
interval1 %()o[]% interval2
interval1 %()o[)% interval2
interval1 %()o(]% interval2
interval1 %()o()% interval2
```

Arguments

`interval1`, `interval2`
: vector, 2-column matrix, list, or `NULL`: the interval end points of two (sets) of closed intervals to compare.

Details

The operators define the open/closed nature of the lower/upper limits of the intervals on the left and right hand side of the `o` in the middle.

The overlap of two closed intervals, [a1, b1] and [a2, b2], is evaluated by the `%[o]%` (alias for `%[]o[]%`) operator (a1 <= b1, a2 <= b2). Endpoints can be defined as a vector with two values (`c(a1, b1)`)or can be stored in matrix-like objects or a lists in which case comparisons are made element-wise. If lengths do not match, shorter objects are recycled. These value-to-interval operators work for numeric (integer, real) and ordered vectors, and object types which are measured at least on ordinal scale (e.g. dates), see Examples. Note: interval endpoints are sorted internally thus ensuring the conditions a1 <= b1 and a2 <= b2 is not necessary. `%)o(%` is used for the negation of two closed interval overlap, directional evaluation is done via the operators `%[<o]%` and `%[o>]%`.

The overlap of two open intervals is evaluated by the `%(o)%` (alias for `%()o()%`). `%]o[%` is used for the negation of two open interval overlap, directional evaluation is done via the operators `%(<o)%` and `%(o>)%`.

Overlap operators with mixed endpoint do not have negation and directional counterparts.

Value

A logical vector, indicating if `interval1` overlaps `interval2`. Values are `TRUE`, `FALSE`, or `NA`.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

See help page for relational operators: `Comparison`.

See `%[]%` for relational operators for value-to-interval comparisons.

See `factor` for the behavior with factor arguments. See also `%in%` for value matching and `%ni%` for negated value matching for factors.

See `Syntax` for operator precedence.

Examples

```
## motivating examples from example(lm)

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept
## compare 95% confidence of the 2 groups to each other
(CI.D90 <- confint(lm.D90))
CI.D90[1,] %[o]% CI.D90[2,]

## simple interval comparisons
c(2:3) %[o]% c(0:1)

## vectorized comparisons
c(2:3) %[o]% list(0:4, 1:5)
```

```
c(2:3) %[o]% cbind(0:4, 1:5)
c(2:3) %[o]% data.frame(a=0:4, b=1:5)
list(0:4, 1:5) %[o]% c(2:3)
cbind(0:4, 1:5) %[o]% c(2:3)
data.frame(a=0:4, b=1:5) %[o]% c(2:3)

list(0:4, 1:5) %[o]% cbind(rep(2,5), rep(3,5))
cbind(rep(2,5), rep(3,5)) %[o]% list(0:4, 1:5)

cbind(rep(3,5),rep(4,5)) %)o(% cbind(1:5, 2:6)
cbind(rep(3,5),rep(4,5)) %[<o]% cbind(1:5, 2:6)
cbind(rep(3,5),rep(4,5)) %[o>]% cbind(1:5, 2:6)

## open intervals

list(0:4, 1:5) %(o)% cbind(rep(2,5), rep(3,5))
cbind(rep(2,5), rep(3,5)) %(o)% list(0:4, 1:5)

cbind(rep(3,5),rep(4,5)) %]o[% cbind(1:5, 2:6)
cbind(rep(3,5),rep(4,5)) %(<o)% cbind(1:5, 2:6)
cbind(rep(3,5),rep(4,5)) %(o>)% cbind(1:5, 2:6)

dt1 <- as.Date(c("2000-01-01", "2000-03-15"))
dt2 <- as.Date(c("2000-03-15", "2000-06-07"))

dt1 %[]o[]% dt2
dt1 %[]o[)% dt2
dt1 %[]o(]% dt2
dt1 %[]o()% dt2
dt1 %[)o[]% dt2
dt1 %[)o[)% dt2
dt1 %[)o(]% dt2
dt1 %[)o()% dt2
dt1 %(]o[]% dt2
dt1 %(]o[)% dt2
dt1 %(]o(]% dt2
dt1 %(]o()% dt2
dt1 %()o[]% dt2
dt1 %()o[)% dt2
dt1 %()o(]% dt2
dt1 %()o()% dt2

## watch precedence
(2 * c(1, 3)) %[o]% (c(2, 4) * 2)
(2 * c(1, 3)) %[o]% c(2, 4) * 2
2 * c(1, 3) %[o]% (c(2, 4) * 2)
2 * c(1, 3) %[o]% c(2, 4) * 2
```

`%[c]%` *Dividing a Range Into 3 Intervals*

Description

Functions for evaluating if values of vectors are within intervals, or less than or higher than interval endpoints. The c within the brackets refer to cut, a similar function.

Usage

```
x %[c]% interval
x %[c)% interval
x %(c]% interval
x %(c)% interval
```

Arguments

x	vector or NULL: the values to be compared to interval endpoints.
interval	vector, 2-column matrix, list, or NULL: the interval end points.

Value

Values of x are compared to interval endpoints a and b (a <= b) (see %[]% for details). The functions return an integer vector taking values -1L (value of x is less than or equal to a, depending on the interval type), 0L (value of x is inside the interval), or 1L (value of x is greater than or equal to b, depending on the interval type).

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

Similar functions (but not quite): sign, cut, .bincode, findInterval.

See relational operators for intervals: %[]%.

See Syntax for operator precedence.

Examples

```
x <- 1:5
x %[c]% c(2,4)
x %[c)% c(2,4)
x %(c]% c(2,4)
x %(c)% c(2,4)
```

`%ni%` *Negated Value Matching*

Description

`%ni%` is the negation of `%in%`, which returns a logical vector indicating if there is a non-match or not for its left operand. `%nin%` and `%notin%` are aliases for better code readability (`%in%` can look very much like `%ni%`).

Usage

```
x %ni% table
x %nin% table
x %notin% table
```

Arguments

`x`	vector or `NULL`: the values to be matched.
`table`	vector or `NULL`: the values to be matched against.

Value

A logical vector, indicating if a non-match was located for each element of `x`: thus the values are `TRUE` or `FALSE` and never `NA`.

Author(s)

Peter Solymos <solymos@ualberta.ca>

See Also

All the opposite of what is written for `%in%`.

See relational operators for intervals: `%[]%`.

See `Syntax` for operator precedence.

Examples

```
1:10 %ni% c(1,3,5,9)
1:10 %nin% c(1,3,5,9)
1:10 %notin% c(1,3,5,9)

sstr <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")
sstr[sstr %ni% c(letters, LETTERS)]
```

Index