Package 'extraoperators'

October 13, 2022

Title Extra Binary Relational and Logical Operators

Version 0.1.1

Date 2019-11-01

URL http://joshuawiley.com/extraoperators,
https://github.com/JWiley/extraoperators

BugReports https://github.com/JWiley/extraoperators/issues

Description Speed up common tasks, particularly logical or
relational comparisons and routine follow up tasks such as finding the
indices and subsetting. Inspired by mathematics, where something like:
3 < x < 6 is a standard, elegant and clear way to assert that
x is both greater than 3 and less than 6
(see for example <https://en.wikipedia.org/wiki/Relational_operator>),
a chaining operator is implemented. The chaining operator, %c%,
allows multiple relational operations to be used in quotes on the right
hand side for the same object, on the left hand side.
The %e% operator allows something like set-builder notation
(see for example <https://en.wikipedia.org/wiki/Set-builder_notation>)
to be used on the right hand side.
All operators have built in prefixes defined for all, subset, and which
to reduce the amount of code needed for common tasks, such as return those
values that are true.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat (>= 2.1.0), covr, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Joshua F. Wiley [aut, cre] (<https://orcid.org/0000-0002-0271-6702>)

Maintainer Joshua F. Wiley <jwiley.psych@gmail.com>

Repository CRAN

Date/Publication 2019-11-04 17:30:02 UTC

R topics documented:

`logical all` *Several ways to evaluate whether all values meet logical conditions including logical range comparison helpers*

Description

Several ways to evaluate whether all values meet logical conditions including logical range comparison helpers

Usage

```
e1 %agele% e2

e1 %agel% e2

e1 %agle% e2

e1 %agl% e2

e1 %age% e2

e1 %ag% e2

e1 %ale% e2

e1 %al% e2

e1 %ain% e2

e1 %a!in% e2

e1 %anin% e2

e1 %a==% e2

e1 %a!=% e2
```

```
e1 %ac% e2

e1 %ae% e2
```

Arguments

e1	A number of vector to be evaluated
e2	A vector of one or two numbers used to denote the limits for logical comparison.

Value

A logical value whether all e1 meet the logical conditions.

Examples

```
1:5 %agele% c(2, 4)
1:5 %agele% c(4, 2) # order does not matter uses min / max

1:5 %agel% c(2, 4)
1:5 %agel% c(4, 2) # order does not matter uses min / max

1:5 %agle% c(2, 4)
1:5 %agle% c(4, 2) # order does not matter uses min / max

1:5 %agl% c(2, 4)
1:5 %agl% c(4, 2) # order does not matter uses min / max

1:5 %age% 2
1:5 %age% 4

1:5 %ag% 2
1:5 %ag% 4

1:5 %ale% 2
1:5 %ale% 4

1:5 %al% 2
1:5 %al% 4

1:5 %ain% c(2, 99)
c("jack", "jill", "john", "jane") %ain% c("jill", "jane", "bill")

1:5 %a!in% c(2, 99)
c("jack", "jill", "john", "jane") %a!in% c("jill", "jane", "bill")

1:5 %a==% 1:5
1:5 %a==% 5:1

1:5 %a!=% 1:5
1:5 %a!=% 5:1
1:5 %a!=% c(5, 4, 1, 3, 2)
```

```
## define a variable
sample_data <- c(1, 3, 9, 5, NA, -9)

## suppose that we expect that values should fall in [1, 10]
## unless they are special character, -9 used for unknown / refused
sample_data %ac% "( >= 1 & <= 10 ) | == -9"

## we might expect some missing values and be OK as long as
## above conditions are met or values are missing
sample_data %ac% "( >= 1 & <= 10 ) | == -9 | is.na"

## equally we might be expecting NO missing values
## and want missing values to come up as FALSE
sample_data %ac% "(( >= 1 & <= 10 ) | == -9) & !is.na"

## clean up
rm(sample_data)
## define a variable
sample_data <- c(1, 3, 9, 5, -9)

sample_data %ae% "(-8, 1] | [2, 9)"
sample_data %ae% "(-Inf, Inf)"

## clean up
rm(sample_data)
```

`logical indexes (which)`

Several ways to return an index based on logical range comparison helpers

---

Description

Several ways to return an index based on logical range comparison helpers

Usage

```
e1 %?gele% e2

e1 %?gel% e2

e1 %?gle% e2

e1 %?gl% e2

e1 %?ge% e2

e1 %?g% e2
```

```
e1 %?le% e2

e1 %?l% e2

e1 %?in% e2

e1 %?!in% e2

e1 %?nin% e2

e1 %?==% e2

e1 %?!=% e2

e1 %?c% e2

e1 %?e% e2
```

Arguments

e1	A number of vector to be evaluated and for which the indices will be returned
e2	A vector of one or two numbers used to denote the limits for logical comparison.

Value

A vector of the indices identifying which values of e1 meet the logical conditions.

Examples

```
1:5 %?gele% c(2, 4)
1:5 %?gele% c(4, 2) # order does not matter uses min / max

1:5 %?gel% c(2, 4)
1:5 %?gel% c(4, 2) # order does not matter uses min / max

1:5 %?gle% c(2, 4)
1:5 %?gle% c(4, 2) # order does not matter uses min / max

1:5 %?gl% c(2, 4)
1:5 %?gl% c(4, 2) # order does not matter uses min / max

1:5 %?ge% 2
1:5 %?ge% 4

1:5 %?g% 2
1:5 %?g% 4

1:5 %?le% 2
1:5 %?le% 4
```

```
1:5 %?l% 2
1:5 %?l% 4

1:5 %?in% c(2, 99)
c("jack", "jill", "john", "jane") %?in% c("jill", "jane", "bill")

1:5 %?!in% c(2, 99)
c("jack", "jill", "john", "jane") %?!in% c("jill", "jane", "bill")

1:5 %?nin% c(2, 99)
c("jack", "jill", "john", "jane") %snin% c("jill", "jane", "bill")

11:15 %?==% c(11, 1, 13, 15, 15)

11:15 %?!=% c(11, 1, 13, 15, 15)
## define a variable
sample_data <- c(1, 3, 9, 5, NA, -9)

## suppose that we expect that values should fall in [1, 10]
## unless they are special character, -9 used for unknown / refused
sample_data %?c% "( >= 1 & <= 10 ) | == -9"

## we might expect some missing values and be OK as long as
## above conditions are met or values are missing
sample_data %?c% "( >= 1 & <= 10 ) | == -9 | is.na"

## equally we might be expecting NO missing values
## and want missing values to come up as FALSE
sample_data %?c% "(( >= 1 & <= 10 ) | == -9) & !is.na"

## clean up
rm(sample_data)
## define a variable
sample_data <- c(1, 3, 9, 5, -9)

sample_data %?e% "(-8, 1] | [2, 9)"

## clean up
rm(sample_data)
```

`logicals` *Several logical range comparison helpers*

Description

Several logical range comparison helpers

Usage

```
e1 %gele% e2
```

```
e1 %gel% e2

e1 %gle% e2

e1 %gl% e2

e1 %g% e2

e1 %ge% e2

e1 %l% e2

e1 %le% e2

e1 %!in% e2

e1 %nin% e2

e1 %flipIn% e2
```

Arguments

e1	A number of vector to be evaluated
e2	A vector of one or two numbers used to denote the limits for logical comparison.

Value

A logical vector of the same length as e1.

Examples

```
1:5 %gele% c(2, 4)
1:5 %gele% c(4, 2) # order does not matter uses min / max

1:5 %gel% c(2, 4)
1:5 %gel% c(4, 2) # order does not matter uses min / max

1:5 %gle% c(2, 4)
1:5 %gle% c(4, 2) # order does not matter uses min / max

1:5 %gl% c(2, 4)
1:5 %gl% c(4, 2) # order does not matter uses min / max

1:5 %g% c(2)

1:5 %ge% c(2)

1:5 %l% c(2)
```

```
1:5 %le% c(2)

1:5 %!in% c(2, 99)
c("jack", "jill", "john", "jane") %!in% c("jill", "jane", "bill")
```

---

subsetting *Several ways to subset based on logical range comparison helpers*

Description

Several ways to subset based on logical range comparison helpers

Usage

```
e1 %sgele% e2

e1 %sgel% e2

e1 %sgle% e2

e1 %sgl% e2

e1 %sge% e2

e1 %sg% e2

e1 %sle% e2

e1 %sl% e2

e1 %sin% e2

e1 %s!in% e2

e1 %snin% e2

e1 %s==% e2

e1 %s!=% e2

e1 %sc% e2

e1 %se% e2
```

Arguments

e1	A number of vector to be evaluated and subset
e2	A vector of one or two numbers used to denote the limits for logical comparison.

Value

A subset of e1 that meets the logical conditions.

Examples

```
1:5 %sgele% c(2, 4)
1:5 %sgele% c(4, 2) # order does not matter uses min / max

1:5 %sgel% c(2, 4)
1:5 %sgel% c(4, 2) # order does not matter uses min / max

1:5 %sgle% c(2, 4)
1:5 %sgle% c(4, 2) # order does not matter uses min / max

1:5 %sgl% c(2, 4)
1:5 %sgl% c(4, 2) # order does not matter uses min / max

1:5 %sge% 2
1:5 %sge% 4

1:5 %sg% 2
1:5 %sg% 4

1:5 %sle% 2
1:5 %sle% 4

1:5 %sl% 2
1:5 %sl% 4

1:5 %sin% c(2, 99)
c("jack", "jill", "john", "jane") %sin% c("jill", "jane", "bill")

1:5 %s!in% c(2, 99)
c("jack", "jill", "john", "jane") %s!in% c("jill", "jane", "bill")

1:5 %s==% 1:5
1:5 %s==% c(1:4, 1)

1:5 %s!=% 1:5
1:5 %s!=% c(1:4, 1)
## define a variable
sample_data <- c(1, 3, 9, 5, NA, -9)

## suppose that we expect that values should fall in [1, 10]
## unless they are special character, -9 used for unknown / refused
sample_data %sc% "( >= 1 & <= 10 ) | == -9"

## we might expect some missing values and be OK as long as
## above conditions are met or values are missing
sample_data %sc% "( >= 1 & <= 10 ) | == -9 | is.na"
```

```
## equally we might be expecting NO missing values
## and want missing values to come up as FALSE
sample_data %sc% "(( >= 1 & <= 10 ) | == -9) & !is.na"

## clean up
rm(sample_data)
## define a variable
sample_data <- c(1, 3, 9, 5, -9)

sample_data %se% "(-8, 1] | [2, 9)"

## clean up
rm(sample_data)
```

`%c%` *Chain Operator*

Description

This operator allows operators on the right hand side to be chained together. The intended use case is when you have a single object on which you want to perform several operations. For example, testing whether a variable is between 1 and 5 or equals special number 9, which might be used to indicate that someone responded to a question (i.e., its not missing per se) but that they preferred not to answer or did not know the answer.

Usage

```
e1 %c% e2
```

Arguments

`e1`	The values to be operated on, on the left hand side
`e2`	A character string (it MUST be quoted) containing all the operators and their values to apply to ‘e1‘. Note that in this character string, operators can be chained together using either ‘|‘ or ‘&‘. Parentheses are also supported and work as expected. See examples for more information on how this function is used.

Details

‘ ‘is.na‘, ‘!is.na‘, ‘is.nan‘, and ‘!is.nan‘. These do not need any values supplied but they work as expected to add those logical assessments into the chain of operators.

Value

a logical vector

Examples

```
## define a variable
sample_data <- c(1, 3, 9, 5, NA, -9)

## suppose that we expect that values should fall in [1, 10]
## unless they are special character, -9 used for unknown / refused
sample_data %c% "( >= 1 & <= 10 ) | == -9"

## we might expect some missing values and be OK as long as
## above conditions are met or values are missing
sample_data %c% "( >= 1 & <= 10 ) | == -9 | is.na"

## equally we might be expecting NO missing values
## and want missing values to come up as FALSE
sample_data %c% "(( >= 1 & <= 10 ) | == -9) & !is.na"

c(1, 3, 9, 5, NA, -9) %c% "is.na & (( >= 1 & <= 10 ) | == -9)"

## clean up
rm(sample_data)
```

`%e%` *Element In Set Operator*

Description

This operator allows use of set notation style definitions

Usage

```
e1 %e% e2
```

Arguments

`e1`	The values to be operated on, on the left hand side
`e2`	A character string containing set notation style defined ranges on the real number line. Separate sets with the "&" or "|" operator for AND or OR.

Value

a logical vector

Examples

```
c(-1, 0, 1, 9, 10, 16, 17, 20) %e% "(-Inf, 0) | [1, 9) | [10, 16] | (17, Inf]"
table(mtcars$mpg %e% "(0, 15.5) | [22.8, 40)")
table(mtcars$mpg %e% "(0, 15) | [16, 18] | [30, 50)")
c(-1, 0, 1) %e% "(-Inf, Inf) & [0, 0] | [1, 1]"

z <- max(mtcars$mpg)
table(mtcars$mpg %e% "(-Inf, z)")

## clean up
rm(z)
```

Index