

Package ‘invctr’

October 13, 2022

Title Infix Functions For Vector Operations

Version 0.2.0

Maintainer Fred Hasselman <fred.hasselmann@ru.nl>

Description Vector operations between grapes: An infix-only package! The ‘invctr’ functions perform common and less common operations on vectors, data frames matrices and list objects:

- Extracting a value (range), or, finding the indices of a value (range).
- Trimming, or padding a vector with a value of your choice.
- Simple polynomial regression.
- Set and membership operations.
- General check & replace function for NAs, Inf and other values.

Imports rlang (>= 0.1.2), plyr

ByteCompile true

Encoding UTF-8

RoxygenNote 7.2.1

License GPL-3

Language en-US

Suggests knitr, spelling, testthat

VignetteBuilder knitr

URL <https://github.com/FredHasselmann/invctr>

BugReports <https://github.com/FredHasselmann/invctr/issues>

NeedsCompilation no

Author Fred Hasselman [aut, cre] (<<https://orcid.org/0000-0003-1384-8361>>)

Repository CRAN

Date/Publication 2022-08-16 13:50:06 UTC

R topics documented:

Counters	2
extractors	3

fINdExers	5
insiders	7
outsiders	8
padders	10
regressors	11
trimmers	12
%00%	13

Index	15
--------------	-----------

Counters	<i>Counters</i>
----------	-----------------

Description

Counters

Signed increment

Non-negative increment

Usage

```
counter %+-% increment
```

```
counter %++% increment
```

Arguments

counter If counter and increment are both (signed/positive) integers counter will change by the value of increment.

increment An integer value $\neq 0$ to add to counter

Examples

```
## Not run:
# Signed increment
# Notice the difference between passing an object and a value for counter

# Value
(10 %+-% -5)
(10 %+-% -5)

# Object
i <- 10
(i %+-% -5)
(i %+-% -5)

# This means we can use the infix in a while ... statement
# WARNING: As is the case for any while ... statement, be careful not to create an infinite loop!
```

```

i <- 10
while(i > -5){
  i %+-% -5
  print(i)
}

# Non-negative increment
# Notice the difference between passing an object and a value for counter

# Value
(0 %+++ 5)
(0 %+++ 5)

# Object
i <- 0
(i %+++ 5)
(i %+++ 5)

# This means we can use the infix in a while ... statement
# WARNING: As is the case for any while ... statement, be careful not to create an infinite loop!

i <- 0
while(i < 20){
  i %+++ 5
  print(i)
}

## End(Not run)

```

extractors

Extract vectors by index or value occurrence

Description

Extract front or rear of vector 'x' up and until an index 'i', the first or last occurrence of a value 'v', or, extract values based on quantile 'q', first, middle, or, last index 'j'.

Usage

```
x %[f% v
```

```
x %[l% v
```

```
x [% i
```

```
x %] i
```

```

x %f]% v
x %l]% v
x %q]% q
x %q)% q
x %[q% q
x %(q% q
x %:% j

```

Arguments

x	A vector
v	A value of which the first or last occurrence in x will be used as an index
i	An index or two element vector c(lo, hi) indicating a range to extract
q	A percentile value (between '0' and '1')
j	A character indicating to extract the first 'f', middle 'm' or last 'l' value of 'x'.

Value

A vector extracted from the front, rear, or, range of 'x'. Either based on an index or the first or last occurrence of a value or the first, middle, or, ;ast element of a vector.

Note

The function provided for symmetry, character lengths of x%]i and x[1:i] are equal.

Examples

```

z <- letters

# Extract front by first occurrence of value
z %[f% "n"

# Extract front by index
x <- rnorm(100)
x [% 10

# Extract rear by index
x %] 90

# Extract rear by index
x %] 90

# Extract by indices if a range is provided

```

```

x %] c(4,30)
z [% c(6,10)

# Extract last/middle value of x
x %:% "l"
z %:% "m"

# Extract by percentile
seq(1,10,.5) %(q% .5 # infix
seq(1,10,.5)[seq(1,10,.5) < quantile(seq(1,10,.5),.5)] # regular syntax

seq(1,10,.5) %q] .5 # infix
seq(1,10,.5)[seq(1,10,.5) >= quantile(seq(1,10,.5),.5)] # regular syntax

```

fINDEXers

Find row or column by name or index

Description

Find row or column by name or index
 Column by name or index
 Row by name or number
 Matrix cell index by name or number
 Return all indices of a (range of) values
 Is element of... with multiple input types

Usage

```

c %ci% d

r %ri% d

rc %mi% d

nv %ai% d

x %e% y

```

Arguments

c	Column name or index
d	A named vector, list, matrix, or data frame
r	Row name or index

<code>rc</code>	A 2-element numeric or character vector representing $c(r, c)$. Names (character) and indices (numeric) vectors can be mixed if <code>rc</code> is passed as a 2-element list object.
<code>nv</code>	A numeric value, or vector of values of which you want to know the indices in <code>d</code> .
<code>x</code>	A vector, data frame or list containing numbers and/or characters that could be elements of <code>y</code>
<code>y</code>	An object that could contain values in <code>x</code>

Value

If `r/c/rc` is numeric, the name corresponding to the row/column index of `d`, if `r/c/rc` is a character vector, the row/column index corresponding to the row/column name. If `dimnames(d) == NULL`, but `names(d) != NULL` then `%ci%` and `%ri%` will look up `r/c` in `names(d)`

Logical vector indicating which `x` are an element of `y`

Author(s)

Fred Hasselman

Examples

```
# data frame
d <- data.frame(x=1:5,y=6,row.names=paste0("ri",5:1))

"y" %ci% d # y is the 2nd column of d
  2 %ci% d # the name of the second column of d is "y"

  2 %ri% d
"ri5" %ri% d

# change column name
colnames(d)["y" %ci% d] <- "Yhat"

# mi works on data frames, matrices, tibbles, etc.
c(5,2) %mi% d
list(r="ri1",c=2) %mi% d

# matrix row and column indices
m <- matrix(1:10,ncol=2, dimnames = list(paste0("ri",0:4),c("xx","yy")))

  1 %ci% m
  5 %ci% m # no column 5

  1 %ri% m
  5 %ri% m

c(5,1)%mi%m
c(1,5)%mi%m
```

```

# For list and vector objects ri and ci return the same values
l <- list(a=1:100,b=LETTERS)

  2 %ci% l
"a" %ci% l

  2 %ri% l
"a" %ri% l

# named vector
v <- c("first" = 1, "2nd" = 1000)

"2nd" %ci% v
  1 %ci% v

"2nd" %ri% v
  1 %ri% v

# get all indices of the number 1 in v
1 %ai% v

# get all indices of the number 3 and 6 in d
c(3,6) %ai% d

# get all indices of values: Z < -1.96 and Z > 1.96
Z <- rnorm(100)
Z[Z%](%c(-1.96,1.96)] %ai% Z

```

insiders

Inside interval

Description

Decide if a value x falls inside an interval $j[1], j[2]$ that can be open or closed on the left and/or the right. Either a logical vector equal to x , or the actual values are extracted, when the ‘.’-versions are used.

Usage

```

x %[]% j

x %()% j

x %[]% j

x %[]% j

```

```
x %[]% j
x %()% j
x %[]% j
x %().% j
```

Arguments

x	A vector
j	A 2-element numeric vector indicating a range

Value

Logical vector of length x, or, values in the range j

Note

Package ‘DescTools’ provides similar functions

Examples

```
# Closed interval
0:5 %[]% c(1,5) # logical vector
0:5 %[]% c(1,5) # extract values

# Open interval
0:5 %()% c(1,5)
0:5 %().% c(1,5)

# Closed interval left
0:5 %[]% c(1,5)
0:5 %[]% c(1,5)

# Closed interval right
0:5 %[]% c(1,5)
0:5 %().% c(1,5)
```

outsiders

Outside interval

Description

Decide if a value x falls outside an interval $j[1], j[2]$ that can be open or closed on the left and/or the right. Either a logical vector equal to x, or the actual values are extracted,

Usage

```
x %][% j
x %)(% j
x %](% j
x %)][% j
x %].[% j
x %).( % j
x %].( % j
x %).[ % j
```

Arguments

x	A vector
j	A range

Value

logical vector of length x, or, values of x outside the range j

Note

Package ‘DescTools‘ provides similar functions

Examples

```
# Closed interval
5%][%c(1,5)
5%].[%c(1,5)

# Open interval
5%)(%c(1,5)
5%).( %c(1,5)

# Half-losed interval left
5%](%c(1,5)
5%].( %c(1,5)

# Half-losed interval right
5%][%c(1,5)
5%].[ %c(1,5)
```

padders

*Padd vector by index***Description**

Padd vector by index

Pad vector front

Pad vector rear

Pad vector front + rear

Usage

x %[+% j

x %+]% j

x %[+]% j

Arguments

x A vector

j A one, or two element vector. One element: Pad front or rear by j 0s, or, front by floor(j/2) and rear by ceiling(j/2). Two elements: Pad j[1] times the value passed in j[2].

Value

A padded version of x

Examples

```
x <- rnorm(100)

# Pad front with 10 zeros
x%[%10
# Same as
x%[%c(10,0)

# Pad rear with zeros
x%+]10
# Same as
x%+]c(10,0)

# Pad front + rear with NA
x%[%c(NA,10)
```

```
# Pad front + rear of a character vector
"yes"%[+]c(2,"no")
"yes"%[+]c(1,"no")
"yes"%[+]c(0,"no")
```

regressors

Regress vectors

Description

Regress vectors

Correlate x and y

Polynomial regression of degree 1

Polynomial regression of degree 2

Polynomial regression of degree 3

Polynomial regression of degree 4

,

Usage

x %/r% y

x %/1% y

x %/2% y

x %/3% y

x %/4% y

x %/n% yn

Arguments

x Numeric vectors

y Numeric vector

yn List of length 2, first element is a vector y, the second element an integer denoting the order of the polynomial regression.

Examples

```

x <- rnorm(100)
y <- x + x^2 + x^3

# Correlate x with y
x%/r%y

# Polynomial regression degree 1 .. 4
x%/1%y
x%/2%y
x%/3%y
x%/4%y

anova(x%/1%y, x%/2%y, x%/3%y, x%/4%y)

# Order n
x%/n%list(y,10)

```

trimmers

Trim vector by index

Description

Trim vector by index
Trim vector front
Trim vector rear
Trim vector front + rear

Usage

```

x %[-% i
x %-]% i
x %[-]% j

```

Arguments

x	A vector
i	A 1 element vector by which the rear of x will be trimmed
j	A one, or two element numeric vector. One element: Trim front by $\text{floor}(i/2)$ and rear by $\text{ceiling}(i/2)$. Two elements: Trim $i[1]$ from the front and $i[2]$ from the rear.

Value

A trimmed version of x

Examples

```
x <- rnorm(100)

# Trim front
x%[-%5

# Trim rear
x%[-]%5

# Trim front + rear
x%[-]%c(2,10)

x%[-]%7
```

%00%

Rose tinted infix

Description

When your functions wear these rose tinted glasses, the world will appear to be a nicer, fluffier place.

Usage

```
x %00% y
```

Arguments

- x If (an element of) x is any of Inf, -Inf, NA, NaN, NULL, length(x)==0, it will return/replace the value of y; otherwise x.
- y The value to return/replace for x in case of catastrophe >00<

Author(s)

Fred Hasselman

See Also

purrrr::

Examples

```
Inf %00% NA
```

```
numeric(0) %00% ''
```

```
NA %00% 0
```

```
NaN %00% NA
```

```
c(1, NaN) %00% NA
```

```
NULL %00% NA
```

```
c(1, NULL) %00% NA # can't see second element
```

Index

%) .[% (outsiders), 8
%)[% (outsiders), 8
%+-% (Counters), 2
%+-% (Counters), 2
%+)% (padders), 10
%-)% (trimmers), 12
%/1% (regressors), 11
%/2% (regressors), 11
%/3% (regressors), 11
%/4% (regressors), 11
%/n% (regressors), 11
%/r% (regressors), 11
%:% (extractors), 3
%[% (insiders), 7
%[+% (padders), 10
%[+)% (padders), 10
%[-)% (trimmers), 12
%[-)% (trimmers), 12
%[.)% (insiders), 7
%[.]% (insiders), 7
%[% (extractors), 3
%[% (insiders), 7
%[f% (extractors), 3
%[l% (extractors), 3
%[q% (extractors), 3
%].[% (outsiders), 8
%][% (outsiders), 8
%]% (extractors), 3
%ai% (fINDEXers), 5
%ci% (fINDEXers), 5
%e% (fINDEXers), 5
%f)% (extractors), 3
%l)% (extractors), 3
%mi% (fINDEXers), 5
%q)% (extractors), 3
%q)% (extractors), 3
%ri% (fINDEXers), 5
%00%, 13

Counters, 2

extractors, 3

fINDEXers, 5

insiders, 7

outsiders, 8

padders, 10

regressors, 11

trimmers, 12