Package: invctr (via r-universe)

October 22, 2024

Title Infix Functions For Vector Operations

Version 0.2.0

Maintainer Fred Hasselman <fred.hasselman@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

LazyData true

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

License GPL-3

Language en-US

Suggests knitr, spelling, testthat

VignetteBuilder knitr

URL https://github.com/FredHasselman/invctr

BugReports https://github.com/FredHasselman/invctr/issues

Repository https://fredhasselman.r-universe.dev

RemoteUrl https://github.com/fredhasselman/invctr

RemoteRef HEAD

RemoteSha 5f9c962dce35b7303c54a8cbc0919b4b4af888ea

Contents

`Counters` *Counters*

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 vectors by index or value occurrence

Extract front by first occurrence of a value `%[f%`

Extract front by last occurrence of a value `%[l%`

Extract vector front by index `%[%`

Extract vector rear by index `%]%`

Extract first occurrence of a value to vector rear `%f]%`

Extract last occurrence of a value to vector rear `%l]%`

Extract values at percentile and higher `%q]%`

Extract values higher than percentile `%q)%`

Extract values at percentile and smaller `%[q%`

Extract values smaller than percentile `%(q%`

Extract first, middle or last value of vector `%:%`

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, last 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: `%ci%`

Row by name or number: `%ri%`

Matrix cell index by name or number: `%mi%`

Return all indices of a (range of) values: `%ai%`

Is element of... with multiple input types: `%e%`

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

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, tiblles, 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 *Values inside interval*

Description

Values inside interval

In closed interval: `%[]%`

In open interval: `%()%`

In half-closed interval (left): `%[)%`

In half-closed interval (right): `%(]%`

Return x in closed interval: `%[.]%`

Return x in open interval: `%(.)%`

Return x in half-closed interval (left): `%[.)%`

Return x in half-closed interval (right): `%(.]%`

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` *Values outside interval*

Description

Values outside interval

Not in closed interval: `%][%`

Not in open interval: `%)(%`

Not in half-closed interval (left): `%](%`

Not in half-closed interval (right): `%)[%`

Return x not in closed interval: `%].[%`

Return x not in open interval: `%).(%`

Return x not in half-closed interval (left): `%].(%`

Return x not in half-closed interval (right): `%).[%`

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` `0`s, 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: `%/r%`

Polynomial regression of degree 1: `%/1%`

Polynomial regression of degree 2: `%/2%`

Polynomial regression of degree 3: `%/3%`

Polynomial regression of degree 4: `%/4%`
`%/n%` Polynomial regression of degree n: `%/n%`

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 `floor(i/2)` and rear by `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