

# Package ‘operators’

January 2, 2012

**Type** Package

**Title** Additional binary operators for the R language

**Version** 0.1-6

**Date** 2010-04-11

**Author** Romain Francois <romain@r-enthusiasts.com>

**Maintainer** Romain Francois <romain@r-enthusiasts.com>

**Suggests** RUnit

**Description** A set of operators for common tasks such as regex manipulation.

**License** GPL-3

**URL** <http://romainfrancois.blog.free.fr>

**Repository** CRAN

**Date/Publication** 2010-04-11 12:57:52

## R topics documented:

operators-package . . . . .	2
but . . . . .	2
character decorator . . . . .	3
files . . . . .	4
notIn . . . . .	5
pattern . . . . .	6
patternDivision . . . . .	8
patternFilter . . . . .	9
patternSubstitution . . . . .	10
pipe . . . . .	12
plusEqual . . . . .	13
withOptions . . . . .	14
%o\textasciitilde% . . . . .	15
%of% . . . . .	16
%without% . . . . .	17

**Index****18**

---

operators-package      *Additional binary operators*

---

**Description**

Additional binary operators for R

**Details**

Package: operators  
Type: Package  
Version: 0.1-6  
Date: 2010-04-11  
License: GPL-3

**Author(s)**

Romain Francois <romain@r-enthusiasts.com>

Maintainer: Romain Francois <romain@r-enthusiasts.com>

---

but      *Modification of function arguments*

---

**Description**

Modifies the arguments of a function

**Usage**

```
fun %but% x
```

**Arguments**

fun      Function to modify

x      Modifier

**Value**

A function with the same body as the fun argument but with a different list of arguments.

**Note**

The %but% operator is S3-generic with the following methods:

- A default method which does nothing more than returning the fun function.
- A character method. In that case, x describes the logical arguments of the function. x is a single character string containing one or several token of the form ab where b is the first letter of the logical argument we wish to modify and a is an optional modifier. a can be empty or +, in which case the argument will be set to TRUE; - in which case the argument will be set to FALSE; or ! in which case the argument will be the opposite of the current value in fun
- A list. In that case, arguments that are part of the formal arguments of fun and elements of the list x are updated to the element in x

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[args](#), [formals](#)

**Examples**

```
### default method, nothing is done
rnorm %but% 44

### character method, operating on logical arguments
grep %but% "pf"      # grep, with perl and fixed set to TRUE
grep %but% "i-e"    # grep, ignoring the case but not using extended regular expressions
( grep %but% "vp" )( "blue", colors() )

### list method
rnorm %but% list( mean = 3 )
rnorm %but% list( nonsense = 4 )
```

---

character decorator      *Creates string decorators by repeating a pattern*

---

**Description**

Creates string decorators by repeating a pattern either a given number of times or so that it takes a given number of character

**Usage**

```
txt %x=% n
txt %x=|% length.out
strrep( txt, n, length.out=getOption("width") )
```

**Arguments**

txt	Pattern to repeat
n	Number of times to repeat the pattern
length.out	number of character the output should be

**Value**

A character string

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[paste](#), [sprintf](#), [rep](#)

**Examples**

```
"=" %x=% 80
"<+>" %x=| % 80
strrep( ".-", n = 10 )
strrep( ".-", length.out = 50 )
strrep( ".-" )
```

---

files

*Read or write an R object to/from a file*

---

**Description**

A set of functions to quickly redirect output to a file or read character vectors from a file.

**Usage**

```
object %>% file
object %>>% file
```

```
object %2>% file
object %2>>% file
```

```
object %*>% file
object %*>>% file
```

```
object %<% file
object %<<% file
```

**Arguments**

object	R object to print to the file or to read from the file
file	file in which to read or write

**Details**

%>% sends the object to the file. The object is printed to the file according to the function specified in the operators.print option supplied with this package, most likely to be the [print](#) function. See examples.

%>>% appends the output to the file.

%2>% sends the message stream to the file by [sinking](#) the message stream to the file. See [sink](#) for details. %2>>% appends the message stream to the file.

%\*>% sends both output and message streams to the file. %\*>>% appends them.

%<% reads the content of the file into the object. %<<% appends the content of the file to the object.

**Value**

NULL, used for the side effects.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[file](#)

**Examples**

```
## Not run:
rnorm(30) %>% "test.txt"
stop("problem") %2>>% "test.txt"
x %<% "test.txt"
x

## End(Not run)
```

---

notIn

*Not in*

---

**Description**

Negation of the %in% operator.

**Usage**

```
x %!in% table
```

**Arguments**

x	The values to be matched
table	The values to <i>not</i> be matched against

**Value**

Logical vector, negation of the %in% operators on the same arguments.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
1:10 %!in% c(1,3,5,9)
```

---

pattern

*Pattern matching operators*

---

**Description**

Set of convenience functions to handle strings and pattern matching. These are basically companion binary operators for the classic R function [grep](#) and [regexpr](#).

**Usage**

```
x %~% rx
x %!~% rx

x %~*% rx
x %!~*% rx

x %~+% rx
x %!~+% rx
```

**Arguments**

x	text to manipulate
rx	regular expression

**Value**

`%~%` : gives a logical vector indicating which elements of `x` match the regular expression `rx`. `%!~%` is the negation of `%~%`

`%~*%` : gives a *single* logical indicating if *all* the elements of `x` are matching the regular expression `rx`. `%!~*%` is the negation of `%~*%`.

`%~+%` : gives a *single* logical indicating if *any* element of `x` matches the regular expression `rx`. `%!~+%` is the negation of `%~+%`.

**Note**

The matching is done using a modified version of the `regexpr` function. The modification is performed by applying the operators `.regexpr` option to the `regexpr` function via the `%but%` operator.

The default version of `regexpr` enables the `perl` and extended options. See `%but%` for details.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[grep](#), [gsub](#), [%textasciitilde|%](#) for regular expression filters

**Examples**

```
txt <- c("arm", "foot", "lefroo", "bafooobar")
txt %~% "foo"
txt %!~% "foo"
txt %~*% "foo"
txt %~+% "foo"
txt %!~*% "foo"
txt %!~+% "foo"

txt %~% "[a-z]"
txt %!~% "[a-z]"
txt %~*% "[a-z]"
txt %~+% "[a-z]"
txt %!~*% "[a-z]"
txt %!~+% "[a-z]"

cols <- colors()
cols[ cols %~% "^blue" ]
# see also %~|%

## needs perl regular expression for the \\d, see %but%
with( options( operators.regexpr = "p" ), {
  cols[ cols %!~% "\\d$" ]
} )
```

---

patternDivision	<i>Divide by a pattern</i>
-----------------	----------------------------

---

## Description

split a character vector by a regular expression

## Usage

```
txt %/~% rx
```

## Arguments

txt	text to manipulate
rx	regular expression

## Value

A character vector. For convenience, this function does not return a list as [strsplit](#) does.

## Note

`%/~%` uses [strsplit](#) to split the strings. Logical arguments of [strsplit](#) can be indirectly modified using the operators `.strsplit` option declared as part of this package. For example, it uses perl regular expressions by default. See [%but%](#) for a description.

## Author(s)

Romain Francois <francoisromain@free.fr>

## See Also

[grep](#), [gsub](#)

## Examples

```
"Separate these words by spaces" %/~% " +"  
  
### From ?strsplit  
unlist(strsplit("a.b.c", "\\."))  
"a.b.c" %/~% "\\."
```

---

patternFilter                      *Regular expression filters*

---

## Description

Filters a character vector by a regular expression.

## Usage

```
x %~|% rx
x %!~|% rx
```

## Arguments

x	text to manipulate
rx	regular expression

## Value

'%~|%' : a character vector containing all the elements of x that match the regular expression rx or NULL if there is no match.

'%!~|%' : a character vector containing all the elements of x that *do not* match the regular expression rx.

## Note

The filtering is done using the `regexr` function. Logical arguments of `regexr` can be indirectly used by `%~|%` or `%!~|%` by using the operators `.regexr` option declared with this package. See [%but%](#) for a description of this mechanism.

## Author(s)

Romain Francois <francoisromain@free.fr>

## See Also

[grep](#), [gsub](#)

## Examples

```
cols <- colors()
cols %~|% "^blue"

### blue colors that don't finish with a digit
( a1 <- cols %~|% "blue" %!~|% "\\d$" )
```

```

( a2 <- cols %~|% "blue[^0-9]*$" )
( a3 <- grep( "blue[^0-9]*", cols, value = TRUE ) )

# using perl regular expressions

### not necessary since p is in the default of the package
with( options( operators.regexpr = "p" ), {
  ( a4 <- grep( "blue[^\\d]*", cols, value = TRUE, perl = TRUE ) )
  ( a5 <- cols %~|% "blue[^\\d]*$" )
})

### blue colors that contain a r
cols %~|% "blue" %~|% "r"
grep( "r", grep( "blue", cols, value = TRUE ), value = TRUE )

### blue colors that don't contain a r
cols %~|% "blue" %!~|% "r"
cols %~|% "^[^r]*blue[^r]*$"

grep( "^[^r]*$", grep( "blue", cols, value = TRUE ), value = TRUE ) # tricky and verbose
# or in two steps, ... laborious
bluecols <- grep( "blue", cols, value = TRUE )
bluecols[ -grep( "r", bluecols) ]

```

---

patternSubstitution     *Remove a pattern from a character vector*

---

### Description

Removes a pattern from a character vector.

### Usage

```

txt %~|% pattern
txt %~| pattern
txt %o~| pattern

```

### Arguments

txt	text to manipulate
pattern	regular expression

### Value

%~%: Removes the pattern rx from the character vector x. It is equivalent of using gsub( rx, "", x ).

`%~|%` does a two-step operation. First, it selects the elements of `x` that match the pattern `rx` and then it removes the pattern from the rest.

`%o~|%` does a slightly more complicated two-step operation. It first gets the elements of `txt` that match the pattern and then keeps only the part that matches the pattern. Similar to the `grep -o` in recent versions of unix.

### Note

`%~%` does the substitution via the `gsub` function. One can pass arguments to the `gsub` function using the operators `.gsub` option declared by this package. See `%but%` for a description of this mechanism.

The filtering in `%~|%` is performed by `%\textasciitilde|%` and therefore options can be passed to `regexpr` using the operators `.regexpr` option.

For `%o~|%`, if the pattern given does not contain opening and closing round brackets, the entire matching space is retained, otherwise only the part that is contained between the brackets is retained, see the example below.

`%s~%` is an attempt to provide some of the functionality of the unix's `sed`. The pattern is split by `/` and used as follows: the first part is the regular expression to replace, the second is the replacement, and the (optional) third gives modifiers to the `gsub` function used to perform the replacement. Modifiers are passed to `gsub` with the `%but%` operator. The `"g"` modifier can also be used in order to control if the `gsub` function is used for global replacement or the `sub` function to only replace the first match. *At the moment `/` cannot be used in the regular expressions.*

### Author(s)

Romain Francois <francoisromain@free.fr>

### See Also

[grep](#), [gsub](#)

### Examples

```
txt <- c("arm", "foot", "lefroo", "bafoobar")
txt %~% "foo"
txt %o~|% "foo"

### Email of the R core team members
rcore <- readLines(file.path(R.home("doc"), "AUTHORS"))
rcore

### or this way
# angle brackets are retained here
rcore %o~|% "<.*@.*>"
rcore %o~|% "<.*@.*>" %~% "[<>]"

# allows to perform the match using < and > but strips them from the result
```

```

rcore %o~|% "<(.@.*>"

# really silly english to french translator
pinks <- colors() %~|% "pink"
pinks %s~% "/pink/rose/"
gsub( "pink", "rose", pinks )

# perl regex pink shouter
pinks %s~% "/(pink)/\\U\\1/p"
gsub( "(pink)", "\\U\\1", pinks, perl = TRUE )

# see ?gsub
gsub("(\\w)(\\w*)", "\\U\\1\\L\\2", "a test of capitalizing", perl=TRUE)
"a test of capitalizing" %s~% "/(\\w)(\\w*)/\\U\\1\\L\\2/gp"

```

---

pipe

*Pipe an R object to a unix command*

---

## Description

The operator `prints` the R object into a temporary file and then executes the unix command through a `pipe`

## Usage

```
r %|% u
```

## Arguments

r	Any R object
u	character string representing the unix command

## Value

An object of S3-class `unixoutput`. The print method for `unixoutput` objects simply `cat` the string.

## Author(s)

Romain Francois <francoisromain@free.fr>

## See Also

[pipe](#)

**Examples**

```
## Not run:
rnorm(30) %|% 'head -n2'
rnorm(30) %|% 'sed "s/^ *\\[[0-9]*\\]/g" '

if( require(R4X ) ){
  x <- xml( '<root>
    <child id="1">
      <subchild id = "sub1" >foo</subchild>
      <subchild id = "sub2" >bar</subchild>
    </child>
    <child id="2">
      <subchild id="a">blah</subchild>
      <subchild id="b">bob</subchild>
      <something id="c" />
    </child>
    <fruits>
      <fruit>banana</fruit>
      <fruit>mango</fruit>
    </fruits>
  </root>' )
  x %|% "xml_pp | highlight -S xml -A"
}

## End(Not run)
```

---

plusEqual

*Plus Equal Operators*


---

**Description**

Plus equal operator

**Usage**

```
object %+=% value
```

**Arguments**

object	object to which to add something
value	object to add

**Value**

NULL. Used for the side effect of changing the value of object

**Note**

The operator %+=% is S3-generic with a single default method implemented at the moment.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
### standard examples
x <- 4
x %+=% 4
x

### XML examples with the R4X package
## Not run:
require("R4X")
x <- xmlNode( "test" )
x %+=% '<foo><bar/></foo>'
x

## End(Not run)
```

---

withOptions

*Alternative option mechanism*

---

**Description**

options is a slight rework on [options](#) that gives a S3 class options to the result. This allows the definition of a with method for the options. This is useful to execute a block of code *with* a set of options.

**Usage**

```
## S3 method for class 'options'
with(data, expr, ...)
options(...)
```

**Arguments**

...	Options to use. See <a href="#">options</a> for details.
data	Options to use. This is typically a call to the options function
expr	Code to execute.

## Details

The result of the expression that is evaluated is modified in order to keep the option context it is associated with. The class of the object created by the expression is expanded to include the `withOptions` class and the `withOptions` attribute that keeps the context in which the object has been created.

This mechanism has been implemented specially for the automatic printing of objects that happens outside the call to the `with.options` function and not reflect the options requested by the user when the object is printed.

## Value

For the function `with.options`, the result of the expression given in `expr` is returned. See details below.

## Author(s)

Romain Francois <francoisromain@free.fr>

## See Also

The original [options](#) function in the base package.

## Examples

```
# part of ?glm
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family=poisson())

summary( glm.D93 )

with( options(show.signif.stars = FALSE,show.coef.Pvalues=FALSE),
summary( glm.D93) )

a <- try(
  with( options( warn = 2) , warning( "more than a warning" ) ),
  silent = TRUE )
class( a )
```

---

%o\textasciitilde%

*Only keeps the matching part of a regular expression*

---

## Description

The operator `%o~%` is used to retain the only the part of the `txt` that matches the regular expression.

**Usage**

```
txt %o~% pattern
```

**Arguments**

txt	Character vector
pattern	Regular expression

**Value**

In case where parts of the regular expression are surrounded by brackets, the operator returns a matrix with as many lines as the length of txt and as many columns as chunks of regular expressions.

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
x <- c("foobar", "barfoooooooooooooobar")
x %o~% "fo+"
```

---

%of%	<i>Is an object of a given class</i>
------	--------------------------------------

---

**Description**

Operator to check if an object is of a given class

**Usage**

```
x %of% y
```

**Arguments**

x	R object
y	Character string, the class to check against.

**Value**

Logical value indicating the result

**Author(s)**

Romain Francois <francoisromain@free.fr>

**See Also**

[inherits](#)

**Examples**

```
iris %of% "data.frame"
```

---

%without%

*Remove certain elements from a vector*

---

**Description**

Remove the elements in table from s

**Usage**

```
x %without% table
```

**Arguments**

x	Vector
table	Elements to remove from x

**Value**

x without the elements of table

**Author(s)**

Romain Francois <francoisromain@free.fr>

**Examples**

```
letters %without% "a"
```

# Index

- \*Topic **character**
  - %without%, 17
- \*Topic **file**
  - files, 4
  - pipe, 12
- \*Topic **manip**
  - %\textasciitilde%, 15
  - %of%, 16
  - character decorator, 3
  - pipe, 12
- \*Topic **package**
  - operators-package, 2
- \*Topic **programming**
  - withOptions, 14
- \*Topic **utilities**
  - but, 2
  - files, 4
  - notIn, 5
  - pattern, 6
  - patternDivision, 8
  - patternFilter, 9
  - patternSubstitution, 10
  - plusEqual, 13
- %!~\*% (pattern), 6
- %!~+% (pattern), 6
- %!~% (pattern), 6
- %!in% (notIn), 5
- %.>>% (files), 4
- %.>% (files), 4
- %.+=% (plusEqual), 13
- %.~% (patternSubstitution), 10
- %/~% (patternDivision), 8
- %.2>>% (files), 4
- %.2>% (files), 4
- %.<\*>% (files), 4
- %.<2% (files), 4
- %.<<\*>% (files), 4
- %.<<2% (files), 4
- %.<<% (files), 4
- %.<% (files), 4
- %.>>% (files), 4
- %.>% (files), 4
- %.~\*% (pattern), 6
- %.~+% (pattern), 6
- %.~% (pattern), 6
- %.but% (but), 2
- %.s~% (patternSubstitution), 10
- %.x=% (character decorator), 3
- %.but%, 7–9, 11
- %.~\textasciitilde%, 15
- %.of%, 16
- %.without%, 17
- args, 3
- but, 2
- cat, 12
- character decorator, 3
- file, 5
- files, 4
- formals, 3
- grep, 6–9, 11
- gsub, 7–9, 11
- inherits, 17
- notIn, 5
- operators (operators-package), 2
- operators-package, 2
- options, 14, 15
- options (withOptions), 14
- paste, 4
- pattern, 6
- patternDivision, 8
- patternFilter, 9

patternSubstitution, [10](#)  
pipe, [12](#), [12](#)  
plusEqual, [13](#)  
print, [5](#), [12](#)  
  
regexpr, [6](#), [7](#)  
rep, [4](#)  
  
sink, [5](#)  
sprintf, [4](#)  
strrep (character decorator), [3](#)  
strsplit, [8](#)  
  
with.options (withOptions), [14](#)  
withOptions, [14](#)