Package 'functional'

June 24, 2026

Type Package

Encoding UTF-8

Title Curry, Compose, and Other Higher-Order Functions

Version 0.7

Date 2026-06-23

Description Provides a small collection of higher-order functions for functional-style programming in R, including 'Curry' (partial application), 'CurryL' (lazy partial application), 'Compose' (function composition, including a multi-argument variant), 'Identity', 'Negate' and 'Swap'.

License GPL (>= 2)

URL https://github.com/klutometis/R-functional

BugReports https://github.com/klutometis/R-functional/issues

LazyLoad yes

Imports lisp (>= 0.2)

NeedsCompilation no

Author Peter Danenberg [aut, cre]

Maintainer Peter Danenberg <pcd@roxygen.org>

Repository CRAN

Date/Publication 2026-06-24 07:00:02 UTC

Contents

`Compose` *Compose an arbitrary number of functions.*

Description

My Happy Hacking keyboard gave out during the writing of this procedure; moment of silence, please.

Usage

```
Compose(...)
```

Arguments

`...`	the functions to be composed

Details

Had to make this slightly more complex to handle the nullary case; also included a fast-path for the trivial case.

Value

A composed function

Examples

```
car <- function(list) list[[1]]
cdr <- function(list) list[2:length(list)]
cadr <- Compose(cdr, car)
stopifnot(cadr(c(1,2,3)) == 2)
```

`Curry` *Pre-specify a procedures named parameters, returning a new procedure.*

Description

Thanks, Byron Ellis; and Aaron McDaid modified it to preserve names across invocation.

Usage

```
Curry(FUN, ...)
```

Arguments

FUN	the function to be curried
...	the determining parameters

Details

https://stat.ethz.ch/pipermail/r-devel/2007-November/047318.html

Value

A new function partially determined

Examples

```
double <- Curry(`*`, e1=2)
stopifnot(double(4) == 8)
```

CurryL *Lazy curry; thanks, Jamie! <https://github.com/klutometis/R-functional/issues/1>*

Description

Lazy curry; thanks, Jamie! <https://github.com/klutometis/R-functional/issues/1>

Usage

```
CurryL(FUN, ...)
```

Arguments

FUN	the function to be curried
...	the determining parameters

Value

A new function that, when called, lazily evaluates FUN with the pre-specified and subsequent arguments.

Examples

```
# day is not defined; thanks, Jamie Folson.
CurryL(function(...) match.call(),
       x=5,
       y=as.Date(day))(z=as.Date(day,"%Y"))
```

Identity *Identity function.*

Description

Is concatenation benign?

Usage

```
Identity(...)
```

Arguments

...	tautological arguments

Value

The tautologized arguments, concatenated

Examples

```
list.copy <- function(list)
  Reduce(Identity, list)

list <- c(1, 2, 3)
stopifnot(list.copy(list) == list)
```

multi.argument.Compose *Composition with multiple arguments.*

Description

Thanks, Alexander Davis!

Usage

```
multi.argument.Compose(...)
```

Arguments

...	the functions to be composed

Value

A composed function

Examples

```
f <- function(x, y) x+y
g <- function(x) x*2
stopifnot(multi.argument.Compose(f, g)(1,1) == 4)
```

`Negate` *Negate a function; borrowed from src/library/base/R/funprog.R for pre-2.7 Rs.*

Description

Negate a function; borrowed from src/library/base/R/funprog.R for pre-2.7 Rs.

Usage

```
Negate(f)
```

Arguments

`f`	the function to be negated

Value

The negated function

Examples

```
is.even <- function(a) a%%2 == 0
is.odd <- Negate(is.even)
stopifnot(Reduce(`&&`, Map(is.odd, c(1, 3, 5))))
```

`Swap` *Thanks, Gabor; see <http://stackoverflow.com/a/23726989>: swaps the first two arguments in a function.*

Description

Thanks, Gabor; see <http://stackoverflow.com/a/23726989>: swaps the first two arguments in a function.

Usage

```
Swap(f)
```

Arguments

`f`	The function whose arguments to swap

Value

A function with swapped arguments

Index