Package 'fcp'

May 8, 2026

Title Function Composition

Version 0.1.0

Date 2023-11-27

Description A function composition operator to chain a series of calls into a single function, mimicking the math notion of (f o g o h)(x) = h(g(f(x))). Inspired by 'pipeOp' ('|>') since R4.1 and 'magrittr pipe' ('%>%'), the operator build a pipe without putting data through, which is best for anonymous function accepted by utilities such as apply() and lapply().

Depends R (>= 3.5.0)

License GPL (>= 2)

URL https://github.com/xiaoran831213/R_fun_comp

Encoding UTF-8

Author Xiaoran Tong [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4648-3330>)

Maintainer Xiaoran Tong <xiaoran.tong.cn@gmail.com>

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-12-05 17:20:02 UTC

Contents

fcp 2

Index **4**

`fcp` *Function Composition*

Description

Given calls `f(., a=A)` and `g(., b=B)`, compose `function(.) g(f(., a=A), b=B)`.

Usage

```
fcp(f, g) # or f %.% g

f %.% g
```

Arguments

`f`	left–operand: inner call to giver a result first.
`g`	right-operand: outer call to receirve that result.

Details

By default, `fcp` is asigned to operator `%.%` so a final function can be constructed from a chain of more than two calls, just like a chain of |> without input or a chain of %>% with `.` as the special input.

`fcp` is best used to quickly create anonymous functions from existing ones as input of functional programming tools like base::apply, base::sapply, and base::lapply, etc. Should the composed function be saved for reuse, its body shall retain the original syntax of component function calls as closely as possible.

Like %>%, `fcp` allows a downstream call to use `.` to refer to input from its immediate upstream call, and pipe the input into any slot in the downstream call's list of arguments while transforming the same input through expressions of `.`; when no transformation is needed, `fcp` allows unchanged input be piped into all empty slots in the argument list (e.g., fun(x=1, y=, z=2) for named slots, or fun(1, , z=2) for positinal slots).

Like %>%, when no explicit reference to the output of the uptream call is used, the results is then piped into the first available positional argument of the downstream call, and a call with a lone input can be shorterned to function names without parentheses.

Later calls in a `fcp` chain can refer to earlier calls by numerical symbols, so `n` refers to the result of nth call in the chain and`0` refers to initial input, allowing more flexablity than a strice "upstream to downstream" pipe.

Value

composite function g(f(x, ...), ...)

Examples

```
## ex1: match and extract date (pipe the initial input at differnt stage).
x <- c("2023-01-01", "2022/12/31", "2002-07-02")
p <- "^([0-9]{2,4})[-/]([0-9]{1,2})[-/]([0-9]{1,2})$"
## reference usage: input x appeared twice
(a0 <- do.call(rbind, regmatches(x, regexec(p, x))))
## composed function 1: blank as input from upstream, `0` as initial input.
(f1 <- regexec(p,  ) %.% regmatches(`0`,  ) %.% do.call(rbind,  ))
(a1 <- f1(x))
## composed function 2: dot refer to upstream, `0` as initial input.
(f2 <- regexec(p, .) %.% regmatches(`0`, .) %.% do.call(rbind, .))
(a2 <- f2(x))
## composed function 3: use of named argument, `0` as initial input.
(f3 <- regexec(pa=p) %.% regmatches(x =`0`) %.% do.call(wh=rbind))
(a3 <- f3(x))

## ex2: date given days since 2000-010-1 (shorthand form of function calls)
(g1 <- as.Date(origin="2000-01-01") %.% f2()) # reused composed `f2`
(b1 <- g1(364))
(g2 <- as.Date(origin="2000-01-01") %.% f2)   # omit ()
(b2 <- g2(364))

## ex3: wrap, pad, and upcase the strings (`sapply` uses composed function)
words <- c("Hellow World!", "Good Morning!")
s0 <- sapply(words, function(x)
{
    toupper(paste(format(strwrap(x, 8), w=12, just="c"), collapse="\n"))
})
cat(s0, sep="\n---- s0 ----\n")
s1 <- sapply(words, strwrap(8) %.% format(w=12, just="c") %.%
                    paste(collapse="\n") %.% toupper)
cat(s1, sep="\n---- s1 ----\n")

## check equivalance
all(a1==a0) && all(a2==a0) && all(a3==a0) && all(b1==b2) && all(s1==s0)
```

Index

%.%*(*fcp*)*, 2
%>%, *2*

base::apply, *2*
base::lapply, *2*
base::sapply, *2*

fcp, 2