Package 'dub'

May 8, 2026

Type Package

Title Unpacking Assignment for Lists via Pattern Matching

Version 0.2.0

Description Provides an operator for assigning nested components of a list to names via a concise pattern matching syntax. This is especially convenient for assigning individual names to the multiple values that a function may return in the form of a list, and for extracting deeply nested list components.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

Suggests covr, testthat

URL https://github.com/egnha/dub

BugReports https://github.com/egnha/dub/issues

Collate 'util.R' 'assert.R' 'tree.R' 'names.R' 'assign.R'

RoxygenNote 6.1.0

NeedsCompilation no

Author Eugene Ha [aut, cre]

Maintainer Eugene Ha <eha@posteo.de>

Repository CRAN

Date/Publication 2018-10-27 12:10:02 UTC

Contents

`%<<-%` *Assign nested components of a list to names*

Description

The `%<<-%` operator assigns multiple (nested) components of a list or vector to names via pattern matching ("unpacking assignment"). Think of the "dub(ble) arrow" `<<-` as a pictograph representing multiple `<-`'s.

`%<<-%` is especially convenient for:

- assigning individual names to the multiple values that a function may return in the form of a list;
- extracting deeply nested list components.

Usage

```
pattern %<<-% value

value %->>% pattern
```

Arguments

`pattern`	Pattern of names that the components of `value` are assigned to (see below).
`value`	List or vector.

Value

Returns `value` invisibly.

Pattern-matching names

Names are matched to the (nested) components of a list using a concise pattern matching syntax that mirrors the structure of the list. Apart from names, the syntax consists of two classes of symbols:

- **List constructors** — Use a pair of parentheses to indicate a list, and a colon, rather than a comma, to indicate successive names.
- **Wildcards** — Use a dot (`.`) to skip assignment of a specific component, or dots (`...`) to skip assignment of a range of components.

See the examples for an illustration of common use cases.

Prior art

Unpacking/multiple assignment appears in other languages (e.g., Python, JavaScript, Clojure). While R has no such feature, using a custom operator to do this has long been a folklore method. An early implementation is due to Gabor Grothendieck (2004), cf. `list` in the gsubfn package.

Examples

```
# Assign successive components
(one : two : three) %<<-% list(1, 2, 3)
stopifnot(one == 1, two == 2, three == 3)

# Assign nested components
(p : (q : r : (s : t))) %<<-% list(1, list(2, 3, list(4, 5)))
(P : (Q : R : S)) %<<-% list(1, list(2, 3, list(4, 5)))
stopifnot(p == 1, q == 2, r == 3, s == 4, t == 5,
          P == 1, Q == 2, R == 3, identical(S, list(4, 5)))

# Unpack nested components with nested parentheses
(w) %<<-% list(1:3)
(((z))) %<<-% list(list(list("z")))
((x : y)) %<<-% list(list("x", "y"))
stopifnot(w == 1:3, x == "x", y == "y", z == "z")

# Skip a component with a dot (.)
(a : . : b) %<<-% list("a", "skip this", "b")
((c : .) : .) %<<-% list(list("c", "skip this"), "skip this")
stopifnot(a == "a", b == "b", c == "c")

# Skip a range of components with dots (...)
(first : ... : last) %<<-% letters
(. : second : ...) %<<-% letters
(mpg : cyl : ...) %<<-% mtcars
stopifnot(
  first == "a", second == "b", last == "z",
  mpg == mtcars$mpg, cyl == mtcars$cyl
)
```

Index