Package ‘adverbial’

May 13, 2025

Type Package

Title Enhanced Adverbial Functions

Version 0.2.0

Description Provides new_partialised() and new_composed(), which extend partial() and compose() functions of 'purrr' to make it easier to extract and replace arguments and functions. It also has additional adverbial functions.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, pillar, purrr, rlang, vctrs

RoxygenNote 7.3.2

URL `https://github.com/UchidaMizuki/adverbial`

BugReports `https://github.com/UchidaMizuki/adverbial/issues`

Suggests lifecycle, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Mizuki Uchida [aut, cre]

Maintainer Mizuki Uchida <`uchidamizuki@vivaldi.net`>

Repository CRAN

Date/Publication 2025-05-13 08:30:06 UTC

Contents

`as_step` *Wrap a function to be used as a step*

Description

[Experimental]

Usage

```
as_step(f, name = NULL)
```

Arguments

`f`	A function to be wrapped.
`name`	The name of the step. If `NULL`, the step does not proceed but the function is applied.

Details

`as_step()` wraps a function to be used as a step in a step-by-step process.

Value

A function that takes a step-by-step object and additional arguments, and returns the updated step-by-step object.

`end_step` *End a step-by-step process*

Description

[Experimental]

Usage

```
end_step(object)
```

Arguments

`object`	The object to end the step-by-step process for.

Details

`end_step()` ends the step-by-step process and removes the step-by-step attributes from the object.

Value

The object with the step-by-step attributes removed.

`new_composed` *Create composed functions*

Description

Create composed functions

Usage

```
new_composed(fns, dir = NULL, ..., class = character())
```

Arguments

`fns`	A list of functions to compose.
`dir`	Direction of composition, either `"forward"` or `"backward"`. By default, the functions are composed in the forward direction. Passed to `purrr::compose()`.
`...`	Additional arguments for attributes.
`class`	Name of subclass.

Value

A composed function that inherits from `adverbial_function_compose`.

See Also

`purrr::compose()`

Examples

```
square <- function(x) x ^ 2
cdist <- new_composed(list(square = square, sum = sum, sqrt = sqrt))
cdist(1:10)

cdist$sum <- new_partialised(sum, list(na.rm = TRUE))
cdist(c(1:10, NA))
```

new_partialised *Create partialised functions*

Description

Create partialised functions

Usage

```
new_partialised(f, args, ..., class = character())
```

Arguments

`f`	A function.
`args`	A list of default arguments.
`...`	Additional arguments for attributes.
`class`	Name of subclass.

Value

A `adverbial_function_partial` function.

See Also

`purrr::partial()`

Examples

```
dist <- function(x, y) {
  sqrt(x ^ 2 + y ^ 2)
}
pdist <- new_partialised(dist, list(x = 3))
pdist(y = 4)
```

step_by_step *Create a step-by-step object*

Description

[Experimental]

Usage

```
step_by_step(steps)
```

Arguments

`steps`	A named vector of steps to be completed. The names of the vector are the names of the steps, and the values are the descriptions of the steps.

Details

`step_by_step()` creates a step-by-step object that can be used to track the progress of a process. It is useful for long-running processes where you want to keep track of the steps that have been completed and the steps that are still to be done.

Value

A function that takes an object and returns a step-by-step object.

Index