Package 'maybe'

May 8, 2026

Title The Maybe Monad

Version 1.1.0

Description The maybe type represents the possibility of some value or nothing.
It is often used instead of throwing an error or returning `NULL`.
The advantage of using a maybe type over `NULL` is that it is both composable
and requires the developer to explicitly acknowledge the potential absence
of a value, helping to avoid the existence of unexpected behaviour.

License MIT + file LICENSE

URL https://github.com/armcn/maybe, https://armcn.github.io/maybe/

BugReports https://github.com/armcn/maybe/issues

Encoding UTF-8

RoxygenNote 7.2.2

Imports magrittr

Suggests testthat (>= 3.0.0), quickcheck, covr

Config/testthat/edition 3

NeedsCompilation no

Author Andrew McNeil [aut, cre]

Maintainer Andrew McNeil <andrew.richard.mcneil@gmail.com>

Repository CRAN

Date/Publication 2023-08-07 13:30:02 UTC

Contents

---

and *Combine predicate functions to check if all are TRUE*

Description

Combine predicate functions to check if all are TRUE

Usage

```
and(...)
```

Arguments

...	Predicate functions

Value

A predicate function

Examples

```
and(not_null, not_na)(1)
and(not_null, not_na)(NULL)
```

and_then *Evaluate a maybe returning function on a maybe value*

Description

Evaluate a maybe returning function on a maybe value

Usage

```
and_then(.m, .f, ...)

bind(.m, .f, ...)
```

Arguments

.m	A maybe value
.f	A maybe returning function to apply to the maybe value
...	Named arguments for the function .f

Value

A maybe value

Examples

```
safe_sqrt <- maybe(sqrt, ensure = not_infinite)

just(9) %>% and_then(safe_sqrt)
just(-1) %>% and_then(safe_sqrt)
nothing() %>% and_then(safe_sqrt)
```

and_then2 *Evaluate a binary maybe returning function on two maybe values*

Description

Evaluate a binary maybe returning function on two maybe values

Usage

```
and_then2(.m1, .m2, .f, ...)
```

Arguments

.m1	A maybe value
.m2	A maybe value
.f	A binary maybe returning function to apply to the maybe values
...	Named arguments for the function .f

Value

A maybe value

Examples

```
and_then2(just(1), just(2), maybe(`+`))
and_then2(nothing(), just(2), maybe(`/`))
```

---

and_then3 *Evaluate a ternary maybe returning function on three maybe values*

Description

Evaluate a ternary maybe returning function on three maybe values

Usage

```
and_then3(.m1, .m2, .m3, .f, ...)
```

Arguments

.m1	A maybe value
.m2	A maybe value
.m3	A maybe value
.f	A ternary maybe returning function to apply to the maybe values
...	Named arguments for the function .f

Value

A maybe value

Examples

```
safe_sum <- maybe(function(x, y, z) sum(x, y, z))

and_then3(just(1), just(2), just(3), safe_sum)
and_then3(nothing(), just(2), just(3), safe_sum)
```

filter_justs *Filter and unwrap a list of 'Just' values*

Description

Filter and unwrap a list of 'Just' values

Usage

```
filter_justs(.l)
```

Arguments

.l	List of maybe values

Value

A list of values

Examples

```
filter_justs(list(just(1), nothing(), just("a")))
```

filter_map *Map a function over a list and filter only 'Just' values*

Description

Map a function over a list and filter only 'Just' values

Usage

```
filter_map(.l, .f, ...)
```

Arguments

.l	List of values
.f	A maybe returning function to apply to the maybe values
...	Named arguments for the function .f

Value

A list of values

Examples

```
filter_map(list(-1, "2", 9), maybe(sqrt))
```

from_just *Unwrap a 'Just' value or throw an error*

Description

Unwrap a 'Just' value or throw an error

Usage

```
from_just(.m)
```

Arguments

.m	A maybe value

Value

The unwrapped 'Just' value

Examples

```
just(1) %>% from_just()
```

is_just *Check if an object is a 'Just' value*

Description

Check if an object is a 'Just' value

Usage

```
is_just(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
is_just(1)
is_just(just(1))
is_just(nothing())
```

is_maybe *Check if an object is a maybe value*

Description

Check if an object is a maybe value

Usage

```
is_maybe(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
is_maybe(1)
is_maybe(just(1))
is_maybe(nothing())
```

is_nothing *Check if an object is a 'Nothing' value*

Description

Check if an object is a 'Nothing' value

Usage

```
is_nothing(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
is_nothing(1)
is_nothing(just(1))
is_nothing(nothing())
```

just *Create a 'Just' variant of a maybe value*

Description

Create a 'Just' variant of a maybe value

Usage

```
just(a)
```

Arguments

a	A value to wrap in a 'Just' container

Value

A 'Just' variant of a maybe value

Examples

```
just(1)
just("hello")
```

maybe *Modify a function to return a maybe value*

Description

Wrapping a function in `maybe` will modify it to return a maybe value. If the function would normally return an error or warning the modified function will return a 'Nothing' value, otherwise it will return a 'Just' value. If a predicate function is provided with the parameter `ensure`, if the predicate returns `TRUE` when evaluated on the return value of the function, then a 'Just' value will be returned by the modified function, otherwise it will return a 'Nothing' value.

Usage

```
maybe(.f, ensure = function(a) TRUE, allow_warning = FALSE)
```

Arguments

.f	A function to modify
ensure	A predicate function
allow_warning	Whether warnings should result in 'Nothing' values

Value

A function which returns maybe values

Examples

```
maybe(mean)(1:10)
maybe(mean, allow_warning = TRUE)("hello")
maybe(sqrt)("hello")
maybe(sqrt, ensure = not_infinite)(-1)
```

`maybe_case` *Unwrap and call a function on a maybe value or return a default*

Description

Unwrap and call a function on a maybe value or return a default

Usage

```
maybe_case(.m, .f, default)
```

Arguments

`.m`	A maybe value
`.f`	A function to apply to the maybe value in the case of 'Just'
`default`	A default value to return in the case of 'Nothing'

Value

The return value of the 'Just' function or the default value

Examples

```
just(1:10) %>% maybe_case(mean, 0)
nothing() %>% maybe_case(mean, 0)
```

maybe_contains *Check if a maybe value contains a specific value*

Description

If the maybe value is a 'Nothing' variant FALSE will be returned. If it is a 'Just' variant the contents will be unwrapped and compared to the value argument using base::identical.

Usage

```
maybe_contains(.m, value)
```

Arguments

.m	A maybe value
value	A value to check

Value

TRUE or FALSE

Examples

```
just(1) %>% maybe_contains(1)
just("a") %>% maybe_contains(1)
nothing() %>% maybe_contains(1)
```

maybe_equal *Check if two maybe values are equal*

Description

If both values are 'Nothing' variants or both values are 'Just' variants with identical contents TRUE will be returned, otherwise FALSE.

Usage

```
maybe_equal(.m1, .m2)
```

Arguments

.m1	A maybe value
.m2	A maybe value

Value

TRUE or FALSE

Examples

```
maybe_equal(just(1), just(1))
maybe_equal(just(1), just(2))
maybe_equal(nothing(), nothing())
```

---

maybe_flatten *Flatten a nested maybe value*

Description

Flatten a nested maybe value

Usage

```
maybe_flatten(.m)

join(.m)
```

Arguments

.m	A maybe value

Value

A maybe value

Examples

```
just(just(1)) %>% maybe_flatten()
just(nothing()) %>% maybe_flatten()
just(1) %>% maybe_flatten()
nothing() %>% maybe_flatten()
```

maybe_map *Evaluate a function on a maybe value*

Description

Evaluate a function on a maybe value

Usage

```
maybe_map(.m, .f, ...)

fmap(.m, .f, ...)
```

Arguments

.m	A maybe value
.f	A function to apply to the maybe value
...	Named arguments for the function .f

Value

A maybe value

Examples

```
just(9) %>% maybe_map(sqrt)
nothing() %>% maybe_map(sqrt)
```

maybe_map2 *Evaluate a binary function on two maybe values*

Description

Evaluate a binary function on two maybe values

Usage

```
maybe_map2(.m1, .m2, .f, ...)
```

Arguments

.m1	A maybe value
.m2	A maybe value
.f	A binary function to apply to the maybe values
...	Named arguments for the function .f

Value

A maybe value

Examples

```
maybe_map2(just(1), just(2), `+`)
maybe_map2(nothing(), just(2), `/`)
```

---

maybe_map3 *Evaluate a ternary function on three maybe values*

Description

Evaluate a ternary function on three maybe values

Usage

```
maybe_map3(.m1, .m2, .m3, .f, ...)
```

Arguments

`.m1`	A maybe value
`.m2`	A maybe value
`.m3`	A maybe value
`.f`	A ternary function to apply to the maybe values
`...`	Named arguments for the function `.f`

Value

A maybe value

Examples

```
maybe_map3(just(1), just(2), just(3), function(x, y, z) x + y + z)
maybe_map3(nothing(), just(2), just(3), function(x, y, z) x / y * z)
```

nothing *Create a 'Nothing' variant of a maybe value*

Description

Create a 'Nothing' variant of a maybe value

Usage

```
nothing()
```

Value

A 'Nothing' variant of a maybe value

Examples

```
nothing()
```

not_empty *Check if a vector or data frame is empty*

Description

Check if a vector or data frame is empty

Usage

```
not_empty(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_empty(integer())
not_empty(list())
not_empty(1:10)
not_empty(data.frame())
not_empty(data.frame(a = 1:10))
```

not_infinite *Check if an object is not infinite*

Description

Check if an object is not infinite

Usage

```
not_infinite(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_infinite(Inf)
not_infinite(1)
```

not_na *Check if an object is not NA*

Description

Check if an object is not NA

Usage

```
not_na(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_na(NA)
not_na(1)
```

not_nan *Check if an object is not NaN*

Description

Check if an object is not NaN

Usage

```
not_nan(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_nan(NaN)
not_nan(1)
```

not_null *Check if an object is not NULL*

Description

Check if an object is not NULL

Usage

```
not_null(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_null(NULL)
not_null(1)
```

not_undefined *Check if an object is not undefined*

Description

In this case 'undefined' values include NULL, NaN, all NA variants, and infinite values.

Usage

```
not_undefined(a)
```

Arguments

a	Object to check

Value

TRUE or FALSE

Examples

```
not_undefined(NA)
not_undefined(NULL)
not_undefined(1)
```

or *Combine predicate functions to check if any are TRUE*

Description

Combine predicate functions to check if any are TRUE

Usage

```
or(...)
```

Arguments

...	Predicate functions

Value

A predicate function

Examples

```
or(not_null, not_na)(1)
or(not_null, not_na)(NULL)
```

perhaps *Modify a function to return the value or a default value*

Description

Wrapping a function in perhaps will modify it to return the expected value or a default value in some circumstances. If the function would normally return an error or warning the modified function will return a default value, otherwise it will return the expected value. If a predicate function is provided with the parameter ensure, if the predicate returns TRUE when evaluated on the return value of the function, then the expected value will be returned by the modified function, otherwise it will return the default value.

Usage

```
perhaps(.f, default, ensure = function(a) TRUE, allow_warning = FALSE)
```

Arguments

.f	A function to modify
default	A default value
ensure	A predicate function
allow_warning	Whether warnings should result in the default value

Value

A function which returns the expected value or the default value

Examples

```
perhaps(mean, default = 0)(1:10)
perhaps(mean, default = 0, allow_warning = TRUE)("hello")
perhaps(sqrt, default = 0)("hello")
perhaps(sqrt, default = 0, ensure = not_infinite)(-1)
```

with_default *Unwrap a maybe value or return a default*

Description

Unwrap a maybe value or return a default

Usage

```
with_default(.m, default)

from_maybe(.m, default)
```

Arguments

.m	A maybe value
default	A default value to return if the maybe value is 'Nothing'

Value

The unwrapped maybe value or the default value

Examples

```
just(1) %>% with_default(default = 0)
nothing() %>% with_default(default = 0)
```

Index