

Package ‘result’

November 21, 2023

Title Result Type for Safely Handling Operations that can Succeed or Fail

Version 0.1.0

Description Allows wrapping values in `success()` and `failure()` types to capture the result of operations, along with any status codes. Risky expressions can be wrapped in `as_result()` and functions wrapped in `result()` to catch errors and assign the relevant result types. Monadic functions can be bound together as pipelines or transaction scripts using `then_try()`, to gracefully handle errors at any step.

Encoding UTF-8

RoxygenNote 7.2.3

License MIT + file LICENSE

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/soumyaray/result>

BugReports <https://github.com/soumyaray/result/issues>

NeedsCompilation no

Author Soumya Ray [aut, cre, cph] (<<https://orcid.org/0000-0002-7497-3281>>)

Maintainer Soumya Ray <soumya.ray@gmail.com>

Repository CRAN

Date/Publication 2023-11-21 18:40:02 UTC

R topics documented:

<code>as_result</code>	2
<code>bind</code>	3
<code>failure</code>	4
<code>is_failure</code>	4
<code>is_result</code>	5
<code>is_success</code>	5

result	6
status	7
success	7
value	8

Index	9
--------------	----------

as_result	<i>Wraps an expression in result type, choosing between success and failure based on the outcome of the expression.</i>
-----------	-------------------------------------------------------------------------------------------------------------------------

Description

Use `as_result` on expressions whose outcomes are not known in advance or not safe to be examined. The expression will be evaluated immediately and wrapped in `success` if it produces a value or `failure` if it produces an error. If the expression produces a warning, it will be wrapped in `success` or `failure` depending on the `fail_on_warning` argument.

Usage

```
as_result(.expr, detect_warning = TRUE, fail_on_warning = TRUE)
```

Arguments

`.expr` expression to evaluate

`detect_warning` logical, whether to detect warnings; note `as_result()` cannot capture the outcome of an expression if it catches warnings, so use `detect_warning = TRUE` only if you want to capture the warning message (e.g., after a side-effect).

`fail_on_warning` logical, whether to treat warnings as failure or success.

Value

result object of subclass `success` or `failure`

Examples

```
as_result(42)
as_result(1 + 1)

stopper <- as_result(stop("This is my error message"))
is_failure(stopper)
value(stopper)

as_result(warning("You've been warned")) |> is_success()
as_result(warning("You've been warned"), fail_on_warning = FALSE) |> value()
```

bind	<i>Binds a result with another result or function to return a result</i>
------	--------------------------------------------------------------------------

Description

If the second object is a function, its return value will be wrapped in a result object of subclass success or failure depending on whether the function produces an error or warning. Bind is aliased with [then_try](#).

If the second object is a function, its return value will be wrapped in a result object of subclass success or failure depending on whether the function produces an error or warning. [then_try](#) is aliased with [bind](#).

Usage

```
bind(last_result, next_obj, ...)
```

```
then_try(last_result, next_obj, ...)
```

Arguments

last_result	result object of subclass success or failure
next_obj	result monad or plain function to bind with
...	additional arguments to pass to next_obj

Value

result object of subclass success or failure
result object of subclass success or failure

See Also

[then_try](#)

[bind](#)

Examples

```
times3 <- function(x, succeeds = TRUE) {  
  if (succeeds) success(x * 3)  
  else failure("func1 failed")  
}  
  
success(5) |> bind(times3) |> value()  
success(5) |> bind(times3, succeeds = FALSE) |> is_failure()  
failure("failed from the start") |> bind(times3) |> is_failure()  
failure("failed from the start") |> bind(times3) |> value()  
times3 <- function(x, succeeds = TRUE) {  
  if (succeeds) success(x * 3)
```

```

    else failure("func1 failed")
  }

success(5) |> then_try(times3) |> value()
success(5) |> then_try(times3, succeeds = FALSE) |> is_failure()
failure("failed from the start") |> then_try(times3) |> is_failure()
failure("failed from the start") |> then_try(times3) |> value()

```

failure	<i>Wraps a value in failure type of result</i>
---------	------------------------------------------------

Description

Wraps a value in failure type of result

Usage

```
failure(value = "failed", status = "error")
```

Arguments

value	any object to wrap
status	character string of the result (typically short)

Value

result object of subclass failure

Examples

```
failure()
failure(42)
```

is_failure	<i>Checks if an object is of failure class</i>
------------	------------------------------------------------

Description

Checks if an object is of failure class

Usage

```
is_failure(obj)
```

Arguments

obj	object to check
-----	-----------------

Value

TRUE if object is of failure class, FALSE otherwise

Examples

```
is_failure(success())  
is_failure(failure())
```

is_result *Checks if an object is of result class*

Description

Checks if an object is of result class

Usage

```
is_result(obj)
```

Arguments

obj object to check

Value

TRUE if object is of result class, FALSE otherwise

Examples

```
is_result(success())  
is_result(failure())  
is_result(42)
```

is_success *Checks if an object is of success class*

Description

Checks if an object is of success class

Usage

```
is_success(obj)
```

Arguments

obj object to check

Value

TRUE if object is of success class, FALSE otherwise

Examples

```
is_success(success())
is_success(failure())
```

result

Wraps a function in an result monad for later evaluation.

Description

Use result on functions whose outcomes are not known in advance or not safe to be examined. The function will not be evaluated until the monad is explicitly called.

Usage

```
result(.fn, detect_warning = TRUE, fail_on_warning = TRUE)
```

Arguments

`.fn` function to wrap

`detect_warning` logical, whether to detect warnings; note result cannot capture the outcome value if it catches warnings, so use `detect_warning = TRUE` only if you want to capture the warning message (e.g., after a side-effect).

`fail_on_warning` logical, whether to treat warnings as failure or success.

Value

function that returns a result object of subclass success or failure

Examples

```
crashy <- function() stop("Go no further")
safely_call_crashy <- result(crashy)
safely_call_crashy() |> is_failure()

calculate <- function(x, y) x + y
safely_calculate <- result(calculate)
safely_calculate(1, 2) |> value()
```

status	<i>Extracts the status of a result</i>
--------	----------------------------------------

Description

Extracts the status of a result

Usage

```
status(obj)
```

Arguments

obj	result object
-----	---------------

Value

status of the result

Examples

```
status(success("datafile.md", status = "created"))
```

success	<i>Wraps a value in success type of result</i>
---------	------------------------------------------------

Description

success is a constructor function for result class.

Usage

```
success(value = "done", status = "ok")
```

Arguments

value	any object to wrap
status	character string of the result (typically short)

Value

result object of subclass success

Examples

```
success()  
success(42)
```

value	<i>Extracts the value of a result</i>
-------	---------------------------------------

Description

Extracts the value of a result

Usage

```
value(obj)
```

Arguments

obj	result object
-----	---------------

Value

value object wrapped by result

Examples

```
value(success(42))
```


Index

`as_result`, 2

`bind`, 3, 3

`failure`, 4

`is_failure`, 4

`is_result`, 5

`is_success`, 5

`result`, 6

`status`, 7

`success`, 7

`then_try`, 3

`then_try(bind)`, 3

`value`, 8