

# Package ‘attempt’

April 20, 2018

**Title** Easy Condition Handling

**Version** 0.2.1

**Description** A friendlier condition handler, inspired by 'purrr' mappers and based on 'rlang'. 'attempt' extends and facilitates condition handling by providing a consistent grammar, and provides a set of easy to use functions for common tests and conditions. 'attempt' only depends on 'rlang', and focuses on speed, so it can be easily integrated in other functions and used in data analysis.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**Imports** rlang

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Colin Fay [aut, cre]

**Maintainer** Colin Fay <contact@colinfay.me>

**Repository** CRAN

**Date/Publication** 2018-04-20 11:43:03 UTC

## R topics documented:

attempt-package . . . . .	2
attempt . . . . .	2
if_all . . . . .	3
if_then . . . . .	3
silently . . . . .	4
silent_attempt . . . . .	5
stop_if . . . . .	5
surely . . . . .	7
try_catch . . . . .	7
with_message . . . . .	8

**Index****10**


---

attempt-package	<i>attempt package</i>
-----------------	------------------------

---

**Description**

A friendlier condition handler, inspired by purrr mappers and based on rlang. 'attempt' extends and facilitates condition handling by providing a consistent grammar, and provides a set of easy to use functions for common tests and conditions. 'attempt' only depends on rlang, and focuses on speed, so it can be easily integrated in other functions and used in data analysis.

**Author(s)**

colin <contact@colinfay.me>

---

attempt	<i>Attempt</i>
---------	----------------

---

**Description**

A wrapper around base try that allows you to set a custom message when an error/warning occurs. attempt returns the value if there is no error nor message.

**Usage**

```
attempt(expr, msg = NULL, verbose = FALSE, silent = FALSE)
```

**Arguments**

expr	the expression to be evaluated
msg	the message to return if an error occurs
verbose	wether or not to return to expression producing the error
silent	wether or not the error should be kept under silence

**Examples**

```
## Not run:
attempt(log("a"), msg = "Nop !")

## End(Not run)
```

---

if_all	<i>Test for all, any or none</i>
--------	----------------------------------

---

**Description**

Test for all, any or none

**Usage**

```
if_all(.l, .p = isTRUE, .f)
```

```
if_any(.l, .p = isTRUE, .f)
```

```
if_none(.l, .p = isTRUE, .f)
```

**Arguments**

.l	the list to test.
.p	the predicate for testing. Default is isTRUE.
.f	a mapper or a function run if .p(x) is TRUE.

**Value**

If .p(x) is TRUE, .f() is run.

**Examples**

```
if_all(1:10, ~ .x < 11, ~ return(letters[1:10]))
if_any(1:10, is.numeric, ~ return(letters[1:10]))
if_none(1:10, is.numeric, ~ return(letters[1:10]))
```

---

if_then	<i>If this, then that</i>
---------	---------------------------

---

**Description**

If this, then that

**Usage**

```
if_then(.x, .p = isTRUE, .f)
```

```
if_not(.x, .p = isTRUE, .f)
```

```
if_else(.x, .p = isTRUE, .f, .else)
```

**Arguments**

.x                   the object to test. If NULL (the default), only .p is evaluated.  
 .p                   the predicate for testing. Default is isTRUE.  
 .f                   a mapper or a function run if .p(x) is TRUE  
 .else                a mapper or a function run if .p(x) is not TRUE

**Value**

Depending on whether or not .p(x) is TRUE, .f() or .else() is run.

**Note**

If you want these function to return a value, you need to wrap these values into a mapper / a function. E.g, to return a vector, you'll need to write `if_then(1, is.numeric, ~ "Yay")`.

**Examples**

```
a <- if_then(1, is.numeric, ~ "Yay")
a <- if_not(1, is.character, ~ "Yay")
a <- if_else(.x = TRUE, .f = ~ "Yay", .else = ~ "Nay")
```

---

silently

*Silently*


---

**Description**

silently returns a new function that will returns an error or a warning if any, or else returns nothing.

**Usage**

```
silently(.f)
```

**Arguments**

.f                   the function to silence

**Value**

an error if any, a warning if any. The result is never returned.

**Examples**

```
## Not run:
silent_log <- silently(log)
silent_log(1)
silent_log("a")

## End(Not run)
```

---

silent_attempt	<i>Silently attempt</i>
----------------	-------------------------

---

**Description**

A wrapper around silently and attempt

**Usage**

```
silent_attempt(...)
```

**Arguments**

... the expression to evaluate

**Value**

an error if any, a warning if any.

**Examples**

```
## Not run:  
silent_attempt(warn("nop!"))  
  
## End(Not run)
```

---

stop_if	<i>Warn if</i>
---------	----------------

---

**Description**

Friendlier messaging functions.

**Usage**

```
stop_if(.x, .p = isTRUE, msg = NULL)  
  
stop_if_any(.l, .p = isTRUE, msg = NULL)  
  
stop_if_all(.l, .p = isTRUE, msg = NULL)  
  
stop_if_none(.l, .p = isTRUE, msg = NULL)  
  
stop_if_not(.x, .p = isTRUE, msg = NULL)  
  
warn_if(.x, .p = isTRUE, msg = NULL)
```

```
warn_if_any(.l, .p = isTRUE, msg = NULL)
warn_if_all(.l, .p = isTRUE, msg = NULL)
warn_if_none(.l, .p = isTRUE, msg = NULL)
warn_if_not(.x, .p = isTRUE, msg = NULL)
message_if(.x = NULL, .p = isTRUE, msg = NULL)
message_if_any(.l, .p = isTRUE, msg = NULL)
message_if_all(.l, .p = isTRUE, msg = NULL)
message_if_none(.l, .p = isTRUE, msg = NULL)
message_if_not(.x, .p = isTRUE, msg = NULL)
```

### Arguments

<code>.x</code>	the element to evaluate. It can be a predicate function (i.e a function returning TRUE).
<code>.p</code>	the predicate with the condition to test on <code>.x</code> or <code>.l</code> . Default is <code>isTRUE</code> .
<code>msg</code>	the message to return. If NULL (default), the built-in message is printed.
<code>.l</code>	the list of elements to evaluate

### Examples

```
## Not run:
x <- 12
stop_if(x, is.numeric)
stop_if_not(x, is.character)

a <- "this is not numeric"
warn_if(a, is.character )
warn_if_not(a, is.numeric )
b <- 20
warn_if(b, ~ . > 10 ,
        msg = "Wow, that's a lot of b")
c <- "a"
message_if(c, is.character,
          msg = "You entered a character element")

## End(Not run)
```

---

surely	<i>surely</i>
--------	---------------

---

**Description**

Wrap a function in a try

**Usage**

```
surely(.f)
```

**Arguments**

`.f` the function to wrap

**Value**

an error if any, a warning if any, the result if any

**Examples**

```
## Not run:  
sure_log <- surely(log)  
sure_log(1)  
sure_log("a")  
  
## End(Not run)
```

---

try_catch	<i>Try Catch</i>
-----------	------------------

---

**Description**

Friendlier try catch functions

**Usage**

```
try_catch(expr, .e = NULL, .w = NULL, .f = NULL)  
  
try_catch_df(expr)  
  
map_try_catch(l, fun, .e = NULL, .w = NULL, .f = NULL)  
  
map_try_catch_df(l, fun)
```

**Arguments**

expr	for simple try catch, the expression to be evaluated
.e	a one side formula or a function evaluated when an error occurs
.w	a one side formula or a function evaluated when a warning occurs
.f	a one side formula or an expression evaluated before returning or exiting
l	for map_* function, a list of arguments
fun	for map_* function, a function to try with the list l

**Details**

try\_catch handles errors and warnings the way you specify. try\_catch\_df returns a tibble with the call, the error message if any, the warning message if any, and the value of the evaluated expression.

**Examples**

```
## Not run:
try_catch(log("a"), .e = ~ paste0("There was an error: ", .x))
try_catch(log(1), .f = ~ print("finally"))
try_catch(log(1), .f = function() print("finally"))

## End(Not run)
```

---

with\_message

*Manipulate messages and warnings*


---

**Description**

with\_message and with\_warning add a warning or a message to a function. without\_message and without\_warning turn the warning and message off.

**Usage**

```
with_message(.f, msg)
```

```
with_warning(.f, msg)
```

```
without_message(.f)
```

```
without_warning(.f)
```

**Arguments**

.f	the function to wrap
msg	the message to print



*with\_message*

9

### **Value**

a function

### **Examples**

```
msg_as_num <- with_message(as.numeric, msg = "Numeric conversion")  
warn_as_num <- with_warning(as.numeric, msg = "Numeric conversion")
```

# Index

attempt, 2  
attempt-package, 2

if\_all, 3  
if\_any (if\_all), 3  
if\_else (if\_then), 3  
if\_none (if\_all), 3  
if\_not (if\_then), 3  
if\_then, 3

map\_try\_catch (try\_catch), 7  
map\_try\_catch\_df (try\_catch), 7  
message\_if (stop\_if), 5  
message\_if\_all (stop\_if), 5  
message\_if\_any (stop\_if), 5  
message\_if\_none (stop\_if), 5  
message\_if\_not (stop\_if), 5

silent\_attempt, 5  
silently, 4  
stop\_if, 5  
stop\_if\_all (stop\_if), 5  
stop\_if\_any (stop\_if), 5  
stop\_if\_none (stop\_if), 5  
stop\_if\_not (stop\_if), 5  
surely, 7

try\_catch, 7  
try\_catch\_df (try\_catch), 7

warn\_if (stop\_if), 5  
warn\_if\_all (stop\_if), 5  
warn\_if\_any (stop\_if), 5  
warn\_if\_none (stop\_if), 5  
warn\_if\_not (stop\_if), 5  
with\_message, 8  
with\_warning (with\_message), 8  
without\_message (with\_message), 8  
without\_warning (with\_message), 8