Package ‘retry’

April 23, 2020

Type Package
Title Repeated Evaluation
Version 0.1.0
Description Provide simple mechanism to repeatedly evaluate an expression until either it succeeds or timeout exceeded. It is useful in situations that random failures could happen.
License MIT + file LICENSE
URL https://github.com/randy3k/retry
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Suggests testthat (>= 2.1.0), covr
Imports rlang, later
NeedsCompilation no
Author Randy Lai [aut, cre]
Maintainer Randy Lai <randy.cs.lai@gmail.com>
Repository CRAN
Date/Publication 2020-04-23 16:40:03 UTC

R topics documented:

  retry-package .......................................................... 2
  retry ................................................................. 2
  wait_until ............................................................ 4

Index                      5
retry-package  retry: Repeated Evaluation

Description

Provide simple mechanism to repeatedly evaluate an expression until either it succeeds or timeout exceeded. It is useful in situations that random failures could happen.

Author(s)

Maintainer: Randy Lai <randy.cs.lai@gmail.com>

See Also

Useful links:

- https://github.com/randy3k/retry

retry  Repeatedly evaluate an expression

Description

Repeatedly evaluate an expression until a condition is met or timeout is exceeded.

Usage

retry(
  expr,
  upon = "error",
  when = NULL,
  until = NULL,
  envir = parent.frame(),
  silent = FALSE,
  timeout = Inf,
  max_tries = Inf,
  interval = 0.1,
  later_run_now = FALSE
)
retry

Arguments

- **expr**: an expression to be evaluated, supports quasiquotation.
- **upon**: a vector of condition classes. The expression will be evaluated again after the delay if a condition is thrown. See the classes parameter of `rlang::catch_cnd`.
- **when**: regular expression pattern that matches the message of the condition. It is used to decide if we need to evaluate `expr`.
- **until**: a function of two arguments. This function is used to check if we need to evaluate `expr`. The first argument is the result of `expr` and the second argument is the condition thrown when `expr` was evaluated. It could be also a one-sided formula that is later converted to a function using `rlang::as_function`.
- **envir**: the environment in which the expression is to be evaluated.
- **silent**: suppress messages and warnings
- **timeout**: raise an error if this amount of time in seconds has passed.
- **max_tries**: maximum number of attempts
- **interval**: delay between retries.
- **later_run_now**: execute `later::run_now()` periodically when `later` is loaded?

Examples

```r
capture the error of "a" + 1
retry(f("a"), when = "random error")

# keep retrying when there is a random error
retry(f(1), when = "random error")

# keep retrying until a condition is met
retry(f(1), until = function(val, cnd) val == 2)

# or using one sided formula
retry(f(1), until = ~ . == 2)

try{
  # it doesn’t capture the error of "a" + 1
  retry(f("a"), when = "random error")
})

try{
  # an error is raised after 1 second
  retry(stop("foo"), when = "foo", timeout = 1)
})

try{
  # timeout also works for indefinite R code
  retry(while(TRUE) {}, until = ~FALSE, timeout = 1)
})
```
wait_until

Wait until a condition is met

Description
Block the current runtime until the expression returns TRUE.

Usage
wait_until(
  expr,
  envir = parent.frame(),
  timeout = Inf,
  interval = 0.1,
  later_run_now = TRUE
)

Arguments
expr: an expression to check, supports quasiquotation.
envir: the environment in which the expression is to be evaluated.
timeout: raise an error if this amount of time in second has passed.
interval: delay between retries.
later_run_now: execute later::run_now() periodically later is loaded?

Examples
s <- Sys.time()
system.time(wait_until(Sys.time() - s > 1))

z <- 0
later::later(function() z <<- 1, 1)
wait_until(z == 1)
z == 1
Index

retry, 2
retry-package, 2
wait_until, 4