Package ‘timechecker’

May 8, 2026

Type Package

Title Visualization of Processing Time with Standard Output

Version 1.1.5

Description Displays processing time in a clear and structured way. One function supports iterative workflows by predicting and showing the total time required, while another reports the time taken for individual steps within a process.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Toda Yusuke [aut, cre]

Maintainer Toda Yusuke <yusuke0320.research@gmail.com>

Repository CRAN

Date/Publication 2025-11-24 17:50:02 UTC

Contents

`set_loop_timechecker` *Print elapsed and remaining time in iterative processes.*

Description

`set_loop_timechecker` returns a function that records and prints processing time in iterative processes.

Usage

```
set_loop_timechecker(
  n_iter,
  overwrite = TRUE,
  timestep = 0.5,
  show_timestamp = TRUE,
  verbose = TRUE
)
```

Arguments

`n_iter`	The number of iterations.
`overwrite`	Logical. Should the message be overwritten?
`timestep`	The smallest time step of the output (sec).
`show_timestamp`	Logical. Should the time stamp be added to the message?
`verbose`	Logical. Should the progress be printed in the console?

Details

Provided with the number of iterations, this function creates a function named `loop_timechecker` which records and prints processing time in iterative process. In actual usage, it is recommended to call this function and creating `loop_timechecker` right before the iteration process, and place the `loop_timechecker` at the end of the iteration process.

If you want to keep all printed records in your console, please set `overwrite = FALSE`.

The `timestep` argument determines the frequency of updating printed information since too fast updates will decrease visibility.

Value

A function `loop_timechecker`. When placed at the end of the iterations, it records and prints the progress of iteration, elapsed time, and predicted remaining time. You can add arbitrary strings to the printed messages by using `char_pre` and `char_post` arguments.

See Also

`set_step_timechecker`

Examples

```
iters <- 1:1000
ans <- NULL
tc <- set_loop_timechecker(length(iters))
for (i in iters) {
  ans <- c(ans, i)
  Sys.sleep(0.002)
  tc()
}

# For multiple loops, overwrite and char_pre arguments can be used for readability
iters1 <- 1:3
iters2 <- 1:100
ans <- NULL
tc1 <- set_loop_timechecker(length(iters1), overwrite = FALSE)
for (i in iters1) {
  tc2 <- set_loop_timechecker(length(iters2))
  for (j in iters2) {
    ans <- c(ans, i * j)
    Sys.sleep(0.004)
    tc2(char_pre = '-- ')
  }
  tc1()
}

# char_pre or char_post can also be used to check name of current process
iters <- paste0('case', LETTERS[1:10])
tc <- set_loop_timechecker(length(iters))
for (i in iters) {
  Sys.sleep(1)
  tc(char_post = paste0('  Processing ', i))
}
```

`set_step_timechecker` *Print elapsed and remaining time in each code block.*

Description

`set_step_timechecker` returns a function that records and prints processing time in each code block.

Usage

```
set_step_timechecker(
  char_pre = "",
  char_post = "",
  show_timestamp = TRUE,
  verbose = TRUE
)
```

Arguments

char_pre	String to be added before messages.
char_post	String to be added after messages.
show_timestamp	Logical. Should the time stamp be added to the message?
verbose	Logical. Should the progress be printed in the console?

Details

This function creates a function named step_timechecker which records and prints processing time in code blocks. The created function step_timechecker should be placed at the head of each code block with its name. Also, this function should be placed at the end of all steps with no argument.

Value

A function step_timechecker. When placed at the head of each code block, it records and prints the progress and elapsed time. An argument step_name, a string representing the name of the steps, should be provided. Otherwise, the function judges that it is the end of the measurement. An argument print_done can also be used if you do not wish to add message when the process of the step ends.

See Also

set_loop_timechecker

Examples

```
f <- function() {

  tc <- set_step_timechecker()

  tc('Simulation')
  df <- data.frame(x = 1:10, y = 1:10 + rnorm(10))
  Sys.sleep(2)

  tc('Data augumentation')
  df$x2 <- df$x ^ 2
  df$x3 <- df$x ^ 3
  Sys.sleep(3)

  tc('Regression')
  lmres <- lm(y ~ ., df)
  Sys.sleep(4)

  tc()
  coef(lmres)

}
ans <- f()
```

```
# when you place set_loop_timechecker in a loop,
# argument char_pre can be used for readability
tcl <- set_loop_timechecker(3, overwrite = FALSE)
for (i in 1:3) {

  tc <- set_step_timechecker(char_pre = '  ')

  tc('Simulation')
  df <- data.frame(x = 1:10, y = 1:10 + rnorm(10))
  Sys.sleep(2)

  tc('Data augumentation')
  df$x2 <- df$x ^ 2
  df$x3 <- df$x ^ 3
  Sys.sleep(3)

  tc('Regression')
  lmres <- lm(y ~ ., df)
  Sys.sleep(4)

  tc()
  tcl()

}

# when you place set_loop_timechecker inside a step in set_step_timechecker,
# argument print_done can be used for readability
f2 <- function() {

  tc <- set_step_timechecker()

  tc('Simulation')
  n <- 10
  df <- data.frame(x = 1:n, y = 1:n + rnorm(10))
  Sys.sleep(2)

  tc('Data augumentation', print_done = FALSE)
  tcl <- set_loop_timechecker(n)
  for (i in seq_len(n)) {
    df$x2[i] <- df$x[i] ^ 2
    df$x3[i] <- df$x[i] ^ 3
    Sys.sleep(0.2)
    tcl(char_pre = '- ')
  }

  tc('Regression')
  lmres <- lm(y ~ ., df)
  Sys.sleep(4)

  tc()
  coef(lmres)

}
```

```
ans <- f2()
```

Index