Package ‘purrrrogress’

July 22, 2019

Title  Add Progress Bars to Mapping Functions
Version  0.1.1
Description  Provides functions to easily add progress bars to apply calls.
License  MIT + file LICENSE
Encoding  UTF-8
LazyData  true
Imports  R6, assertthat, glue, hms, methods, pkgcond, purrr, testextra, utils, rlang
RoxygenNote  6.1.1
Language  en-US
Suggests  covr, datasets, stringi, testthat, tibble
Enhances  dplyr
URL  https://github.com/halpo/purrrrogress
BugReports  https://github.com/halpo/purrrrogress/issues
NeedsCompilation  no
Author  Andrew Redd [aut, cre] (<https://orcid.org/0000-0002-6149-2438>)
Maintainer  Andrew Redd <andrew.redd@hsc.utah.edu>
Repository  CRAN
Date/Publication  2019-07-22 21:10:08 UTC

R topics documented:

  is_purrr_map2_fun ........................................... 2
  progress_bar .............................................. 2
  R6_progress ............................................... 3
  with_progress ............................................. 3
is_purrr_map2_fun  Check if a function is a map2 derived function

Description

Besides the obvious map2 and map2_* variants, this also covers functions based off map2:

- imap and imap_* variants.
- invoke_map and [invoke_map_*] variants.

Usage

is_purrr_map2_fun(fun)

Arguments

fun  function to test.

progress_bar  Create a R6 progress bar directly

Description

Create a R6 progress bar directly

Usage

progress_bar(total, title = "Progress", ..., type = getOption("progress.type", infer_type()))

Arguments

total  the total number of elements

title  the title of the progress bar

...  passed on to the specific constructor determined by type.

type  the type of progress bar to create as a string, or an R6ClassGenerator object for a class that inherits from the "R6 Progress Base Class".
Examples

pb_win <- progress_bar(100, "Windows Progress", type = 'win')

pb_txt <- progress_bar(100, "Text Progress", type = 'txt')
pb_txt$init() # starts the timer and shows the bar.
pb_txt$step() # take 1 step update progress bar.
pb_txt$step(25) # take 24 steps at one time
pb_txt$term() # do finishing tasks for progress bar.

# The following use Unicode characters and may not work with all fonts.
# DejaVu Sans Mono is one font which supports all the characters used
pb_bar <- progress_bar(100, "Bar Progress", type = 'bar')
pb_line <- progress_bar(100, "Line Progress", type = 'line')
pb_box <- progress_bar(100, "Box Progress", type = 'box')
pb_block <- progress_bar(100, "Block Progress", type = 'block')

R6_progress  Base Progress bar Class

Description

This is the base class for all R6 progress bars. It also doubles as a null progress bar that displays no progress bar, but allows for checking values.

Usage

R6_progress

Format

An object of class R6ClassGenerator of length 24.

with_progress  Apply a function with progress bars.

Description

Apply a function with progress bars.

Usage

with_progress(fun, total, ...)
Arguments

fun  The function to be apply

total  The total number of elements to be mapped. If omitted an attempt will be made to infer the correct number.

...  Arguments passed on to progress_bar

total  the total number of elements

title  the title of the progress bar

type  the type of progress bar to create as a string, or an R6ClassGenerator object for a class that inherits from the "R6 Progress Base Class".

Examples

# with purrr functions
long_function <- function(x, how.long=0.05){
  Sys.sleep(how.long)
  x
}

purrr::walk(1:100, with_progress(long_function))
purrr::walk2(1:100, 0.01, with_progress(long_function))

# with dplyr::group_map
if(require(dplyr)){
group_function <- function(x, y, how.long=0.05){
  Sys.sleep(how.long)
  x
}

group_map( group_by(mtcars, cyl, gear)
  , with_progress(group_function, type='line')
  , how.long=1/3)
group_walk( group_by_all(mtcars)
  , with_progress(group_function, type='box')
  , how.long=1)
}

# with standard apply functions
sapply(1:100, with_progress(long_function, type='txt'), 0.001)