Package ‘pbmcapply’

October 14, 2022

Type Package

Title Tracking the Progress of Mc*apply with Progress Bar

Version 1.5.1

Author Kevin Kuang (aut), Quyu Kong (ctb), Francesco Napolitano (ctb)

Maintainer Kevin kuang <kvn.kuang@mail.utoronto.ca>

Description A light-weight package helps you track and visualize
the progress of parallel version of vectorized R functions (mc*apply).
Parallelization (mc.core > 1) works only on *nix (Linux, Unix such as macOS) system due to
the lack of fork() functionality, which is essential for mc*apply, on Windows.

Depends utils, parallel

BugReports https://github.com/kvnkuang/pbmcapply/issues

URL https://github.com/kvnkuang/pbmcapply

License MIT + file LICENSE

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-28 14:50:02 UTC

R topics documented:

pbmclapply ................................................................. 2
pbmcapply ............................................................... 3
progressBar ............................................................. 4

Index 6
pbmclapply

Tracking mclapply with progress bar

Description

pbmclapply is a wrapper around the mclapply function. It adds a progress bar to mclapply function.

Parallelization (mc.core > 1) works only on *nix (Linux, Unix such as macOS) system due to the lack of fork() functionality, which is essential for mcapply, on Windows.

Usage

pbmclapply(X, FUN, ..., mc.style = "ETA", mc.substyle = NA, mc.cores = getOption("mc.cores", 2L), ignore.interactive = getOption("ignore.interactive", F), mc.preschedule = TRUE, mc.set.seed = TRUE, mc.cleanup = TRUE, mc.allow.recursive = TRUE)

Arguments

X a vector (atomic or list) or an expressions vector. Other objects (including classed objects) will be coerced by 'as.list'.

FUN the function to be applied to.

... optional arguments to FUN.

mc.cores see mclapply.

mc.style, mc.substyle style of the progress bar. See progressBar.

ignore.interactive whether the interactive() is ignored. If set to TRUE, the progress bar will be printed even in a non-interactive environment (e.g. called by Rscript). Can be set as an option "ignore.interactive".

mc.preschedule, mc.set.seed, mc.cleanup, mc.allow.recursive See mclapply.

Examples

# A lazy sqrt function which doesn't care about efficiency
lazySqrt <- function(num) {
  # Sleep randomly between 0 to 0.5 second
  Sys.sleep(runif(1, 0, 0.5))
  return(sqrt(num))
}

# On Windows, set cores to be 1
if (.Platform$OS.type == "windows") {
  

pbmcmapply

    cores = 1
} else {
    cores = 2
}

# A lazy and chatty sqrt function.
# An example of passing arguments to pbmclapply.
lazyChattySqrt <- function(num, name) {
    # Sleep randomly between 0 to 0.5 second
    Sys.sleep(runif(1, 0, 0.5))
    return(sprintf("Hello %s, the sqrt of %f is %f.", toString(name), num, sqrt(num)))
}

# Get the sqrt of 1-3 in parallel
result <- pbmclapply(1:3, lazySqrt, mc.cores = cores)
chattyResult <- pbmclapply(1:3, lazyChattySqrt, "Bob", mc.cores = cores)

pbmcmapply

Tracking mcmapply with progress bar

Description

pbmcmapply is a wrapper around the mcmapply function. It adds a progress bar to mcmapply function.

Parallelization (mc.core > 1) works only on *nix (Linux, Unix such as macOS) system due to the lack of fork() functionality, which is essential for mcmapply, on Windows.

Usage

pbmcmapply(FUN, ..., MoreArgs = NULL,
    mc.style = "ETA", mc.substyle = NA,
    mc.cores = getOption("mc.cores", 2L),
    ignore.interactive = getOption("ignore.interactive", F),
    mc.preschedule = TRUE, mc.set.seed = TRUE,
    mc.cleanup = TRUE)

Arguments

FUN 
    the function to be applied in parallel to ...

... 
    arguments to vectorize over (vectors or lists of strictly positive length, or all of zero length).

MoreArgs 
    a list of other arguments to FUN.

mc.cores 
    see mcmapply.

mc.style, mc.substyle 
    style of the progress bar. See progressBar.
ignore.interactive

whether the interactive() is ignored. If set to TRUE, the progress bar will be
printed even in a non-interactive environment (e.g. called by Rscript). Can be
set as an option "ignore.interactive".

mc.preschedule, mc.set.seed, mc.cleanup

See mcmapply.

Examples

# A lazy sqrt function which doesn't care about efficiency
lazySqrt <- function(num) {
  # Sleep randomly between 0 to 0.5 second
  Sys.sleep(runif(1, 0, 0.5))
  return(sqrt(num))
}

# On Windows, set cores to be 1
if (.Platform$OS.type == "windows") {
  cores = 1
} else {
  cores = 2
}

# A lazy and chatty sqrt function.
# An example of passing arguments to pbmcmapply.
lazyChattySqrt <- function(num, name) {
  # Sleep randomly between 0 to 0.5 second
  Sys.sleep(runif(1, 0, 0.5))
  return(sprintf("Hello %s, the sqrt of %f is %f.", toString(name), num, sqrt(num)))
}

# Get the sqrt of 1-3 in parallel
result <- pbmcmapply(lazySqrt, 1:3, mc.cores = cores)
chattyResult <- pbmcmapply(lazyChattySqrt, 1:3, MoreArgs = list("Bob"), mc.cores = cores)

progressBar

Progress bar with the estimated time to completion (ETA).

Description

This is an extended version of the txtProgressBar function with the estimated time to comple-
tion (ETA). Please refer to that for documentation (help(utils::txtProgressBar)). The original
utils::setTxtProgressBar can be used to update the bar. Use help(setTxtProgressBar,
"utils") to get help about the original function.

Usage

progressBar(min = 0, max = 1, initial = 0, style = "ETA", substyle = NA,
char = ", width = NA, file = ")
Arguments

- min, max, initial
  - see `txtProgress Bar`.
- style
  - style of the progress bar - see 'Details'.
- substyle
  - substyle of the progress bar - only needed when style is set to certain value (see 'Details').
- char, width, file
  - see `txtProgress Bar`.

Details

When style = "txt", it performs exactly the same as the original `txtProgress Bar`. In this case, substyle shall be treated as the style in the original `txtProgress Bar`. Please refer to the 'Detail' of `txtProgress Bar` for the meanings of substyles.

When style = "ETA", it shows a progress bar with the estimated time to completion (ETA). Substyle is not used in this case. However, when running in a terminal and the width of the terminal windows is smaller than 40 characters, the progress bar will not be displayed.

Value

An object of class "txtProgress Bar".

Note

Code derived from library pbarETA (https://github.com/franapoli/pbarETA) by Francesco Napolitano <franapoli@gmail.com>.

See Also

- `txtProgress Bar`

Examples

```r
# Test function
testit <- function(x, ...) {
  pb <- progressBar(...)
  for(i in c(0, x, 1)) {
    setTxtProgressBar(pb, i)
  }
  close(pb)
}

# Txt progress bar
testit(sort(runif(10)), style = "txt", substyle = 3)

# ETA progress bar
testit(sort(runif(10)), style = "ETA")
```
Index

mclapply, 2
mcmapply, 3, 4

pbmclapply, 2
pbmcmapply, 3
progressBar, 2, 3, 4

txtProgressBar, 5