Package 'progress'

June 5, 2015

Title Terminal Progress Bars

Version 1.0.2

Author Gabor Csardi [aut, cre]

Maintainer Gabor Csardi <csardi.gabor@gmail.com>

Description Terminal progress bars. They are
configurable, may include percentage, elapsed time, and/or
the estimated completion time. They work in the command line,
in Emacs, R Studio, Windows Rgui and Mac OSX R.app. The package
also provides a C++ API, that works with or without Rcpp.

License MIT + file LICENSE

LazyData true

URL https://github.com/gaborcsardi/progress

BugReports https://github.com/gaborcsardi/progress/issues

Imports prettyunits, R6

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2015-06-05 01:14:18

R topics documented:

progress_bar *Progress bar in the terminal*

Description

Progress bars are configurable, may include percentage, elapsed time, and/or the estimated completion time. They work in the command line, in Emacs and in R Studio. The progress package was heavily influenced by https://github.com/tj/node-progress

Creating the progress bar

A progress bar is an R6 object, that can be created with `progress_bar$new()`. It has the following arguments:

format The format of the progress bar. A number of tokens can be used here, see them below. It defaults to `"[:bar] :percent"`, which means that the progress bar is within brackets on the left, and the percentage is printed on the right.

total Total number of ticks to complete. Defaults to 100.

width Width of the progress bar. Default is the current terminal width (see `options()` and `width`) minus two.

stream The output stream to put the progress bar on. It defaults to `stderr()`, except in R Studio that has a bug when printing on the standard error, so there we use `stdout`. If the output stream is not a terminal and we are not in R Studio, then no progress bar is printed.

complete Completion character, defaults to `=`.

incomplete Incomplete character, defaults to `-`.

callback Callback function to call when the progress bar finishes. The progress bar object itself is passed to it as the single parameter.

clear Whether to clear the progress bar on completion. Defaults to `TRUE`.

show_after Amount of time in seconds, after which the progress bar is shown on the screen. For very short processes, it is probably not worth showing it at all. Defaults to two tenth of a second.

force Whether to force showing the progress bar, even if the given (or default) stream does not seem support it.

Using the progress bar

Two functions can update a progress bar. `progress_bar$tick()` increases the number of ticks by one (or another specified value). `progress_bar$update()` sets a given ratio.

The progress bar is displayed after the first ‘tick‘ command. This might not be desirable for long computations, because nothing is shown before the first tick. It is good practice to call ‘tick(0)‘ at the beginning of the computation or download, which shows the progress bar immediately.

Tokens

They can be used in the `format` argument when creating the progress bar.

:bar The progress bar itself.

:current Current tick number.

:total Total ticks.

:elapsed Elapsed time in seconds.

:eta Estimated completion time in seconds.

:percent Completion percentage.

:rate Download rate, bytes per second. See example below.

:bytes Shows :current, formatted as bytes. Useful for downloads or file reads if you don't know the size of the file in advance. See example below.

Custom tokens are also supported, and you need to pass their values to `progress_bar$tick()` or `progress_bar$update()`, in a named list. See example below.

Examples

```
## Basic
pb <- progress_bar$new(total = 100)
for (i in 1:100) {
  pb$tick()
  Sys.sleep(1 / 100)
}

## ETA
pb <- progress_bar$new(
  format = "  downloading [:bar] :percent eta: :eta",
  total = 100, clear = FALSE, width= 60)
for (i in 1:100) {
  pb$tick()
  Sys.sleep(1 / 100)
}

## Elapsed time
pb <- progress_bar$new(
  format = "  downloading [:bar] :percent in :elapsed",
  total = 100, clear = FALSE, width= 60)
for (i in 1:100) {
  pb$tick()
  Sys.sleep(1 / 100)
}

## Custom tokens
pb <- progress_bar$new(
  format = "  downloading :what [:bar] :percent eta: :eta",
  clear = FALSE, total = 200, width = 60)
f <- function() {
  for (i in 1:100) {
```

```
    pb$tick(tokens = list(what = "foo   "))
    Sys.sleep(2 / 100)
  }
  for (i in 1:100) {
    pb$tick(tokens = list(what = "foobar"))
    Sys.sleep(2 / 100)
  }
}
f()

## Download (or other) rates
pb <- progress_bar$new(
  format = "  downloading foobar at :rate, got :bytes in :elapsed",
  clear = FALSE, total = 1e7, width = 60)
f <- function() {
  for (i in 1:100) {
    pb$tick(sample(1:100 * 1000, 1))
    Sys.sleep(2/100)
  }
  pb$tick(1e7)
  invisible()
}
f()
```

Index