

Package ‘profvis’

February 22, 2018

Title Interactive Visualizations for Profiling R Code

Version 0.3.5

Description Interactive visualizations for profiling R code.

Depends R (>= 3.0)

Imports htmlwidgets (>= 0.3.2), stringr

License GPL-3 | file LICENSE

Suggests knitr, ggplot2, rmarkdown, testthat, devtools

RoxygenNote 6.0.1.9000

URL <https://rstudio.github.io/profvis/>

NeedsCompilation yes

Author Winston Chang [aut, cre],
Javier Luraschi [aut],
RStudio [cph],
jQuery Foundation [cph] (jQuery library),
jQuery contributors [ctb, cph] (jQuery library; authors listed in
inst/www/shared/jquery-AUTHORS.txt),
Mike Bostock [ctb, cph] (D3 library),
D3 contributors [ctb] (D3 library),
Ivan Sagalaev [ctb, cph] (highlight.js library)

Maintainer Winston Chang <winston@rstudio.com>

Repository CRAN

Date/Publication 2018-02-22 04:24:31 UTC

R topics documented:

parse_rprof	2
pause	2
print.profvis	3
profvis	3
profvisOutput	5
renderProfvis	5
Index	6

parse_rprof	<i>Parse Rprof output file for use with profvis</i>
-------------	---

Description

Parse Rprof output file for use with profvis

Usage

```
parse_rprof(path = "Rprof.out", expr_source = NULL)
```

Arguments

path	Path to the Rprof output file.
expr_source	If any source refs in the profiling output have an empty filename, that means they refer to code executed at the R console. This code can be captured and passed (as a string) as the <code>expr_source</code> argument.

pause	<i>Pause an R process</i>
-------	---------------------------

Description

This function pauses an R process for some amount of time. It differs from [Sys.sleep](#) in that time spent in `pause` will show up in profiler data. Another difference is that `pause` uses up 100 whereas `Sys.sleep` does not.

Usage

```
pause(seconds)
```

Arguments

seconds	Number of seconds to pause.
---------	-----------------------------

Examples

```
# Wait for 0.5 seconds  
pause(0.5)
```

print.profvis	<i>Print a profvis object</i>
---------------	-------------------------------

Description

Print a profvis object

Usage

```
## S3 method for class 'profvis'
print(x, ..., width = NULL, height = NULL, split = NULL)
```

Arguments

x	The object to print.
...	Further arguments to passed on to other print methods.
width	Width of the htmlwidget.
height	Height of the htmlwidget
split	Direction of split. Either "v" (the default) for vertical, or "h" for horizontal. This is the orientation of the split bar.

profvis	<i>Profile an R expression and visualize profiling data</i>
---------	---

Description

This function will run an R expression with profiling, and then return an htmlwidget for interactively exploring the profiling data.

Usage

```
profvis(expr = NULL, interval = 0.01, prof_output = NULL,
        prof_input = NULL, width = NULL, height = NULL, split = c("h", "v"),
        torture = 0)
```

Arguments

expr	Code to profile. Not compatible with prof_input.
interval	Interval for profiling samples, in seconds. Values less than 0.005 (5 ms) will probably not result in accurate timings
prof_output	Name of an Rprof output file or directory in which to save profiling data. If NULL (the default), a temporary file will be used and automatically removed when the function exits. For a directory, a random filename is used.

prof_input	The path to an Rprof data file. Not compatible with <code>expr</code> or <code>prof_output</code> .
width	Width of the htmlwidget.
height	Height of the htmlwidget
split	Direction of split. Either "v" (the default) for vertical, or "h" for horizontal. This is the orientation of the split bar.
torture	Triggers garbage collection after every <code>torture</code> memory allocation call. Note that memory allocation is only approximate due to the nature of the sampling profiler and garbage collection: when garbage collection triggers, memory allocations will be attributed to different lines of code. Using <code>torture = steps</code> helps prevent this, by making R trigger garbage collection after every <code>torture</code> memory allocation step.

Details

An alternate way to use `profvis` is to separately capture the profiling data to a file using `Rprof()`, and then pass the path to the corresponding data file as the `prof_input` argument to `profvis()`.

See Also

[print.profvis](#) for printing options.

[Rprof](#) for more information about how the profiling data is collected.

Examples

```
# Only run these examples in interactive R sessions
if (interactive()) {

# Profile some code
profvis({
  dat <- data.frame(
    x = rnorm(5e4),
    y = rnorm(5e4)
  )

  plot(x ~ y, data = dat)
  m <- lm(x ~ y, data = dat)
  abline(m, col = "red")
})

# Save a profile to an HTML file
p <- profvis({
  dat <- data.frame(
    x = rnorm(5e4),
    y = rnorm(5e4)
  )

  plot(x ~ y, data = dat)
  m <- lm(x ~ y, data = dat)
  abline(m, col = "red")
})
}
```

```

  })
  htmlwidgets::saveWidget(p, "profile.html")

  # Can open in browser from R
  browseURL("profile.html")

}

```

profvisOutput *Widget output function for use in Shiny*

Description

Widget output function for use in Shiny

Usage

```
profvisOutput(outputId, width = "100%", height = "600px")
```

Arguments

outputId	Output variable for profile visualization.
width	Width of the htmlwidget.
height	Height of the htmlwidget

renderProfvis *Widget render function for use in Shiny*

Description

Widget render function for use in Shiny

Usage

```
renderProfvis(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	An expression that returns a profvis object.
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with <code>quote()</code>)?

Index

`parse_rprof`, [2](#)
`pause`, [2](#)
`print.profvis`, [3](#), [4](#)
`profvis`, [3](#)
`profvisOutput`, [5](#)

`quote`, [5](#)

`renderProfvis`, [5](#)
`Rprof`, [2](#), [4](#)

`Sys.sleep`, [2](#)