Package 'GUIProfiler'

January 20, 2025

Type Package

Title Graphical User Interface for Rprof()

Version 2.0.1

Date 2015-08-23

Author Fernando de Villar<fdevillar@gmail.com> and Angel Rubio <arubio@ceit.es>

Depends Nozzle.R1, MASS, proftools, Rgraphviz, graph

Imports methods, utils, rstudioapi,grDevices

Maintainer Fernando de Villar<fdevillar@gmail.com>

Description Show graphically the results of profiling R functions by tracking their execution time.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2015-08-23 10:33:01

Contents

`GUIProfiler-package` *Graphical User Interface for Rprof()*

Description

Show graphically the results of profiling R functions by tracking their execution time.

Details

Package:	GUIProfiler
Type:	Package
Version:	2.0.1
Date:	2015-08-23
License:	GPL

This package mimics the behavior of the Matlab profiler: after the code in a file is executed, it is generated an HTML report. This HTML report includes information on the time spent on each of the lines of the profiled code and hyperlinks to jump across the included functions. The graphical interface makes it easy to identify which are the specific lines that may slow down the code.

Author(s)

Fernando de Villar and Angel Rubio

Maintainer: Fernando de Villar <fdevillar@gmail.com>

See Also

RRprofStart, RRprofStop, RRprofReport, Rprof

Examples

```
temp<-tempdir()
# Definition of two functions
normal.solve <- function(A,b) {
  Output <- solve(crossprod(A), t(A)%*%b)
}

chol.solve <- function(A,b) {
  L <- chol(crossprod(A))
  Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
  Output2 <- backsolve(L, Output1)
}

compareMethods <- function() {
  library(MASS)
  # Call the functions
  source(paste(temp,"/normal.solve.R",sep=""))
  source(paste(temp,"/chol.solve.R",sep=""))
  # Solving a big system of equations
  nrows <- 1000
  ncols <- 500
  A <- matrix(rnorm(nrows*ncols),nrows,ncols)
  b <- rnorm(nrows)
  # Testing different possibilities
  Sol1 <- qr.solve(A,b) # Using QR factorization
  Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
  Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
```

```
  Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
  Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
}

# Dump these functions to three different files

dump("normal.solve",file=paste(temp,"/normal.solve.R",sep=""))
dump("chol.solve",file=paste(temp,"/chol.solve.R",sep=""))
dump("compareMethods",file=paste(temp,"/compareMethods.R",sep=""))
source(paste(temp,"/compareMethods.R",sep=""))

# Profile the code

RRprofStart()
compareMethods()
RRprofStop()
# Uncomment to open the report
#RRprofReport()
```

RRprofReport *RRprofReport*

Description

Generate the report based on the output of the R profiler.

Usage

```
RRprofReport(file.name = "RRprof.out", notepad.path =
"C:/Program Files/Notepad++/notepad++.exe",reportname = "my_report")
```

Arguments

file.name	Name of a file produced by RRprofStart()
notepad.path	Path where notepad++.exe is
reportname	Name of the html file to be generated

Details

This function generates a profiling report as a html file in the working directory.

The report consists of two main groups of tables: a summary of the called functions with the time spent for each of them and a second group of tables with the time spent on each line of code for each function. Furthermore, it includes a graph of the different hierarchical relationships between the called functions generated using functions in the package 'proftools'.

In the RStudio environment, this report is shown in the viewer pane. In addition, the markers panel indicates the lines of code where more time was spent to ease the navigation across the source code by simply clicking on them.

When it is opened in a browser, a convenient implemented feature is that the line numbers of the functions are clickable. If Notepad++ is installed, once a line number is clicked, the corresponding file is opened with the cursor on the selected line.

Note

It is advisable to open the Report using Internet Explorer, because other browsers can block the clickable line numbers feature

Author(s)

Fernando de Villar and Angel Rubio

See Also

RRprofStop, RRprofStart, Rprof

Examples

```
temp<-tempdir()
# Definition of two functions
normal.solve <- function(A,b) {
  Output <- solve(crossprod(A), t(A)%*%b)
}

chol.solve <- function(A,b) {
  L <- chol(crossprod(A))
  Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
  Output2 <- backsolve(L, Output1)
}

compareMethods <- function() {
  library(MASS)
  # Call the functions
  source(paste(temp,"/normal.solve.R",sep=""))
  source(paste(temp,"/chol.solve.R",sep=""))
  # Solving a big system of equations
  nrows <- 1000
  ncols <- 500
  A <- matrix(rnorm(nrows*ncols),nrows,ncols)
  b <- rnorm(nrows)
  # Testing different possibilities
  Sol1 <- qr.solve(A,b) # Using QR factorization
  Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
  Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
  Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
  Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
}

# Dump these functions to three different files

dump("normal.solve",file=paste(temp,"/normal.solve.R",sep=""))
```

```
dump("chol.solve",file=paste(temp,"/chol.solve.R",sep=""))
dump("compareMethods",file=paste(temp,"/compareMethods.R",sep=""))
source(paste(temp,"/compareMethods.R",sep=""))

# Profile the code

RRprofStart()
compareMethods()
RRprofStop()
# Uncomment to open the report
#RRprofReport()
```

RRprofStart *RRprofStart*

Description

Rprof() is activated and started

Usage

```
RRprofStart(filename = "RRprof.out", interval = 0.02, numfiles = 100L, bufsize = 10000L)
```

Arguments

filename	The file to be used for recording the profiling results.
interval	real: time interval between samples.
numfiles	integers: line profiling memory
bufsize	allocation

Note

The profiler interrupts R asynchronously, and it cannot allocate memory to store results as it runs. This affects line profiling, which needs to store an unknown number of file pathnames. The numfiles and bufsize arguments control the size of pre-allocated buffers to hold these results: the former counts the maximum number of paths, the latter counts the numbers of bytes in them. If the profiler runs out of space it will skip recording the line information for new files, and issue a warning when Rprof(NULL) is called to finish profiling.

The timing interval cannot be too small, for the time spent in each profiling step is added to the interval. What is feasible is machine-dependent, but 10ms seemed as small as advisable on a 1GHz machine.

Author(s)

Fernando de Villar and Angel Rubio

See Also

RRprofStop, RRprofReport, Rprof

Examples

```
temp<-tempdir()
# Definition of two functions
normal.solve <- function(A,b) {
  Output <- solve(crossprod(A), t(A)%*%b)
}

chol.solve <- function(A,b) {
  L <- chol(crossprod(A))
  Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
  Output2 <- backsolve(L, Output1)
}

compareMethods <- function() {
  library(MASS)
  # Call the functions
  source(paste(temp,"/normal.solve.R",sep=""))
  source(paste(temp,"/chol.solve.R",sep=""))
  # Solving a big system of equations
  nrows <- 1000
  ncols <- 500
  A <- matrix(rnorm(nrows*ncols),nrows,ncols)
  b <- rnorm(nrows)
  # Testing different possibilities
  Sol1 <- qr.solve(A,b) # Using QR factorization
  Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
  Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
  Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
  Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
}

# Dump these functions to three different files

dump("normal.solve",file=paste(temp,"/normal.solve.R",sep=""))
dump("chol.solve",file=paste(temp,"/chol.solve.R",sep=""))
dump("compareMethods",file=paste(temp,"/compareMethods.R",sep=""))
source(paste(temp,"/compareMethods.R",sep=""))

# Profile the code

RRprofStart()
compareMethods()
RRprofStop()
# Uncomment to open the report
#RRprofReport()
```

RRprofStop *RRprofStop*

Description

Once the R expressions to be profiled have finished, the profile is stopped.

Usage

```
RRprofStop()
```

Author(s)

Fernando de Villar and Angel Rubio

See Also

RRprofStart, RRprofReport, Rprof

Examples

```
temp<-tempdir()
# Definition of two functions
normal.solve <- function(A,b) {
  Output <- solve(crossprod(A), t(A)%*%b)
}

chol.solve <- function(A,b) {
  L <- chol(crossprod(A))
  Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
  Output2 <- backsolve(L, Output1)
}

compareMethods <- function() {
  library(MASS)
  # Call the functions
  source(paste(temp,"/normal.solve.R",sep=""))
  source(paste(temp,"/chol.solve.R",sep=""))
  # Solving a big system of equations
  nrows <- 1000
  ncols <- 500
  A <- matrix(rnorm(nrows*ncols),nrows,ncols)
  b <- rnorm(nrows)
  # Testing different possibilities
  Sol1 <- qr.solve(A,b) # Using QR factorization
  Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
  Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
  Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
  Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
}
```

```
# Dump these functions to three different files

dump("normal.solve",file=paste(temp,"/normal.solve.R",sep=""))
dump("chol.solve",file=paste(temp,"/chol.solve.R",sep=""))
dump("compareMethods",file=paste(temp,"/compareMethods.R",sep=""))
source(paste(temp,"/compareMethods.R",sep=""))

# Profile the code

RRprofStart()
compareMethods()
RRprofStop()
# Uncomment to open the report
#RRprofReport()
```

Index