Package 'proftools'

October 14, 2022

Title Profile Output Processing Tools for R

Version 0.99-3

Author Luke Tierney and Riad Jarjour

Description Tools for examining Rprof profile output.

Maintainer Luke Tierney <luke-tierney@uiowa.edu>

Suggests graph,Rgraphviz,boot,knitr

VignetteBuilder knitr

License GPL

ByteCompile yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-08 06:20:23 UTC

R topics documented:

rgl-package *Tools for examining R profile output*

Description

Tools for examining and displaying ouptut from the Rprof R profiling tool.

Details

proftools provides a set of tools for summarizing and displaying time profile outpus produced by R's Rprof.

The starting point for a profiling analysis using **proftools** is to profile code using Rprof and then use readProfileData to read in the profile data into a sutable format for furhter processing. An alternative is to use the profileExpr function to handle profiling and reading in one step. The function filterProfileData can be used to narrow the profile data to particular regions of interest.

The summary functions funSummary and callSummary produce summaries at the function and call level. pathSummary produces a summary for each unique call stack, or path; and hotPaths identifies produces path data ordered to show the hottest paths first.

If source information is recorded when profiling then srcSummary to show profiling by source lines, and annotateSource produces an annotated version of the source files.

The plot method for profile data objects can produce call graphs, tree maps, flame graphs, and time graphs; the type argument choses the particular visualization to produce. These graphs can also be produced by the functions plotProfileCallGraph, calleeTreeMap, and flameGraph.

The function writeCallgrindFile writes a file for use by the codekcachegrind program available on some operating systems.

flatProfile

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
funSummary(pd)
callSummary(pd)
pathSummary(pd)
hotPaths(pd)
plot(pd)
plot(filterProfileData(pd, focus = "glm", self.pct=1, total.pct=10))
```

annotateSource *Annotate Source Files*

Description

Annotates source files with profile information.

Usage

```
annotateSource(pd, value = c("pct", "time", "hits"), GC = TRUE,
               sep = ":  ", show = TRUE, ...)
```

Arguments

pd	profile data as returned by `readProfileData`.
value	character; show result as percentage, time, or hits.
GC	logical; include GC information or not.
sep	character; separator between profile info and source lines.
show	logical; if true, show files with `file.show`.
...	additional arguments for `file.show`.

Details

For lines that appear in the stack trace the percent time and, optionally, GC time are shown before each line.

Value

A list of character vectors of the annotated file lines.

Author(s)

Luke Tierney

See Also

`Rprof`, `summaryRprof`, `flatProfile`, `filterProfileData`, `readProfileData`, `srcSummary`

Examples

```
## This defines the function rw()
source(system.file("samples", "rw.R", package="proftools"))

## Execute the function and collect profile data
Rprof(tmp <- tempfile(), gc.profiling = TRUE, line.profiling = TRUE)
w <- rw(200000)
Rprof(NULL)
```

```
pd <- readProfileData(tmp)
unlink(tmp)

## Annotate the sources
annotateSource(pd)
```

`callTreeGraphs` *Call Tree Graphs*

Description

Produce a flame graph or a callee tree map for the call tree in a profile stack trace.

Usage

```
flameGraph(pd, svgfile, order = c("hot", "alpha", "time"),
           colormap = NULL, srclines = FALSE, cex = 0.75,
           main = "Call Graph", tooltip = FALSE)

calleeTreeMap(pd, srclines = FALSE, cex = 0.75, colormap = NULL,
              main = "Callee Tree Map", squarify = FALSE, border = NULL)

## S3 method for class 'proftools_calleeTreeMap'
identify(x, n = 1, print = FALSE, ...)

## S3 method for class 'proftools_flameGraph'
identify(x, n = 1, print = FALSE,
                                       outline = FALSE, ...)
```

Arguments

`pd`	profile data as returned by `readProfileData`.
`svgfile`	character; name for SVG output file.
`order`	character; see details below.
`colormap`	a function or `NULL`; see details below.
`srclines`	logical; include source information, if available, or not.
`cex`	numeric character expansion value.
`main`	character; plot title.
`tooltip`	logical; whether SVG should show details in tooltips.
`border`	character or `NULL`; border color for rectangles.
`squarify`	logical; whether the squarified tiling algorithm should be used.
`x`	`flameGraph` of `calleeTreeMap` object to use.
`n`	integer; number of items to identify.
`print`	logical; whether to print result on each click.
`outline`	logical; whether to outline rectangles corresponding to identified call.
`...`	Further arguments for the identify methods; currently ignored.

Details

`calleeTreeMap` shows a tree map of the calls in a stack trace's call tree. The tiling algorithm used depends on the `squarify` argument. If `squarify` is `TRUE` then the *squarified* algorithm is used; otherwise, the longer side is partitioned.

`flameGraph` produces a *flame graph* of the call tree. The vertical positions of rectangles represent call depth on the stack. The widths of the rectangles represent the amount of time spent in a call at a particular call or set of calls at a particular depth. The `order` argument determines the ordering of call rectangles at a particular level within a call at the lower level. The `"alpha"` ordering orders the calls alphabetically; `"hot"` uses the hot path ordering with the call with the largest amount of time first. The `"time"` ordering preserves the original time ordering within the stack trace file.

Default colors are based on the `rainbow` palette. Alternative colors can be specified by a `colormap` function. This function is called with three argument vectors, with one element for each rectangle. The arguments are the function name, call depth, and number of hits.

The results returned by `flameGraph` and `calleeTreeMap` can be passed to `identify`. The `identify` method for `calleeTreeMap` returns a list of the call stacks for the identified rectangles. The `identify` method for `flameGraph` returns a character vector of the labels of the identified rectangles.

Value

Objects that can be used with `identify`.

Author(s)

Luke Tierney

See Also

`Rprof`, `summaryRprof`, `flatProfile`, `filterProfileData`, `readProfileData`, `plotProfileCallGraph`, `profileCallGraph2Dot`, `hotPaths`

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
flameGraph(pd)
calleeTreeMap(pd)
```

---

filterProfileData *Filter Profile Data*

Description

Allow profile data to be filtered on several criteria.

Usage

```
filterProfileData(pd, ..., normalize = FALSE, regex = FALSE)
```

Arguments

pd	profile data as returned by readProfileData.
...	filter specifications is filter = value form as described below.
normalize	logical; if true the total hit count is set to the total number of hits in the reduced profile data; otherwise the original value is retained.
regex	logical; if true the specifications in select, and omit, and focus filters are treated as regular expressions; otherwise exact matches are required.

Details

This function can be used to make plots and summaries more readable or relevant by removing functions that are not of direct interest or have low hit counts.

Filters are specified in filter = value form with value typically specifying a filter level or argument. Possible filters and their argument values are:

select character vector specifying names of functions; call stacks not containing functions matching any of these names are dropped.

omit character vector specifying names of functions; call stacks containing functions matching any of these names are dropped.

focus character vector specifying names of functions; call stacks not containing functions matching any of these names are dropped, and functions at the bottom of the stack not matching the focus specification are dropped.

skip integer; the number of elements to trim from the bottom of the stacks.

maxdepth integer; stacks are truncated to have at most maxdepth elements.

self.pct numeric; functions at the bottom of the stacks with self percentages below this value are removed.

total.pct numeric; functions at the top of the stacks with total percentages below this value are removed.

interval inter vector of length 2 specifying first and last sample to use.

merge.pct numeric; functions at the top of the stacks are removed and stack traces merged until each retained trace accounts for at least this percentage of run time.

Value

A reduced profile data structure.

Author(s)

Luke Tierney

See Also

Rprof, summaryRprof, flatProfile, readProfileData, plotProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
plotProfileCallGraph(pd)
plotProfileCallGraph(filterProfileData(pd, self.pct = 1))
plotProfileCallGraph(filterProfileData(pd, self.pct = 1, total.pct = 10))
plotProfileCallGraph(filterProfileData(pd, select = "glm", self.pct=1,
                                       total.pct=10))
```

`flatProfile` *Flat Profile for Rprof Profile Data*

Description

Computes a flat profile reflecting time spent in functions themselves (self) and functions plus callees (total).

Usage

```
flatProfile(pd, byTotal = TRUE, GC = TRUE)
```

Arguments

`pd`	profile data as returned by `readProfileData`.
`byTotal`	logical; sort by total time if true, self time if not.
`GC`	logical; include GC information or not.

Details

If `byTotal` is true then the result is analogous to the `by.total` component of the result returned by `summaryRprof`. Otherwise, the result is analogous to the `by.self` component returned by `summaryRprof` but with an additional cumulative self times column. The result returned when `byTotal` is not true is analogous to the flat profile produced by `gprof`.

Value

A matrix with one row per function recorded in the profile data.

Author(s)

Luke Tierney

References

User manual for `gprof`, the GNU profiler.

See Also

Rprof, summaryRprof, readProfileData, plotProfileCallGraph, printProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
flatProfile(pd)
flatProfile(pd, FALSE)
```

---

hotPaths *Hot Paths in Profile Data*

Description

Computes and displays hot paths in profiling data.

Usage

```
hotPaths(pd, value = c("pct", "time", "hits"), self = TRUE,
         srclines = TRUE, GC = FALSE, memory = FALSE, maxdepth = 10,
         self.pct = 0, total.pct = 0, short = ". ", nlines = NA)
```

Arguments

pd	profile data as returned by readProfileData.
value	character; show result as percentage, time, or hits.
self	logical; include self time for each stack in the result.
srclines	logical; include source information, if available, or not.
GC	logical; include GC information or not.
memory	logical; include memory use information or not.
maxdepth	integer; stacks are truncated to have at most maxdepth elements.
self.pct	numeric; stacks with self percent values below this level are dropped.
total.pct	numeric; stacks with total percent values below this level are dropped.
short	character; abbreviation to be used for functions lower on the stack.
nlines	integer; number of lines to show. The lines shown are the ones with the highest total percentage.

Details

The hot path ordering sorts stacks in the profile data first by the frequency with which the bottom functions on the stack are called, with highest frequency first, then within each bottom function by the frequency of the bottom two, and so on. Examining the result of hotPaths starting with low values of maxdepth and then moving to higher levels is a useful way to expore where the computational effort is concentrated.

Value

A data frame designed to produce a useful printed result.

Author(s)

Luke Tierney

See Also

Rprof, summaryRprof, flatProfile, filterProfileData, readProfileData, plotProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
hotPaths(pd)
hotPaths(pd, maxdepth = 8)
```

---

plot.proftools_profData

Plot Profile Data

Description

Plot method for objects of class "proftools_profData".

Usage

```
## S3 method for class 'proftools_profData'
plot(x, type = c("call", "tree", "flame", "time"),
                                    ...)
```

Arguments

x	an object of class "proftools_profData".
type	the type of plot to be drawn; default is a call graph.
...	additional arguments for specific plot types.

Details

Depending on the type argument the profile data plot method calls link{plotProfileCallGraph}, calleeTreeMap, or flameGraph with order = "hot" or orter = "time".

See Also

plotProfileCallGraph, calleeTreeMap, flameGraph

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
plot(pd)
plot(pd, style = plain.style)
plot(pd, type = "call")
plot(pd, type = "flame")
```

---

plotProfileCallGraph *Plot Call Graph for Rprof Profile Data*

Description

Uses the **graph** and **Rgraphviz** packages to plot a call graph for profile data produced by `Rprof`.

Usage

```
plotProfileCallGraph(pd, layout = "dot",
                     score = c("none", "total", "self"),
                     transfer = function(x) x, nodeColorMap = NULL,
                     edgeColorMap = NULL, mergeCycles = FALSE,
                     edgesColored = FALSE, rankDir = c("TB", "LR"),
                     nodeDetails = FALSE, edgeDetails = FALSE,
                     nodeSizeScore = c("none", "total", "self"),
                     edgeSizeScore = c("none", "total"),
                     shape = "ellipse", style, GC = TRUE,
                     maxnodes = NA, total.pct = 0, ...)
```

Arguments

`pd`	profile data as returned by `readProfileData`.
`layout`	The layout method to use: One of `"dot"`, `"neato"`, and `"twopi"`.
`score`	character string specifying whether to use total time or self time for coloring nodes/edges; no color used if missing.
`transfer`	function; maps score values in unit interval to unit interval
`nodeColorMap,edgeColorMap`	character vectors of color specifications as produced by `rainbow`; `transfer` of score is mapped to color
`mergeCycles`	logical; whether to merge each cycle of recursion into a single node
`edgesColored`	logical; whether to color edges
`rankDir`	The direction that the plot is laid out in, one of either `"TB"` for Top-to-Bottom or `"LR"` for Left-to-Right. The default value is `"LR"`. This argument is only useful for `dot` layouts.
`nodeDetails,edgeDetails`	logical; whether count information should be shown.

nodeSizeScore	character; value to encode in the size of the nodes.
edgeSizeScore	character; value to encode in the width of the edges.
shape	character; node shape.
style	named list of values for arguments score through layout to use if not explicitly supplied.
...	additional arguments for the graphNEL plot method.
GC	logical; include GC information or not.
maxnodes	integer; maximal number of nodes to use; nodes with lower total hit counts are dropped.
total.pct	numeric; if positive, nodes with hit percentages below this level are dropped.

Details

Color is used to encode the fraction of total or self time spent in each function or call. The scores used correspond to the values in the printed representation produced by printProfileCallGraph. For now, see the gprof manual for further details. The color encoding for a score s and a color map m is m[ceiling(length(m) * transfer(s))]

A style can be specified to set options to a set of cvalues that work well together.

Value

Used for side effect.

Note

Because of lazy evaluation, nested calls like f(g(x)) appear in the profile graph as f or one of its callees calling g.

Calls to functions with names containing a | character are dropped as they cause problems in the **graph** package.

Author(s)

Luke Tierney

References

User manual for gprof, the GNU profiler.

Graphviz: https://graphviz.gitlab.io/download/

See Also

Rprof, summaryRprof, readProfileData, flatProfile, profileCallGraph2Dot printProfileCallGraph plain.style google.style

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
plotProfileCallGraph(pd)
plotProfileCallGraph(pd, score = "none")
plotProfileCallGraph(pd, style = plain.style, score = "total")
```

`printProfileCallGraph` *Print Call Graph for Rprof Profile Data*

Description

Prints a representation of the call graph for profile data produced by `Rprof`. Output can be directed to a connection or a file.

Usage

```
printProfileCallGraph(pd, file = stdout(), percent = TRUE, GC = TRUE,
                      maxnodes = NA, total.pct = 0)
```

Arguments

`pd`	profile data as returned by `readProfileData`.
`file`	a connection or the name of the file where the profile graph will be written.
`percent`	logical; if true use percent of total time; otherwise use time in seconds
`GC`	logical; include GC information or not.
`maxnodes`	integer; maximal number of nodes to use; nodes with lower total hit counts are dropped.
`total.pct`	numeric; if positive, nodes with hit percentages below this level are dropped.

Details

`printProfileCallGraph` produces a printed representation of the call graph for profile data produced by `Rprof`. The representation is analogous to the call graph produced by `gprof` with a few minor changes. Eventually more complete documentation of the format will be provided here; for now, reading the `gprof` manual section on the call graph should help understanding this output. The output is similar enough to gprof output for the `cgprof` script to be able to produce a visual representation of the call graph via Graphviz.

Value

Used for side effect.

Note

Because of lazy evaluation, nested calls like `f(g(x))` appear in the profile graph as `f` or one of its callees calling `g`.

Author(s)

Luke Tierney

References

User manual for gprof, the GNU profiler.

cgprof: http://mvertes.free.fr/

Graphviz: http://www.research.att.com/sw/tools/graphviz/

See Also

Rprof, summaryRprof, flatProfile, readProfileData, plotProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
printProfileCallGraph(pd)
## Not run:
## If you have graphviz and cgprof installed on a UNIX-like system
## then in R do:

pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
printProfileCallGraph(pd, "foo.graph")

## and then in a shell do (to use the interactive dotty):

cgprof -TX foo.graph

## or (to create a postscript version and view with gv):

cgprof -Tps foo.graph > foo.ps
gv foo.ps

## End(Not run)
```

---

profileCallGraph2Dot *Write Call Graph for Rprof Profile Data to Graphviz Dot File*

Description

Prints a Graphviz .dot file representation of the call graph for profile data produced by Rprof.

Usage

```
profileCallGraph2Dot(pd, score = c("none", "total", "self"),
                     transfer = function(x) x, nodeColorMap = NULL,
                     edgeColorMap = NULL, filename = "Rprof.dot",
                     landscape = FALSE, mergeCycles = FALSE,
```

```
                edgesColored = FALSE,
                rankDir = c("TB", "LR"),
                nodeDetails = FALSE, edgeDetails = FALSE,
                nodeSizeScore = c("none", "total", "self"),
                edgeSizeScore = c("none", "total"),
                center = FALSE, size, shape = "ellipse",
                layout = "dot", style, GC = TRUE,
                maxnodes = NA, total.pct = 0)
```

Arguments

pd	profile data as returned by readProfileData.
score	character string specifying whether to use total time or self time for coloring nodes/edges; no color used if missing.
transfer	function; maps score values in unit interval to unit interval
nodeColorMap,edgeColorMap	character vectors of color specifications as produced by rainbow; transfer of score is mapped to color
filename	name of .dot file
landscape	logical; whether to add the rotate=90 option to the .dot file
mergeCycles	logical; whether to merge each cycle of recursion into a single node
edgesColored	logical; whether to color edges
rankDir	character; value to use for the rankdir= option to specify the direction that the plot is laid out using the dot layout; must be either "TB" for Top-to-Bottom or "LR" for Left-to-Right. The default value is "LR".
nodeDetails,edgeDetails	logical; whether count information should be shown.
nodeSizeScore	character; value to encode in the size of the nodes.
edgeSizeScore	character; value to encode in the width of the edges.
center	logical; whether to add the center=1 option to the .dot file.
size	character; string to add as size= option in the .dot file.
shape	character; node shape.
layout	character; layout method to use.
style	named list of values for arguments score through layout to use if not explicitly supplied.
GC	logical; include GC information or not.
maxnodes	integer; maximal number of nodes to use; nodes with lower total hit counts are dropped.
total.pct	numeric; if positive, nodes with hit percentages below this level are dropped.

Details

Writes the call graph as a Graphviz `.dot` file. Color is used to encode the fraction total or self time spent in each function or call. The scores used correspond to the values in the printed representation produced by `printProfileCallGraph`. For now, see the `gprof` manual for further details. The color encoding for a score `s` and a color map `m` is `ceiling(length(m) * transfer(s))`

A style can be specified to set options to a set of cvalues that work well together.

Value

Used for side effect.

Note

Because of lazy evaluation, nested calls like `f(g(x))` appear in the profile graph as `f` or one of its callees calling `g`.

Calls to functions with names containing a `|` character are dropped as they cause problems in the **graph** package.

Author(s)

Luke Tierney

References

User manual for `gprof`, the GNU profiler.

Graphviz: https://graphviz.gitlab.io/download/

See Also

`Rprof`, `summaryRprof`, `readProfileData`, `flatProfile`, `plotProfileCallGraph`, `printProfileCallGraph` `plain.style` `google.style`

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
tmp <- tempfile()
profileCallGraph2Dot(pd, filename = tmp)
file.show(tmp)
unlink(tmp)

## Not run:
## If you have graphviz installed on a UNIX-like system
## then in R do:

tmp.dot <- tempfile()
tmp.pdf <- tempfile()

profileCallGraph2Dot(pd, filename = tmp.dot)
system(sprintf("dot -Tpdf -o
browseURL(sprintf("file://
```

```
profileCallGraph2Dot(pd, filename = tmp.dot)
system(sprintf("dot -Tpdf -o
browseURL(sprintf("file://

unlink(tmp.dot)
unlink(tmp.pdf)

## End(Not run)
```

`profileExpr` *Read Rprof Profile Data*

Description

Return profile data collected while evaluating an expression.

Usage

```
profileExpr(expr, GC = TRUE, srclines = TRUE, memory = TRUE)
```

Arguments

`expr`	Valid R expression to be profiled.
`srclines`	logical; include source information, if available, or not.
`GC`	logical; include GC information or not.
`memory`	logical; include memory use information or not.

Details

`profileExpr` uses `Rprof` to profile execution of `expr` and returns the profile data read in using `readProfileData`. By default GC and source information are included in the profile data. If memory profiling is supported, memory use information is also included by default.

Value

R representation of Rprof data.

Author(s)

Luke Tierney

See Also

`Rprof`, `funSummary`, `srcSummary`, `plotProfileCallGraph`,

Examples

```
## This defines the function rw()
source(system.file("samples", "rw.R", package="proftools"))

## Execute the function and collect profile data
pd <- profileExpr(rw(200000))

## Examine the profiel data
funSummary(pd)
callSummary(pd)
hotPaths(pd)
srcSummary(pd)
plotProfileCallGraph(pd)
```

---

readProfileData *Read Rprof Profile Data*

Description

Reads in Rprof profile data for further processing.

Usage

```
readProfileData(filename = "Rprof.out")
readRStudioProfileCacheData()
```

Arguments

filename	Name of a file produced by Rprof()

Details

readProfileData reads the data in the file produced by Rprof into a data structure for processing by other functions. The details of the structure are subject to change.

readRStudioProfileCacheData returns the data for the most recent profiling run in the RStudio profile cache, or NULL if no data is available.

Value

R representation of Rprof data,

Author(s)

Luke Tierney

See Also

Rprof, summaryRprof, flatProfile, plotProfileCallGraph, printProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
flatProfile(pd)
flatProfile(pd, FALSE)
```

`styles` *Style Specifications for Call Graphs*

Description

Styles providing coordinated settings of display parameters for the call graph display functions `plotProfileCallGraph` and `profileCallGraph2Dot`.

Usage

```
plain.style
google.style
```

Details

The `plain.style` style corresponds to the default parameter settings in the display functions. It can be used as the basis for creating a new custom style.

The `google.style` style is based on the display style used in the `pprof` tool from the Google Performance Tools suite.

Value

A list containing the following components:

`layout`	The layout method to use: One of `"dot"`, `"neato"`, and `"twopi"`.
`score`	character string specifying whether to use total time or self time for coloring nodes/edges; no color used if missing.
`transfer`	function; maps score values in unit interval to unit interval
`nodeColorMap,edgeColorMap`	character vectors of color specifications as produced by `rainbow`; `transfer` of score is mapped to color
`mergeCycles`	logical; whether to merge each cycle of recursion into a single node
`edgesColored`	logical; whether to color edges
`rankDir`	The direction that the plot is laid out in, one of either `"TB"` for Top-to-Bottom or `"LR"` for Left-to-Right. The default value is `"LR"`. This argument is only useful for `dot` layouts.
`nodeDetails,edgeDetails`	logical; whether count information should be shown.
`nodeSizeScore`	character; value to encode in the size of the nodes.

edgeSizeScore	character; value to encode in the width of the edges.
shape	character; node shape.
maxnodes	integer; maximal number of nodes to use; nodes with lower total hit counts are dropped.
total.pct	numeric; if positive, nodes with hit percentages below this level are dropped.

Author(s)

Luke Tierney

References

https://gperftools.github.io/gperftools/cpuprofile.html.

See Also

Rprof, flatProfile, summaryRprof, readProfileData, plotProfileCallGraph, printProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
plotProfileCallGraph(pd, style = plain.style)
plotProfileCallGraph(pd, style = google.style)
```

---

summaries *Basic Profile Data Summaries*

Description

Functions to summarize profile data.

Usage

```
funSummary(pd, byTotal = TRUE, value = c("pct", "time", "hits"),
         srclines = TRUE, GC = TRUE, memory = FALSE, self.pct = 0, total.pct = 0)
callSummary(pd, byTotal = TRUE, value = c("pct", "time", "hits"),
          srclines = TRUE, GC = TRUE, memory = FALSE, self.pct = 0, total.pct = 0)
pathSummary(pd, value = c("pct", "time", "hits"),
            srclines = FALSE, GC = TRUE, memory = FALSE, total.pct = 0, ...)
srcSummary(pd, byTotal = TRUE, value = c("pct", "time", "hits"),
           GC = TRUE, memory = FALSE, total.pct = 0, source = TRUE,
           width = getOption("width"))
```

Arguments

pd	profile data as returned by readProfileData.
byTotal	logical; sort by total or by self time.
value	character; show result as percentage, time, or hits.
srclines	logical; include source information, if available, or not.
GC	logical; include GC information or not.
memory	logical; include memory use information or not.
self.pct	numeric; functions at the bottom of the stacks with self percentages below this value are removed.
total.pct	numeric; functions at the top of the stacks with total percentages below this value are removed.
source	logical; if true, include source lines if available.
width	maximal line length to use; source lines are abbreviated to fit if necessary.
...	arguments to control path formatting; not useful at this time.

Details

funSummary returns a summary of the time spent in each function, or each call site if source information is available and requested. It is similar to flatProfile.

callSummary provides a breakdown by calls, again with an option of distinguishing call and callee sites if source information is available.

pathSummary returns a summary of time spent in each unique call path contained in the profile data.

For profile data containing source information srcSummary returns a summary of time spent in each file line recorded in the profile data.

Value

A data frame of summary information.

Author(s)

Luke Tierney

See Also

Rprof, flatProfile, summaryRprof, readProfileData, plotProfileCallGraph, printProfileCallGraph, profileCallGraph2Dot

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
funSummary(pd)
callSummary(pd)
pathSummary(pd)
```

utils *Profile Data Utilities*

Description

Utility functions for developing new profile data analysis tools.

Usage

```
profileDataCycles(pd, GC)
```

Arguments

pd	profile data as returned by readProfileData.
GC	logical; include GC information or not.

Details

These functions are experimental and subject to change.

Author(s)

Luke Tierney

writeCallgrindFile *Write Out Profile Data in Callgrind Format*

Description

Writes the profile data in callgrind format suitable for use with kcachegrind or qcachegrind.

Usage

```
writeCallgrindFile(pd, file = "Rprof.cg", GC = TRUE, dropSource = TRUE)
```

Arguments

pd	profile data as returned by readProfileData.
file	a connection or the name of the file where the callgrind output will be written.
GC	logical; if true include GC information.
dropSource	logical; if true drop initial stack entried from a source call and add a top level marker.

Details

The callgrind format is used by Valgrind's callgrind tool. The KDE tool `kcachegrind` can be used to display the file; `kcachegrind` displays the summary statistics, a call graph, and annotated source code if source information is available. `kcachegrind` is available in Linux and Windows; on Mac OSX `qcachegrind` is available.

Value

Used for side effect.

Author(s)

Luke Tierney

See Also

`Rprof`, `summaryRprof`, `flatProfile`, `readProfileData`, `plotProfileCallGraph`, `profileCallGraph2Dot`

Examples

```
pd <- readProfileData(system.file("samples", "glmEx.out", package="proftools"))
tmp <- tempfile()
writeCallgrindFile(pd, file = tmp)
file.show(tmp)
unlink(tmp)
## Not run:
## If you have kcachegrind installed on a UNIX-like system then do:
tmp <- tempfile()
writeCallgrindFile(pd, file = tmp)
system(sprintf("kcachegrind
unlink(tmp)

## End(Not run)
```

Index