R PROFILING AND OPTIMISATION

GÜNTHER SAWITZKI

Contents

Profiling facilitites in R

R provides the basic instruments for profiling, both for time based samplers as for event based intrumentation. However this source of information seems to be rarely used.

Maybe the supporting tools are not adequate. The summaries in R itself reduce the information beyond necessity. The data structures in additional packages are not sufficiently action oriented.

With package ***sprof*** we want to give a data representation that keeps the full profile information. Tools to answer common questions are provided. The data structure should make it easy to extend the tools as required.

The package is currently distributed at r-forge as part of the sintro material.

Date: Typeset: July 2, 2013.
Key words and phrases. R programming, profiling, optimisation, R program language.
An R *vignette for package sprof. Private Version*
.

To install this package directly within R , type
`install.packages("sprof",repos="http://r-forge.r-project.org")`

To install the recent package from source directly within R , type
`install.packages("sprof",repos="http://r-forge.r-project.org",type="source")`

LaTeX Layout Tools and Utilities

Print parameters used here:

```
Input
options(width = 72)
options(digits = 6)
```

We use the R function *xtable()* for output and LaTeX *longtable*. A convenient wrapper to use this in out *Sweave* source is:

```
Input
library(xtable)
prxt <- function (x, digits=3,         caption=NULL,
        label=NULL, ...)  {
        print(
                xtable(x, digits=digits, caption=caption, label=label, ...),
                floating=FALSE, tabular.environment="longtable")
        }
```

This is to be used with `<<print = FALSE, results = tex, label=tab:prxxxx>>=`

1. Profiling

The basic information provided by all profilers in is a protocol of sampled stacks. For each recorded event, the protocol has one record , such as a line with a text string showing the sampled stack.

We use profiles to provide hints on the dynamic behaviour of programs. Most often, this is used to improve or even optimise programs. Sometimes, it is even used to understand some algorithm.

Profiles represent the program flow, which is considered to be laid out by the control structure of a program. The control structure is represented by the control graph, and this leads to the common approach to (re)construct the control graph, map the profile to this graph, and used graph based methods for further analysis. The prime example for this strategy is the GNU profiler ircodegprof (see `http://sourceware.org/binutils/docs/gprof/`) which is used as master plan for many common profilers.

It is only half of the truth that the control graph can serve as a base for the profiled stacks. The fine issues will be discussed below. But for now, we stay with this simplification.

We take a different approach. We take the stacks, as recorded in the profiles as our basic information unit. From this, we ask: what are the actions we need to answer our questions? Representation in graphs may come later, if they can help.

```
Input
options(error = recover)
library(sprof)
```

In this presentation, we will use a small list of examples Since *Rprof* is not implemented on all systems, and since the profiles tend to get very large, we use some prepared examples that are frozen in this vignette and not included in the distribution, but all the code to generate the examples is provided.

1.1. Simple regression example.

Input
```
n <- 10000
x <- runif(n)
err <- rnorm(n)
y <- 2+ 3 * x + err
reg0data <- data.frame(x=x, y=y, err=err)
rm(x,y,err)
```

We will use this example to illustrate the basics. Of course the immediate questions are the variance between varying samples, and the influence of the sample size n. We keep everything fixed, so the only issue for now is the computational performance under strict iid conditions.

Still we have parameters to choose. We can determine the profiling granularity by setting the timing interval, and we can use repeated measurements to increase precision below the timing interval.

The timing interval should depend on the clock speed. Using for example 1ms amounts to some 1000 steps on a current CPU, per kernel.

If we use repeated samples, the usual rules of statistics applies. So taking 100 runs and taking the mean reduces the standard deviation by a factor 1/10.

Input
```
profinterval <- 0.001
simruns <- 100
Rprof(filename="RprofsRegressionExpl.out", interval = profinterval)
for (i in 1:simruns) xxx<- summary(lm(y~x, data=reg0data))
Rprof(NULL)
```

We now have the profile data in a file *RprofsRegressionExpl.out*. For this vignette, we use a frozen version*RprofsRegressionExpl01.out*.

The basic R invites us to get a summary.

Input
```
sumRprofRegressionExpl <- summaryRprof("RprofsRegressionExpl01.out")
str(sumRprofRegressionExpl)
```

Output
```
List of 4
 $ by.self        :'data.frame':        41 obs. of  4 variables:
  ..$ self.time : num [1:41] 0.087 0.057 0.051 0.043 0.042 0.04 0.032 0.026 0.022 0.022 ...
  ..$ self.pct  : num [1:41] 16.67 10.92 9.77 8.24 8.05 ...
  ..$ total.time: num [1:41] 0.113 0.099 0.069 0.043 0.474 0.045 0.033 0.114 0.022 0.022 ...
  ..$ total.pct : num [1:41] 21.65 18.97 13.22 8.24 90.8 ...
 $ by.total       :'data.frame':        62 obs. of  4 variables:
  ..$ total.time: num [1:62] 0.522 0.522 0.521 0.521 0.521 0.521 0.521 0.521 0.521 0.521 ...
  ..$ total.pct : num [1:62] 100 100 99.8 99.8 99.8 ...
  ..$ self.time : num [1:62] 0.006 0 0.001 0 0 0 0 0 0 0 ...
  ..$ self.pct  : num [1:62] 1.15 0 0.19 0 0 0 0 0 0 0 ...
 $ sample.interval: num 0.001
 $ sampling.time  : num 0.522
```

The summary reduces the information contained in the profile to marginal statistics per node.

The file contains several spurious recordings: nodes that have been recorded only few times. It is worth noting these, but then they better be discarded. We use a time limit of 4ms, which given our sampling interval of 1ms means we require more than four observations.

```
                              Input
prxt(sumRprofRegressionExpl$by.self,
        caption="by.self as final stack entry, all records",
        label="tab:prSRREbs")
```

	self.time	self.pct	total.time	total.pct
"lm.fit"	0.087	16.670	0.113	21.650
"[.data.frame"	0.057	10.920	0.099	18.970
"model.matrix.default"	0.051	9.770	0.069	13.220
"as.character"	0.043	8.240	0.043	8.240
"lm"	0.042	8.050	0.474	90.800
"summary.lm"	0.040	7.660	0.045	8.620
"structure"	0.032	6.130	0.033	6.320
"na.omit.data.frame"	0.026	4.980	0.114	21.840
"anyDuplicated.default"	0.022	4.210	0.022	4.210
"as.list.data.frame"	0.022	4.210	0.022	4.210
"na.omit"	0.020	3.830	0.134	25.670
"model.response"	0.013	2.490	0.056	10.730
"model.frame.default"	0.012	2.300	0.168	32.180
"rep.int"	0.007	1.340	0.007	1.340
"<Anonymous>"	0.006	1.150	0.522	100.000
"list"	0.005	0.960	0.005	0.960
"vapply"	0.003	0.570	0.023	4.410
"unique"	0.003	0.570	0.004	0.770
"lapply"	0.002	0.380	0.030	5.750
".deparseOpts"	0.002	0.380	0.004	0.770
"lazyLoadDBfetch"	0.002	0.380	0.003	0.570
"!"	0.002	0.380	0.002	0.380
"mean.default"	0.002	0.380	0.002	0.380
"mode"	0.002	0.380	0.002	0.380
"names"	0.002	0.380	0.002	0.380
"pmatch"	0.002	0.380	0.002	0.380
"eval"	0.001	0.190	0.521	99.810
"anyDuplicated"	0.001	0.190	0.023	4.410
"sapply"	0.001	0.190	0.014	2.680
"match"	0.001	0.190	0.011	2.110
"[[.data.frame"	0.001	0.190	0.008	1.530
"FUN"	0.001	0.190	0.007	1.340
"%in%"	0.001	0.190	0.004	0.770
"deparse"	0.001	0.190	0.002	0.380
"$"	0.001	0.190	0.001	0.190
"as.list.default"	0.001	0.190	0.001	0.190
"as.name"	0.001	0.190	0.001	0.190
"coef"	0.001	0.190	0.001	0.190
"file"	0.001	0.190	0.001	0.190
"NCOL"	0.001	0.190	0.001	0.190
"terms.formula"	0.001	0.190	0.001	0.190

Table 1: by.self as final stack entry, all records

Input

```
prxt(sumRprofRegressionExpl$by.total[sumRprofRegressionExpl$by.total$total.time>0.004,],
        caption="by.total, total time > 0.004s",
        label="tab:prSRREbt")
```

	total.time	total.pct	self.time	self.pct
"<Anonymous>"	0.522	100.000	0.006	1.150
"Sweave"	0.522	100.000	0.000	0.000
"eval"	0.521	99.810	0.001	0.190
"doTryCatch"	0.521	99.810	0.000	0.000
"evalFunc"	0.521	99.810	0.000	0.000
"try"	0.521	99.810	0.000	0.000
"tryCatch"	0.521	99.810	0.000	0.000
"tryCatchList"	0.521	99.810	0.000	0.000
"tryCatchOne"	0.521	99.810	0.000	0.000
"withVisible"	0.521	99.810	0.000	0.000
"summary"	0.520	99.620	0.000	0.000
"lm"	0.474	90.800	0.042	8.050
"model.frame.default"	0.168	32.180	0.012	2.300
"model.frame"	0.168	32.180	0.000	0.000
"na.omit"	0.134	25.670	0.020	3.830
"na.omit.data.frame"	0.114	21.840	0.026	4.980
"lm.fit"	0.113	21.650	0.087	16.670
"[.data.frame"	0.099	18.970	0.057	10.920
"["	0.099	18.970	0.000	0.000
"model.matrix.default"	0.069	13.220	0.051	9.770
"model.matrix"	0.069	13.220	0.000	0.000
"model.response"	0.056	10.730	0.013	2.490
"summary.lm"	0.045	8.620	0.040	7.660
"as.character"	0.043	8.240	0.043	8.240
"structure"	0.033	6.320	0.032	6.130
"lapply"	0.030	5.750	0.002	0.380
".getXlevels"	0.026	4.980	0.000	0.000
"vapply"	0.023	4.410	0.003	0.570
"anyDuplicated"	0.023	4.410	0.001	0.190
"as.list"	0.023	4.410	0.000	0.000
"anyDuplicated.default"	0.022	4.210	0.022	4.210
"as.list.data.frame"	0.022	4.210	0.022	4.210
"sapply"	0.014	2.680	0.001	0.190
"match"	0.011	2.110	0.001	0.190
"[[.data.frame"	0.008	1.530	0.001	0.190
"[["	0.008	1.530	0.000	0.000
"rep.int"	0.007	1.340	0.007	1.340
"FUN"	0.007	1.340	0.001	0.190
"list"	0.005	0.960	0.005	0.960

Table 2: by.total, total time > 0.004s

In contrast, in our implementation we take a two step approach. First we read in the profile file to an internal representation.

Input
```
sprofRegressionExpl <- readRprof("RprofsRegressionExpl01.out")
str(sprofRegressionExpl, max.level=2)
```

Output
```
List of 4
 $ info    :'data.frame':       1 obs. of  8 variables:
  ..$ id       : Factor w/ 1 level "\"RprofsRegressionExpl01.out\" 2013-06-13 23:46:04": 1
  ..$ date     : POSIXct[1:1], format: "2013-07-02 17:30:21"
  ..$ nrnodes  : int 62
  ..$ nrstacks : int 50
  ..$ nrrecords: int 522
  ..$ firstline: Factor w/ 1 level "sample.interval=1000": 1
  ..$ ctllines : Factor w/ 1 level "sample.interval=1000": 1
  ..$ ctllinenr: num 1
 $ nodes   :'data.frame':       62 obs. of  5 variables:
  ..$ name      : Factor w/ 62 levels "!","..getNamespace",..: 1 2 3 4 5 6 7 8 9 10 ...
  ..$ self.time : num [1:62] 2 0 2 0 0 57 0 1 1 6 ...
  ..$ self.pct  : num [1:62] 0.38 0 0.38 0 0 ...
  ..$ total.time: num [1:62] 2 1 4 26 99 99 8 8 4 522 ...
  ..$ total.pct : num [1:62] 0.03 0.01 0.05 0.34 1.29 1.29 0.1 0.1 0.05 6.79 ...
 $ stacks  :'data.frame':       50 obs. of  7 variables:
  ..$ nodes         :List of 50
  ..$ shortname     : Factor w/ 50 levels "S<A>eFttCtCLtCOdTCwVeesleem.m..n.n...[[.",..: 27 17 19 1 35
  ..$ refcount      : num [1:50] 1 5 26 55 13 43 51 87 1 15 ...
  ..$ stacklength   : int [1:50] 19 20 19 21 14 15 15 14 15 18 ...
  ..$ stackheadnodes: int [1:50] 52 52 52 52 52 52 52 52 52 52 ...
  ..$ stackleafnodes: int [1:50] 27 28 41 6 39 14 38 30 27 49 ...
  ..$ stackssrc     : Factor w/ 50 levels "! [.data.frame [ na.omit.data.frame na.omit model.frame.defa
 $ profiles:List of 4
  ..$ data    : int [1:522] 1 2 2 3 4 4 5 5 6 7 ...
  ..$ mem     : NULL
  ..$ malloc  : NULL
  ..$ timesRLE:List of 2
  .. ..- attr(*, "class")= chr "rle"
 - attr(*, "class")= chr [1:2] "sprof" "list"
```

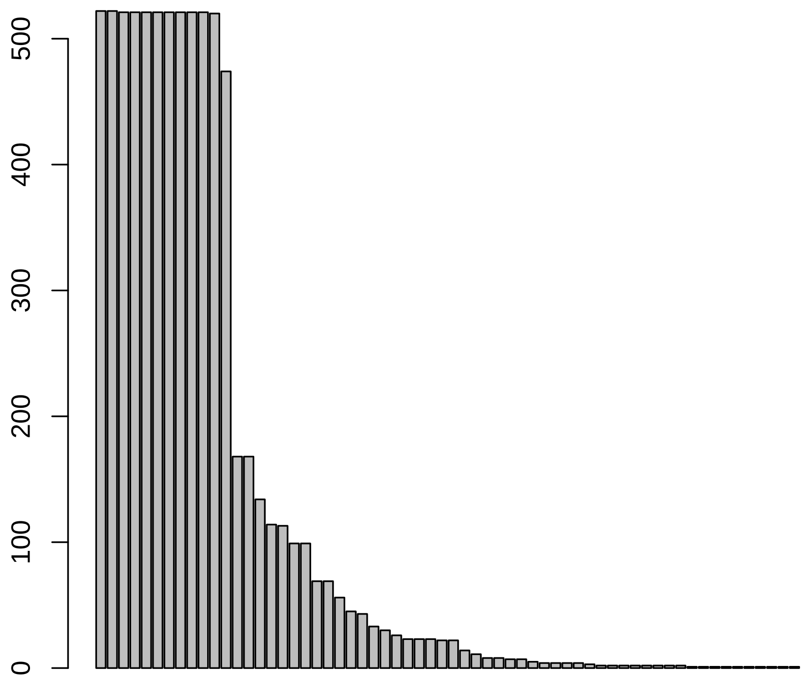
Input
```
# xtable cannot handle posix
str(sprofRegressionExpl$info,
        caption="info",
        label="tab:prSREinfo")
```

Output
```
'data.frame':       1 obs. of  8 variables:
 $ id       : Factor w/ 1 level "\"RprofsRegressionExpl01.out\" 2013-06-13 23:46:04": 1
 $ date     : POSIXct, format: "2013-07-02 17:30:21"
 $ nrnodes  : int 62
 $ nrstacks : int 50
 $ nrrecords: int 522
 $ firstline: Factor w/ 1 level "sample.interval=1000": 1
 $ ctllines : Factor w/ 1 level "sample.interval=1000": 1
 $ ctllinenr: num 1
```

As a convention, we do not re-arrange items for ad-hoc choices, but use a permutation vector instead.

Input

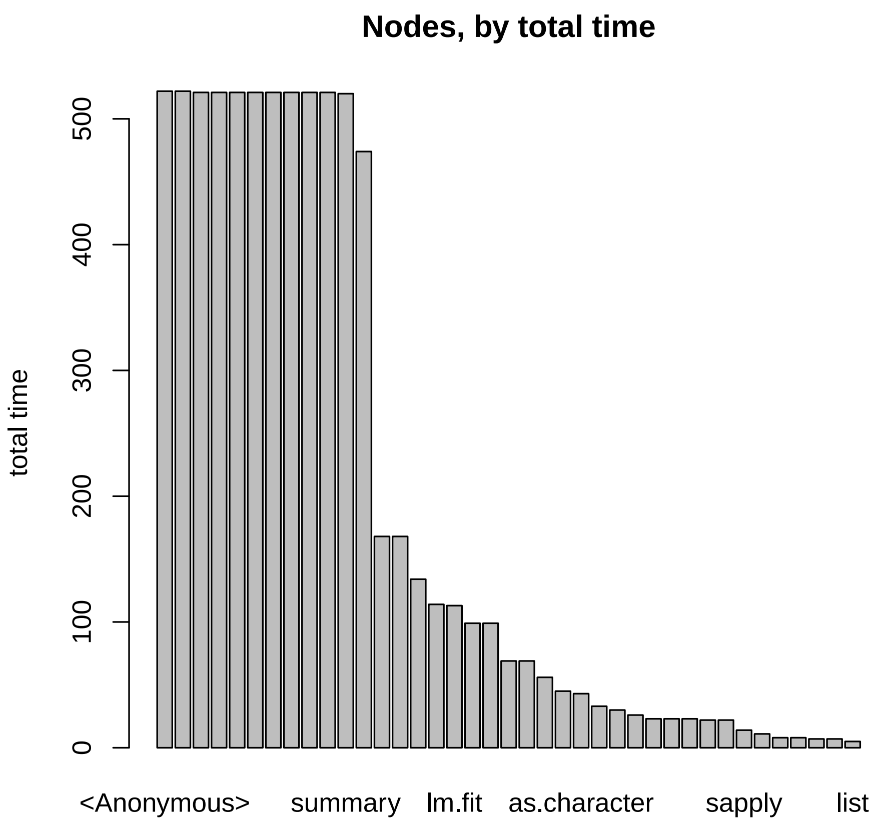
```
rownames(sprofRegressionExpl$nodes) <- sprofRegressionExpl$nodes$names
nodesperm <- order(sprofRegressionExpl$nodes$total.time,decreasing=TRUE)
barplot(sprofRegressionExpl$nodes$total.time[nodesperm])
```



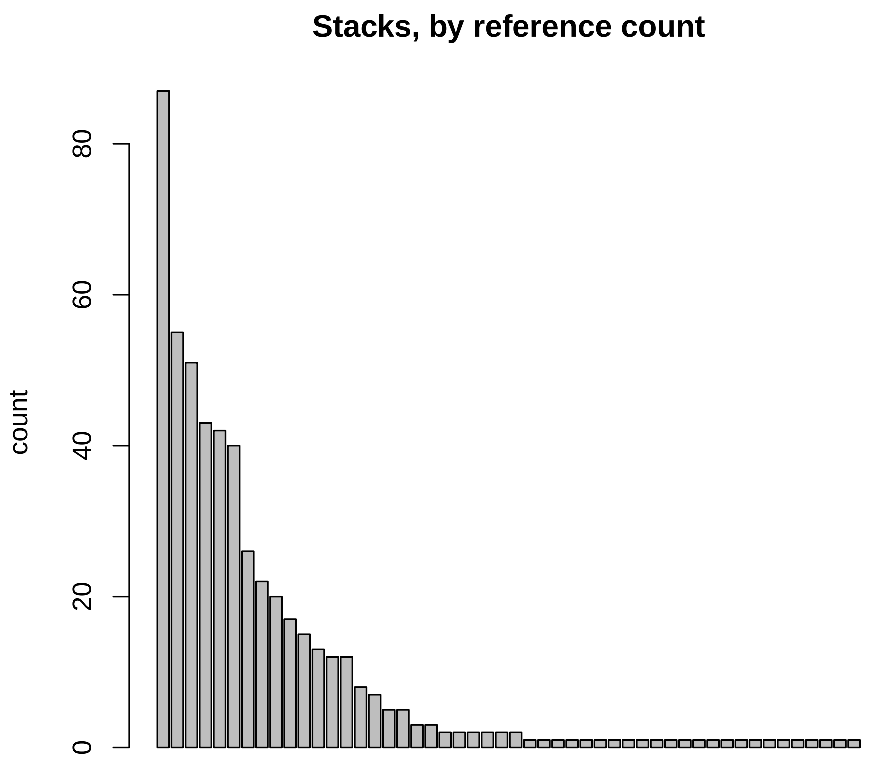
Selections are recorded as selection vectors, with reference to the original order. This needs some caution to align them with the order choices.

Input

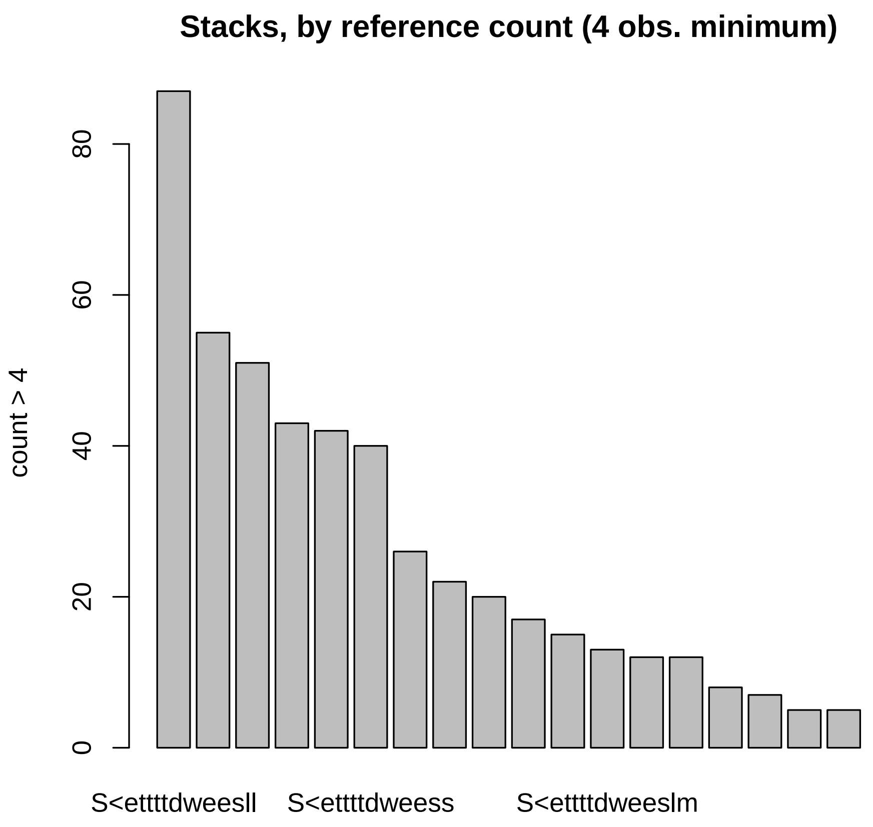
```
nodesnrobsok <- sprofRegressionExpl$nodes$total.time > 4
sp <- sprofRegressionExpl$nodes$total.time[nodesperm][nodesnrobsok[nodesperm]]
names(sp) <- sprofRegressionExpl$nodes$name[nodesperm][nodesnrobsok[nodesperm]]
barplot(sp,
 main="Nodes, by total time", ylab="total time")
```



Input
```
#rownames(sprofRegressionExpl$nodes) <- sprofRegressionExpl$nodes$names
stacksperm <- order(sprofRegressionExpl$stacks$refcount,decreasing=TRUE)
barplot(sprofRegressionExpl$stacks$refcount[stacksperm],main="Stacks, by reference count", ylab="count
```

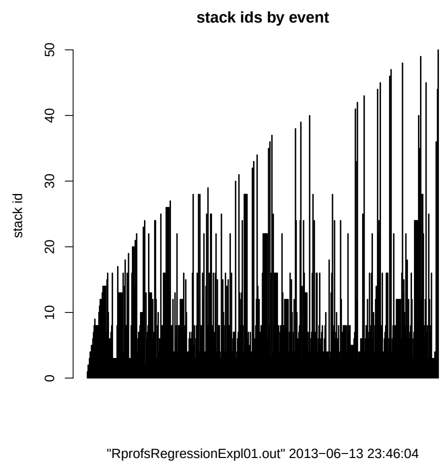


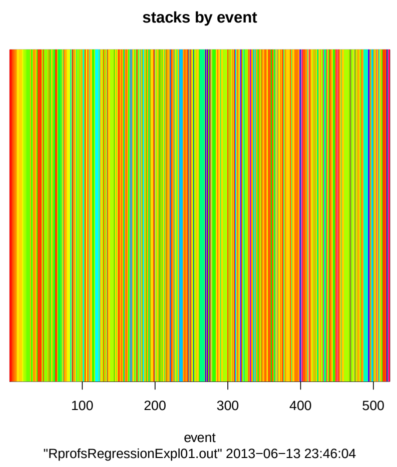
Input
```
stacksnrobsok <- sprofRegressionExpl$stacks$refcount > 4
sp4 <- sprofRegressionExpl$stacks$refcount[stacksperm][stacksnrobsok[stacksperm]]
names(sp4) <- sprofRegressionExpl$stacks$shortname[stacksperm][stacksnrobsok[stacksperm]]
barplot(sp4,
 main="Stacks, by reference count (4 obs. minimum)", ylab="count > 4")
```



On the first look, information on the profile level is not informative. Profile records are just recordings of some step, taken at regular intervals. We get a minimal information, if we encode the stacks in colour.

Input
```
par(mfrow=c(1,2))
plot_profiles(sprofRegressionExpl)
```

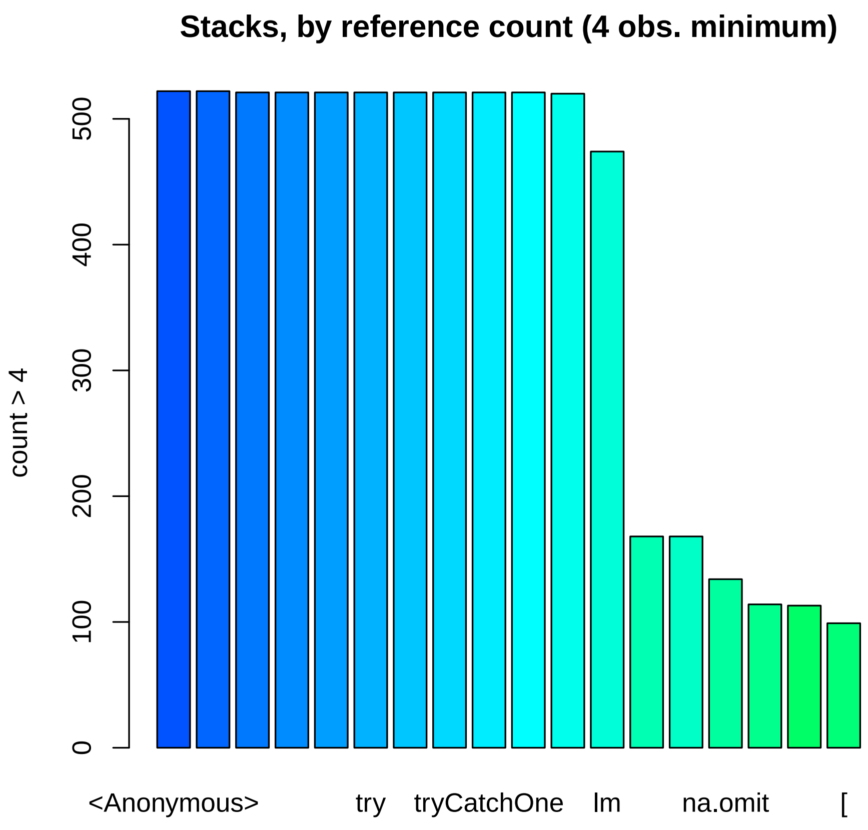




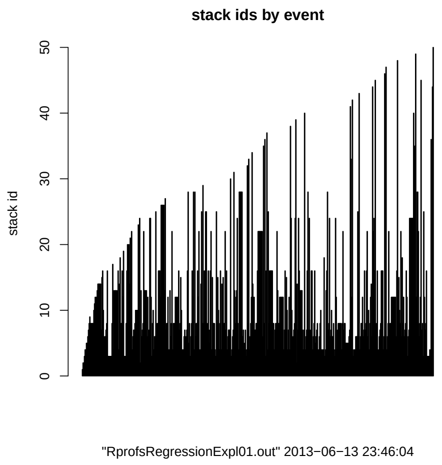
We now do a step down analysis. Agregating the information from the profilingn events, we have the frequency of stack references. On the stack level, we encode the frequncy in colour, and linking propagates this to tne profile level.

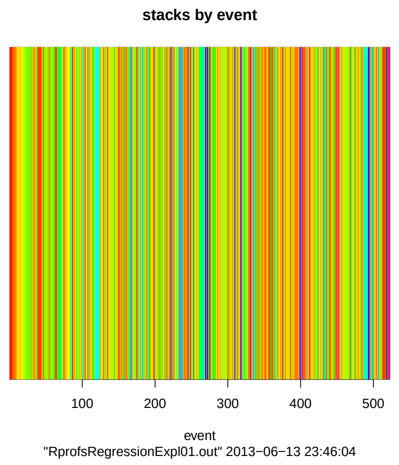
Input

```
stackfreqscore <- rank(sprofRegressionExpl$stacks$refcount,ties.method="random")
stackfreqscore4<- stackfreqscore[stacksperm][stacksnrobsok[stacksperm]]
barplot(sp[stacksnrobsok[stacksperm]], main="Stacks, by reference count (4 obs. minimum)", ylab="count
col=rainbow(80)[stackfreqscore4])
```



Input
```
par(mfrow=c(1,2))
plot_profiles(sprofRegressionExpl)
```


stack ids by event
stack id
0
10
20
30
40
50
"RprofsRegressionExpl01.out" 2013−06−13 23:46:04


stacks by event
100
200
300
400
500
event
"RprofsRegressionExpl01.out" 2013−06−13 23:46:04

Input
```
prxt(sprofRegressionExpl$nodes,
        caption="nodes",
        label="tab:prSREnodes")
```

	name	self.time	self.pct	total.time	total.pct
1	!	2.000	0.380	2.000	0.030
2	..getNamespace	0.000	0.000	1.000	0.010
3	.deparseOpts	2.000	0.380	4.000	0.050
4	.getXlevels	0.000	0.000	26.000	0.340
5	[	0.000	0.000	99.000	1.290
6	[.data.frame	57.000	10.920	99.000	1.290
7	[[	0.000	0.000	8.000	0.100
8	[[.data.frame	1.000	0.190	8.000	0.100
9	%in%	1.000	0.190	4.000	0.050
10	<Anonymous>	6.000	1.150	522.000	6.790
11	$	1.000	0.190	1.000	0.010
12	anyDuplicated	1.000	0.190	23.000	0.300
13	anyDuplicated.default	22.000	4.210	22.000	0.290
14	as.character	43.000	8.240	43.000	0.560
15	as.list	0.000	0.000	23.000	0.300
16	as.list.data.frame	22.000	4.210	22.000	0.290
17	as.list.default	1.000	0.190	1.000	0.010
18	as.name	1.000	0.190	1.000	0.010
19	coef	1.000	0.190	1.000	0.010
20	deparse	1.000	0.190	2.000	0.030
21	doTryCatch	0.000	0.000	521.000	6.780
22	eval	1.000	0.190	521.000	6.780
23	evalFunc	0.000	0.000	521.000	6.780
24	file	1.000	0.190	1.000	0.010
25	FUN	1.000	0.190	7.000	0.090
26	lapply	2.000	0.380	30.000	0.390
27	lazyLoadDBfetch	2.000	0.380	3.000	0.040
28	list	5.000	0.960	5.000	0.070
29	lm	42.000	8.050	474.000	6.160
30	lm.fit	87.000	16.670	113.000	1.470
31	match	1.000	0.190	11.000	0.140
32	mean	0.000	0.000	2.000	0.030
33	mean.default	2.000	0.380	2.000	0.030
34	mode	2.000	0.380	2.000	0.030
35	model.frame	0.000	0.000	168.000	2.180
36	model.frame.default	12.000	2.300	168.000	2.180
37	model.matrix	0.000	0.000	69.000	0.900
38	model.matrix.default	51.000	9.770	69.000	0.900
39	model.response	13.000	2.490	56.000	0.730
40	na.omit	20.000	3.830	134.000	1.740
41	na.omit.data.frame	26.000	4.980	114.000	1.480
42	names	2.000	0.380	2.000	0.030
43	NCOL	1.000	0.190	1.000	0.010

44	paste	0.000	0.000	1.000	0.010
45	pmatch	2.000	0.380	2.000	0.030
46	rep.int	7.000	1.340	7.000	0.090
47	sapply	1.000	0.190	14.000	0.180
48	simplify2array	0.000	0.000	4.000	0.050
49	structure	32.000	6.130	33.000	0.430
50	summary	0.000	0.000	520.000	6.760
51	summary.lm	40.000	7.660	45.000	0.590
52	Sweave	0.000	0.000	522.000	6.790
53	terms	0.000	0.000	2.000	0.030
54	terms.formula	1.000	0.190	1.000	0.010
55	try	0.000	0.000	521.000	6.780
56	tryCatch	0.000	0.000	521.000	6.780
57	tryCatchList	0.000	0.000	521.000	6.780
58	tryCatchOne	0.000	0.000	521.000	6.780
59	unique	3.000	0.570	4.000	0.050
60	unlist	0.000	0.000	1.000	0.010
61	vapply	3.000	0.570	23.000	0.300
62	withVisible	0.000	0.000	521.000	6.780

Table 3: nodes

Input
```
str(sprofRegressionExpl$stacks, max.level=1)
```

Output
```
'data.frame':       50 obs. of  7 variables:
 $ nodes         :List of 50
 $ shortname     : Factor w/ 50 levels "S<A>eFttCtCLtCOdTCwVeesleem.m..n.n...[[.",..: 27 17 19 1 35 36 3
 $ refcount      : num  1 5 26 55 13 43 51 87 1 15 ...
 $ stacklength   : int  19 20 19 21 14 15 15 14 15 18 ...
 $ stackheadnodes: int  52 52 52 52 52 52 52 52 52 52 ...
 $ stackleafnodes: int  27 28 41 6 39 14 38 30 27 49 ...
 $ stackssrc     : Factor w/ 50 levels "! [.data.frame [ na.omit.data.frame na.omit model.frame.default
```

Input

Input
```
str(sprofRegressionExpl$profiles, max.level=1)
```

Output
```
List of 4
 $ data    : int [1:522] 1 2 2 3 4 4 5 5 6 7 ...
 $ mem     : NULL
 $ malloc  : NULL
 $ timesRLE:List of 2
  ..- attr(*, "class")= chr "rle"
```

Input

A summary is provided on request.

Input
```
sumsprofRegressionExpl <- summary.sprof(sprofRegressionExpl)
str(sumsprofRegressionExpl, max.level=2)
```

Output
```
List of 4
 $ info    :'data.frame':       1 obs. of  8 variables:
  ..$ id       : Factor w/ 1 level "\"RprofsRegressionExpl01.out\" 2013-06-13 23:46:04": 1
  ..$ date     : POSIXct[1:1], format: "2013-07-02 17:30:21"
  ..$ nrnodes  : int 62
  ..$ nrstacks : int 50
  ..$ nrrecords: int 522
  ..$ firstline: Factor w/ 1 level "sample.interval=1000": 1
  ..$ ctllines : Factor w/ 1 level "sample.interval=1000": 1
  ..$ ctllinenr: num 1
 $ nodes   :'data.frame':       62 obs. of  5 variables:
  ..$ name      : Factor w/ 62 levels "!","..getNamespace",..: 1 2 3 4 5 6 7 8 9 10 ...
  ..$ self.time : num [1:62] 2 0 2 0 0 57 0 1 1 6 ...
  ..$ self.pct  : num [1:62] 0.38 0 0.38 0 0 ...
  ..$ total.time: num [1:62] 2 1 4 26 99 99 8 8 4 522 ...
  ..$ total.pct : num [1:62] 0.03 0.01 0.05 0.34 1.29 1.29 0.1 0.1 0.05 6.79 ...
 $ stacks  :'data.frame':       50 obs. of  7 variables:
  ..$ nodes         :List of 50
  ..$ shortname     : Factor w/ 50 levels "S<A>eFttCtCLtCOdTCwVeesleem.m..n.n...[[.",..: 27 17 19 1 35 3
  ..$ refcount      : num [1:50] 1 5 26 55 13 43 51 87 1 15 ...
  ..$ stacklength   : int [1:50] 19 20 19 21 14 15 15 14 15 18 ...
  ..$ stackheadnodes: int [1:50] 52 52 52 52 52 52 52 52 52 52 ...
  ..$ stackleafnodes: int [1:50] 27 28 41 6 39 14 38 30 27 49 ...
  ..$ stackssrc     : Factor w/ 50 levels "! [.data.frame [ na.omit.data.frame na.omit model.frame.defau
 $ profiles:List of 4
  ..$ data    : int [1:522] 1 2 2 3 4 4 5 5 6 7 ...
  ..$ mem     : NULL
  ..$ malloc  : NULL
  ..$ timesRLE:List of 2
  .. ..- attr(*, "class")= chr "rle"
 - attr(*, "class")= chr [1:2] "sprof" "list"
```

The classical approach hides the work that has been done. Actually it breaks down the data to record items. This figure is not reported anywhere. In our case, it can be reconstructed. The profile data have 8456 words in 524 lines.

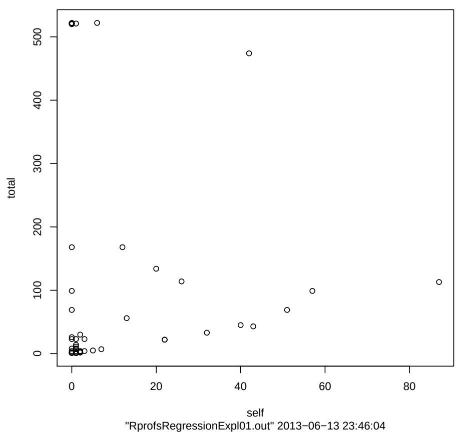
In our approach, we break down the information. Two lines of control information are split off. We have 522 lines of profile with 50 unique stacks, referencing 62 nodes. Instead of reducing it to a summary, we keep the full information. Information is always kept on its original level.

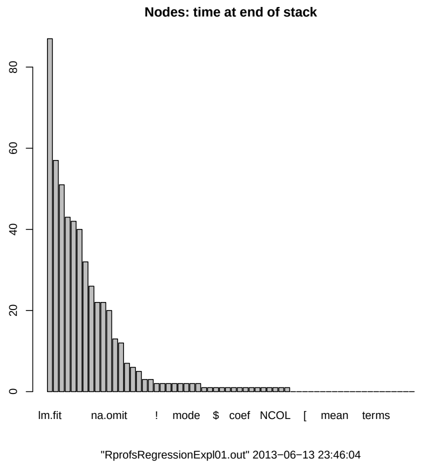
On the profiles level, we know the sample interval length, and the id of the stack recorded. On the stack level, for each stack we have a reference count, with the sample interval lengths used as weights. This reference count is added up for each node in the stack to give the node timings.

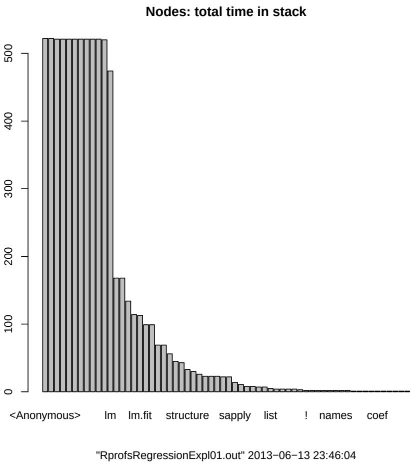
Cheap statistics are collected as the come by. For example, from the stacks table it is cheap to identify root and leaf nodes, and this mark is propagated to the nodes table.

1.1.1. *Plot.* Looking at lists of numbers is not too informative. We get a first impression by plotting the data.

Input
```
#plot_nodes(sprofRegressionExpl, col=nodescol[nodescore])
par(mfrow=c(2,2))
plot_nodes(sprofRegressionExpl)
```

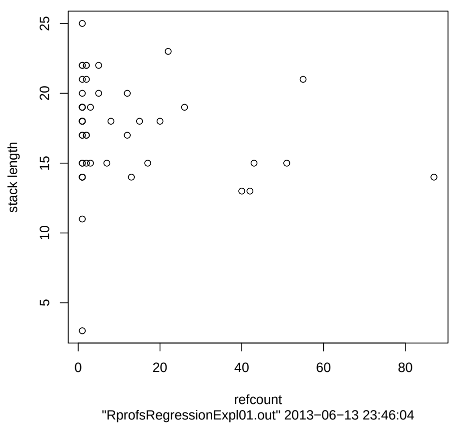


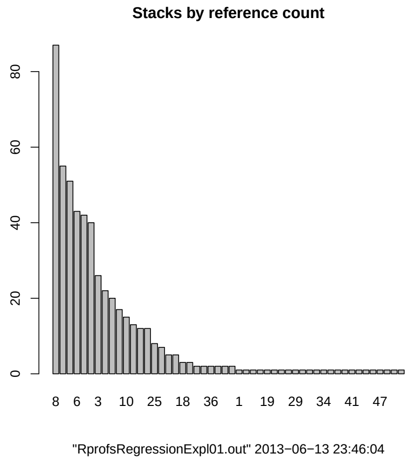




Input

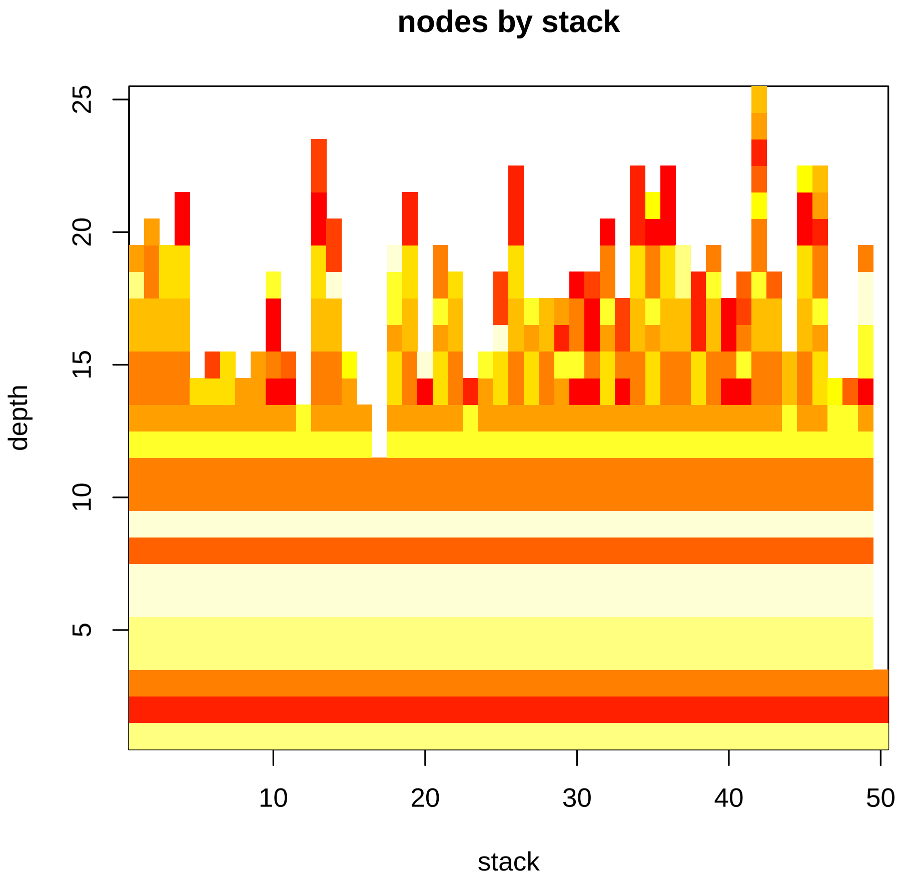
```
par(mfrow=c(1,2))
plot_stacks(sprofRegressionExpl)
```



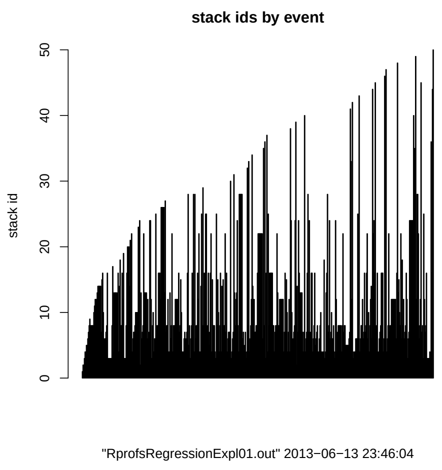


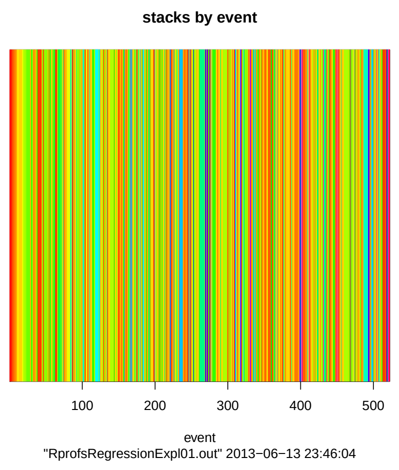
Input
```
stacks_nodes <- list.as.matrix(sprofRegressionExpl$stacks$nodes)
```

Input
```
image(x=1:ncol(stacks_nodes),y=1:nrow(stacks_nodes),
t(stacks_nodes),xlab="stack", ylab="depth", main="nodes by stack")
```



Input
```
par(mfrow=c(1,2))
plot_profiles(sprofRegressionExpl)
```



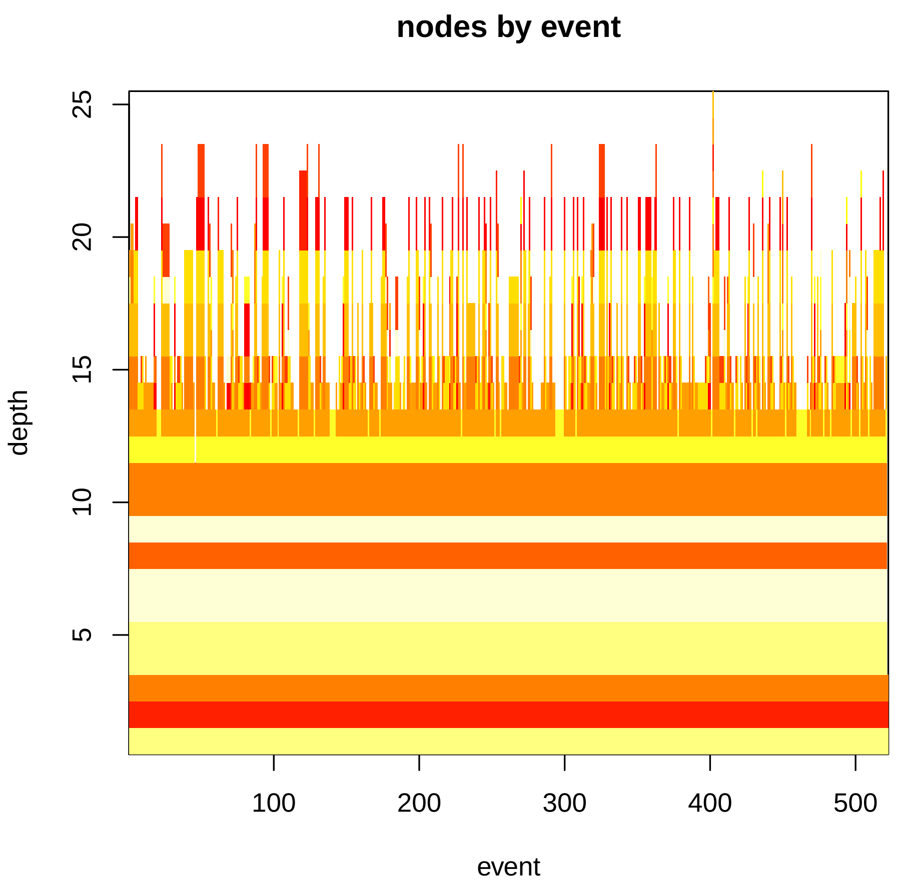


Input

```
profile_nodes <- profiles_matrix(sprofRegressionExpl)
```

Input

```
image(x=1:ncol(profile_nodes),y=1:nrow(profile_nodes),
t(profile_nodes),xlab="event", ylab="depth", main="nodes by event")
```



The rtop] Topic **misc**!plot@`plot`*`plot()`* rfun]plot@`plot`|textit method for *`sprof`* objects concatenates these three functions.

1.2. **analysis.**

```
Input
profile_nodes_rrle <- rrle(profile_nodes)
str(profile_nodes_rrle)
```

```
Output
List of 25
 $ :List of 2
  ..$ lengths: int 522
  ..$ values : int 52
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int 522
  ..$ values : int 10
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 23 24
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 55 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 56 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 57 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 58 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 21 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 62 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 22 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:2] 521 1
  ..$ values : int [1:2] 22 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:4] 45 1 475 1
  ..$ values : int [1:4] 50 NA 50 NA
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:62] 18 1 3 23 1 14 1 22 1 13 ...
  ..$ values : int [1:62] 29 27 51 29 NA 29 51 29 51 29 ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:361] 6 3 1 7 1 1 1 1 1 6 ...
  ..$ values : int [1:361] 22 39 37 30 4 2 NA NA NA 22 ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:407] 6 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:407] 22 NA NA 14 38 NA 27 NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:427] 6 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:427] 35 NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:427] 6 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:427] 36 NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:450] 1 2 3 1 1 1 1 1 1 1 ...
  ..$ values : int [1:450] 53 22 40 NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:466] 1 2 3 1 1 1 1 1 1 1 ...
  ..$ values : int [1:466] 27 22 41 NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:489] 1 2 1 2 1 1 1 1 1 1 ...
  ..$ values : int [1:489] NA 28 NA 5 NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:494] 1 1 1 1 2 1 1 1 1 1 ...
  ..$ values : int [1:494] NA NA NA NA 6 NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:508] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:508] NA NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:512] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:512] NA NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:522] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:522] NA NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
 $ :List of 2
  ..$ lengths: int [1:522] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ values : int [1:522] NA NA NA NA NA NA NA NA NA NA ...
  ..- attr(*, "class")= chr "rle"
```

1.3. **trimming.**

Input
```
trimstacks <- function(sprof, level){
lapply(sprof$stacks$nodes, function(x) {x[-(1:level)]})
}
```

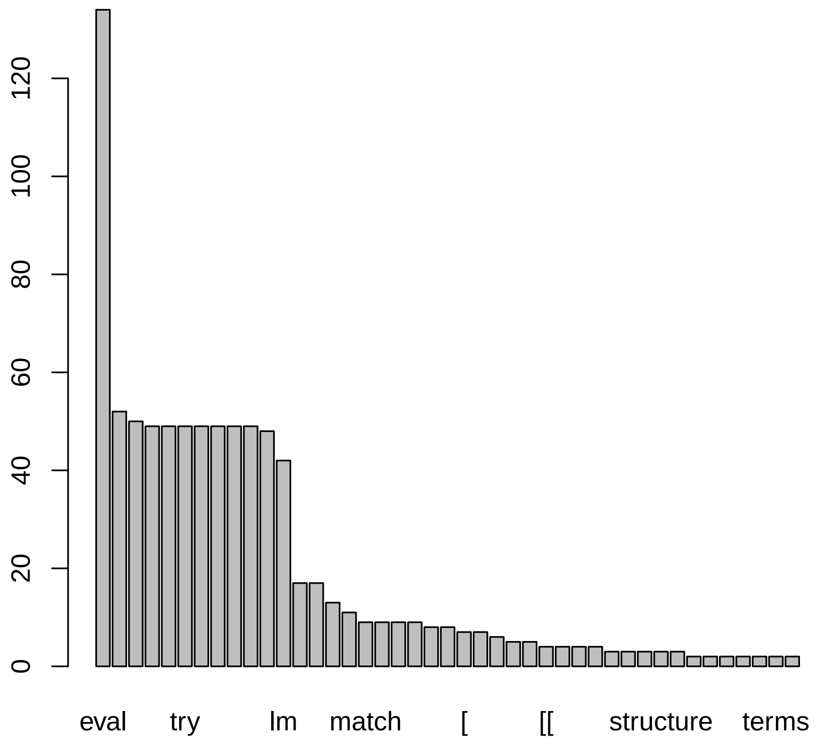
Input
```
sprofRegressionExplTr <- trimstacks(sprofRegressionExpl, 11)
#profile_nodesTr <- profiles_matrix(sprofRegressionExplTr)
#image(x=1:ncol(profile_nodesTr),y=1:nrow(profile_nodesTr), t(profile_nodesTr),xlab="event", ylab="dep
```

Input

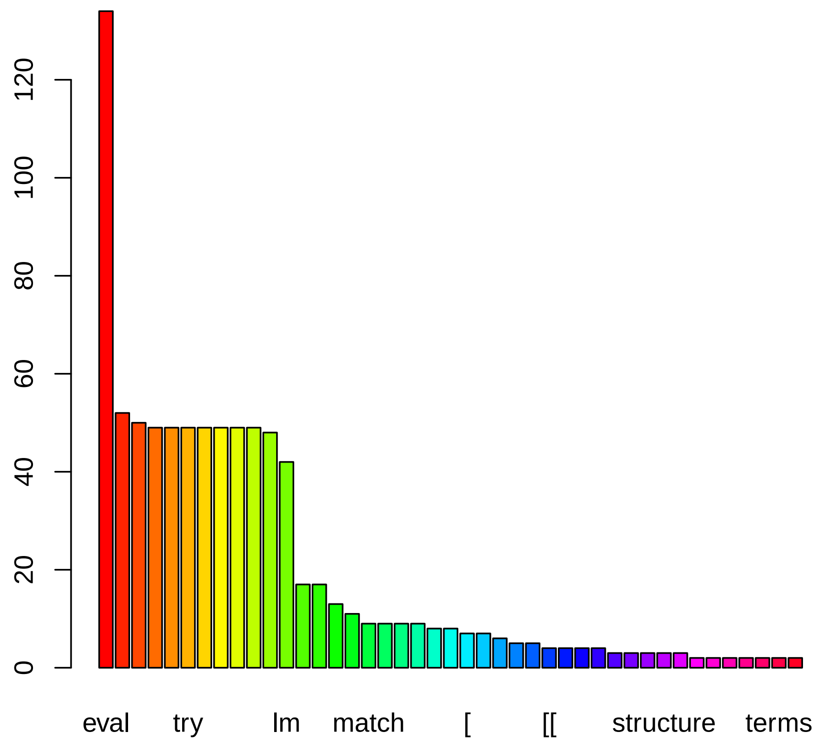
```
nodefreq <- rep(0,length(sprofRegressionExpl$nodes$name))
for (i in (1:length(sprofRegressionExpl$stacks$nodes))){
        nodefreq <- nodefreq +
                table( factor(sprofRegressionExpl$stacks$nodes[[i]],
                        levels <- 1:length(sprofRegressionExpl$nodes$name) ,
                        ordered=FALSE))
        }
names(nodefreq) <- sprofRegressionExpl$nodes$name
```

Top frequent nodes.

Input
```
ndf <- nodefreq[nodefreq>1]
ondf <- order(ndf,decreasing=TRUE)
barplot(ndf[ondf])
```

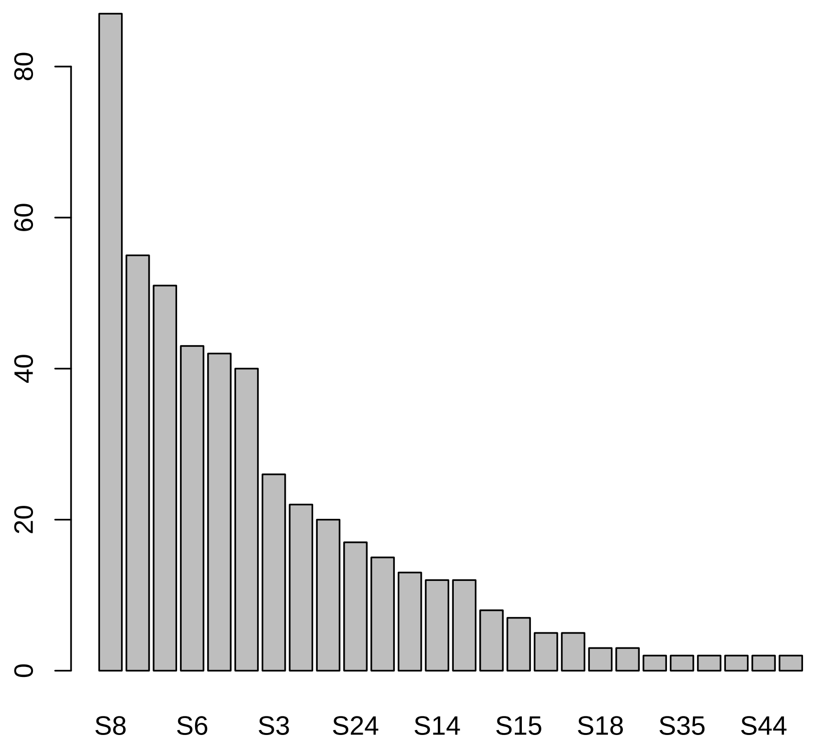


Input
```
barplot(ndf[ondf], col=rainbow(length(ondf)))
```

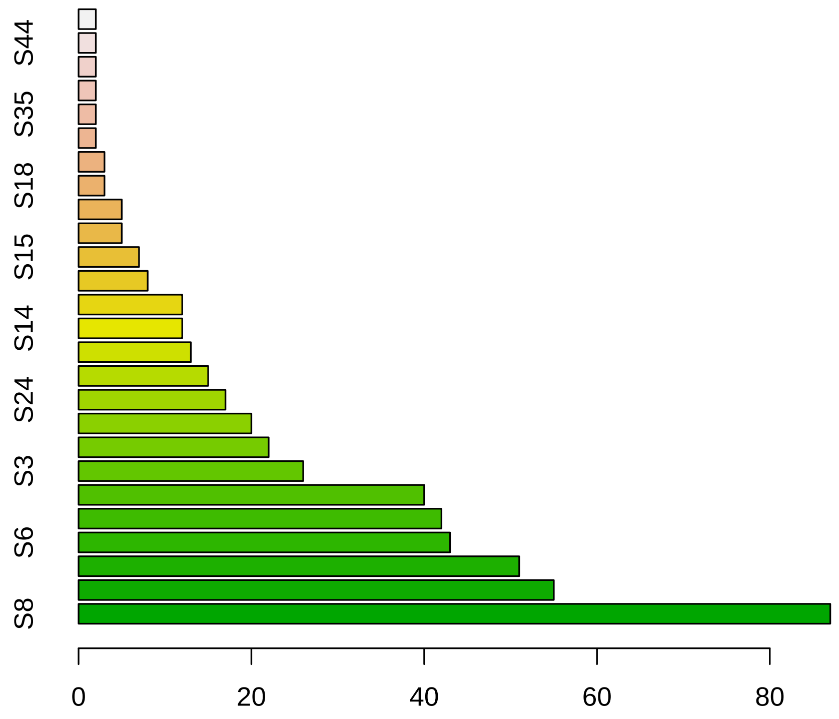


Top frequent stacks.

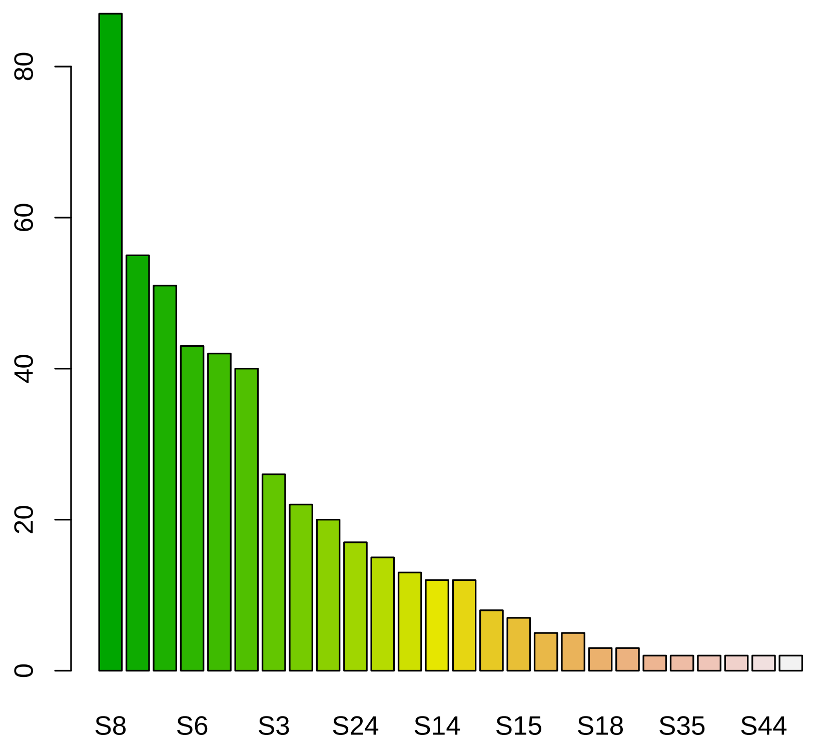
Input
```
x <- sprofRegressionExpl
xsrc <- as.matrix(x$stacks$refcount)
rownames(xsrc) <- rownames(xsrc, do.NULL=FALSE, prefix="S")
#stf <- x$stacks$refcount[x$stacks$refcount>1]
#names(stf) <-  x$stacks$shortname[x$stacks$refcount>1]
stf <- xsrc[xsrc>1]
names(stf) <- rownames(xsrc)[xsrc>1]
ostf <- order(stf,decreasing=TRUE)
barplot(stf[ostf])
```



Input
```
barplot(stf[ostf], col=terrain.colors(length(ostf)), horiz=TRUE)
```



Input
```
barplot(stf[ostf], col=terrain.colors(length(ostf)))
```



There is no statistics on profiles. Profiling are our elementary data. However we can lnk to our derived data to get a more informative display. For example, going one step back we can encode stacks and use these colour codes in the display of a profile.

Or going two steps back, we can encode nodes in colour, giving coloured stacks, and use these in the display of profile data.

2. Example data

Input

```
getwd()
```

Output

```
[1] "/Users/gs/projects/rforge/sintro/pkg/sprof/work/vignettes"
```

Input

```
dir()
```

Output

```
 [1] "gssda_old.sty"
 [2] "Makefile"
 [3] "rpo.out"
 [4] "Rprof.out"
 [5] "Rprofsr01.out"
 [6] "RprofsRegressionExpl.out"
 [7] "RprofsRegressionExpl01.out"
 [8] "SIntro_old.sty"
 [9] "sprof01.Rdata"
[10] "sprofiling-barplotNodes.pdf"
[11] "sprofiling-barplotNodes4.pdf"
[12] "sprofiling-barplotStacks.pdf"
[13] "sprofiling-barplotStacks4.pdf"
[14] "sprofiling-nwrpoadj.pdf"
[15] "sprofiling-rpoadjNEL.pdf"
[16] "sprofiling-rpoadjvizcirco.pdf"
[17] "sprofiling-rpoadjvizdot.pdf"
[18] "sprofiling-rpoadjvizfdp.pdf"
[19] "sprofiling-rpoadjvizneato.pdf"
[20] "sprofiling-rpoadjviztwopi.pdf"
[21] "sprofiling-rpoig.pdf"
[22] "sprofiling-rpoplotnodes.pdf"
[23] "sprofiling-rpoplotprofiles.pdf"
[24] "sprofiling-rpoplotstacks.pdf"
[25] "sprofiling-sREimgprofiles.pdf"
[26] "sprofiling-sREimgstacks.pdf"
[27] "sprofiling-sREplotnodefreq.pdf"
[28] "sprofiling-sREplotnodefreq1.pdf"
[29] "sprofiling-sREplotnodes.pdf"
[30] "sprofiling-sREplotprofiles.pdf"
[31] "sprofiling-sREplotprofiles242.pdf"
[32] "sprofiling-sREplotprofiles323.pdf"
[33] "sprofiling-sREplotprofiles4.pdf"
[34] "sprofiling-sREplotstackfreq.pdf"
[35] "sprofiling-sREplotstackfreq1.pdf"
[36] "sprofiling-sREplotstackfreq2.pdf"
[37] "sprofiling-sREplotstacks.pdf"
[38] "sprofiling-sREplotstacks4col.pdf"
[39] "sprofiling.aux"
[40] "sprofiling.idx"
[41] "sprofiling.ilg"
[42] "sprofiling.ind"
[43] "sprofiling.log"
[44] "sprofiling.pdf"
[45] "sprofiling.tex"
[46] "sprofiling.toc"
```

3. A better grip on profile information

The basic information provided by all profilers in R is a protocol of sampled stacks. For each recorded event, the protocol records one line with a text string showing the sampled stack (in reverse order: most recent first). The stack lines may be preceded by header lines with event specific information. The protocol may be interspersed with control information, such as information about the timing interval used.

We know that the structural information, static information as well as dynamic information, can be represented with the help of a graph. For a static analysis, the graph representation may be the first choice. For a dynamic analysis, the stack information is our first information. A stack is a connected path in the program graph. If we start with nodes and edges, we loose information which is readily available in record of stacks.

As we know that we are working with stacks, we know that they have their peculiarities. Stacks tend to grow and shrink. Subsequent events will have extensions and shrinkages of stacks (if the recording is on a fine scale), or stack sharing common stumps (if the recording is on a coarser scale).

There have always been interrupts, and these show up in profiles. In R, this is related problem (GC)

The graph is a second instance that is (re)constructed from the stack recording.

Here is the way we represent the profile information:

The profile log file is sanitised:

- Control lines are extracted and recorded in a separate list.
- Head parts, if present, ere extracted and recorded in a matrix that is kept line-aligned with the remainder
- Line content is standardised, for example by removing stray quotation marks etc.

After this, the sanitised lines are encoded as a vector of stacks, and references to this.

If necessary, these steps are done by chunks to reduce memory load.

From the vector of stacks, a vector of nodes (or rather node names) is derived.

The stacks are now encoded by references to the nodes table. For convenience, we keep the (sanitised) textual representation of the stacks.

So far, texts are in reverse order. For each stack, we record the trailing leaf, and then we reverse order. The top of stack is now on first position.

Several statistics can be accumulated easily as a side effect.

Conceptually, the data structure consist of three tables (the implementation may differ, and is subject to change).

The profiles table is the representation of the input file. Control lines are are collected in a special table. With the control lines removed, the rest is a table, one row per input line. The body of the line, the stack, is encoded as a reference to a stacks table (obligatory) and header information (optional).

The stacks table contains the collected stacks, each stack encoded as a list of references to the node table. This is obligatory. This list is kept in reverse order (root at position 1). A source line representing the stack information may be kept (optional).

The nodes table keeps the names at the nodes.

To illustrate our data structure, we use *`Rprofsr01.out`* as provided in section 1.1 on page 3.

vref:

section 1.1 on page 3

This is a temporary hack to get a most recent private recent version of *`library(sprof)`*.

`Input`

```
 rpo <- readRprof("Rprofsr01.out")
 str_prof(rpo)
```

```
________________________________ Output ________________________________
First line:
0 Sampling intervals in micros: NULL
64 nodes in 47 stacks
36 Terminals
1 Roots:
         .deparseOpts          .getXlevels       .row_names_info
                    0                    0                     0
                    [          [.data.frame                    [[
                    0                    0                     0
        [[.data.frame                 %in%           <Anonymous>
                    0                    0                     0
                    $        anyDuplicated anyDuplicated.default
                    0                    0                     0
         as.character              as.list    as.list.data.frame
                    0                    0                     0
            as.vector                    c                   cat
                    0                    0                     0
             chol2inv              deparse            doTryCatch
                    0                    0                     0
                 eval             evalFunc                   FUN
                    0                    0                     0
            getOption               lapply        lazyLoadDBfetch
                    0                    0                     0
                 list                   lm                lm.fit
                    0                    0                     0
                match            match.fun                  mean
                    0                    0                     0
         mean.default                 mode           model.frame
                    0                    0                     0
  model.frame.default         model.matrix  model.matrix.default
                    0                    0                     0
       model.response              na.omit    na.omit.data.frame
                    0                    0                     0
                names              options                 paste
                    0                    0                     0
               pmatch              rep.int                sapply
                    0                    0                     0
       simplify2array            structure               summary
                    0                    0                     0
           summary.lm               Sweave                 terms
                    0                   47                     0
        terms.formula                  try              tryCatch
                    0                    0                     0
         tryCatchList          tryCatchOne                unique
                    0                    0                     0
       unique.default               unlist                vapply
                    0                    0                     0
          withVisible
                    0
```

```
rpo  Structure: List of 4
$ info    :'data.frame':        1 obs. of  8 variables:
$ nodes   :'data.frame':        64 obs. of  5 variables:
$ stacks  :'data.frame':        47 obs. of  7 variables:
$ profiles:List of 4
- attr(*, "class")= chr [1:2] "sprof" "list"

stacks Structure: 'data.frame':        47 obs. of  7 variables:
$ nodes         :List of 47
$ shortname     : Factor w/ 47 levels "S<A>eFttCtCLtCOdTCwVeesleem.m..n.n...[[.",..: 3 18 14 20 23 ...
$ refcount      : num  1 1 16 23 1 ...
$ stacklength   : int  13 18 17 19 21 ...
$ stackheadnodes: int  53 53 53 53 53 ...
$ stackleafnodes: int  27 27 37 42 27 ...
$ stackssrc     : Factor w/ 47 levels ".deparseOpts FUN lapply sapply match model.matrix.default model
```

4. Standard output

4.1. Summary.

Input
```
summary_nodes(rpo)
```

Output
```
                      shortname root leaf self.time self.pct total.time
.deparseOpts               .dpO    - LEAF         1      Inf          0
.getXlevels                .gtX    -    -         0      NaN          0
.row_names_info            .r__    - LEAF         1      Inf          0
[                             [    -    -         0      NaN          0
[.data.frame               [.d.    - LEAF        48      Inf          0
[[                           [[    -    -         0      NaN          0
[[.data.frame              [[..    - LEAF         2      Inf          0
%in%                       %in%    - LEAF         1      Inf          0
<Anonymous>                <An>    -    -         0      NaN          0
$                             $    - LEAF         2      Inf          0
anyDuplicated              anyD    - LEAF         1      Inf          0
anyDuplicated.default      anD.    - LEAF         9      Inf          0
as.character               as.c    - LEAF        35      Inf          0
as.list                    as.l    - LEAF         1      Inf          0
as.list.data.frame         a...    - LEAF        33      Inf          0
as.vector                  as.v    -    -         0      NaN          0
c                             c    - LEAF         1      Inf          0
cat                         cat    - LEAF         1      Inf          0
chol2inv                   chl2    - LEAF         2      Inf          0
deparse                    dprs    -    -         0      NaN          0
doTryCatch                 dTrC    -    -         0      NaN          0
eval                       eval    -    -         0      NaN          0
evalFunc                   evlF    -    -         0      NaN          0
FUN                         FUN    - LEAF         1      Inf          0
getOption                  gtOp    -    -         0      NaN          0
lapply                     lppl    -    -         0      NaN          0
lazyLoadDBfetch            lLDB    - LEAF         7      Inf          0
list                       list    - LEAF        14      Inf          0
lm                           lm    - LEAF        43      Inf          0
```

```
lm.fit                    lm.f    - LEAF     114      Inf          0
match                     mtch    - LEAF       2      Inf          0
match.fun                 mtc.    -    -       0      NaN          0
mean                      mean    -    -       0      NaN          0
mean.default              mn.d    - LEAF       1      Inf          0
mode                      mode    - LEAF       1      Inf          0
model.frame              mdl.f    -    -       0      NaN          0
model.frame.default      mdl.f.   - LEAF      16      Inf          0
model.matrix             mdl.m    -    -       0      NaN          0
model.matrix.default     mdl.m.   - LEAF      55      Inf          0
model.response           mdl.r    - LEAF       2      Inf          0
na.omit                   n.mt    - LEAF      15      Inf          0
na.omit.data.frame        n...    - LEAF      23      Inf          0
names                     nams    - LEAF       1      Inf          0
options                   optn    - LEAF       1      Inf          0
paste                     past    -    -       0      NaN          0
pmatch                    pmtc    - LEAF       1      Inf          0
rep.int                   rp.n    - LEAF       1      Inf          0
sapply                    sppl    -    -       0      NaN          0
simplify2array            smp2    -    -       0      NaN          0
structure                 strc    - LEAF      33      Inf          0
summary                   smmr    -    -       0      NaN          0
summary.lm                smm.    - LEAF      36      Inf          0
Sweave                    Swev ROOT    -       0      NaN          0
terms                     trms    -    -       0      NaN          0
terms.formula             trm.    - LEAF       1      Inf          0
try                        try    -    -       0      NaN          0
tryCatch                  tryC    -    -       0      NaN          0
tryCatchList              trCL    -    -       0      NaN          0
tryCatchOne               trCO    -    -       0      NaN          0
unique                    uniq    -    -       0      NaN          0
unique.default            unq.    - LEAF       1      Inf          0
unlist                    unls    - LEAF       1      Inf          0
vapply                    vppl    -    -       0      NaN          0
withVisible               wthV    -    -       0      NaN          0
                     total.pct
.deparseOpts               NaN
.getXlevels                NaN
.row_names_info            NaN
[                          NaN
[.data.frame               NaN
[[                         NaN
[[.data.frame              NaN
%in%                       NaN
<Anonymous>                NaN
$                          NaN
anyDuplicated              NaN
anyDuplicated.default      NaN
as.character               NaN
as.list                    NaN
as.list.data.frame         NaN
as.vector                  NaN
c                          NaN
```

```
cat                         NaN
chol2inv                    NaN
deparse                     NaN
doTryCatch                  NaN
eval                        NaN
evalFunc                    NaN
FUN                         NaN
getOption                   NaN
lapply                      NaN
lazyLoadDBfetch             NaN
list                        NaN
lm                          NaN
lm.fit                      NaN
match                       NaN
match.fun                   NaN
mean                        NaN
mean.default                NaN
mode                        NaN
model.frame                 NaN
model.frame.default         NaN
model.matrix                NaN
model.matrix.default        NaN
model.response              NaN
na.omit                     NaN
na.omit.data.frame          NaN
names                       NaN
options                     NaN
paste                       NaN
pmatch                      NaN
rep.int                     NaN
sapply                      NaN
simplify2array              NaN
structure                   NaN
summary                     NaN
summary.lm                  NaN
Sweave                      NaN
terms                       NaN
terms.formula               NaN
try                         NaN
tryCatch                    NaN
tryCatchList                NaN
tryCatchOne                 NaN
unique                      NaN
unique.default              NaN
unlist                      NaN
vapply                      NaN
withVisible                 NaN
```

Input

```
summary_stacks(rpo)
```

Output

```
   len refcount root leafs
1   13        1   53    27
```

```
2   18        1   53    27
3   17       16   53    37
4   19       23   53    42
5   21        1   53    27
6   21       48   53     5
7   20        1   53    27
8   15        1   53    27
9   18       25   53    15
10  15       55   53    39
11  14        1   53    27
12  14      114   53    30
13  15       23   53    50
14  13       43   53    29
15  18       10   53    50
16  14        1   53    27
17  14        2   53    19
18  13       36   53    52
19  25        1   53    35
20  15       35   53    13
21  18       15   53    41
22  20        1   53     1
23  20        7   53    15
24  14        2   53    40
25  20        1   53    62
26  23        9   53    12
27  20       14   53    28
28  19        1   53    24
29  22        1   53    17
30  15        1   53    47
31  14        2   53    10
32  21        1   53     7
33  17        1   53    15
34  16        1   53    43
35  15        1   53    34
36  22        1   53    11
37  19        1   53    55
38  21        1   53    46
39  16        1   53     7
40  18        1   53     3
41  24        1   53    31
42  16        1   53     8
43  19        1   53    44
44  19        1   53    31
45  17        1   53    14
46  20        1   53    61
47   3        1   53    18
```

Input

```
summary_profiles(rpo)
```

Output

```
$id
[1] "Profile Summary Tue Jul  2 17:30:23 2013"
```

```
$len
[1] 508

$uniquestacks
[1] 47

$nr_runs
[1] 373
```

The rtop] Topic **misc**!summary@`summary`*`summary()`* rfun]summary@`summary`|textit method for ***`sprof`*** objects concatenates these three functions.

4.2. **Print.**

```
Input
print_nodes(rpo)
```

```
Output
                      shortname root leaf self.time self.pct total.time
.deparseOpts               .dpO    - LEAF         1      Inf          0
.getXlevels                .gtX    -    -         0      NaN          0
.row_names_info            .r__    - LEAF         1      Inf          0
[                             [    -    -         0      NaN          0
[.data.frame               [.d.    - LEAF        48      Inf          0
[[                           [[    -    -         0      NaN          0
[[.data.frame              [[..    - LEAF         2      Inf          0
%in%                       %in%    - LEAF         1      Inf          0
<Anonymous>                <An>    -    -         0      NaN          0
$                             $    - LEAF         2      Inf          0
anyDuplicated              anyD    - LEAF         1      Inf          0
anyDuplicated.default      anD.    - LEAF         9      Inf          0
as.character               as.c    - LEAF        35      Inf          0
as.list                    as.l    - LEAF         1      Inf          0
as.list.data.frame         a...    - LEAF        33      Inf          0
as.vector                  as.v    -    -         0      NaN          0
c                             c    - LEAF         1      Inf          0
cat                         cat    - LEAF         1      Inf          0
chol2inv                   chl2    - LEAF         2      Inf          0
deparse                    dprs    -    -         0      NaN          0
doTryCatch                 dTrC    -    -         0      NaN          0
eval                       eval    -    -         0      NaN          0
evalFunc                   evlF    -    -         0      NaN          0
FUN                         FUN    - LEAF         1      Inf          0
getOption                  gtOp    -    -         0      NaN          0
lapply                     lppl    -    -         0      NaN          0
lazyLoadDBfetch            lLDB    - LEAF         7      Inf          0
list                       list    - LEAF        14      Inf          0
lm                           lm    - LEAF        43      Inf          0
lm.fit                     lm.f    - LEAF       114      Inf          0
match                      mtch    - LEAF         2      Inf          0
match.fun                  mtc.    -    -         0      NaN          0
mean                       mean    -    -         0      NaN          0
mean.default               mn.d    - LEAF         1      Inf          0
mode                       mode    - LEAF         1      Inf          0
model.frame               mdl.f    -    -         0      NaN          0
model.frame.default      mdl.f.    - LEAF        16      Inf            0
model.matrix              mdl.m    -    -         0      NaN            0
model.matrix.default     mdl.m.    - LEAF        55      Inf            0
model.response            mdl.r    - LEAF         2      Inf            0
na.omit                    n.mt    - LEAF        15      Inf            0
na.omit.data.frame         n...    - LEAF        23      Inf            0
names                      nams    - LEAF         1      Inf            0
options                    optn    - LEAF         1      Inf            0
paste                      past    -    -         0      NaN            0
pmatch                     pmtc    - LEAF         1      Inf            0
rep.int                    rp.n    - LEAF         1      Inf            0
sapply                     sppl    -    -         0      NaN            0
simplify2array             smp2    -    -         0      NaN            0
structure                  strc    - LEAF        33      Inf            0
summary                    smmr    -    -         0      NaN            0
summary.lm                 smm.    - LEAF        36      Inf            0
Sweave                     Swev ROOT    -         0      NaN            0
terms                      trms    -    -         0      NaN            0
terms.formula              trm.    - LEAF         1      Inf            0
try                         try    -    -         0      NaN            0
tryCatch                   tryC    -    -         0      NaN            0
tryCatchList               trCL    -    -         0      NaN            0
tryCatchOne                trCO    -    -         0      NaN            0
unique                     uniq    -    -         0      NaN            0
unique.default             unq.    - LEAF         1      Inf            0
unlist                     unls    - LEAF         1      Inf            0
vapply                     vppl    -    -         0      NaN            0
withVisible                wthV    -    -         0      NaN            0
                      total.pct
.deparseOpts                NaN
.getXlevels                 NaN
.row_names_info             NaN
[                           NaN
[.data.frame                NaN
[[                          NaN
[[.data.frame               NaN
%in%                        NaN
<Anonymous>                 NaN
$                           NaN
anyDuplicated               NaN
anyDuplicated.default       NaN
as.character                NaN
as.list                     NaN
as.list.data.frame          NaN
as.vector                   NaN
c                           NaN
cat                         NaN
chol2inv                    NaN
deparse                     NaN
doTryCatch                  NaN
eval                        NaN
evalFunc                    NaN
FUN                         NaN
```

```
getOption                   NaN
lapply                      NaN
lazyLoadDBfetch             NaN
list                        NaN
lm                          NaN
lm.fit                      NaN
match                       NaN
match.fun                   NaN
mean                        NaN
mean.default                NaN
mode                        NaN
model.frame                 NaN
model.frame.default         NaN
model.matrix                NaN
model.matrix.default        NaN
model.response              NaN
na.omit                     NaN
na.omit.data.frame          NaN
names                       NaN
options                     NaN
paste                       NaN
pmatch                      NaN
rep.int                     NaN
sapply                      NaN
simplify2array              NaN
structure                   NaN
summary                     NaN
summary.lm                  NaN
Sweave                      NaN
terms                       NaN
terms.formula               NaN
try                         NaN
tryCatch                    NaN
tryCatchList                NaN
tryCatchOne                 NaN
unique                      NaN
unique.default              NaN
unlist                      NaN
vapply                      NaN
withVisible                 NaN
```

Input

```
 print_stacks(rpo)
```

Output

```
   len refcount root leafs
1   13        1   53    27
2   18        1   53    27
3   17       16   53    37
4   19       23   53    42
5   21        1   53    27
6   21       48   53     5
7   20        1   53    27
8   15        1   53    27
```

```
9   18       25   53    15
10  15       55   53    39
11  14        1   53    27
12  14      114   53    30
13  15       23   53    50
14  13       43   53    29
15  18       10   53    50
16  14        1   53    27
17  14        2   53    19
18  13       36   53    52
19  25        1   53    35
20  15       35   53    13
21  18       15   53    41
22  20        1   53     1
23  20        7   53    15
24  14        2   53    40
25  20        1   53    62
26  23        9   53    12
27  20       14   53    28
28  19        1   53    24
29  22        1   53    17
30  15        1   53    47
31  14        2   53    10
32  21        1   53     7
33  17        1   53    15
34  16        1   53    43
35  15        1   53    34
36  22        1   53    11
37  19        1   53    55
38  21        1   53    46
39  16        1   53     7
40  18        1   53     3
41  24        1   53    31
42  16        1   53     8
43  19        1   53    44
44  19        1   53    31
45  17        1   53    14
46  20        1   53    61
47   3        1   53    18
```

Input

```
 print_profiles(rpo)
```

Output

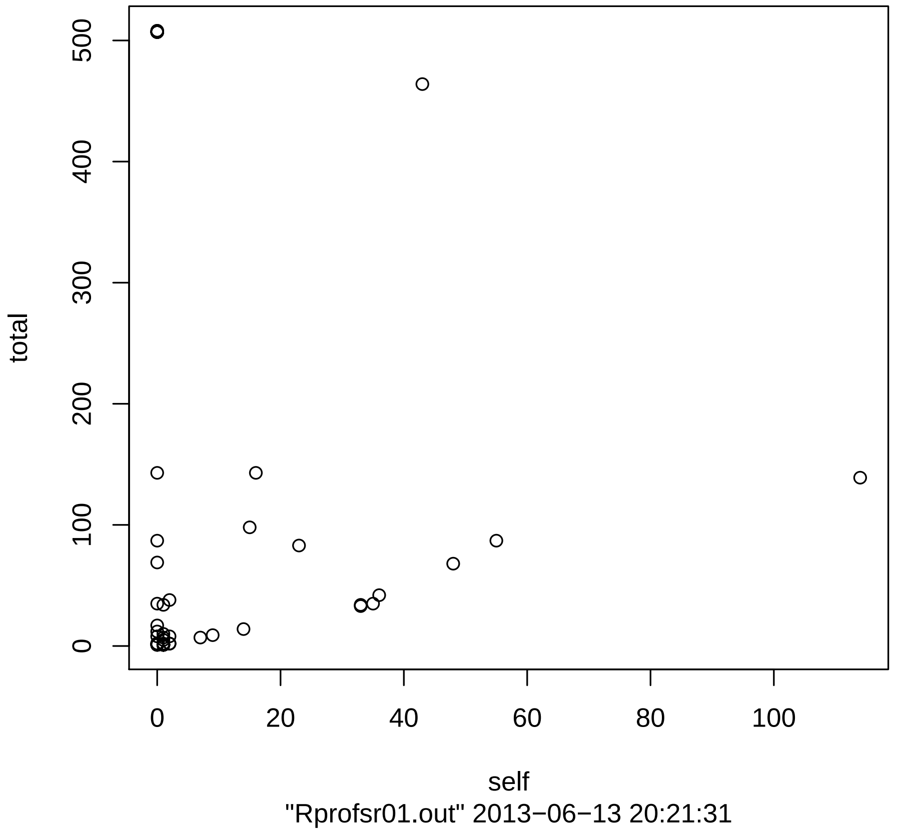
```
$id
[1] "Profile Summary Tue Jul  2 17:30:23 2013"

$len
[1] 508

$uniquestacks
[1] 47
```

```
$nr_runs
[1] 373
```

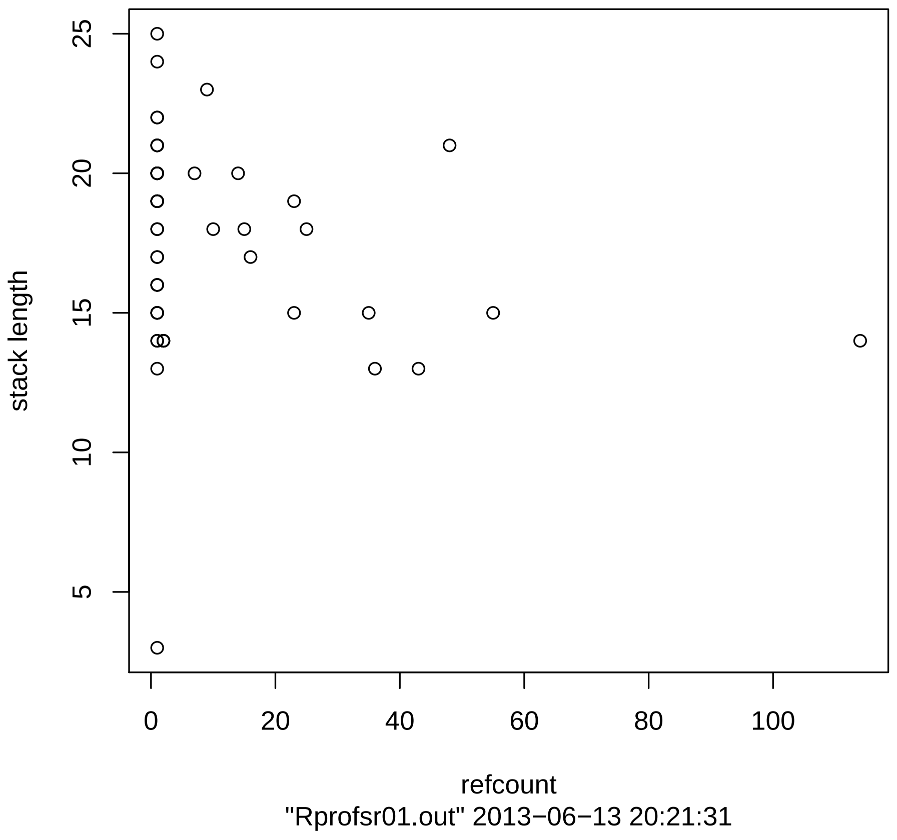
The rtop] Topic **misc**!print@`print`*`print()`* rfun]print@`print`|textit method for *`sprof`* objects concatenates these three functions.

4.3. **Plot.**

```
Input
plot_nodes(rpo)
```

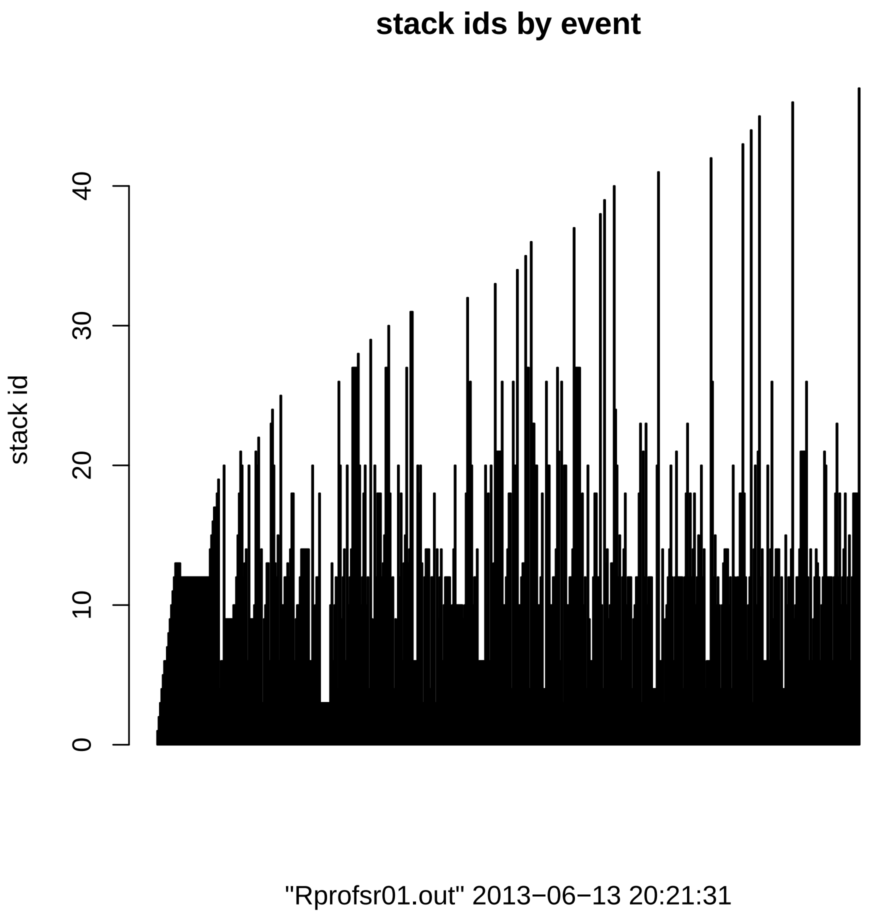


```
Input
plot_stacks(rpo)
```



Input

```
plot_profiles(rpo)
```



The rtop] Topic **misc**!plot@`plot`*`plot()`*rfun]plot@`plot`|textit method for *`sprof`* objects concatenates these three functions.

4.4. **Graph.** To interface *`sprof`* to a graph handling package, rtop] Topic **adjacency**!until@`until`*`until()`*rfun]until@`until`|textit can extract the adjacency matrix from the profile.

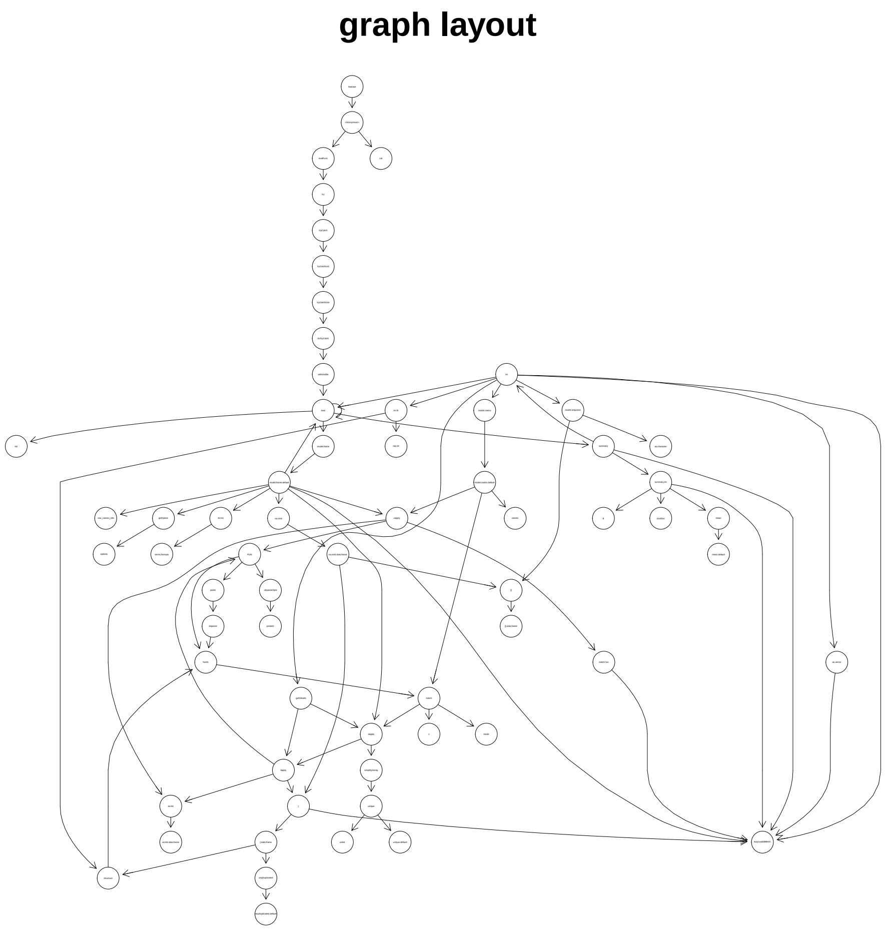
```
Input
rpoadj <- adjacency(rpo)
```

This is a format any graph package can handle.

4.4.1. *graph Package.*

Input
```
library(graph)
rpoadjNEL <- as(rpoadj,"graphNEL")
```

Input
```
plot(rpoadjNEL,  main="graph layout", cex.main=5)
#detach("package:graph")
```



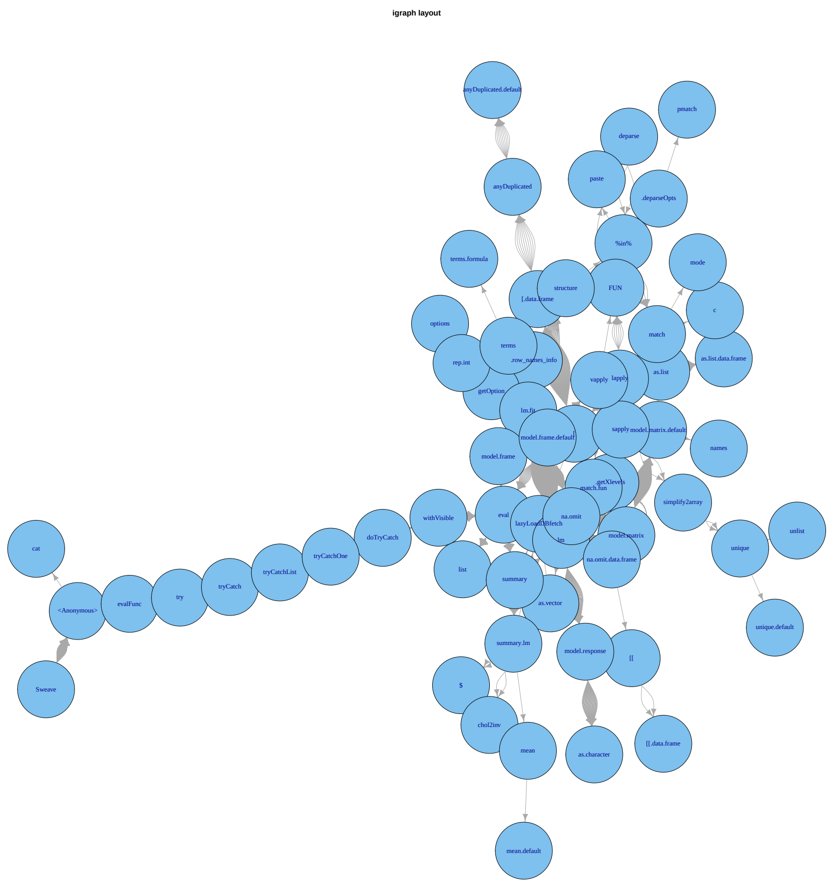
4.4.2. *igraph Package.*

Input

```
library(igraph)
rpoig <- graph.adjacency(rpoadj)
```

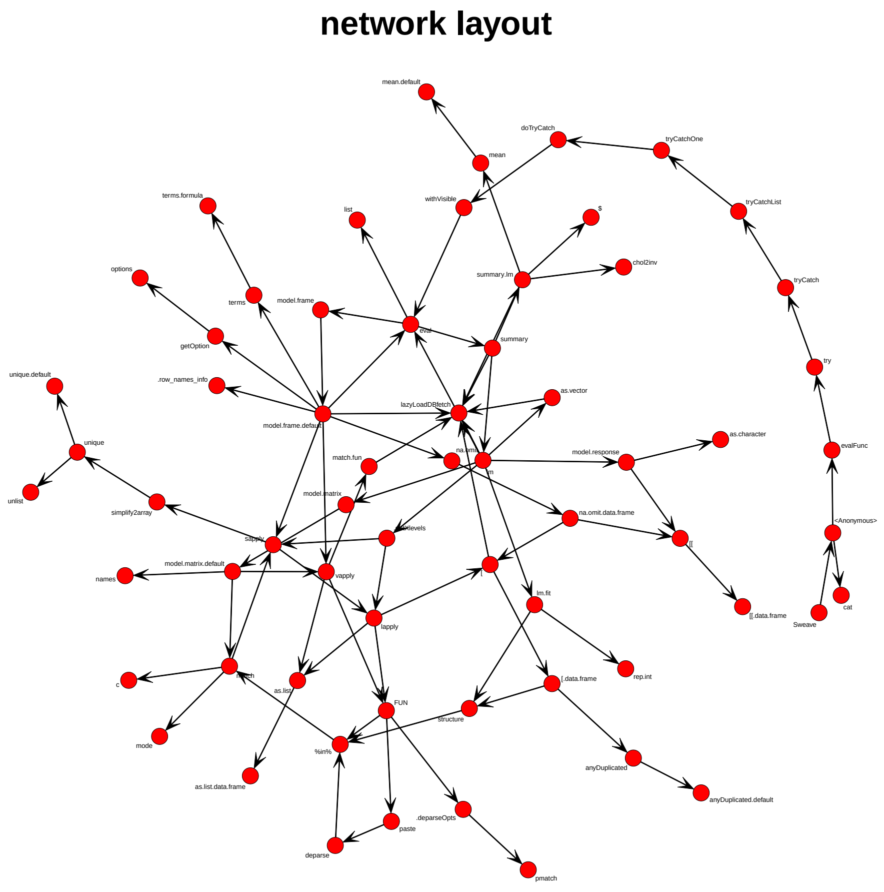
Input

```
#plot(rpoig, main="igraph layout", cex.main=5)
plot(rpoig, main="igraph layout")
detach("package:igraph")
```



4.4.3. *network Package.*

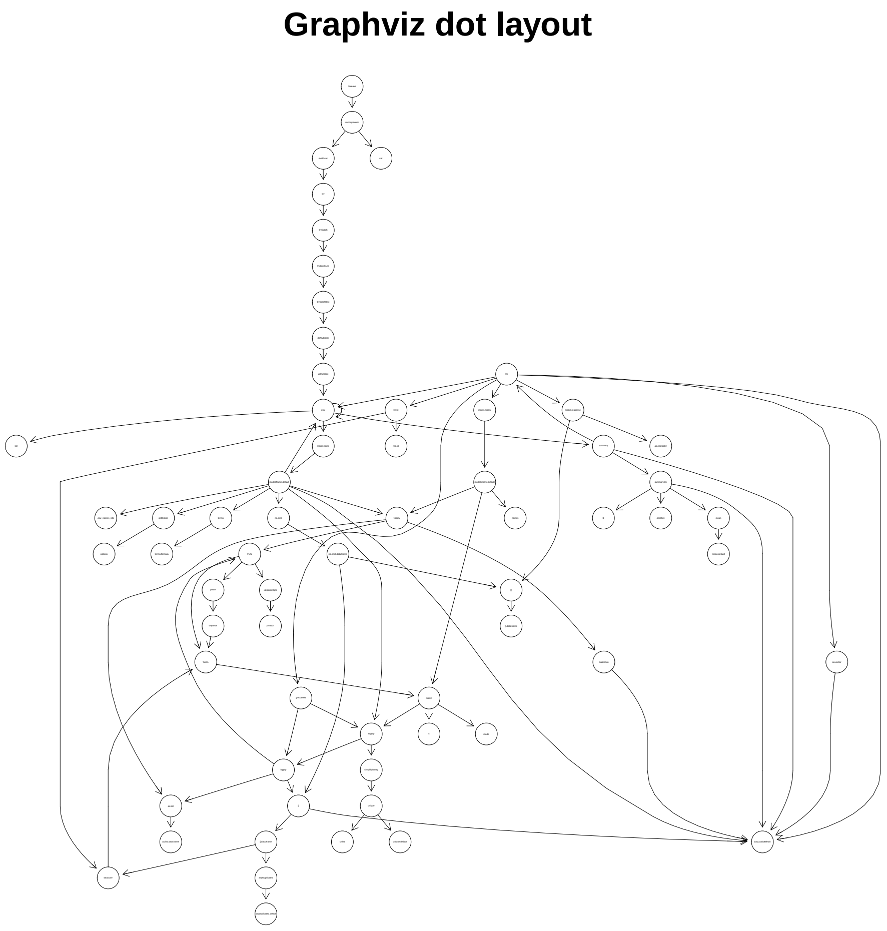
Input
```
library(network)
nwrpoadj <- as.network(rpoadj) # names is not imported
network.vertex.names(nwrpoadj) <- rownames(rpoadj) # not honoured by plot
plot(nwrpoadj, label=rownames(rpoadj), main="network layout", cex.main=5)
#plot(nwrpoadj, label=rownames(rpoadj), edge.lwd=rpoadj)
detach("package:network")
```



4.4.4. *Rgraphviz Package.*

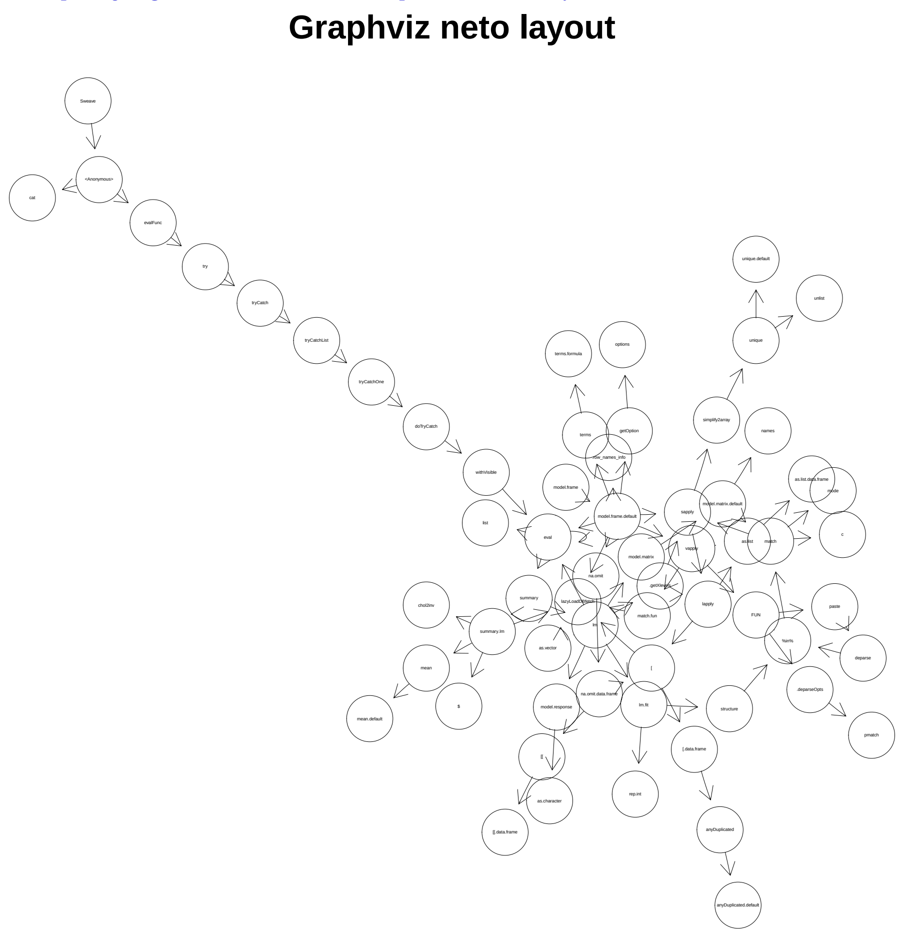
Input
```
library(Rgraphviz)
rpoadjRag <- agopen(rpoadjNEL, name="Rprof Example")
```

Input
```
plot(rpoadjRag, main="Graphviz dot layout", cex.main=5)
```



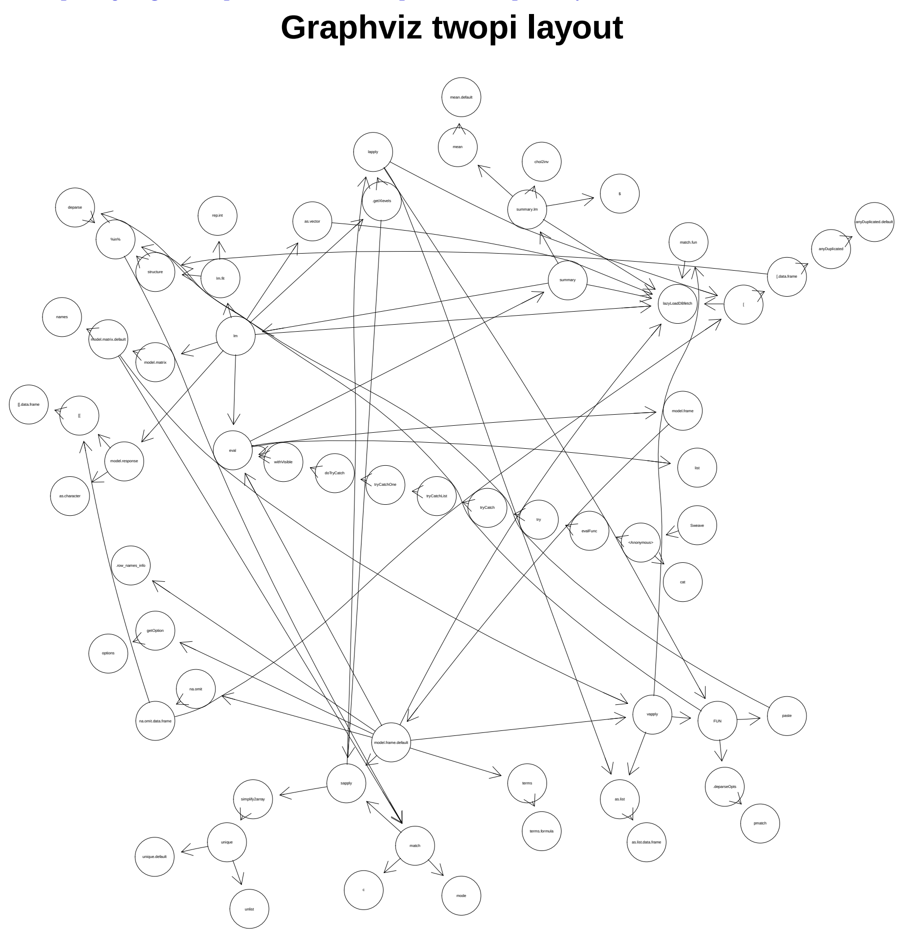
Input

```
plot(rpoadjRag,"neato", main="Graphviz neto layout", cex.main=5)
```



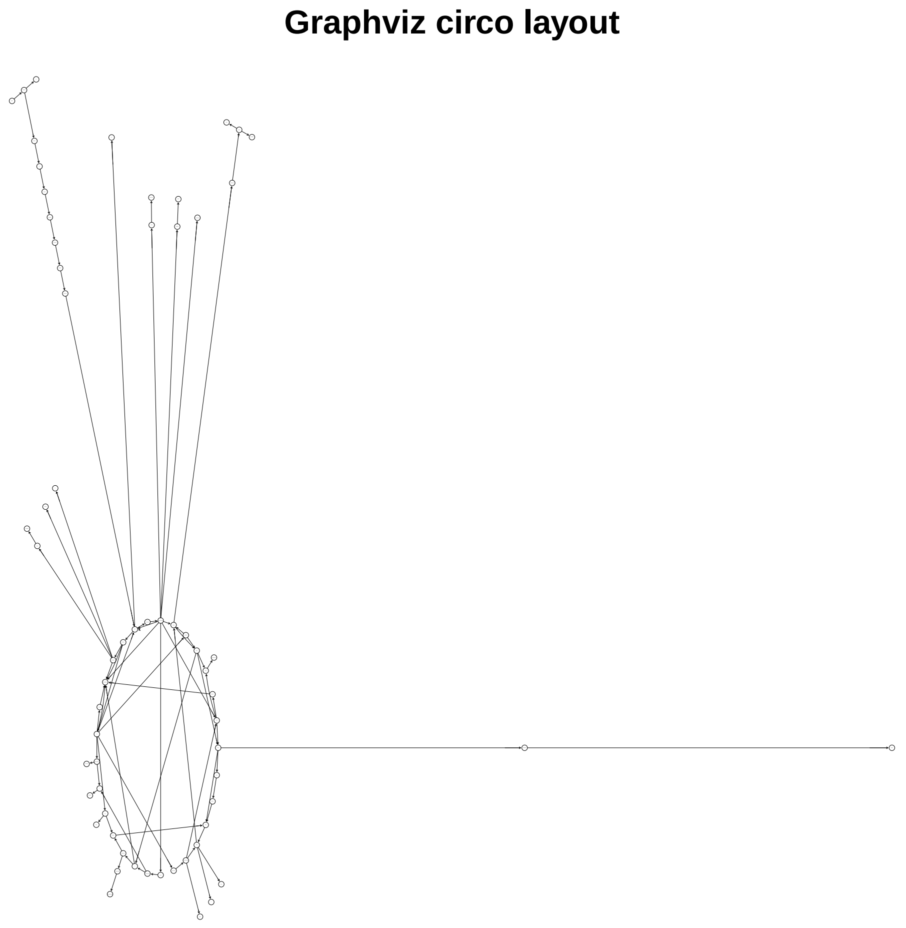
Input

```
plot(rpoadjRag,"twopi", main="Graphviz twopi layout", cex.main=5)
```



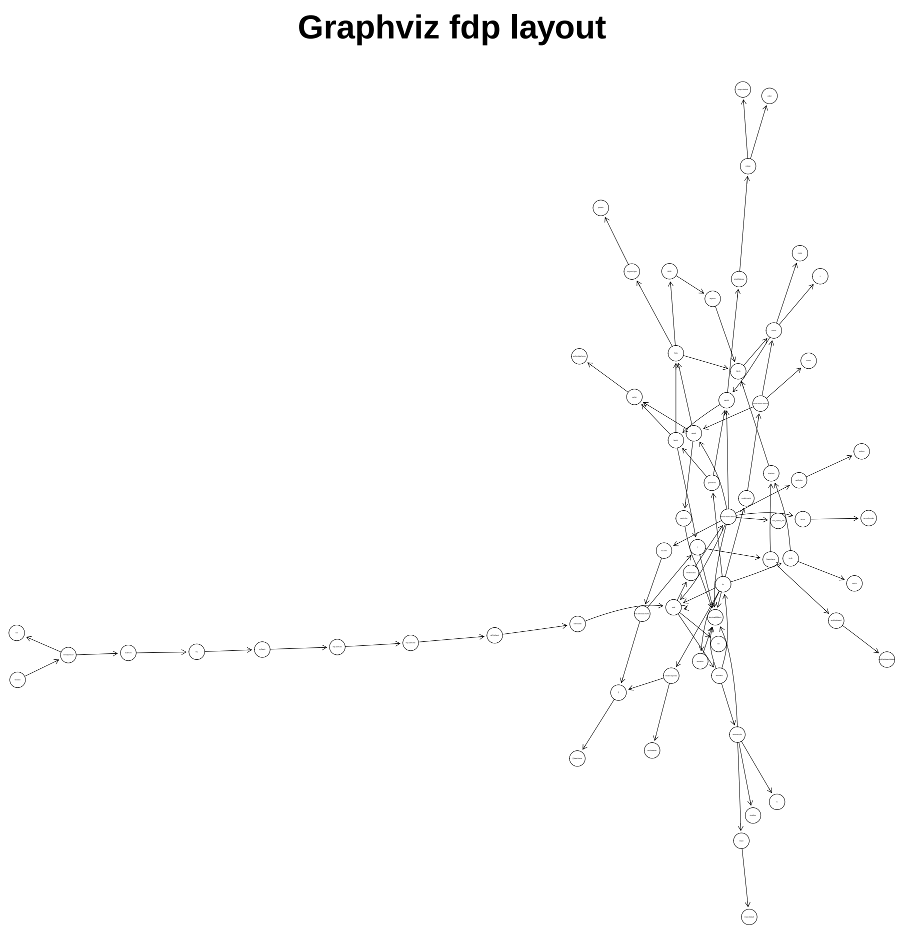
Input

```
plot(rpoadjRag,"circo", main="Graphviz circo layout", cex.main=5)
```



Input

```
plot(rpoadjRag,"fdp", main="Graphviz fdp layout", cex.main=5)
```



Index

- R version 3.0.1 (2013-05-16), `x86_64-apple-darwin10.8.0`
- Locale:
  `en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8`
- Base packages: base, datasets, graphics, grDevices, grid, methods, parallel, stats, utils
- Other packages: graph 1.38.2, Rgraphviz 2.4.0, sprof 0.0-4, stringdist 0.5.0, xtable 1.7-1
- Loaded via a namespace (and not attached): BiocGenerics 0.6.0, igraph 0.6.5-2, network 1.7.2, stats4 3.0.1, tools 3.0.1

```
$Source: /u/math/j40/cvsroot/lectures/src/insider/profile/Rnw/profile.Rnw,v $

$Revision: 1.1 $

$Date: 2013/05/20 20:24:04 $

$name:  $

$Author: j40 $
```

Günther Sawitzki
StatLab Heidelberg
Im Neuenheimer Feld 294
D 69120 Heidelberg

E-mail address: `gs@statlab.uni-heidelberg.de`
URL: `http://sintro.r-forge.r-project.org/`