Package 'microbenchmark'

January 24, 2018

Title Accurate Timing Functions

Description Provides infrastructure to accurately measure and compare the execution time of R expressions.

URL https://github.com/joshuaulrich/microbenchmark/

BugReports https://github.com/joshuaulrich/microbenchmark/issues/

License BSD_2_clause + file LICENSE

Imports graphics, stats

Suggests ggplot2, multcomp

ByteCompile yes

LazyData yes

Version 1.4-4

RoxygenNote 6.0.1

NeedsCompilation yes

Author Olaf Mersmann [aut],
Claudia Beleites [ctb],
Rainer Hurling [ctb],
Ari Friedman [ctb],
Joshua M. Ulrich [cre]

Maintainer Joshua M. Ulrich <josh.m.ulrich@gmail.com>

Repository CRAN

Date/Publication 2018-01-24 12:17:05 UTC

R topics documented:

`autoplot.microbenchmark`

Autoplot method for microbenchmark objects: Prettier graphs for microbenchmark using ggplot2

Description

Uses ggplot2 to produce a more legible graph of microbenchmark timings

Usage

```
autoplot.microbenchmark(object, ..., log = TRUE, y_max = 1.05 *
  max(object$time))
```

Arguments

`object`	A microbenchmark object
`...`	Ignored
`log`	If `TRUE` the time axis will be on log scale.
`y_max`	The upper limit of the y axis (defaults to 5 percent more than the maximum value)

Value

A ggplot2 plot

Author(s)

Ari Friedman, Olaf Mersmann

Examples

```
if (require("ggplot2")) {

tm <- microbenchmark(rchisq(100, 0),
                     rchisq(100, 1),
                     rchisq(100, 2),
                     rchisq(100, 3),
                     rchisq(100, 5), times=1000L)
autoplot(tm)
}
```

boxplot.microbenchmark *Boxplot of* microbenchmark *timings.*

Description

Boxplot of microbenchmark timings.

Usage

```
## S3 method for class 'microbenchmark'
boxplot(x, unit = "t", log = TRUE, xlab, ylab,
  horizontal = FALSE, ...)
```

Arguments

x	A microbenchmark object.
unit	Unit in which the results be plotted.
log	Should times be plotted on log scale?
xlab	X axis label.
ylab	Y axis label.
horizontal	Switch X and Y axes.
...	Passed on to boxplot.formula.

Author(s)

Olaf Mersmann

get_nanotime *Return the current value of the platform timer.*

Description

The current value of the most accurate timer of the platform is returned. This can be used as a time stamp for logging or similar purposes. Please note that there is no common reference, that is, the timer value cannot be converted to a date and time value.

Usage

```
get_nanotime()
```

Author(s)

Olaf Mersmann

`microbenchmark` *Sub-millisecond accurate timing of expression evaluation.*

Description

`microbenchmark` serves as a more accurate replacement of the often seen `system.time(replicate(1000, expr))` expression. It tries hard to accurately measure only the time it takes to evaluate `expr`. To achieved this, the sub-millisecond (supposedly nanosecond) accurate timing functions most modern operating systems provide are used. Additionally all evaluations of the expressions are done in C code to minimize any overhead.

Usage

```
microbenchmark(..., list = NULL, times = 100L, unit, check = NULL,
  control = list())
```

Arguments

`...`	Expressions to benchmark.
`list`	List of unevaluated expression to benchmark.
`times`	Number of times to evaluate the expression.
`unit`	Default unit used in `summary` and `print`.
`check`	Function to check if the expressions are equal. By default `NULL` which omits the check.
`control`	List of control arguments. See Details.

Details

This function is only meant for micro-benchmarking small pieces of source code and to compare their relative performance characteristics. You should generally avoid benchmarking larger chunks of your code using this function. Instead, try using the R profiler to detect hot spots and consider rewriting them in C/C++ or FORTRAN.

The `control` list can contain the following entries:

order the order in which the expressions are evaluated. "random" (the default) randomizes the execution order, "inorder" executes each expression in order and "block" executes all repetitions of each expression as one block.

warmup the number of warm-up iterations performed before the actual benchmark. These are used to estimate the timing overhead as well as spinning up the processor from any sleep or idle states it might be in. The default value is 2.

Value

Object of class 'microbenchmark', a data frame with columns `expr` and `time`. `expr` contains the deparsed expression as passed to `microbenchmark` or the name of the argument if the expression was passed as a named argument. `time` is the measured execution time of the expression in nanoseconds. The order of the observations in the data frame is the order in which they were executed.

Note

Depending on the underlying operating system, different methods are used for timing. On Windows the `QueryPerformanceCounter` interface is used to measure the time passed. For Linux the `clock_gettime` API is used and on Solaris the `gethrtime` function. Finally on MacOS X the, undocumented, `mach_absolute_time` function is used to avoid a dependency on the CoreServices Framework.

Before evaluating each expression `times` times, the overhead of calling the timing functions and the C function call overhead are estimated. This estimated overhead is subtracted from each measured evaluation time. Should the resulting timing be negative, a warning is thrown and the respective value is replaced by `0`. If the timing is zero, a warning is raised. Should all evaluations result in one of the two error conditions described above, an error is raised.

One platform on which the clock resolution is known to be too low to measure short runtimes with the required precision is Oracle® Solaris on some SPARC® hardware. Reports of other platforms with similar problems are welcome. Please contact the package maintainer.

Author(s)

Olaf Mersmann

See Also

`print.microbenchmark` to display and `boxplot.microbenchmark` or `autoplot.microbenchmark` to plot the results.

Examples

```
## Measure the time it takes to dispatch a simple function call
## compared to simply evaluating the constant \code{NULL}
f <- function() NULL
res <- microbenchmark(NULL, f(), times=1000L)

## Print results:
print(res)

## Plot results:
boxplot(res)

## Pretty plot:
if (require("ggplot2")) {
  autoplot(res)
}

## Example check usage
my_check <- function(values) {
  all(sapply(values[-1], function(x) identical(values[[1]], x)))
}

f <- function(a, b)
  2 + 2
```

```
a <- 2
## Check passes
microbenchmark(2 + 2, 2 + a, f(2, a), f(2, 2), check=my_check)
## Not run:
a <- 3
## Check fails
microbenchmark(2 + 2, 2 + a, f(2, a), f(2, 2), check=my_check)

## End(Not run)
```

`microtiming_precision` *Estimate precision of timing routines.*

Description

This function is currently experimental. Its main use is to judge the quality of the underlying timer implementation of the operating system. The function measures the overhead of timing a C function call `rounds` times and returns all non-zero timings observed. This can be used to judge the granularity and resolution of the timing subsystem.

Usage

```
microtiming_precision(rounds = 100L, warmup = 2^18)
```

Arguments

`rounds`	Number of measurements used to estimate the precision.
`warmup`	Number of iterations used to warm up the CPU.

Value

A vector of observed non-zero timings.

Author(s)

Olaf Mersmann

print.microbenchmark *Print* microbenchmark *timings.*

Description

Print microbenchmark timings.

Usage

```
## S3 method for class 'microbenchmark'
print(x, unit, order, signif, ...)
```

Arguments

x	An object of class microbenchmark.
unit	What unit to print the timings in. Default value taken from to option microbenchmark.unit (see example).
order	If present, order results according to this column of the output.
signif	If present, limit the number of significant digits shown.
...	Passed to print.data.frame

Note

The available units are nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s") and evaluations per seconds ("eps") and relative runtime compared to the best median time ("relative").

If the multcomp package is available a statistical ranking is calculated and displayed in compact letter display from in the cld column.

Author(s)

Olaf Mersmann

See Also

boxplot.microbenchmark and autoplot.microbenchmark for a plot methods.

Examples

```
a1 <- a2 <- a3 <- a4 <- numeric(0)

res <- microbenchmark(a1 <- c(a1, 1),
                      a2 <- append(a2, 1),
                      a3[length(a3) + 1] <- 1,
                      a4[[length(a4) + 1]] <- 1,
                      times=100L)
print(res)
```

```
## Change default unit to relative runtime
options(microbenchmark.unit="relative")
print(res)
## Change default unit to evaluations per second
options(microbenchmark.unit="eps")
print(res)
```

---

summary.microbenchmark *Summarize* microbenchmark *timings.*

Description

Summarize microbenchmark timings.

Usage

```
## S3 method for class 'microbenchmark'
summary(object, unit, ...)
```

Arguments

object	An object of class microbenchmark.
unit	What unit to print the timings in. If none is given, either the unit attribute of object or the option microbenchmark.unit is used and if neither is set "t" is used.
...	Passed to print.data.frame

Value

A data.frame containing the aggregated results.

Note

The available units are nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s") and evaluations per seconds ("eps") and relative runtime compared to the best median time ("relative").

See Also

print.microbenchmark

Index