Package 'benchr'

October 12, 2022

Title High Precise Measurement of R Expressions Execution Time

Version 0.2.5

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

URL https://gitlab.com/artemklevtsov/benchr

BugReports https://gitlab.com/artemklevtsov/benchr/issues

License GPL (>= 2)

Encoding UTF-8

ByteCompile yes

LazyData yes

Imports Rcpp (>= 0.12.11), RcppProgress, stats, graphics

Suggests tinytest, ggplot2 (>= 3.3.0)

LinkingTo Rcpp, RcppProgress

SystemRequirements C++11

RoxygenNote 7.0.2

Collate 'RcppExports.R' 'order.R' 'utils.R' 'benchmark.R' 'benchr-package.R' 'boxplot.R' 'mean.R' 'plot.R' 'units.R' 'summary.R' 'print.R' 'zzz.R'

NeedsCompilation yes

Author Artem Klevtsov [aut, cre] (<https://orcid.org/0000-0003-0492-6647>), Anton Antonov [ctb], Philipp Upravitelev [ctb]

Maintainer Artem Klevtsov <a.a.klevtsov@gmail.com>

Repository CRAN

Date/Publication 2020-03-07 06:30:03 UTC

R topics documented:

---

benchmark *High Precise Measurement of R Expressions Execution Time*

Description

benchmark 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 achieve 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 measurment error.

Usage

```
benchmark(
  ...,
  times = 100L,
  order = c("random", "inorder", "block"),
  envir = parent.frame(),
  progress = TRUE,
  gcFirst = TRUE,
  gcDuring = FALSE
)
```

Arguments

...	Captures any number of unevaluated expressions passed to benchmark as named or unnamed arguments.
times	Integer. Number of times to evaluate each expression.
order	Character. The order in which the expressions are evaluated.
envir	The environment in which the expressions will be evaluated.
progress	Logical. Show progress bar during expressions evaluation.
gcFirst	Logical. Should a garbage collection be performed immediately before the timing?
gcDuring	Logical. Should a garbage collection be performed immediately before each iteration, as produced by times? (very slow)

Details

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 also raised. Should all evaluations result in one of the two error conditions described above, an error is raised.

Value

Object of class `benchmark`, which is a `data.frame` with a number of additional attributes and contains the following columns:

`expr`	The deparsed expression as passed to `benchmark` or the name of the argument if the expression was passed as a named argument.
`time`	The measured execution time of the expression in seconds. The order of the observations in the data frame is the order in which they were executed.

An object of class benchmark also contains the following attributes:

`precision`	Timer precision in seconds.
`error`	Timer error (overhead) in seconds.
`units`	Units for time intervals (by default, "s" – seconds).
`times`	Number of repeats for each measurement.
`order`	Execution regime.
`gc`	Whether garbage collection took place before each execution.

The order in which the expressions are evaluated

"random" (the default) randomizes the execution order

"inorder" executes each expression in order

"block" executes all repetitions of each expression as one block.

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

`summary.benchmark()`, `mean.benchmark()`, `print.benchmark()`, `plot.benchmark()`, `boxplot.benchmark()`

Examples

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

## Print results:
```

```
print(res)

## Plot results
boxplot(res)
```

---

benchr *High Precise Measurement of R Expressions Execution Time*

Description

Package **benchr** provides an infrastructure (framework) for precise measurement of R expressions execution time.

Details

To measure execution time, **benchr** provides function `benchmark()`, as well as a number of additional methods for analysis and representation of results.

For precise time measurement we use a cross-platform monotone clock, provided by C++11 standard in header file `chrono`. The timer accuracy depends on the implementation by the compiler in use, the OS and the hardware. Most commonly, the precision is one micro- or nanosecond. We provide the opportunity to get the timer accuracy (time interval between two consecutive timer ticks) via function `timer_precision()`. This accuracy is also listed in the output of implicit or explicit `print` call.

We estimate the timer overhead before the actual measurement by running multiple (`2*10^5` by default) calls to an empty function. By doing that, we not only estimate the overhead, but also produce a warm-up effect on the processor, taking it out from idle state. After the actual measurement results are adjusted by the timer overhead.

Time intervals are measured in seconds and stored as `long double`, which lets us capture a wide range of possible values from `.Machine$double.xmin` to `.Machine$double.xmax`. This is quite enough to operate both within very small (nanoseconds) and very big time frames (e.g. the maximum time interval possible is `.Machine$double.xmax / 3600` hours).

It should be noted that the R session is not an isolated container with strictly bounded resources, therefore the execution time can be influenced by various factors, which may lead to outliers. In order to increase measurement reliability, we repeat executions multiple times (100 repetitions for each expression by default). This approach allows to collect enough data for statistical analysis in time difference.

We have also implemented several execution regimes in order to minimize the probability of systematic errors in measurements. By default, a random order of execution is being used. There is also a block order of execution, when the first expression is repeated a fixed number of times, then the second and so on. In such regime one can decrease the influence of allocators and garbage collection, since the memory is allocated only at the beginning of each block. The third option is to execute expressions in the order, provided by the user.

Note that we do not make any checks regarding return objects, i.e. one can compare not only algorithms with the same result, but also the same algorithm with different input parameters (e.g. input data sets of different size).

We also do not check whether the expressions are `language` objects (see `is.language()`) and do not coerce to that type.

Package options

accessible through function arguments. We allow to modify these parameters via package options. We have tried to set optimal default values, which you may consider changing in some cases. Here's a complete list of package options:

'`benchr.warmup`' Number of iterations for timer overhead estimation (`2*10^5` by default).

'`benchr.print_details`' Whether additional information on the measurement parameters should be displayed (`FALSE` by default).

'`benchr.use_ggplot`' Whether **ggplot2** package should be used to produce plots, if the package is installed (`TRUE` by default).

Author(s)

Maintainer: Artem Klevtsov <a.a.klevtsov@gmail.com> (ORCID)

Other contributors:

- Anton Antonov <tonytonov@gmail.com> [contributor]
- Philipp Upravitelev <upravitelev@gmail.com> [contributor]

See Also

Useful links:

- https://gitlab.com/artemklevtsov/benchr
- Report bugs at https://gitlab.com/artemklevtsov/benchr/issues

Examples

```
# Benchmark expressions
res <- benchmark(
  rep(1:10, each = 10),
  rep.int(1:10, rep.int(10, 10))
)
# Aggregated statistics
mean(res)
summary(res)
# Plot results
boxplot(res)
```

`boxplot.benchmark` *Boxplot method for the* `benchmark` *timings*

Description

Displays measurement results as box plots, with R expressions on X axis and execution time on Y axis.

Usage

```
## S3 method for class 'benchmark'
boxplot(
  x,
  units = "auto",
  log = TRUE,
  xlab,
  ylab,
  horizontal = FALSE,
  violin = FALSE,
  ...
)
```

Arguments

x	An object of class benchmark.
units	Character. The units to be used in printing the timings. The available units are nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s").
log	Logical. Should times be plotted on log scale?
xlab	Character. X axis label.
ylab	Character. Y axis label.
horizontal	Logical. If set to TRUE, X and Y axes will be switched. Defaults to FALSE.
violin	Logical. Use ggplot2::geom_violin() instead ggplot2::geom_boxplot() for the ggplot plots.
...	Arguments passed on to boxplot.default().

Details

If ggplot2 package is available, it will be used. In order to switch to default boxplot from the graphics package set option benchr.use_ggplot to FALSE: options(benchr.use_ggplot = FALSE).

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

plot.benchmark()

Examples

```
timings <- benchmark(
  rchisq(100, 0), rchisq(100, 1), rchisq(100, 2),
  rchisq(100, 3), rchisq(100, 5),
  times = 1000L
)
boxplot(timings)
```

mean.benchmark *Mean method for the* benchmark *timings.*

Description

This method computes aggregated statistics (sample mean and confidence intervals) for each expression.

Usage

```
## S3 method for class 'benchmark'
mean(x, trim = 0.05, conf.level = 0.95, relative = "mean", ...)
```

Arguments

x	An object of class benchmark.
trim	Numeric. The fraction (0 to 0.5) of observations to be trimmed before the mean is computed.
conf.level	Numeric. Confidence level of the interval.
relative	Character. The name or index of the column whose values will be used to compute relative timings.
...	Not currently used.

Value

The method returns a data.frame with additional attributes, which contains these columns:

expr	The deparsed expression as passed to benchmark or the name of the argument if the expression was passed as a named argument.
mean	Sample mean for timing results.
trimmed	Trimmed sample mean for timing results (a fraction of observations to be trimmed is defined by the argument trim).
lw.ci	Lower boundary for the confidence level (confidence level is specified by the argument conf.level).
up.ci	Upper boundary for the confidence level (confidence level is specified by the argument conf.level).
relative	Relative difference across expressions compared to a minimal value in the column, specified by the argument relative.

Additional attributes:

units	Units for time intervals.
conf.level	Confidence level.
trim	Fraction of observations that was trimmed before the trimmed mean was computed.

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

summary.benchmark()

Examples

```
timings <- benchmark(
  rchisq(100, 0), rchisq(100, 1), rchisq(100, 2),
  rchisq(100, 3), rchisq(100, 5),
  times = 1000L
)
mean(timings)
```

---

plot.benchmark *Plot method for the* benchmark *timings.*

Description

Displays measurement results as a scatter plot, with R expressions on X axis and execution time on Y axis. Each expression is highlighted by its own colour.

Usage

```
## S3 method for class 'benchmark'
plot(x, units = "auto", log = TRUE, xlab, ylab, ...)
```

Arguments

x	An object of class benchmark.
units	Character. The units to be used in printing the timings. The available units are nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s").
log	Logical. Should times be plotted on log scale?
xlab	Character. X axis label.
ylab	Character. Y axis label.
...	Not currently used.

Details

If ggplot2 package is available, it will be used. In order to switch to default boxplot from the graphics package set option benchr.use_ggplot to FALSE: options(benchr.use_ggplot = FALSE).

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

boxplot.benchmark()

Examples

```
timings <- benchmark(
  rchisq(100, 0), rchisq(100, 1), rchisq(100, 2),
  rchisq(100, 3), rchisq(100, 5),
  times = 1000L
)
plot(timings)
```

---

print.benchmark *Print method for the* benchmark *timings.*

Description

This is universal method of measurement results representation, which can be called either implicitly or explicitly. The method uses summary method to compute aggregated statistics for benchmarking results. print also provides additional information about the timer precision and overhead, the execution regime and the number of repeats.

Usage

```
## S3 method for class 'benchmark'
print(
  x,
  units = "auto",
  order = "none",
  relative = "median",
  details = FALSE,
  ...
)

## S3 method for class 'summaryBenchmark'
print(x, units = "auto", order = "none", ...)
```

Arguments

x	An object of class benchmark, summaryBenchmark or meanBenchmark.
units	Character. The units to be used in printing the timings. The available units are nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s").
order	Character. Order results according to this column of the output.

relative	Character. The name or index of the column whose values will be used to compute relative timings.
details	Logical. Show additional detauls about measurement process.
...	Arguments passed on to print.data.frame().

Value

Apart from the table output produced by summary, the method also prints additional information about the benchmarking process (with details = TRUE):

Timer precision	Timer precision in seconds.
Timer error	Timer error (overhead) in seconds.
Replications	Number of repeats for each expression.
Expressions order	Execution regime.
Garbage collection	Whether garbage collection took place before each execution.

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

summary.benchmark(), mean.benchmark()

Examples

```
a1 <- a2 <- a3 <- a4 <- numeric(0)
res <- benchmark(
  a1 <- c(a1, 1),
  a2 <- append(a2, 1),
  a3[length(a3) + 1] <- 1,
  a4[[length(a4) + 1]] <- 1
)
print(res)
```

---

summary.benchmark *Summary method for the* benchmark *timings.*

Description

This method computes aggregated statistics (quantiles, means and sums) for each expression.

Usage

```
## S3 method for class 'benchmark'
summary(object, relative = "median", ...)
```

Arguments

object	An object of class benchmark.
relative	Character. The name or index of the column whose values will be used to compute relative timings.
...	Not currently used.

Value

The method returns a data.frame with additional attributes, which contains these columns:

expr	The deparsed expression as passed to benchmark or the name of the argument if the expression was passed as a named argument.
n.eval	Number of successful measurements.
min	Minimal timing measurement for this expression.
lw.qu	First quartile of measurements for this expression.
mean	Sample mean of measurements for this expression.
median	Sample median of measurements for this expression.
up.qu	Third quartile of measurements for this expression.
max	Maximal timing measurement for this expression.
total	Total (summed) measured time for this expression.
relative	Relative difference across expressions compared to a minimal value in the column, specified by the argument relative.

Additional attributes:

units	Units for time intervals.

Author(s)

Artem Klevtsov <a.a.klevtsov@gmail.com>

See Also

mean.benchmark()

Examples

```
timings <- benchmark(
  rchisq(100, 0), rchisq(100, 1), rchisq(100, 2),
  rchisq(100, 3), rchisq(100, 5),
  times = 1000L
)
summary(timings)
```

`timer_precision` *Get timer precision.*

Description

Get timer precision.

Usage

```
timer_precision()
```

Value

Number of seconds from one clock tick to the next.

Index