Package: rbenchmark (via r-universe)

June 15, 2026

Type Package

Title Benchmarking Routine for R

Version 1.0.1

Date 2026-06-14

Description Benchmarking of arbitrary R code, inspired by similar functionality in a Perl module, is implemented via a simple wrapper around system.time(). Given a specification of the benchmarking process (counts of replications, evaluation environment) and an arbitrary number of expressions, the function evaluates each of the expressions in the specified environment, replicating the evaluation as many times as specified, and returning the results conveniently wrapped into a data frame.

License GPL (>= 2)

URL https://github.com/eddelbuettel/rbenchmark

BugReports https://github.com/eddelbuettel/rbenchmark/issues

Repository https://eddelbuettel.r-universe.dev

Date/Publication 2026-06-14 22:16:15 UTC

RemoteUrl https://github.com/eddelbuettel/rbenchmark

RemoteRef HEAD

RemoteSha 808caa5c8bdc359f2a0a6201ad3608d49d1771b9

Contents

rbenchmark-package *Benchmarking Routine for R*

Description

Benchmarking of arbitrary R code, inspired by similar functionality in a Perl module, is implemented via a simple wrapper around system.time(). Given a specification of the benchmarking process (counts of replications, evaluation environment) and an arbitrary number of expressions, the function evaluates each of the expressions in the specified environment, replicating the evaluation as many times as specified, and returning the results conveniently wrapped into a data frame.

Package Content

Index: This package was not yet installed at build time.

Maintainer

Dirk Eddelbuettel <edd@debian.org>

Author(s)

Wacek Kusnierczyk [aut], Dirk Eddelbuettel [aut, cre] (ORCID: <https://orcid.org/0000-0001-6419-907X>), Berend Hasselman [aut]

benchmark *A simple routine for benchmarking R code*

Description

`benchmark` is a simple wrapper around `system.time`.

Given a specification of the benchmarking process (counts of replications, evaluation environment) and an arbitrary number of expressions, `benchmark` evaluates each of the expressions in the specified environment, replicating the evaluation as many times as specified, and returning the results conveniently wrapped into a data frame.

Usage

```
benchmark(..., columns = c("test", "replications", "elapsed", "relative",
                           "user.self", "sys.self", "user.child", "sys.child"),
          order = "test", replications = 100, environment = parent.frame(),
          relative = "elapsed")
```

Arguments

`...`	captures any number of unevaluated expressions passed to benchmark as named or unnamed arguments.
`columns`	a character or integer vector specifying which columns should be included in the returned data frame (see below).
`order`	a character or integer vector specifying which columns should be used to sort the output data frame. Any of the columns that can be specified for `columns` (see above) can be used, even if it is not included in `columns` and will not appear in the output data frame. If `order=NULL`, the benchmarks will appear in the order of the replication counts and expressions provided in the call to `benchmark`, without sorting.
`replications`	a numeric vector specifying how many times an expression should be evaluated when the runtime is measured. If `replications` consists of more than one value, each expression will be benchmarked multiple times, once for each value in replications.
`environment`	the environment in which the expressions will be evaluated.
`relative`	the name or index of the column whose values will be used to compute relative timings (see below). If `relative` is not given, it defaults to `'elapsed'`.

Details

The parameters `columns`, `order`, `replications`, and `environment` are optional and have the following default values:

- columns = c('test', 'replications', 'elapsed', 'relative', 'user.self', 'sys.self', 'user.child', 'sys.child')

  By default, the returned data frame will contain all columns generated internally in benchmark. These named columns will contain the following data:

  - `test`: a character string naming each individual benchmark. If the corresponding expression was passed to benchmark in a named argument, the name will be used; otherwise, the expression itself converted to a character string will be used.
  - `replications`: a numeric vector specifying the number of replications used within each individual benchmark.
  - `elapsed`, `user.self`, `sys.self`, `user.child`, and `sys.child` are columns containing values reported by system.time; see Sec. 7.1 Operating system access in The R language definition, or see system.time.
  - `relative`: a column containing benchmark values relative to the shortest benchmark value. The benchmark values used in this computation are taken from the column specified with the `relative` argument.

- order = 'test'

  By default, the data frame is sorted by the column test (the labels of the expressions or the expressions themselves; see above).

- replications = 100

  By default, each expression will be benchmarked once, and will be evaluated 100 times within the benchmark.

- environment = parent.frame()
  By default, all expressions will be evaluated in the environment in which the call to benchmark is made.
- relative = 'elapsed'
  By default, relative timings are given based on values from the column 'elapsed'.

Value

The value returned from a call to `benchmark` is a data frame with rows corresponding to individual benchmarks, and columns as specified above.

An individual benchmark corresponds to a unique combination (see below) of an expression from `...` and a replication count from `replications`; if there are n expressions in `...` and m replication counts in `replication`, the returned data frame will consist of n*m rows, each corresponding to an individual, independent (see below) benchmark.

If either `...` or `replications` contain duplicates, the returned data frame will contain multiple benchmarks for the involved expression-replication combinations. Note that such multiple benchmarks for a particular expression-replication pair will, in general, have different timing results, since they will be evaluated independently (unless the expressions perform side effects that can influence each other's performance).

Note

Not all expressions, if passed as unnamed arguments, will be cast to character strings as you might expect:

```
benchmark({x = 5; 1:x^x})
# the benchmark will be named '{'
```

benchmark performs no smart argument-parameter matching. Any named argument whose name is not exactly 'replications', 'environment', 'columns', or 'order' will be treated as an expression to be benchmarked:

```
benchmark(1:10^5, repl=1000)
# there will be a benchmark named 'repl'
```

Author(s)

Wacek Kusnierczyk, Dirk Eddelbuettel, Berend Hasselman

Examples

```
library(rbenchmark)

# Example 1
# Benchmarking the allocation of one 10^6-element numeric vector,
# by default replicated 100 times
```

```
benchmark(1:10^6)

# simple test functions used in subsequent examples
random.array <- function(rows, cols, dist=rnorm) array(dist(rows*cols), c(rows, cols))
random.replicate <- function(rows, cols, dist=rnorm) replicate(cols, dist(rows))

# Example 2
# Benchmarking an expression multiple times with the same replication count,
# output with selected columns only
benchmark(replications=rep(100, 3),
          random.array(100, 100),
          random.replicate(100, 100),
          columns=c('test', 'elapsed', 'replications'))

# Example 3
# Benchmarking two named expressions with three different replication
# counts, output sorted by test name and replication count,
# with additional column added after the benchmark
within(benchmark(rep=random.replicate(100, 100),
                 arr=random.array(100, 100),
                 replications=10^(1:3),
                 columns=c('test', 'replications', 'elapsed'),
                 order=c('test', 'replications')), {
                     average = elapsed/replications
                 })

# Example 4
# Benchmarking a list of arbitrary predefined expressions
tests <- list(rep=expression(random.replicate(100, 100)),
              arr=expression(random.array(100, 100)))
do.call(benchmark, c(tests, list(replications=100,
                                 columns=c('test', 'elapsed', 'replications'),
                                 order='elapsed')))
```

Index