Package: RcppClock (via r-universe)

September 5, 2024

Type Package

Title Seamless 'Rcpp' Benchmarking

Version 1.2

Date 2021-11-24

Author Zach DeBruine

Maintainer Zach DeBruine <zacharydebruine@gmail.com>

Description Time the execution of overlapping (or not) 'Rcpp' code chunks using convenient methods, seamlessly write timing results to an 'RcppClock' object in the R global environment, and summarize and/or plot the results in R using 'RcppClock' methods.

License GPL (>= 2)

Imports Rcpp (>= 1.0.7), ggplot2

LinkingTo Rcpp

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

RemoteUrl https://github.com/zdebruine/rcppclock

RemoteRef HEAD

RemoteSha ba4e664cb22dba93549ddd12536a422d544f6132

Contents

`fibonacci` *Simple RcppClock example*

Description

Time the computation of fibonacci numbers

Usage

```
fibonacci(n, reps = 10L)
```

Arguments

`n`	vector giving integers for which to compute the fibonacci sum
`reps`	number of replicates for timing

Details

The function being timed is the following:

```
int fib(int n) { return ((n <= 1) ? n : fib(n - 1) + fib(n - 2)); }
```

Runtime for computations less than `n = 25` is nearly unmeasurable.

Examples

```
fibonacci(n = c(25:35), reps = 10)
# this function creates a global environment variable "clock"
#   that is an S3 RcppClock object
clock
plot(clock)
summary(clock, units = "ms")
```

`RcppClock` *RcppClock*

Description

Time Rcpp functions and summarize, print, and plot runtime statistics

Usage

```
## S3 method for class 'RcppClock'
summary(object, units = "auto", ...)

## S3 method for class 'RcppClock'
print(x, ...)

## S3 method for class 'RcppClock'
plot(x, ...)
```

Arguments

object	RcppClock object
units	nanoseconds ("ns"), microseconds ("us"), milliseconds ("ms"), seconds ("s"), or auto ("auto")
...	arguments to other functions
x	RcppClock object

Details

See https://github.com/zdebruine/RcppClock/readme.md for information on how to use the package.

RcppClock functions

See the RcppClock README on `https://github.com/zdebruine/RcppClock#readme` for basic usage examples.

When the Rcpp `Rcpp::clock::stop()` method is called in Rcpp code, an S3 `RcppClock` object will be created in the global environment. This object contains three methods:

- `summary`: computes runtime summary statistics and returns a `data.frame`
- `print`: runs `summary` and then prints the resulting `data.frame`
- `plot`: a ggplot2 violin plot with jitter points showing runtimes for each expression

The `fibonacci` function is a simple example of how to use RcppClock. See the source code on `github.com/zdebruine/RcppClock/src/fibonacci.cpp`

See Also

`fibonacci`

Examples

```
library(RcppClock)
fibonacci(n = 25:35, reps = 10)
# this function creates a global environment variable "clock"
#   that is an S3 RcppClock object
clock
plot(clock)
summary(clock, units = "ms")

## Not run:
# this is the Rcpp code behind the "fibonacci" example function

```{Rcpp}
//[[Rcpp::depends(RcppClock)]]
#include <RcppClock.h>

int fib(int n) {
return ((n <= 1) ? n : fib(n - 1) + fib(n - 2));
```
```

```
}

//[[Rcpp::export]]
void fibonacci(std::vector<int> n, int reps = 10) {
  Rcpp::Clock clock;
  while(reps-- > 0){
    for(auto number : n){
      clock.tick("fib" + std::to_string(number));
      fib(number);
      clock.tock("fib" + std::to_string(number));
    }
 }
 clock.stop("clock");
}
```

End(Not run)

Index