Package: hashr (via r-universe)

October 28, 2026

Maintainer Mark van der Loo <mark.vanderloo@gmail.com>

License GPL-3

Title Hash R Objects to Integers Fast

Type Package

LazyLoad yes

Description Apply an adaptation of the SuperFastHash algorithm to any R object. Hash whole R objects or, for vectors or lists, hash R objects to obtain a set of hash values that is stored in a structure equivalent to the input. See <http://www.azillionmonkeys.com/qed/hash.html> for a description of the hash algorithm.

Version 0.1.4

URL https://github.com/markvanderloo/hashr

BugReports https://github.com/markvanderloo/hashr/issues

Suggests tinytest

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation yes

Author Mark van der Loo [aut, cre], Paul Hsieh [ctb]

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

Date/Publication 2021-09-02 06:30:02 UTC

RemoteUrl https://github.com/cran/hashr

RemoteRef HEAD

RemoteSha 6683ade02cd21f0ec6d13ffeaa36bb7cd5add059

Contents

hashr-package 2
hash 2

Index **5**

hashr-package *Hash R Objects Quickly*

Description

This package exports Paul Hsies's SuperFastHash C-code to R. It can be used to hash either whole R objects or, for vectors or lists, R objects can be hashed recursively so one obtains a set of hash values that is stored in a structure equivalent to the input.

hash *Hash R objects to 32bit integers*

Description

Hash R objects to 32bit integers

Usage

```
hash(x, ...)

## Default S3 method:
hash(x, ...)

## S3 method for class 'character'
hash(
  x,
  recursive = TRUE,
  what = c("string", "pointer"),
  nthread = getOption("hashr_num_thread"),
  ...
)

## S3 method for class 'list'
hash(x, recursive = TRUE, nthread = getOption("hashr_num_thread"), ...)
```

Arguments

x	Object to hash
...	Arguments to be passed to other methods. In particular, for the default method, these arguments are passed to serialize.
recursive	hash each element separately?
what	Hash the string or the pointer to the string (faster, but not reproducible over R sessions)
nthread	maximum number of threads used.

Details

The default method serializes the input to a single raw vector which is then hashed to a single signed integer. This is also true for character vectors when recursive=FALSE. When recursive=TRUE each element of a character vector is hashed separately, based on the underlying char representation in C.

Parallelization

On systems supporting openMP, this function is able to use multiple cores. By default, a sensible number of cores is chosen. See the entry on OpenMP Support in the writing R extensions manual to check whether your system supports it.

Hash function

The hash function used is Paul Hsieh's' SuperFastHash function which is described on his website. As the title of the algorithm suggests, this hashing algorithm is not aimed to be used as a secure hash, and it is probably a bad idea to use it for that purpose.

Examples

```
# hash some complicated R object (not a list).
m <- lm(height ~ weight, data=women)
hash(m)

# hash a character vector element by element:
x <- c("Call any vegetable"
     , "and the chances are good"
     , "that the vegetable will respond to you")
hash(x)

# hash a character vector as one object:
hash(x, recursive=FALSE)

# hash a list recursively
L <- strsplit(x," ")
hash(L)

# recursive really means recursive, so nested lists are recursed over:
L <- list(
  x = 10
  , y = list(
    foo = "bob"
    , bar = lm(Sepal.Width ~ Sepal.Length, data=iris)
  )
)

hash(L)
hash(L,recursive=FALSE)
```

Index

hash, 2
hashr-package, 2

raw, *3*

serialize, *2*, *3*