Package 'R.cache'

May 3, 2025

Version 0.17.0

Depends R (>= 2.14.0)

Imports utils, R.methodsS3 (>= 1.8.1), R.oo (>= 1.24.0), R.utils (>= 2.10.1), digest (>= 0.6.13)

Title Fast and Light-Weight Caching (Memoization) of Objects and Results to Speed Up Computations

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <henrikb@braju.com>

Description Memoization can be used to speed up repetitive and computational expensive function calls. The first time a function that implements memoization is called the results are stored in a cache memory. The next time the function is called with the same set of parameters, the results are momentarily retrieved from the cache avoiding repeating the calculations. With this package, any R object can be cached in a key-value storage where the key can be an arbitrary set of R objects. The cache memory is persistent (on the file system).

License LGPL (>= 2.1)

LazyLoad TRUE

URL https://github.com/HenrikBengtsson/R.cache

BugReports https://github.com/HenrikBengtsson/R.cache/issues

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2025-05-02 22:20:02 UTC

Contents

R.cache-package 2
addMemoization 3
evalWithMemoization 4
getCacheRootPath 6

loadCache 7
memoizedCall 8
saveCache 9
setCacheRootPath 11

Index **12**

`R.cache-package` *Package R.cache*

Description

Memoization can be used to speed up repetitive and computational expensive function calls. The first time a function that implements memoization is called the results are stored in a cache memory. The next time the function is called with the same set of parameters, the results are momentarily retrieved from the cache avoiding repeating the calculations. With this package, any R object can be cached in a key-value storage where the key can be an arbitrary set of R objects. The cache memory is persistent (on the file system).

Installation and updates

To install this package and all of its dependent packages, do: `install.packages("R.cache")`

To get started

loadCache, saveCache Methods for loading and saving objects from and to the cache.

getCacheRootPath, setCacheRootPath Methods for getting and setting the directory where cache files are stored.

How to cite this package

Whenever using this package, please cite [1] as

```
Bengtsson, H. The R.oo package - Object-Oriented Programming with References Using
Standard R Code, Proceedings of the 3rd International Workshop on Distributed
Statistical Computing (DSC 2003), ISSN 1609-395X, Hornik, K.; Leisch, F. & Zeileis,
A. (ed.), 2003
```

Wishlist

Here is a list of features that would be useful, but which I have too little time to add myself. Contributions are appreciated.

- Add a functionality to identify cache files that are no longer of use. For now, there is an extra header field for arbitrary comments which can be used, but maybe more formal fields are useful, e.g. keywords, user, etc?

If you consider implement some of the above, make sure it is not already implemented by downloading the latest "devel" version!

Related work

See also the **filehash** package, and the `cache()` function in the **Biobase** package of Bioconductor.

License

The releases of this package is licensed under LGPL version 2.1 or newer.

References

[1] H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. https://www.r-project.org/conferences/DSC-2003/Proceedings/

Author(s)

Henrik Bengtsson

`addMemoization` *Creates a copy of an existing function such that its results are memoized*

Description

Creates a copy of an existing function such that its results are memoized.

Usage

```
## Default S3 method:
addMemoization(fcn, envir=parent.frame(), ...)
```

Arguments

`fcn`	A `function` (or the name of a function) that should be copied and have memoization added.
`envir`	The `environment` from where to look for the function.
`...`	Additional arguments for controlling the memoization, i.e. all arguments of `memoizedCall`() that are not passed to `do.call`().

Details

The new function is setup such that the the memoized call is done in the environment of the caller (the parent frame of the function).

If the `function` returns `NULL`, that particular function call is *not* memoized.

Value

Returns a function.

Author(s)

Henrik Bengtsson

See Also

The returned function utilized memoizedCall() internally.

---

evalWithMemoization *Evaluates an R expression with memoization*

Description

Evaluates an R expression with memoization such that the same objects are assigned to the current environment and the same result is returned, if any.

Usage

```
evalWithMemoization(expr, key=NULL, ..., envir=parent.frame(),
  drop=c("srcref", "srcfile", "wholeSrcref"), force=FALSE)
```

Arguments

expr	The expression to be evaluated.
key	Additional objects to uniquely identify the evaluation.
...	Additional arguments passed to loadCache() and saveCache().
envir	The environment in which the expression should be evaluated.
drop	character vector of expr attributes to drop. The default is to drop all source-reference information.
force	If TRUE, existing cached results are ignored.

Value

Returns the value of the evaluated expr expression, if any.

Author(s)

Henrik Bengtsson

See Also

Internally, eval() is used to evaluate the expression.

Examples

```
for (kk in 1:5) {
  cat(sprintf("Iteration #%d:\n", kk))
  res <- evalWithMemoization({
    cat("Evaluating expression...")
    a <- 1
    b <- 2
    c <- 4
    Sys.sleep(1)
    cat("done\n")
    b
  })
  print(res)

  # Sanity checks
  stopifnot(a == 1 && b == 2 && c == 4)

  # Clean up
  rm(a, b, c)
} # for (kk ...)


## OUTPUTS:
## Iteration #1:
## Evaluating expression...done
## [1] 2
## Iteration #2:
## [1] 2
## Iteration #3:
## [1] 2
## Iteration #4:
## [1] 2
## Iteration #5:
## [1] 2


############################################################
# WARNING
############################################################
# If the expression being evaluated depends on
# "input" objects, then these must be be specified
# explicitly as "key" objects.
for (ii in 1:2) {
  for (kk in 1:3) {
    cat(sprintf("Iteration #%d:\n", kk))
    res <- evalWithMemoization({
      cat("Evaluating expression...")
      a <- kk
      Sys.sleep(1)
      cat("done\n")
      a
    }, key=list(kk=kk))
```

```
    print(res)

    # Sanity checks
    stopifnot(a == kk)

    # Clean up
    rm(a)
  } # for (kk ...)
} # for (ii ...)

## OUTPUTS:
## Iteration #1:
## Evaluating expression...done
## [1] 1
## Iteration #2:
## Evaluating expression...done
## [1] 2
## Iteration #3:
## Evaluating expression...done
## [1] 3
## Iteration #1:
## [1] 1
## Iteration #2:
## [1] 2
## Iteration #3:
## [1] 3
```

`getCacheRootPath` *Gets the root path to the file cache directory*

Description

Gets the root path to the file cache directory.

Usage

```
## Default S3 method:
getCacheRootPath(defaultPath=NULL, ...)
```

Arguments

`defaultPath`	The default path, if no user-specified directory has been given.
`...`	Not used.

Value

Returns the path as a `character` string.

Author(s)

Henrik Bengtsson

See Also

Too set the directory where cache files are stored, see setCacheRootPath().

Examples

```
print(getCacheRootPath())
```

---

loadCache *Loads data from file cache*

Description

Loads data from file cache, which is unique for an optional key object.

Usage

```
## Default S3 method:
loadCache(key=NULL, sources=NULL, suffix=".Rcache", removeOldCache=TRUE, pathname=NULL,
  dirs=NULL, ..., onError=c("warning", "error", "message", "quiet", "print"))
```

Arguments

key	An optional object from which a hexadecimal hash code will be generated and appended to the filename.
sources	Optional source objects. If the cache object has a timestamp older than one of the source objects, it will be ignored and removed.
suffix	A character string to be appended to the end of the filename.
removeOldCache	If TRUE and the cache is older than the sources, the cache file is removed, otherwise not.
pathname	The pathname to the cache file. If specified, arguments key and suffix are ignored. Note that this is only needed in order to read a cache file for which the key is unknown, for instance, in order to investigate an unknown cache file.
dirs	A character vector constituting the path to the cache subdirectory (of the *cache root directory* as returned by getCacheRootPath()) to be used. If NULL, the path will be the cache root path.
...	Not used.
onError	A character string specifying what the action is if an exception is thrown.

Details

The hash code calculated from the key object is a 32 characters long hexadecimal MD5 hash code. For more details, see getChecksum().

Value

Returns an R object or NULL, if cache does not exist.

Author(s)

Henrik Bengtsson

See Also

saveCache().

Examples

```
simulate <- function(mean, sd) {
  # 1. Try to load cached data, if already generated
  key <- list(mean, sd)
  data <- loadCache(key)
  if (!is.null(data)) {
    cat("Loaded cached data\n")
    return(data);
  }

  # 2. If not available, generate it.
  cat("Generating data from scratch...")
  data <- rnorm(1000, mean=mean, sd=sd)
  Sys.sleep(1)             # Emulate slow algorithm
  cat("ok\n")
  saveCache(data, key=key, comment="simulate()")

  data;
}

data <- simulate(2.3, 3.0)
data <- simulate(2.3, 3.5)
data <- simulate(2.3, 3.0) # Will load cached data

# Clean up
file.remove(findCache(key=list(2.3,3.0)))
file.remove(findCache(key=list(2.3,3.5)))
```

---

memoizedCall *Calls a function with memoization*

Description

Calls a function with memoization, that is, caches the results to be retrieved if the function is called again with the exact same arguments.

Usage

```
## Default S3 method:
memoizedCall(what, ..., envir=parent.frame(), force=FALSE, sources=NULL, dirs=NULL)
```

Arguments

what	The function to be called, or a character string specifying the name of the function to be called, cf. do.call().
...	Arguments passed to the function.
envir	The environment in which the function is evaluated.
force	If TRUE, any cached results are ignored, otherwise not.
sources, dirs	Optional arguments passed to loadCache() and saveCache().

Details

If the function returns NULL, that particular function call is *not* memoized.

Value

Returns the result of the function call.

Author(s)

Henrik Bengtsson

See Also

Internally, loadCache() is used to load memoized results, if available. If not available, then do.call() is used to evaluate the function call, and saveCache() is used to save the results to cache.

---

saveCache *Saves data to file cache*

Description

Saves data to file cache, which is unique for an optional key object.

Usage

```
## Default S3 method:
saveCache(object, key=NULL, sources=NULL, suffix=".Rcache", comment=NULL, pathname=NULL,
  dirs=NULL, compress=NULL, ...)
```

Arguments

object	The object to be saved to file.
key	An optional object from which a hexadecimal hash code will be generated and appended to the filename.
sources	Source objects used for comparison of timestamps when cache is loaded later.
suffix	A character string to be appended to the end of the filename.
comment	An optional character string written in ASCII at the beginning of the file.
pathname	(Advanced) An optional character string specifying the pathname to the cache file. If not specified (default), a unique one is automatically generated from arguments key and suffix among other things.
dirs	A character vector constituting the path to the cache subdirectory (of the *cache root directory* as returned by getCacheRootPath()) to be used. If NULL, the path will be the cache root path.
compress	If TRUE, the cache file will be saved using gzip compression, otherwise not.
...	Additional argument passed to save().

Value

Returns (invisible) the pathname of the cache file.

Compression

The saveCache() method saves a compressed cache file (with filename extension *.gz) if argument compress is TRUE. The loadCache() method locates (via findCache()) and loads such cache files as well.

Author(s)

Henrik Bengtsson

See Also

For more details on how the hash code is generated etc, loadCache().

Examples

```
## Not run: For an example, see ?loadCache
```

setCacheRootPath *Sets the root path to the file cache directory*

Description

Sets the root path to the file cache directory.

Usage

```
## Default S3 method:
setCacheRootPath(path=NULL, ...)
```

Arguments

path	The path.
...	Not used.

Value

Returns (invisibly) the old root path.

Author(s)

Henrik Bengtsson

See Also

getCacheRootPath().

Index

∗ **IO**
  addMemoization, 3
  evalWithMemoization, 4
  getCacheRootPath, 6
  loadCache, 7
  memoizedCall, 8
  saveCache, 9
  setCacheRootPath, 11
∗ **package**
  R.cache-package, 2
∗ **programming**
  addMemoization, 3
  evalWithMemoization, 4
  getCacheRootPath, 6
  loadCache, 7
  memoizedCall, 8
  saveCache, 9
  setCacheRootPath, 11

addMemoization, 3

character, *4*, *6, 7*, *9, 10*

do.call, *3*, *9*

environment, *3, 4*, *9*
eval, *4*
evalWithMemoization, 4
expression, *4*

findCache, *10*
function, *3, 4*, *9*

getCacheRootPath, *2*, 6, *7*, *10, 11*
getChecksum, *7*

loadCache, *2*, *4*, 7, *9, 10*

memoizedCall, *3, 4*, 8

NULL, *3*, *7–10*

R.cache *(*R.cache-package*)*, 2
R.cache-package, 2

save, *10*
saveCache, *2*, *4*, *8, 9*, 9
setCacheRootPath, *2*, *7*, 11

TRUE, *4*, *7*, *9, 10*

vector, *7*, *10*