Package 'tinycache'

August 7, 2026

Type Package

Title Cache Objects in Disk or Memory

Version 0.1

Suggests tinytest

Imports utils

Depends R(>= 4.0.0)

Description Reuse objects with a long processing time either by storing those in disk or memory. Uses caching to identify R objects (e.g., data frames, plots, etc.) and allows repeated access to those. Created with the specific goal of skipping the waiting time for summary tables obtained from large 'SQL' tables. It is extensible to other uses, such as caching plots in 'Tabler' dashboards to reduce waiting times.

License Apache License 2.0

BugReports https://github.com/pachadotdev/tinycache/issues

URL https://pacha.dev/tinycache/

Encoding UTF-8

NeedsCompilation yes

LinkingTo cpp4r

Author Mauricio Vargas Sepulveda [aut, cre] (ORCID: <https://orcid.org/0000-0003-1017-7574>)

Maintainer Mauricio Vargas Sepulveda <m.vargas.sepulveda@gmail.com>

Repository CRAN

Date/Publication 2026-08-07 16:50:02 UTC

Contents

dcache *Create a Disk Cache*

Description

A lightweight disk-backed key-value cache: each value is stored as its own `.rds` file inside `dir`. Pruning is not throttled, and eviction metadata (age, last-access time) comes directly from the files' own modification times rather than being tracked separately.

Usage

```
dcache(
  dir = NULL,
  max_size = Inf,
  max_age = Inf,
  max_n = Inf,
  evict = c("lru", "fifo"),
  missing = key_missing(),
  destroy_on_finalize = FALSE
)
```

Arguments

`dir`	Directory used to store the cached files. If `NULL` (the default), a new temporary directory is created and used.
`max_size`	See `mcache`.
`max_age`	See `mcache`.
`max_n`	See `mcache`.
`evict`	See `mcache`.
`missing`	See `mcache`.
`destroy_on_finalize`	If `TRUE`, the cache directory and all of its contents are deleted from disk when the returned object is garbage collected. Default `FALSE`.

Value

A `dcache` object with the same methods as `mcache`, plus `destroy()`, which deletes the cache directory from disk.

Examples

```
cache <- dcache(dir = tempdir())

fit_model <- function(n, cache) {
  key <- hash(n)
  cached <- cache$get(key)
```

```
  if (!is.key_missing(cached)) {
    return(cached)
  }

  set.seed(123)
  mydata <- data.frame(x = seq_len(n), y = seq_len(n) * 2 + rnorm(n))
  mycoef <- coef(lm(y ~ x, data = mydata))

  cache$set(key, mycoef)
  mycoef
}

fit_model(100, cache) # computed and cached
fit_model(100, cache) # reused from disk, no recomputation
```

---

hash *Hash an R Object*

Description

Computes a fast, well-mixed 128-bit (32 hex character) hash of an R object, similar to `rlang::hash()`. Unlike hashing `serialize(x, connection = NULL)` (see `hash_raw()`), this walks the object's type, length, data bytes, and attributes directly, so it never allocates a serialized byte buffer. Supports atomic vectors (logical, integer, double, character, raw), lists, and their attributes meaning that it covers data frames, factors, and nested lists thereof. Other types (e.g. closures, environments, S4 objects) are not supported.

Usage

```
hash(x)
```

Arguments

x	An R object to hash.

Value

A 32-character lowercase hex string.

hash_raw *Hash Raw Bytes (C++)*

Description

Computes a fast, well-mixed 128-bit (32 hex character) hash of a raw vector, e.g. the output of `serialize()`. For hashing an R object directly, without serializing it first, see `hash()`.

Usage

```
hash_raw(x)
```

Arguments

`x`	A raw vector, e.g. the output of `serialize()`.

Value

A 32-character lowercase hex string.

key_missing *Key Missing Sentinel*

Description

`key_missing()` returns a sentinel object indicating that a requested key was not found in a cache. `is.key_missing()` tests whether a value is this sentinel. Mirrors `cachem::key_missing()` / `fastmap::key_missing()`.

Usage

```
key_missing()
is.key_missing(x)
```

Arguments

`x`	An object to test.

Value

`key_missing()` returns an object of class `"key_missing"`. `is.key_missing()` returns `TRUE` or `FALSE`.

mcache *Create a Memory Cache*

Description

A lightweight in-memory key-value cache. Values are stored directly in an environment (not serialized), so as long as an object is cached it will not be garbage collected.

Usage

```
mcache(
  max_size = Inf,
  max_age = Inf,
  max_n = Inf,
  evict = c("lru", "fifo"),
  missing = key_missing()
)
```

Arguments

max_size	Maximum size of the cache, in bytes, as reported by object.size. Use Inf (the default) for no limit.
max_age	Maximum age of an object, in seconds, before it is evicted. Use Inf (the default) for no limit.
max_n	Maximum number of objects allowed in the cache. Use Inf (the default) for no limit.
evict	Eviction policy used when max_n or max_size is exceeded: "lru" (least recently used, the default) or "fifo" (first in, first out).
missing	Value returned by get() when key is not present in the cache. Defaults to a key_missing() sentinel; test for it with is.key_missing().

Value

An mcache object with methods get(key, missing), set(key, value), exists(key), remove(key), keys(), size(), reset(), and prune().

Examples

```
cache <- mcache()

fit_model <- function(n, cache) {
  key <- hash(n)
  cached <- cache$get(key)
  if (!is.key_missing(cached)) {
    return(cached)
  }
```

```
  set.seed(123)
  mydata <- data.frame(x = seq_len(n), y = seq_len(n) * 2 + rnorm(n))
  mycoef <- coef(lm(y ~ x, data = mydata))

  cache$set(key, mycoef)
  mycoef
}

fit_model(100, cache) # computed and cached
fit_model(100, cache) # reused from disk, no recomputation
```

Index