Package 'memtoc'

September 26, 2026

Title 'Tictoc'-Style Memory Usage Tracking

Version 0.1.1

Description Provides simple start/stop memory tracking functions tic_mem() and toc_mem() that can be nested, inspired by the 'tictoc' package.
Track RAM usage during code execution with support for logging, custom messages, nested tracking blocks, and parallel worker monitoring. Features continuous background polling to estimate peak memory usage across main process and workers. Integrates with the 'future' package ecosystem for automatic worker detection. Designed for monitoring memory consumption in parallel workflows.

License MIT + file LICENSE

URL https://github.com/jcoa05/memtoc

BugReports https://github.com/jcoa05/memtoc/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ps (>= 1.7.0), cli (>= 3.0.0), callr (>= 3.7.0)

Suggests testthat (>= 3.2.0), withr, future, parallelly, knitr, rmarkdown

Config/testthat/edition 3

Language en-US

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Juan Ocampo [aut, cre] (ORCID: <https://orcid.org/0000-0002-9353-7581>)

Maintainer Juan Ocampo <jocampo1997@hotmail.com>

Repository CRAN

Date/Publication 2026-09-26 17:10:02 UTC

Contents

mem_capabilities 2
mem_clear 3
mem_clearlog 3
mem_diagnose 4
mem_log 4
mem_parallel_info 5
mem_print_log 6
mem_recover 6
print.memtoc_result 7
tic_mem 8
toc_mem 9

Index **11**

`mem_capabilities` *Check memtoc capabilities*

Description

Tests which features are available on the current system. This is useful for understanding why certain features might be disabled.

Usage

```
mem_capabilities()
```

Value

A named logical vector with elements:

`memory_queries` Can query process memory via ps package

`background_polling`
: Can spawn background processes via callr

Examples

```
mem_capabilities()
```

mem_clear *Clear the memtoc stack*

Description

Removes all entries from the tracking stack. This is useful if an error occurred before `toc_mem()` could be called, leaving orphaned entries on the stack.

Usage

```
mem_clear()
```

Details

Also stops any running background monitors associated with orphaned entries.

Value

Invisibly returns `NULL`.

Examples

```
tic_mem("will be cleared")
mem_clear()
# Stack is now empty
```

mem_clearlog *Clear the memory tracking log*

Description

Removes all entries from the memtoc log. This does not affect the tracking stack (active tic_mem/toc_mem blocks).

Usage

```
mem_clearlog()
```

Value

Invisibly returns `NULL`.

See Also

`mem_log()` to retrieve the log before clearing

Examples

```
mem_clearlog()
tic_mem("example", interval = NULL, workers = "none")
toc_mem(log = TRUE, quiet = TRUE)
nrow(mem_log())
mem_clearlog()
nrow(mem_log())
```

`mem_diagnose` *Diagnose why background polling might not be working*

Description

Runs detailed diagnostics to identify issues with background process spawning. Useful for troubleshooting when `mem_capabilities()` shows `background_polling = FALSE`.

Usage

```
mem_diagnose()
```

Value

Invisibly returns a list with diagnostic results

Examples

```
mem_diagnose()
```

`mem_log` *Retrieve the memory tracking log*

Description

Returns a data frame containing all logged memory tracking results. Results are added to the log when `toc_mem(log = TRUE)` is called.

Usage

```
mem_log(format = c("data.frame", "list"))
```

Arguments

`format`	Character string specifying the output format.
	"data.frame" (Default) Returns a data frame with one row per logged result
	"list" Returns the raw list of memtoc_result objects

Value

A data frame (default) or list containing logged results. The data frame has columns:

msg Label from tic_mem(), or NA if none provided
mem_start Memory (RSS) in bytes at start
mem_end Memory (RSS) in bytes at end
mem_peak Peak memory in bytes
mem_change Change in memory (bytes)
elapsed Elapsed time in seconds
tic_timestamp When tic_mem() was called
toc_timestamp When toc_mem() was called

See Also

`mem_clearlog()` to clear the log, `toc_mem()` with `log = TRUE`

Examples

```
mem_clearlog()
tic_mem("step 1", interval = NULL, workers = "none")
x <- numeric(100)
toc_mem(log = TRUE, quiet = TRUE)
tic_mem("step 2", interval = NULL, workers = "none")
y <- sum(x)
toc_mem(log = TRUE, quiet = TRUE)
mem_log()
mem_clearlog()
```

`mem_parallel_info` *Get information about current parallel setup*

Description

Returns diagnostic information about the detected parallel backend and any workers that can be monitored.

Usage

```
mem_parallel_info()
```

Value

A list with parallel backend information

Examples

```
info <- mem_parallel_info()
info$main_pid
info$worker_pids
```

mem_print_log *Print formatted log output*

Description

Displays the memory tracking log in a human-readable format, similar to how tictoc's tic.log() output can be printed with writeLines().

Usage

```
mem_print_log()
```

Value

Invisibly returns a character vector of formatted log lines.

Examples

```
mem_clearlog()
tic_mem("step 1", interval = NULL, workers = "none")
x <- numeric(100)
toc_mem(log = TRUE, quiet = TRUE)
tic_mem("step 2", interval = NULL, workers = "none")
y <- sum(x)
toc_mem(log = TRUE, quiet = TRUE)
mem_print_log()
mem_clearlog()
```

mem_recover *Recover data from a crashed monitoring session*

Description

Attempts to recover memory samples from a previous session that crashed or was interrupted before toc_mem() was called. The monitor periodically saves checkpoints to disk, so some data may be recoverable.

Usage

```
mem_recover(pid = NULL, path = NULL)
```

Arguments

pid	Process ID for a checkpoint in the current session's temporary directory. If NULL, lists available recovery files in that directory.
path	Direct path to a recovery file. Overrides pid if provided.

Details

Checkpoints are stored in the current R session's temporary directory. After restarting R, use `path` to locate a surviving checkpoint in the previous session's temporary directory; recovery is not guaranteed if that directory has been removed. Normal completion removes checkpoints. Nested blocks have separate files and can be recovered using `path`.

Value

If pid or path is provided, returns a data frame of recovered samples or NULL if not found. If both are NULL, returns a list of available recovery files with their metadata.

Examples

```
mem_recover()

# Recover a small example checkpoint from a temporary file.
local({
  path <- tempfile(fileext = ".rds")
  on.exit(unlink(path))
  samples <- data.frame(
    timestamp = Sys.time(),
    pid = Sys.getpid(),
    rss = 1024^2
  )
  saveRDS(samples, path)
  recovered <- mem_recover(path = path)
  stopifnot(identical(recovered, samples))
  recovered
})
```

`print.memtoc_result` *Print method for memtoc_result objects*

Description

Print method for memtoc_result objects

Usage

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

Arguments

`x`	A memtoc_result object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

`tic_mem` *Start memory tracking*

Description

Begins a memory tracking block. Call `toc_mem()` to end the block and see the results. Multiple calls to `tic_mem()` can be nested, and each `toc_mem()` will match with the most recent unmatched `tic_mem()`.

Usage

```
tic_mem(msg = NULL, quiet = TRUE, interval = 1, workers = "auto")
```

Arguments

`msg`	Optional character string label for this tracking block. This label is displayed in the output from `toc_mem()` and stored in the log if `log = TRUE` is passed to `toc_mem()`.
`quiet`	Logical. If `TRUE` (default), no message is printed when tracking starts. Set to `FALSE` to see a startup message.
`interval`	Numeric. Polling interval in seconds for background monitoring. Set to `NULL` or `0` to disable background polling and use snapshot-only mode (faster but only captures start/end memory). Default is `1` second.
`workers`	Worker specification for parallel monitoring: • `"auto"` (default): Automatically detect future workers and child processes • `"none"`: Only monitor the main R process • `"children"`: Monitor main process and all child processes • Integer vector: Explicit list of PIDs to monitor

Details

When `interval` is set (default), a background R process is spawned to sample memory usage. The reported peak is the maximum observed sample; allocations between samples may be missed. Starting and running the background process adds overhead.

For very short operations (under 1 second), consider using `interval = NULL` to avoid the background process startup overhead (~200ms).

Parallel Worker Monitoring:

When using `future` for parallel processing, memtoc can automatically detect and monitor worker processes. Set `workers = "auto"` to enable this feature. The trajectory will include memory samples from all workers, and the result will show aggregate statistics.

Value

Invisibly returns the timestamp when tracking started.

See Also

`toc_mem()` to stop tracking, `mem_log()` to retrieve logged results, `mem_parallel_info()` to check parallel backend status

Examples

```
# Track a small allocation using start and end snapshots.
tic_mem("small allocation", interval = NULL, workers = "none")
x <- numeric(1000)
toc_mem()

# Nested blocks are stopped in reverse order.
tic_mem("outer", interval = NULL, workers = "none")
tic_mem("inner", interval = NULL, workers = "none")
y <- sum(x)
toc_mem()
toc_mem()
```

---

toc_mem *Stop memory tracking and report results*

Description

Ends a memory tracking block started by `tic_mem()` and reports the results. By default, prints a summary message showing peak memory, current memory, and elapsed time.

Usage

```
toc_mem(log = FALSE, quiet = FALSE)
```

Arguments

`log`	Logical. If `TRUE`, stores the result in an internal log that can be retrieved later with `mem_log()`. Default is `FALSE`.
`quiet`	Logical. If `TRUE`, suppresses the output message. Default is `FALSE` (message is shown).

Value

Invisibly returns a `memtoc_result` object (a list) containing:

`msg`	The label passed to `tic_mem()`, or NULL
`mem_start`	Memory (RSS) in bytes at start (main process)
`mem_end`	Memory (RSS) in bytes at end (main process)

mem_peak	Peak memory during the interval (total across all processes)
mem_change	Change in memory (end - start) in bytes
elapsed	Elapsed time in seconds
tic_timestamp	POSIXct timestamp when tic_mem() was called
toc_timestamp	POSIXct timestamp when toc_mem() was called
trajectory	Data frame of memory samples (if background polling was used)
n_samples	Number of samples collected
n_workers	Number of worker processes monitored (0 if main process only)
worker_stats	Data frame with per-worker peak memory (if workers monitored)

See Also

tic_mem() to start tracking, mem_log() to retrieve logged results

Examples

```
tic_mem("small allocation", interval = NULL, workers = "none")
x <- numeric(1000)
result <- toc_mem()
result$elapsed
result$mem_peak
result$n_samples

# Brief background polling; sample availability depends on startup time.
tic_mem("polling", interval = 0.1, workers = "none")
Sys.sleep(0.3)
polled <- toc_mem()
if (!is.null(polled$trajectory)) {
  head(polled$trajectory)
}
```

Index

mem_capabilities, 2
mem_clear, 3
mem_clearlog, 3
mem_clearlog(), *5*
mem_diagnose, 4
mem_log, 4
mem_log(), *3*, *9, 10*
mem_parallel_info, 5
mem_parallel_info(), *9*
mem_print_log, 6
mem_recover, 6

print.memtoc_result, 7

tic_mem, 8
tic_mem(), *9, 10*
toc_mem, 9
toc_mem(), *5*, *8, 9*