Package: Rfmalloc (via r-universe)

June 3, 2026

Title Persistent ALTREP Vectors

Version 0.1.0

Description Provides experimental file-backed, ALTREP-style vector allocation for R using the 'fmalloc' library. The package supports persistent and scratch runtimes, durable reference-based serialization, explicit vector lifecycle management, and multiple runtime handles for working with several backing files in one R process.

License GPL (>= 2)

Author Sounkou Mahamane Toure [aut, cre], Kenichi Yasukata [cph] (fmalloc), Wolfram Gloger [cph] (ptmalloc3), Free Software Foundation, Inc. [cph] (selected GNU C Library support files)

Maintainer Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Encoding UTF-8

Imports methods

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests bench, lobstr, tinytest

SystemRequirements GNU make, POSIX threads

URL https://github.com/sounkou-bioinfo/Rfmalloc, https://sounkou-bioinfo.github.io/Rfmalloc/

Depends R (>= 4.4.0)

BugReports https://github.com/sounkou-bioinfo/Rfmalloc/issues

Config/pak/sysreqs make

Repository https://sounkou-bioinfo.r-universe.dev

Date/Publication 2026-06-03 21:41:46 UTC

RemoteUrl https://github.com/sounkou-bioinfo/Rfmalloc

RemoteRef HEAD

RemoteSha f3099c9e37316dd176fd8cae706a59b75bf760ce

Contents

`Rfmalloc-package` *Rfmalloc: Memory-Mapped File Allocation for R*

Description

Rfmalloc provides experimental memory-mapped file allocation capabilities for R using a patched copy of the fmalloc library. The current package exposes ALTREP file-backed vector allocation for logical, integer, numeric, raw, complex, character, and list vectors with fmalloc payload storage.

Main Functions

`open_fmalloc` Open an explicit fmalloc runtime handle.

`init_fmalloc` Open and install a default fmalloc runtime.

`create_fmalloc_vector` Create vectors using fmalloc.

`create_fmalloc_matrix` Create matrix-shaped fmalloc vectors.

`create_fmalloc_array` Create array-shaped fmalloc vectors.

`create_fmalloc_data_frame` Create data.frames from fmalloc-backed columns.

`as_fmalloc_matrix` Convert fmalloc vectors to matrix-shaped objects.

`as_fmalloc_array` Convert fmalloc vectors to array-shaped objects.

`as_fmalloc_data_frame` Convert fmalloc-backed objects to a data.frame.

`list_fmalloc_allocations` List persistent allocation catalog records.

`diagnose_fmalloc_runtime` Summarize persistent allocation catalog state and runtime diagnostics.

`cleanup_fmalloc` Request cleanup of an fmalloc runtime.

Current Scope

- ALTREP file-backed allocation for logical, integer, numeric, raw, complex, character, and list vectors. List elements are restricted to `NULL` or Rfmalloc-backed vectors from the same runtime.
- Large allocations spanning multiple fmalloc chunks.
- Multiple runtime handles in one R process.
- Persistent and scratch runtime modes.
- Reference serialization for persistent fixed-width atomic and character ALTREP vectors.
- Fmalloc-backed ALTREP subset copies for vector indexing operations.
- An in-file allocation catalog for persistent vectors.
- A C-callable API and installed header for other packages.
- Native lifetime tracking so runtime mappings outlive reachable vectors allocated from them.
- Runtime and catalog diagnostics for planning recovery and operational cleanup.

Known Limitations

- ALTREP-backed dispatch now covers core `Ops`, `Summary`, `Math`, `Math2`, and matrix `rowSums/colSums/rowMeans/colMea` workflows through S3 methods for common vector/matrix usage.
- Explicit base-fallback boundaries are:
  - `rowSums()`, `colSums()`, `rowMeans()`, and `colMeans()` when the input is not an exact 2D matrix or `dims != 1L`; these cases now emit a warning and call the corresponding `base::` reducer.
  - Scalar or zero-length results from `Summary`, `Math`, and `Math2` generics (for example `sum(x)` returning a single value) are returned as ordinary R scalars by design.
- Full operator- and method-family coverage is still incomplete for all R generics. Some advanced families may still materialize ordinary R objects in a few edge cases.

Future Work

Future work includes view-based subset representations, catalog compaction and reset tooling, metadata storage for attributes on persisted elements, robust nested-list reference validation, and compaction of recovery metadata.

Author(s)

Maintainer: Sounkou Mahamane Toure <sounkoutoure@gmail.com>

Other contributors:

- Kenichi Yasukata (fmalloc) [copyright holder]
- Wolfram Gloger (ptmalloc3) [copyright holder]
- Free Software Foundation, Inc. (selected GNU C Library support files) [copyright holder]

See Also

Useful links:

- https://github.com/sounkou-bioinfo/Rfmalloc
- https://sounkou-bioinfo.github.io/Rfmalloc/
- Report bugs at https://github.com/sounkou-bioinfo/Rfmalloc/issues

---

`as_fmalloc_array` *Convert a vector to fmalloc array metadata*

Description

Returns an existing vector re-typed as an array by installing array dimensions (and optional dimnames) as metadata.

Usage

```
as_fmalloc_array(x, dim = NULL, dimnames = NULL, copy = TRUE)
```

Arguments

`x`	A vector.
`dim`	Target dimension vector.
`dimnames`	Optional `dimnames` for the resulting array.
`copy`	If TRUE (default), allocate a new fmalloc-backed array object. If FALSE, install metadata in place on the same fmalloc ALTREP payload without allocation (this also updates any aliases of `x`).

Value

An array object, backed by the same payload when `copy = FALSE`.

`as_fmalloc_data_frame` *Convert to data.frame for fmalloc vectors*

Description

Thin convenience wrapper around `data.frame()`.

Usage

```
as_fmalloc_data_frame(
  ...,
  row.names = NULL,
  check.names = TRUE,
  stringsAsFactors = FALSE
)
```

Arguments

`...`	Columns or objects to include in the frame.
`row.names`	Optional row names for the frame.
`check.names`	Whether to enforce syntactic column names.
`stringsAsFactors`	Deprecated: retained for compatibility.

Value

A `data.frame` containing the supplied columns.

`as_fmalloc_matrix` *Convert a vector to fmalloc matrix metadata*

Description

Returns an existing vector re-typed as a matrix by installing matrix dimensions (and optional dimnames) as metadata.

Usage

```
as_fmalloc_matrix(x, nrow = NULL, ncol = NULL, dimnames = NULL, copy = TRUE)
```

Arguments

`x`	A vector.
`nrow`	Optional target row count.
`ncol`	Optional target column count.
`dimnames`	Optional `dimnames` for the resulting matrix.
`copy`	If TRUE (default), allocate a new fmalloc-backed matrix object. If FALSE, install metadata in place on the same fmalloc ALTREP payload without allocation (this also updates any aliases of `x`).

Value

A matrix object, backed by the same payload when `copy = FALSE`.

cleanup_fmalloc *Clean Up fmalloc*

Description

Requests cleanup of an fmalloc runtime. If vectors allocated from the runtime are still reachable, the native mapping is kept alive until those vectors are garbage-collected.

Usage

```
cleanup_fmalloc(runtime = NULL)
```

Arguments

runtime	Optional runtime handle returned by open_fmalloc(). If not supplied, the current default runtime is cleaned up.

Value

NULL (invisibly)

Examples

```
## Not run:
init_fmalloc("data.bin")
v <- create_fmalloc_vector("integer", 100)
rm(v)
gc()
cleanup_fmalloc()

## End(Not run)
```

create_fmalloc_array *Create Array Using fmalloc*

Description

Creates an fmalloc-backed ALTREP array in a single step by allocating vector storage and installing array dimensions (and optional dimnames).

Usage

```
create_fmalloc_array(
  type = "integer",
  dim,
  dimnames = NULL,
  runtime = NULL,
  zero_initialize = TRUE
)
```

Arguments

type	Character string specifying the vector type. Supported values are the same as for create_fmalloc_vector().
dim	Integer dimension vector.
dimnames	Optional dimnames for the array.
runtime	Optional runtime handle returned by open_fmalloc(). If not supplied, the default runtime established by init_fmalloc() is used.
zero_initialize	Logical scalar passed through to payload allocation. See create_fmalloc_vector() for semantics.

Value

An fmalloc-backed ALTREP array.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"))
a <- create_fmalloc_array("numeric", dim = c(2L, 3L), runtime = rt)
cleanup_fmalloc(rt)

## End(Not run)
```

---

create_fmalloc_data_frame

Construct data.frame from fmalloc columns

Description

Thin constructor wrapper around data.frame() that keeps fmalloc vectors as column payloads.

Usage

```
create_fmalloc_data_frame(
  ...,
  row.names = NULL,
  check.names = TRUE,
  stringsAsFactors = FALSE
)
```

Arguments

`...`	Columns to include in the frame.
`row.names`	Optional row names for the frame.
`check.names`	Whether to enforce syntactic column names.
`stringsAsFactors`	Deprecated: retained for compatibility.

Value

A `data.frame` with the provided columns.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"))
x <- create_fmalloc_vector("integer", 3, runtime = rt)
y <- create_fmalloc_vector("character", 3, runtime = rt)
x[] <- 1:3
y[] <- c("a", "b", "c")
df <- create_fmalloc_data_frame(x = x, y = y)
cleanup_fmalloc(rt)

## End(Not run)
```

`create_fmalloc_matrix` *Create Matrix Using fmalloc*

Description

Creates an fmalloc-backed ALTREP matrix in a single step by allocating vector storage and installing matrix dimensions (and optional dimnames).

Usage

```
create_fmalloc_matrix(
  type = "integer",
  nrow,
  ncol,
  dimnames = NULL,
  runtime = NULL,
  zero_initialize = TRUE
)
```

Arguments

type	Character string specifying the vector type. Supported values are the same as for create_fmalloc_vector().
nrow	Integer number of rows.
ncol	Integer number of columns.
dimnames	Optional dimnames list for the matrix.
runtime	Optional runtime handle returned by open_fmalloc(). If not supplied, the default runtime established by init_fmalloc() is used.
zero_initialize	Logical scalar passed through to payload allocation. See create_fmalloc_vector() for semantics.

Value

An fmalloc-backed ALTREP matrix object.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"))
m <- create_fmalloc_matrix("integer", nrow = 2, ncol = 3, runtime = rt)
cleanup_fmalloc(rt)

## End(Not run)
```

---

create_fmalloc_vector *Create Vector Using fmalloc*

Description

Creates an ALTREP vector using a file-backed fmalloc runtime. The returned object is ALTREP from creation time. Fixed-width atomic payload bytes are allocated directly with fmalloc, and ALTREP duplication and vector subsetting keep copy-on-write copies fmalloc-backed without using R's non-API Rf_allocVector3() path.

Usage

```
create_fmalloc_vector(
  type = "integer",
  length,
  runtime = NULL,
  zero_initialize = TRUE
)
```

Arguments

type	Character string specifying the vector type. Supported values are "logical", "integer", "numeric"/"double", "raw", "complex", "character", and "list". Fixed-width atomic types expose a direct writable fmalloc DATAPTR; character vectors store string bytes in fmalloc and materialize R CHARSXP values on demand; list vectors use ALTREP element access with an R-visible reference sidecar for GC safety and only accept NULL or fmalloc-backed vectors from the same runtime as elements. Persistent list containers are serialized by nested reference states when all elements are recoverable from the same runtime.
length	Integer specifying the non-negative length of the vector to create.
runtime	Optional runtime handle returned by open_fmalloc(). If not supplied, the default runtime established by init_fmalloc() is used.
zero_initialize	Logical scalar. If TRUE (default), newly allocated payload bytes are zero-initialized. Set FALSE to skip initialization for faster large allocations when you will fully initialize values yourself.

Value

A vector of the specified type and length, allocated using fmalloc.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"))
v <- create_fmalloc_vector("integer", 1000, runtime = rt)
cleanup_fmalloc(rt)

## End(Not run)
```

---

destroy_fmalloc_vector

Explicitly destroy a fmalloc vector

Description

Releases runtime bookkeeping for a single fmalloc ALTREP vector immediately. In scratch mode, payload memory is immediately reclaimed. In persistent mode, the vector payload is retained by default so existing on-disk state remains durable; optional `unsafe = TRUE` reclaims payload memory and marks metadata as non-recoverable.

Usage

```
destroy_fmalloc_vector(x, unsafe = FALSE)
```

Arguments

`x`	Fmalloc ALTREP vector to destroy.
`unsafe`	Whether to physically free persistent payload bytes. Unsafe destroy is intended for short-lived scratch-like cleanup and will mark the catalog entry as non-recoverable.

Details

Explicit destroy fails when a vector is still referenced by another fmalloc list vector as a child.

Value

Logical value indicating whether a live vector was destroyed.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"), mode = "persistent")
v <- create_fmalloc_vector("integer", 10, runtime = rt)
destroy_fmalloc_vector(v)

## End(Not run)
```

`diagnose_fmalloc_runtime` *Diagnose fmalloc runtime state*

Description

Returns diagnostic metadata for an open runtime handle, including lightweight runtime attributes, the current allocation catalog, and a catalog-level summary useful for estimating reclaimable/fragmented payload regions.

Usage

```
diagnose_fmalloc_runtime(runtime = NULL)
```

Arguments

runtime	Optional runtime handle returned by open_fmalloc(). If not supplied, the current default runtime is used.

Value

A named list with three components:

- runtime: runtime metadata such as file path, UUID, mode, catalog counters, live vectors, and reference state;
- catalog: the full allocation catalog returned by list_fmalloc_allocations();
- summary: a compact set of computed diagnostics and an explicit compaction status note.

See Also

list_fmalloc_allocations()

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"), mode = "persistent")
x <- create_fmalloc_vector("integer", 4, runtime = rt)
y <- create_fmalloc_vector("logical", 2, runtime = rt)
diagnose_fmalloc_runtime(rt)
cleanup_fmalloc(rt)

## End(Not run)
```

---

fmalloc_reduction_methods

Matrix reduction helpers for fmalloc-backed matrices

Description

These S3 methods preserve current fmalloc behavior for matrix summary/reduction operations while returning ordinary R vectors for small results.

Usage

```
rowSums(x, na.rm = FALSE, dims = 1L)

colSums(x, na.rm = FALSE, dims = 1L)

rowMeans(x, na.rm = FALSE, dims = 1L)

colMeans(x, na.rm = FALSE, dims = 1L)
```

Arguments

x	A matrix-like object.
na.rm	Logical scalar controlling NA removal.
dims	Numeric scalar for dimensions.

Details

These implementations keep managed execution for 2D fmalloc matrices with dims = 1L. For unsupported shapes or dims values (for example, non-2D arrays or dims != 1L), the methods warn and delegate to the base R implementations (base::rowSums, base::colSums, base::rowMeans, and base::colMeans).

Value

The reduction result, as either an ordinary R object or a fmalloc vector when result length exceeds getOption("Rfmalloc.reduce_result_length", 1e6).

---

init_fmalloc *Initialize fmalloc*

Description

Compatibility wrapper that opens an fmalloc runtime and installs it as the package default runtime used by create_fmalloc_vector() when no explicit runtime is supplied.

Usage

```
init_fmalloc(filepath, size_gb = NULL, mode = c("persistent", "scratch"))
```

Arguments

filepath	Character string specifying the file path for fmalloc data.
size_gb	Numeric value specifying the size of the backing file in GB (optional). If not specified, uses the package default size for new files or the existing file size.
mode	Runtime mode. "persistent" keeps committed vector payloads in the backing file and serializes fixed-width atomic vectors by reference. "scratch" uses the backing file as a large temporary allocation arena and serializes vectors by value.

Details

For new code, prefer open_fmalloc() and pass the returned runtime handle to create_fmalloc_vector().

Value

Logical indicating whether the file was newly initialized.

Examples

```
## Not run:
alloc_file <- tempfile(fileext = ".bin")
init_fmalloc(alloc_file)
v <- create_fmalloc_vector("integer", 1000)
cleanup_fmalloc()
unlink(alloc_file)

## End(Not run)
```

---

list_fmalloc_allocations

List Persistent fmalloc Allocations

Description

Returns the in-file allocation catalog for a persistent fmalloc runtime. The catalog is stored in the backing file and records physical allocation metadata used to validate serialized persistent references.

Usage

```
list_fmalloc_allocations(runtime = NULL)
```

Arguments

`runtime`	Optional runtime handle returned by `open_fmalloc()`. If not supplied, the default runtime established by `init_fmalloc()` is used.

Details

For successful recovery, look at the `state` column:

- `"committed"`: valid serialized payload exists for that record;
- `"tombstone"`: the payload has been destroyed and is non-recoverable unless the runtime remains open and referenced directly by an existing SEXP;
- other transient states are internal and are generally not expected.

`recoverable` indicates whether the record can be reopened via serialized reference metadata. `payload_offset == 0` or `payload_nbytes == 0` generally indicates a non-payload entry.

Value

A data frame with one row per catalog record and columns describing the catalog record offset, generation, state, vector type, length, payload offset, payload byte size, flags, and whether the record is recoverable by reference serialization.

Examples

```
## Not run:
rt <- open_fmalloc(tempfile(fileext = ".bin"))
v <- create_fmalloc_vector("integer", 10, runtime = rt)
list_fmalloc_allocations(rt)
cleanup_fmalloc(rt)

## End(Not run)
```

`open_fmalloc` *Open an fmalloc Runtime*

Description

Opens a file-backed fmalloc runtime and returns an external-pointer handle. Multiple handles to the same path share a single in-process runtime while the underlying file-backed runtime remains open. Runtime mode controls whether vector payloads are durable persistent allocations or scratch allocations that can be returned to fmalloc when their ALTREP handles are garbage-collected.

Usage

```
open_fmalloc(filepath, size_gb = NULL, mode = c("persistent", "scratch"))
```

Arguments

`filepath`	Character string specifying the file path for fmalloc data.
`size_gb`	Numeric value specifying the size of the backing file in GB (optional). If not specified, uses the package default size for new files or the existing file size.
`mode`	Runtime mode. `"persistent"` keeps committed vector payloads in the backing file and serializes fixed-width atomic vectors by reference. `"scratch"` uses the backing file as a large temporary allocation arena and serializes vectors by value.

Value

An external pointer of class `fmalloc_runtime`.

Index