

Package: jlview (via r-universe)

May 23, 2026

Title Zero-Copy Julia to R Array Bridge via ALTREP

Version 0.1.0

Description Provides zero-copy R views of Julia-owned arrays by implementing ALTREP (Alternative Representations) classes that return pointers directly into Julia's memory. The package integrates with 'JuliaCall' and uses C-level finalizers for safe cross-runtime garbage collection.

License MIT + file LICENSE

URL <https://github.com/tanaylab/jlview>

BugReports <https://github.com/tanaylab/jlview/issues>

Depends R (>= 4.0.0)

SystemRequirements Julia (>= 1.6, <https://julialang.org/downloads/>)

Imports JuliaCall, Matrix, methods

Suggests knitr, matrixStats, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository <https://tanaylab.r-universe.dev>

Date/Publication 2026-03-24 08:38:22 UTC

RemoteUrl <https://github.com/tanaylab/jlview>

RemoteRef HEAD

RemoteSha e4360fb75577bf452846c7bc9020be35f85afdd7

Contents

is_jlvview	2
jlvview	2
jlvview_gc_pressure	3
jlvview_info	4
jlvview_named_matrix	4
jlvview_named_vector	5
jlvview_release	6
jlvview_set_gc_threshold	6
jlvview_sparse	7
with_jlvview	7

Index	9
--------------	----------

is_jlvview	<i>Check if an object is a jlvview ALTREP vector</i>
------------	--

Description

Check if an object is a jlvview ALTREP vector

Usage

```
is_jlvview(x)
```

Arguments

x	An R object
---	-------------

Value

TRUE if x is a jlvview ALTREP vector, FALSE otherwise

jlvview	<i>Create a zero-copy R view of a Julia array</i>
---------	---

Description

Creates an ALTREP vector (or matrix, if 2D+) backed by Julia memory. The resulting R object shares the same memory as the Julia array, avoiding data copying. Modifications to the R object trigger copy-on-write (unless `writeable = TRUE`).

Usage

```
jlvview(julia_array, writeable = FALSE, names = NULL, dimnames = NULL)
```

Arguments

julia_array	A JuliaObject referencing a Julia array
writeable	If TRUE, allow R to write directly to Julia's memory. Use with caution — this enables shared mutation. Default FALSE.
names	Optional character vector of names to attach to the result. Attached atomically during construction to avoid ALTREP materialization.
dimnames	Optional list of dimnames to attach to the result. Attached atomically during construction to avoid ALTREP materialization.

Value

An ALTREP vector backed by Julia memory, or a standard R vector if the Julia type is not supported for zero-copy.

Examples

```
## Not run:
JuliaCall::julia_setup()
# Create a Julia array and view it in R without copying
JuliaCall::julia_command("x = randn(1000)")
x <- jlvview(JuliaCall::julia_eval("x"))
sum(x) # operates directly on Julia memory

## End(Not run)
```

jlvview_gc_pressure *Get current GC pressure information*

Description

Returns the current amount of Julia memory pinned by jlvview objects and the threshold at which forced GC is triggered.

Usage

```
jlvview_gc_pressure()
```

Value

A list with pinned_bytes and threshold

jview_info	<i>Get information about a jview object</i>
------------	---

Description

Returns metadata about a jview ALTREP vector including the Julia element type, length, writeability, and release status.

Usage

```
jview_info(x)
```

Arguments

x	A jview ALTREP vector
---	-----------------------

Value

A named list with components:

type Julia element type (e.g., "Float64")

length Number of elements

writable Whether the view allows direct writes

released Whether the view has been released

materialized Whether COW materialization has occurred

jview_named_matrix	<i>Create a zero-copy R view of a named Julia matrix</i>
--------------------	--

Description

Creates a zero-copy ALTREP view of a Julia NamedArray matrix, preserving row and column names as dimnames.

Usage

```
jview_named_matrix(julia_named_matrix)
```

Arguments

julia_named_matrix	A JuliaObject referencing a Julia NamedArray matrix
--------------------	---

Value

An ALTREP matrix with dimnames set from the Julia NamedArray

Examples

```
## Not run:
JuliaCall::julia_setup()
JuliaCall::julia_command("using NamedArrays")
m <- JuliaCall::julia_eval('NamedArray(randn(3,2), (["a","b","c"], ["x","y"]))')
x <- jview_named_matrix(m)
rownames(x) # returns c("a", "b", "c")
colnames(x) # returns c("x", "y")

## End(Not run)
```

jview_named_vector *Create a zero-copy R view of a named Julia vector*

Description

Creates a zero-copy ALTREP view of a Julia NamedArray vector, preserving the axis names.

Usage

```
jview_named_vector(julia_named_array)
```

Arguments

```
julia_named_array  
                  A JuliaObject referencing a Julia NamedArray vector
```

Value

An ALTREP vector with names set from the Julia NamedArray

Examples

```
## Not run:
JuliaCall::julia_setup()
JuliaCall::julia_command("using NamedArrays")
v <- JuliaCall::julia_eval('NamedArray([1.0, 2.0, 3.0], (["a", "b", "c"],))')
x <- jview_named_vector(v)
names(x) # returns c("a", "b", "c")

## End(Not run)
```

jlvview_release	<i>Explicitly release a jlvview object</i>
-----------------	--

Description

Unpins the Julia array immediately, freeing memory without waiting for R's garbage collector. After release, accessing the data will error.

Usage

```
jlvview_release(x)
```

Arguments

x	A jlvview ALTREP vector
---	-------------------------

Value

Invisible NULL

jlvview_set_gc_threshold	<i>Set the GC pressure threshold</i>
--------------------------	--------------------------------------

Description

When total pinned bytes exceeds this threshold, jlvview forces an R garbage collection to reclaim stale ALTREP objects. Default is 10GB.

Usage

```
jlvview_set_gc_threshold(bytes)
```

Arguments

bytes	Threshold in bytes (numeric)
-------	------------------------------

Value

Invisible NULL

jview_sparse	<i>Create a zero-copy R sparse matrix view of a Julia SparseMatrixCSC</i>
--------------	---

Description

Creates a `dgMatrix-class` backed by a zero-copy ALTREP vector for the nonzero values (x slot). The row indices (i slot) and column pointers (p slot) are copied and shifted from 1-based (Julia) to 0-based (R) indexing in Julia, then returned as plain R integer vectors.

Usage

```
jview_sparse(julia_sparse_matrix, lazy_indices = FALSE)
```

Arguments

<code>julia_sparse_matrix</code>	A JuliaObject referencing a Julia SparseMatrixCSC. The value type must be supported for zero-copy (Float64, Float32, Int32, Int64, Int16, UInt8).
<code>lazy_indices</code>	Ignored. Retained for API compatibility only. Previously controlled lazy vs eager materialization of ALTREP index vectors, which have been removed in favor of simple copy+shift in Julia.

Value

A `dgMatrix-class` sparse matrix.

Examples

```
## Not run:
JuliaCall::julia_setup()
JuliaCall::julia_command("using SparseArrays")
m <- JuliaCall::julia_eval("sprand(Float64, 100, 50, 0.1)")
s <- jview_sparse(m)
class(s) # "dgMatrix"

## End(Not run)
```

<code>with_jview</code>	<i>Use a jview object within a scope, releasing it on exit</i>
-------------------------	--

Description

Creates a `jview` object and ensures it is released when the scope exits, even if an error occurs. This prevents memory leaks from forgotten releases.

Usage

```
with_jlview(  
  julia_array,  
  expr,  
  writeable = FALSE,  
  names = NULL,  
  dimnames = NULL  
)
```

Arguments

julia_array	A JuliaObject referencing a Julia array
expr	An expression to evaluate with the jlview object bound to .x
writeable	Passed to jlview
names	Passed to jlview
dimnames	Passed to jlview

Value

The result of evaluating expr

Examples

```
## Not run:  
JuliaCall::julia_setup()  
JuliaCall::julia_command("big = randn(100000)")  
result <- with_jlview(JuliaCall::julia_eval("big"), {  
  c(mean(.x), sd(.x))  
})  
# .x is automatically released here  
  
## End(Not run)
```

Index

`is_jlview`, 2

`jlview`, 2, 8

`jlview_gc_pressure`, 3

`jlview_info`, 4

`jlview_named_matrix`, 4

`jlview_named_vector`, 5

`jlview_release`, 6

`jlview_set_gc_threshold`, 6

`jlview_sparse`, 7

`with_jlview`, 7