Package 'beachmat.tiledb'

November 11, 2026

Version 1.4.0

Date 2025-07-13

Title beachmat bindings for TileDB-backed matrices

Description Extends beachmat to initialize tatami matrices from TileDB-backed arrays.
This allows C++ code in downstream packages to directly call the TileDB C/C++ library to access array data,
without the need for block processing via DelayedArray.
Developers only need to import this package to automatically extend the capabilities of beachmat::initializeCpp to TileDBArray instances.

Imports methods, beachmat, tiledb, TileDBArray, DelayedArray, Rcpp

Suggests testthat, BiocStyle, knitr, rmarkdown, Matrix

LinkingTo Rcpp, assorthead, beachmat

biocViews DataRepresentation, DataImport, Infrastructure

License GPL-3

NeedsCompilation yes

VignetteBuilder knitr

SystemRequirements C++17

URL https://github.com/tatami-inc/beachmat.tiledb

BugReports https://github.com/tatami-inc/beachmat.tiledb/issues

RoxygenNote 7.3.2

Encoding UTF-8

git_url https://git.bioconductor.org/packages/beachmat.tiledb

git_branch RELEASE_3_23

git_last_commit 0953beb

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Aaron Lun [aut, cre]

Maintainer Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

Contents

initializeCpp 2
initializeOptions 3
loadIntoMemory 4

Index **6**

`initializeCpp` *Initialize TileDB-backed matrices*

Description

Initialize C++ representations of TileDB-backed matrices based on their **TileDBArray** representations.

Usage

```
## S4 method for signature 'TileDBArraySeed'
initializeCpp(
  x,
  tiledb.cache.size = getAutoBlockSize(),
  tiledb.realize = initializeOptions("realize"),
  tiledb.realize.num.threads = initializeOptions("realize.num.threads"),
  tiledb.concurrency.level = initializeOptions("concurrency.level"),
  ...
)
```

Arguments

`x`	A **TileDBArray** seed object.
`tiledb.cache.size`	Integer scalar specifying the size of the cache in bytes during data extraction from a TileDB matrix. Larger values reduce disk I/O during random access to the matrix, at the cost of increased memory usage.
`tiledb.realize`	See the `realize` option in `initializeOptions`.
`tiledb.realize.num.threads`	See the `realize.num.threads` option in `initializeOptions`.
`tiledb.concurrency.level`	See the `concurrency.level` option in `initializeOptions`.
`...`	Further arguments, ignored.

Value

An external pointer that can be used in any **tatami**-compatible function.

Author(s)

Aaron Lun

Examples

```
library(TileDBArray)
y <- matrix(runif(1000), ncol=20, nrow=50)
z <- as(y, "TileDBArray")
ptr <- initializeCpp(z)
```

---

initializeOptions *Options for TileDB matrices*

Description

Options for initializing TileDB matrices in `initializeCpp`.

Usage

```
initializeOptions(option, value)
```

Arguments

`option`	String specifying the name of the option.
`value`	Value of the option.

Details

The following options are supported:

- `realize`, a logical scalar specifying whether to load the matrix data from TileDB into memory with `loadIntoMemory`, and then cache it for future calls with `checkMemoryCache`. This avoids time-consuming disk I/O when performing multiple passes through the matrix, at the expense of increased memory usage.
- `realize.num.threads`, an integer scalar specifying the number of threads that can be used by `loadIntoMemory` outside of TileDB calls. This is only relevant when `realize=TRUE`.
- `concurrency.level`, an integer scalar specifying the number of threads that can be used by the TileDB library. Alternatively `NULL`, in which case TileDB's default (i.e., all available cores on the machine) are used. Greater performance may be achieved when the product of `realize.num.threads` and `concurrency.level` does not exceed the number of available cores.

Value

If `value` is missing, the current setting of `option` is returned.

If `value` is supplied, it is used to set the option, and the previous value of the option is invisibly returned.

Author(s)

Aaron Lun

Examples

```
initializeOptions("realize")
old <- initializeOptions("realize", TRUE) # setting to a new value
initializeOptions("realize") # new option takes affect
initializeOptions("realize", old) # setting it back
```

`loadIntoMemory` *Load a TileDB matrix into memory*

Description

Load a TileDB-backed matrix into memory as an external pointer to a **tatami**-compatible representation. This differs from the (default) behavior of `initializeCpp`, which only loads slices of the matrix on request.

Usage

```
loadIntoMemory(
  x,
  cache.size = getAutoBlockSize(),
  num.threads = 1,
  concurrency.level = NULL
)
```

Arguments

`x`	A **TileDBArray**-derived matrix or seed object.
`cache.size`	Integer scalar specifying the size of the cache in bytes during data extraction from a TileDB matrix.
`num.threads`	Integer scalar specifying the number of threads to use outside of the TileDB library.
`concurrency.level`	Integer scalar specifying the number of threads that can be used by the TileDB library. See the option of the same name in `initializeOptions` for details.

Value

An external pointer that can be used in **tatami**-based functions.

Author(s)

Aaron Lun

Examples

```
library(TileDBArray)
y <- matrix(runif(1000), ncol=20, nrow=50)
z <- as(y, "TileDBArray")
ptr <- loadIntoMemory(z)
```

Index

checkMemoryCache, *3*

initializeCpp, 2, *3*, *4*
initializeCpp,TileDBArraySeed-method
        *(*initializeCpp*)*, 2
initializeOptions, *2*, 3, *4*

loadIntoMemory, *3*, 4