Package 'alabaster.matrix'

May 29, 2024

Title Load and Save Artifacts from File

Version 1.4.0

Date 2024-02-29

License MIT + file LICENSE

Description

Save matrices, arrays and similar objects into file artifacts, and load them back into memory.
This is a more portable alternative to serialization of such objects into RDS files.
Each artifact is associated with metadata for further interpretation;
downstream applications can enrich this metadata with context-specific properties.

Depends alabaster.base

Imports methods, BiocGenerics, S4Vectors, DelayedArray (>= 0.27.2),
S4Arrays, SparseArray, rhdf5 (>= 2.47.1), HDF5Array, Matrix,
Rcpp

Suggests testthat, knitr, BiocStyle, chihaya, BiocSingular,
ResidualMatrix

LinkingTo Rcpp

VignetteBuilder knitr

RoxygenNote 7.3.1

biocViews DataImport, DataRepresentation

git_url https://git.bioconductor.org/packages/alabaster.matrix

git_branch RELEASE_3_19

git_last_commit d0aba74

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Aaron Lun [aut, cre]

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

Contents

AmalgamatedArray   *Amalgamated array class*

Description

Implements an amalgamated array, equivalent to a delayed combination of DelayedArray objects. It allows [stageObject](#) to save a combination of multiple matrices without actually aggregating their data into a single file.

Constructors

AmalgamatedArraySeed(..., along=1) accepts any number of named array-like objects and returns a AmalgamatedArraySeed. Each object corresponds to a block and should be named accordingly; names should be unique and non-empty. The along argument specifies the dimension in which matrices should be combined - for matrices, this is 1 for rows, 2 for columns.

AmalgamatedArray(..., along=1) accepts any number of named array-like objects and returns a AmalgamatedArray. Alternatively, a single AmalgamatedArraySeed may be provided in ....

Functions

componentNames(x) will return a character vector of names of component arrays in a AmalgamatedArray(Seed) object x.

extractComponents(x) will return a named list of array-like objects, corresponding to the component arrays used to construct the AmalgamatedArray(Seed) object x.

[stageObject](#)(x, dir, path, child = FALSE) will save the AmalgamatedArray x and its components into the path inside dir. Each component array is staged into its own subdirectory inside path.

Comments on usage

The AmalgamatedArraySeed is closely related to (and in fact, is a subclass of) the [DelayedAbind](#) class. This means that we can leverage many of the **DelayedArray** methods for handling the delayed bind. In theory, we could just use a DelayedAbind directly and save it with **chihaya** in [stageObject](#) (via [preserveDelayedOperations](#)(TRUE)). However, this provides fewer opportunities for tracking and manipulating the samples. It also saves the per-sample matrices into a single file, which eliminates possibilities for per-file deduplication and linking, e.g., with [recycleHdf5Files](#)(TRUE).

Author(s)

Aaron Lun

Examples

```
first <- Matrix::rsparsematrix(10, 10, 0.1)
second <- Matrix::rsparsematrix(10, 20, 0.1)
mat <- AmalgamatedArray(list(foo = first, bar = second), along=2)
mat

componentNames(mat)
out <- extractComponents(mat)
lapply(out, dim)
```

---

createRawArraySeed   *Array loading utilities*

Description

Utilities for loading an array saved by [stageObject](#).

Usage

```
.createRawArraySeed(info, project, names = TRUE)

.extractArrayDimnames(path, group, ndim)
```

Arguments

info	A named list of metadata for this array.
project	Any argument accepted by the acquisition functions, see ?[acquireFile](#). By default, this should be a string containing the path to a staging directory.
names	Logical scalar indicating whether the seed should be annotated with dimnames (if available).
path	String containing the path to the file containing said array.
group	String containing the name of the group with the dimnames.
ndim	Integer scalar specifying the number of dimensions.

Details

For `.createArraySeed`, the array should be one of:

- `hdf5_dense_array`

- `hdf5_sparse_matrix`

- `hdf5_delayed_array`

- `amalgamated_array`

For delayed arrays, the file may contain a seed array with the `"custom alabaster local array"` type. This should have a `path` dataset containing a relative path to another array in the same `project`, which is loaded and used as the seed for this delayed array. Callers can overwrite this behavior by setting `"custom alabaster local array"` in the knownArrays from **chihaya** before calling `.createRawArraySeed`.

For `.extractArrayDimnames`, `path` is expected to be a HDF5 file with a group specified by `group`. Each child of this group is a string dataset named after a (0-indexed) dimension, containing the names for that dimension.

Value

`.createRawArraySeed` returns a seed that can be used in the [DelayedArray](#) constructor.

`.extractArrayDimnames` returns a list of character vectors or `NULL`, containing the dimnames.

Author(s)

Aaron Lun

Examples

```
# Staging an array as an example:
dir <- tempfile()
dir.create(dir)
mat <- array(rpois(10000, 10), c(50, 20, 10))
meta <- stageObject(mat, dir, "whee")

# Loading it back as a DelayedArray seed:
.createRawArraySeed(meta, project=dir)
```

DelayedMask   *Delayed masking*

Description

Delayed masking of missing values, based on replacement of placeholder values with `NA`. This allows missingness to be encoded in frameworks without the same concept of NA as R.

Usage

```
DelayedMask(x, placeholder)
```

Arguments

x	An existing **DelayedArray** seed.
placeholder	Placeholder value to replace with NA. This should be of the same type as `type`(x).

Details

If `is.na`(placeholder) is true for double-precision x, masking is performed for all values of x where is.na is true. This includes both NaNs and NAs; no attempt is made to distinguish between the NaN payloads.

Currently, an error is raised for any integer x that produces non-missing values of -2^31 without a placeholder of NA_integer_. This is because R cannot distinguish the integer -2^31 from an integer-type NA.

Value

A DelayedMask object, to be wrapped in a `DelayedArray`.

Author(s)

Aaron Lun

Examples

```
original <- DelayedArray(matrix(rpois(40, lambda=2), ncol=5))
original
masked <- DelayedMask(original, 0)
DelayedArray(masked)
```

---

preserveDelayedOperations

Preserve delayed operations during staging

Description

Preserve delayed operations via **chihaya** when staging a DelayedArray with `stageObject`.

Usage

```
preserveDelayedOperations(preserve)
```

Arguments

preserve	Whether to preserve delayed operations using the **chihaya** specification.

Details

By default, any DelayedArray in `stageObject` will be saved as a new dense array or sparse matrix. However, if this option is enabled, DelayedArrays will instead be saved in the **chihaya** specification, where the delayed operations are themselves stored in the HDF5 file (see `https://ltla.github.io/chihaya` for details).

The **chihaya** specification is more complicated to parse but can be helpful in reducing disk usage. One simple example is to avoid sparsity-breaking or integer-to-float operations by storing their delayed representations in the file. If the seed matrix is derived from some immutable reference location, advanced users can even store links to that location instead of duplicating the seed data.

Value

Logical scalar indicating whether delayed operations are to be preserved by the DelayedArray method. If `preserve` is supplied, it is used to set this scalar, and the *previous* value of the scalar is invisibly returned.

Author(s)

Aaron Lun

Examples

```
preserveDelayedOperations()
old <- preserveDelayedOperations(TRUE)
preserveDelayedOperations()
preserveDelayedOperations(old)
```

---

readArray   *Read a dense array from disk*

Description

Read a dense high-dimensional array from its on-disk representation. This is usually not directly called by users, but is instead called by dispatch in `readObject`.

Usage

```
readArray(path, metadata, ...)
```

Arguments

path	String containing a path to a directory, itself created by the `saveObject` method for a dense array.
metadata	Named list of metadata for this object, see `readObject` for more details.
...	Further arguments, ignored.

Value

A dense file-backed [ReloadedArray](#).

Author(s)

Aaron Lun

See Also

["saveObject,array-method"](#), to create the directory and its contents.

Examples

```
arr <- array(rpois(10000, 10), c(50, 20, 10))
dimnames(arr) <- list(
    paste0("GENE_", seq_len(nrow(arr))),
    letters[1:20],
    NULL
)

dir <- tempfile()
saveObject(arr, dir)
readObject(dir)
```

---

readDelayedArray *Read a delayed array from disk*

Description

Read a delayed high-dimensional array from its on-disk representation. This is usually not directly called by users, but is instead called by dispatch in [readObject](#).

Usage

```
readDelayedArray(path, metadata, delayed_array.reload.args = list(), ...)
```

Arguments

path	String containing a path to a directory, itself created by the [saveObject](#) method for a delayed array.
metadata	Named list of metadata for this object, see [readObject](#) for more details.
delayed_array.reload.args	
	Named list of arguments to be passed to [reloadDelayedObject](#).
...	Further arguments, ignored.

Value

A multi-dimensional array-like object.

Author(s)

Aaron Lun

See Also

"`saveObject,DelayedArray-method`", to create the directory and its contents.

Examples

```
arr <- array(rpois(10000, 10), c(50, 20, 10))
dimnames(arr) <- list(
    paste0("GENE_", seq_len(nrow(arr))),
    letters[1:20],
    NULL
)

dir <- tempfile()
saveObject(arr, dir)
readObject(dir)
```

---

readSparseMatrix   *Read a sparse matrix from disk*

Description

Read a sparse matrix from its on-disk representation. This is usually not directly called by users, but is instead called by dispatch in `readObject`.

Usage

```
readSparseMatrix(path, metadata, ...)
```

Arguments

path	String containing a path to a directory, itself created by the `saveObject` method for a spars matrix.
metadata	Named list of metadata for this object, see `readObject` for more details.
...	Further arguments, ignored.

Value

A sparse ReloadedMatrix object.

Author(s)

Aaron Lun

See Also

"`saveObject,sparseMatrix-method`", to create the directory and its contents.

Examples

```
mat <- Matrix::rsparsematrix(100, 200, density=0.2)
rownames(mat) <- paste0("GENE_", seq_len(nrow(mat)))
dir <- tempfile()
saveObject(mat, dir)
readObject(dir)
```

---

recycleHdf5Files *Recycle existing HDF5 files*

Description

Re-use existing files in HDF5-backed arrays rather than reserializing them in `stageObject`.

Usage

```
recycleHdf5Files(recycle)
```

Arguments

recycle   Whether to recycle existing files for HDF5-backed DelayedArrays.

Details

If this options is enabled, `stageObject` will attempt to link/copy existing files for any HDF5-backed DelayedArray instances - most specifically, HDF5Array objects and H5SparseMatrix objects using the 10X format. This avoids re-serialization of the data for faster staging. It also allows advanced users to add their own customizations into the HDF5 file during staging, as long as they do not interfere with `loadArray`.

By default, this option is disabled as the properties of the existing file are not known in the general case. In particular, the file might contain other groups/datasets that are irrelevant, and use up extra disk space if copied; or confidential, and should not be stored in the staging directory. Users should only enable this option if they have full control over the generation and contents of the backing HDF5 files.

Also note that any dimnames on x will be ignored during recycling.

Value

Logical scalar indicating whether HDF5 files are to be reused. If `recycle` is supplied, it is used to set this scalar, and the *previous* value of the scalar is invisibly returned.

Author(s)

Aaron Lun

Examples

```
recycleHdf5Files()
old <- recycleHdf5Files(TRUE)
recycleHdf5Files()
recycleHdf5Files(old)
```

ReloadedArraySeed   *Reloaded* **alabaster** *array*

Description

An array that was reloaded from disk by the readObject function. This allows methods to refer to the existing on-disk representation by inspecting the `path`. For example, saveObject can just copy/link to the files instead of repeating the saving process.

Usage

```
ReloadedArraySeed(path, seed = NULL, ...)

ReloadedArray(path, seed = NULL, ...)
```

Arguments

path	String containing a path to the directory with the on-disk array representation. Alternatively an existing ReloadedArraySeed, which is returned without modification.
seed	Contents of the loaded array, e.g., as an ordinary R array, a DelayedArray or a sparse matrix. If NULL, this is obtained by calling readObject.
...	Further arguments to pass to readObject when seed=NULL.

Details

The ReloadedArraySeed is a subclass of the WrapperArraySeed and will just forward all operations to the underlying `seed`. Its main purpose is to track the `path` that was originally used to generate `seed`, which enables optimizations for methods that need to operate on the files.

One obvious optimization is the specialization of saveObject on ReloadedArray instances. Instead of loading the array data back into the R session and saving it again, the saveObject method can just

link or copy the existing files. This behavior is controlled by the optional `ReloadedArray.reuse.files` option in the `saveObject` method, which can be one of:

- `"copy"`: copy the files from the original directory (as stored in the ReloadedArray object) to the new `path` specified in `saveObject`.
- `"link"`: create a hard link from the files in the original directory to the new `path`. If this fails, we silently fall back to a copy. This mode is the default approach.
- `"symlink"`: create a symbolic link from the files in the original directory to the new `path`.
- `"none"`: ignore existing files and just save the contents by calling "`saveObject,DelayedArray-method`".

Value

For the constructors, an instance of the [ReloadedArraySeed](#) or [ReloadedArray](#).

Examples

```
arr <- array(rpois(10000, 10), c(50, 20, 10))
dir <- tempfile()
saveObject(arr, dir)
obj <- readArray(dir)
obj
DelayedArray::showtree(obj)
```

---

saveArray   *Save a multi-dimensional array to disk*

Description

Save a high-dimensional array to its on-disk representations.

Usage

```
## S4 method for signature 'array'
saveObject(x, path, ...)

## S4 method for signature 'denseMatrix'
saveObject(x, path, ...)
```

Arguments

x	An integer, numeric, logical or character array. Alternatively, any of the [dense-Matrix](#) subclasses from the **Matrix** package.
path	String containing the path to a directory in which to save x.
...	Further arguments, currently ignored.

Value

x is saved to path and NULL is invisibly returned.

Author(s)

Aaron Lun

See Also

[readArray](), to read the directory contents back into the R session.

Examples

```
mat <- array(rpois(10000, 10), c(50, 20, 10))
dimnames(mat) <- list(
   paste0("GENE_", seq_len(nrow(mat))),
   letters[1:20],
   NULL
)

dir <- tempfile()
saveObject(mat, dir)
list.files(dir)
```

---

saveDelayedArray   *Save DelayedArrays to disk*

Description

Save [DelayedArray]() objects to their on-disk representation.

Usage

```
## S4 method for signature 'DelayedArray'
saveObject(
  x,
  path,
  DelayedArray.dispatch.pristine = TRUE,
  DelayedArray.preserve.ops = FALSE,
  DelayedArray.store.args = list(),
  ...
)
```

Arguments

x	A [DelayedArray](#) object.
path	String containing a path to a directory in which to save x.
DelayedArray.dispatch.pristine	
	Logical scalar indicating whether to call the [saveObject](#) methods of seeds of pristine arrays.
DelayedArray.preserve.ops	
	Logical scalar indicating whether delayed operations should be preserved on-disk.
DelayedArray.store.args	
	Named arguments to pass to [storeDelayedObject](#).
...	Further arguments, ignored.

Value

x is saved to path and NULL is invisibly returned.

Author(s)

Aaron Lun

Examples

```
mat <- Matrix::rsparsematrix(100, 200, density=0.2)
rownames(mat) <- paste0("GENE_", seq_len(nrow(mat)))
dmat <- DelayedArray::DelayedArray(mat) * 1

dir <- tempfile()
saveObject(dmat, dir, delayed.preserve.ops=TRUE)
list.files(dir)
```

---

saveSparseMatrix   *Save a sparse matrix to disk*

Description

Save a sparse matrix to its on-disk representations.

Usage

```
## S4 method for signature 'sparseMatrix'
saveObject(x, path, ...)

## S4 method for signature 'SVT_SparseMatrix'
saveObject(x, path, ...)
```

Arguments

x	A sparse matrix of some kind, typically from either the **Matrix** or **SparseArray** packages.
path	String containing the path to a directory in which to save x.
...	Further arguments, currently ignored.

Value

x is saved to path and NULL is invisibly returned.

Author(s)

Aaron Lun

See Also

[readSparseMatrix](), to read the directory contents back into the R session.

Examples

```
mat <- Matrix::rsparsematrix(100, 200, density=0.2)
rownames(mat) <- paste0("GENE_", seq_len(nrow(mat)))

dir <- tempfile()
saveObject(mat, dir)
list.files(dir)
```

---

storeDelayedObject   *Store operations in a DelayedArray*

Description

Store the delayed operations of a [DelayedArray]() in a HDF5 file.

Arguments

x	Any of the delayed operation classes from **DelayedArray**.
file	String containing the path to a HDF5 file.
name	String containing the name of the group to save into.
...	Arguments to be passed to specific methods.

Value

The contents of x are saved to file, and NULL is invisibly returned.

Author(s)

Aaron Lun

Examples

```
library(DelayedArray)
X <- DelayedArray(matrix(runif(100), ncol=20))
Y <- cbind(X, DelayedArray::ConstantArray(value=50, c(5, 10)))

library(rhdf5)
temp <- tempfile()
dir.create(temp)

fpath <- file.path(temp, "foo.h5")
fhandle <- H5Fcreate(fpath)
storeDelayedObject(Y@seed, fhandle, "YAY")
rhdf5::h5ls(fhandle)
H5Fclose(fhandle)

fhandle <- H5Fopen(fpath, "H5F_ACC_RDONLY")
reloadDelayedObject(fhandle, "YAY")
H5Fclose(fhandle)
```

WrapperArraySeed   *DelayedArray wrapper seed*

Description

The WrapperArraySeed is, as the name suggests, a virtual class for a DelayedArray wrapper seed. This forwards most of the DelayedArray generic operations onto an internal seed class, typically a H5SparseMatrixSeed or HDF5ArraySeed objects from readSparseMatrix or readArray. Similarly, the WrapperArray is a virtual DelayedArray class that contains a WrapperArraySeed.

If an **alabaster** application operates on large arrays, developers may can consider defining concrete subclasses of the WrapperArraySeed (and WrapperArray). These subclasses can store application-specific provenance-tracking information that persist throughout the lifetime of the array. Such information is most useful for optimizing saveObject calls, which can instruct the application to link to the existing array rather than creating a new file. Check out the ReloadedArraySeed class for an example of this approach.

Examples

```
# Mocking up a concrete wrapper array class, which contains an
# extra 'foo_id' slot to track the provenance of the data.
setClass("FooArraySeed", contains="WrapperArraySeed",
    slots=c(seed="ANY", foo_id="character"))

y <- Matrix::rsparsematrix(1000, 100, 0.01)
```

```
foo <- new("FooArraySeed", seed=y, foo_id="FOO.0001")

dim(foo)
is_sparse(foo)
extract_array(foo, list(1:10, 1:10))
OLD_extract_sparse_array(foo, list(1:10, 1:10))
```

---

writeSparseMatrix   *Write a sparse matrix*

Description

Writes a sparse matrix to file in a compressed sparse format.

Usage

```
writeSparseMatrix(
  x,
  file,
  name,
  chunk = 10000,
  column = TRUE,
  tenx = FALSE,
  guess.integer = TRUE
)
```

Arguments

x	A sparse matrix of some sort. This includes sparse [DelayedMatrix](#) objects.
file	String containing a path to the HDF5 file. The file is created if it is not already present.
name	String containing the name of the group to store x.
chunk	Integer scalar specifying the chunk size for the indices and values.
column	Logical scalar indicating whether to store as compressed sparse column format.
tenx	Logical scalar indicating whether to use the 10X compressed sparse column format.
guess.integer	Logical scalar specifying whether to guess an appropriate integer type from x.

Details

This writes a sparse matrix to file in various formats:

- column=TRUE and tenx=FALSE uses H5AD's csr_matrix format.
- column=FALSE and tenx=FALSE uses H5AD's csc_matrix format.
- tenx=TRUE uses 10X Genomics' HDF5 matrix format.

For the first two formats, the apparent transposition is deliberate, because columns in R are interpreted as rows in H5AD. This allows us to retain consistency the interpretation of samples (columns in R, rows in H5AD) and features (vice versa). Constructors for classes like [H5SparseMatrix](#) will automatically transpose so no extra work is required.

If guess.integer=TRUE, we attempt to save x's values into the smallest type that will accommodate all of its values. If x only contains unsigned integers, we will attempt to save either 8-, 16- or 32-bit unsigned integers. If x contains signed integers, we will fall back to 32-bit signed integers. For all other values, we will fall back to double-precision floating point values.

We attempt to save x's indices to unsigned 16-bit integers if the relevant dimension of x is small enough. Otherwise we will save it as an unsigned 32-bit integer.

Value

A NULL invisibly. The contents of x are written to name in file.

Author(s)

Aaron Lun

Examples

```
library(Matrix)
x <- rsparsematrix(100, 20, 0.5)
tmp <- tempfile(fileext=".h5")
writeSparseMatrix(x, tmp, "csc_matrix")
writeSparseMatrix(x, tmp, "csr_matrix", column=FALSE)
writeSparseMatrix(x, tmp, "tenx_matrix", tenx = TRUE)

rhdf5::h5ls(tmp)
library(HDF5Array)
H5SparseMatrix(tmp, "csc_matrix")
H5SparseMatrix(tmp, "csr_matrix")
H5SparseMatrix(tmp, "tenx_matrix")
```

Index