Package ‘DelayedArray’

April 25, 2017

Title Delayed operations on array-like objects

Description Wrapping an array-like object (typically an on-disk object) in a DelayedArray object allows one to perform common array operations on it without loading the object in memory. In order to reduce memory usage and optimize performance, operations on the object are either delayed or executed using a block processing mechanism. Note that this also works on in-memory array-like objects like DataFrame objects (typically with Rle columns), Matrix objects, and ordinary arrays and data frames.

Version 0.2.0

Encoding UTF-8

Author Hervé Pagès

Maintainer Hervé Pagès <hpages@fredhutch.org>

biocViews Infrastructure, DataRepresentation, Annotation, GenomeAnnotation

Depends R (>= 3.4), methods, BiocGenerics, S4Vectors, IRanges, matrixStats

Imports stats

Suggests Matrix, HDF5Array, genefilter, BiocStyle

License Artistic-2.0


NeedsCompilation no

R topics documented:

cbind-methods .................................................. 2
DelayedArray-class .......................................... 3
DelayedArray-utils ......................................... 5
realize .......................................................... 7
RleArray-class ................................................ 9

Index 11
cbind-methods

Bind DelayedArray objects along their rows or columns

Description

Methods for binding DelayedArray objects along their rows or columns.

Details

rbind, cbind, arbind, acbind methods are defined for DelayedArray objects. They perform delayed binding along the rows (rbind and arbind) or columns (cbind and acbind) of the objects passed to them.

See Also

• cbind in the base package for rbind/cbind’ing ordinary arrays.
• acbind in the IRanges package for arbind/acbind’ing ordinary arrays.
• DelayedArray-utils for common operations on DelayedArray objects.
• DelayedArray objects.
• HDF5Array objects in the HDF5Array package.
• array objects in base R.

Examples

## ---------------------------------------------------------------------
## rbind/cbind
## ---------------------------------------------------------------------
library(HDF5Array)
toy_h5 <- system.file("extdata", "toy.h5", package="HDF5Array")
h5ls(toy_h5)
M1 <- HDF5Array(toy_h5, "M1")
M2 <- HDF5Array(toy_h5, "M2")
M <- rbind(M1, t(M2))
M
colMeans(M)

## ---------------------------------------------------------------------
## arbind/acbind
## ---------------------------------------------------------------------
a1 <- array(1:60, c(3, 5, 4),
  dimnames=list(NULL, paste0("M1y", 1:5), NULL))
a2 <- array(101:240, c(7, 5, 4),
  dimnames=list(paste0("M2x", 1:7), paste0("M2y", 1:5), NULL))
a3 <- array(10001:10100, c(5, 5, 4),
  dimnames=list(paste0("M3x", 1:5), NULL, paste0("M3z", 1:4)))

A1 <- DelayedArray(a1)
A2 <- DelayedArray(a2)
A3 <- DelayedArray(a3)
A <- arbind(A1, A2, A3)
A

## Sanity check:
stopifnot(identical(arbind(a1, a2, a3), as.array(A)))

---

### Description

Wrapping an array-like object (typically an on-disk object) in a DelayedArray object allows one to perform common array operations on it without loading the object in memory. In order to reduce memory usage and optimize performance, operations on the object are either delayed or executed using a block processing mechanism.

### Usage

DelayedArray(seed)  # constructor function

### Arguments

- **seed**: An array-like object.

### In-memory versus on-disk realization

To realize a DelayedArray object (i.e. to trigger execution of the delayed operations carried by the object and return the result as an ordinary array), call `as.array` on it. However this realizes the full object at once in memory which could require too much memory if the object is big. A big DelayedArray object is preferably realized on disk e.g. by calling `writeHDF5Array` on it (this function is defined in the [HDF5Array](#) package) or coercing it to an HDF5Array object with `as(x, "HDF5Array")`. Other on-disk backends can be supported. This uses a block-processing strategy so that the full object is not realized at once in memory. Instead the object is processed block by block i.e. the blocks are realized in memory and written to disk one at a time. See `?writeHDF5Array` in the [HDF5Array](#) package for more information about this.

### Accessors

DelayedArray objects support the same set of getters as ordinary arrays i.e. `dim()`, `length()`, and `dimnames()`.

Only `dimnames()` is supported as a setter.

### Subsetting

A DelayedArray object can be subsetted like an ordinary object but with the following differences:

- The `drop` argument of the `[` operator is ignored i.e. subsetting a DelayedArray object always returns a DelayedArray object with the same number of dimensions. You need to call `drop()` on the subsetted object to actually drop its ineffective dimensions (i.e. the dimensions equal to 1).
- Linear subsetting (a.k.a. 1D-style subsetting, that is, subsetting with a single subscript `i`) is not supported.

Subsetting with `[[` is supported but only the linear form of it.

DelayedArray objects don’t support subassignment (`[<-` or `[[<-`).
See Also

- `realize` for realizing a `DelayedArray` object in memory or on disk.
- `DelayedArray-utils` for common operations on `DelayedArray` objects.
- `cbind` in this package (`DelayedArray`) for binding `DelayedArray` objects along their rows or columns.
- `RleArray` objects.
- `HDF5Array` objects in the `HDF5Array` package.
- `array` objects in base R.

Examples

```r
## WRAP AN ORDINARY ARRAY IN A DelayedArray OBJECT
a <- array(runif(1500000), dim=c(10000, 30, 5))
A <- DelayedArray(a)
A

toto <- function(x) (5 * x[, , 1] ^ 3 + 1L) * log(x[, , 2])
b <- toto(a)
head(b)
B <- toto(A) # very fast! (operations are delayed)
B # still 3 dimensions (subsetting a DelayedArray object never drops # dimensions)
B <- drop(B)
B

cs <- colSums(b)
CS <- colSums(B)
stopifnot(identical(cs, CS))

## WRAP A DataFrame OBJECT IN A DelayedArray OBJECT
## Generate random coverage and score along an imaginary chromosome:
cov <- Rle(sample(20, 5000, replace=TRUE), sample(6, 5000, replace=TRUE))
score <- Rle(sample(100, nrun(cov), replace=TRUE), runLength(cov))

DF <- DataFrame(cov, score)
A2 <- DelayedArray(DF)
A2
t(A2) # delayed transpose is very fast and very memory efficient because # the matrix data is not copied
colSums(A2)

## A HDF5Array OBJECT IS A (PARTICULAR KIND OF) DelayedArray OBJECT
library(HDF5Array)
A3 <- as(a, "HDF5Array") # write 'a' to an HDF5 file
A3
```
is(A3, "DelayedArray")  # TRUE
B3 <- toto(A3)          # very fast! (operations are delayed)
B3                     # not a HDF5Array object because now it
# carries delayed operations
B3 <- drop(B3)
CS3 <- colSums(B3)
stopifnot(identical(cs, CS3))

## PERFORM THE DELAYED OPERATIONS
as(B3, "HDF5Array")    # "realize" 'B3' on disk

## If this is just an intermediate result, you can either keep going
## with B3 or replace it with its "realized" version:
B3 <- as(B3, "HDF5Array")  # no more delayed operations on new 'B3'

## For convenience, realize() can be used instead of explicit coercion.
## The current "realization backend" controls where realization
## happens e.g. in memory if set to NULL or in an HDF5 file if set
## to "HDF5Array":
D <- cbind(B3, exp(B3))
D
setRealizationBackend("HDF5Array")
D <- realize(D)
D
## See '?realize' for more information about "realization backends".

Description

Common operations on DelayedArray objects.

Details

The operations currently supported on DelayedArray objects are:

Delayed operations:

• all the members of the Ops, Math, and Math2 groups
• ! is.na, is.finite, is.infinite, is.nan
• nchar, tolower, toupper
• pmax2 and pmin2
• rbind and cbind (documented in chbind)

Block-processed operations:

• anyNA, which
• all the members of the **Summary** group
• **mean**
• **apply**
• matrix multiplication (%*%) of an ordinary matrix by a **DelayedMatrix** object

**See Also**

• **is.na**, !, **mean**, **apply**, and %*% in the **base** package for the corresponding operations on ordinary arrays or matrices.
• **rowSums** in the **base** package and **rowMaxs** in the **matrixStats** package for row/col summarization of an ordinary matrix.
• **setRealizationBackend** for how to set a realization backend.
• **writeHDF5Array** in the **HDF5Array** package for writing an array-like object to an HDF5 file and other low-level utilities to control the location of automatically created HDF5 datasets.
• **DelayedArray** objects.
• **HDF5Array** objects in the **HDF5Array** package.
• **S4groupGeneric** in the **methods** package for the members of the **Ops**, **Math**, and **Math2** groups.
• **array** objects in base R.

**Examples**

```r
library(HDF5Array)
toy_h5 <- system.file("extdata", "toy.h5", package="HDF5Array")
h5ls(toy_h5)

M1 <- HDF5Array(toy_h5, "M1")
range(M1)
M1 >= 0.5 & M1 < 0.75
log(M1)

M2 <- HDF5Array(toy_h5, "M2")
pmax2(M2, 0)

M3 <- rbind(M1, t(M2))
M3
```

```r
## MATRIX MULTIPLICATION
## ---------------------------------------------------------------------
## The output matrix is returned as a DelayedMatrix object with no delayed
## operations on it. The exact class of the object depends on the backend
## e.g. it will be HDF5Matrix with "HDF5Array" backend.
```
### Description

Realize a DelayedArray object in memory or on disk. Get or set the realization backend for the current session with getRealizationBackend or setRealizationBackend.
Usage

supportedRealizationBackends()
getRealizationBackend()
setRealizationBackend(BACKEND=NULL)

realize(x, ...)

### S4 method for signature 'ANY'
realize(x, BACKEND=getRealizationBackend())

Arguments

x
The array-like object to realize.

... Additional arguments passed to methods.

BACKEND NULL (the default), or a single string specifying the name of the backend. When the backend is set to NULL, x is realized in memory as an ordinary array by just calling as.array on it.

Details

The realization backend controls where/how realization happens e.g. as an ordinary array if set to NULL, as an RleArray object if set to "RleArray", or in an HDF5 file if set to "HDF5Array".

Value

realize(x) returns a DelayedArray object. More precisely, it returns DelayedArray(as.array(x)) when the backend is set to NULL (the default). Otherwise it returns an instance of the class associated with the specified backend (which should extend DelayedArray).

See Also

- DelayedArray objects.
- RleArray objects.
- HDF5Array objects in the HDF5Array package.
- array objects in base R.

Examples

library(HDF5Array)
toy_h5 <- system.file("extdata", "toy.h5", package="HDF5Array")
h5ls(toy_h5)
M1 <- HDF5Array(toy_h5, "M1")
M2 <- HDF5Array(toy_h5, "M2")
M3 <- rbind(log(M1), t(M2))
supportedRealizationBackends()
getRealizationBackend() # backend is set to NULL
realize(M3) # realization as ordinary array

setRealizationBackend("RleArray")
getRealizationBackend() # backend is set to "RleArray"
realize(M3) # realization as RleArray object
RleArray-class

setRealizationBackend("HDF5Array")
getRealizationBackend() # backend is set to "HDF5Array"
realize(M3) # realization in HDF5 file

RleArray-class  RleArray objects

Description

The RleArray class is an array-like container where the values are stored in a run-length encoding format. RleArray objects support delayed operations and block processing.

Usage

RleArray(rle, dim, dimnames=NULL) # constructor function

Arguments

  rle       An Rle object.
  dim       The dimensions of the object to be created, that is, an integer vector of length
            one or more giving the maximal indices in each dimension.
  dimnames  Either NULL or the names for the dimensions. This must a list of length the
            number of dimensions. Each list element must be either NULL or a character
            vector along the corresponding dimension.

Details

RleArray extends DelayedArray. All the operations available on DelayedArray objects work on RleArray objects.

See Also

  • Rle objects in the S4Vectors package.
  • DelayedArray objects.
  • DelayedArray-utils for common operations on DelayedArray objects.
  • realize for realizing a DelayedArray object in memory or on disk.
  • HDF5Array objects in the HDF5Array package.
  • array objects in base R.

Examples

rle <- Rle(sample(6L, 500000, replace=TRUE), 8)
a <- array(rle, dim=c(50, 20, 4000)) # array() expands the Rle object
    # internally with as.vector()

A <- RleArray(rle, dim=c(50, 20, 4000)) # Rle object is NOT expanded
A

object.size(a)
object.size(A)
stopifnot(identical(a, as.array(A)))

toto <- function(x) (5 * x[ , , 1]^3 + 1L) * log(x[ , 2])
b <- toto(a)
head(b)

B <- toto(A) # very fast! (operations are delayed)
B # still 3 dimensions (subsetting a DelayedArray object never drops # dimensions)
B <- drop(B)
B

stopifnot(identical(b, as.array(B)))

cs <- colSums(b)
CS <- colSums(B)
stopifnot(identical(cs, CS))
Index

!,DelayedArray-method (DelayedArray-utils), 5

*Topic classes
DelayedArray-class, 3
RleArray-class, 9

*Topic methods
  cbind-methods, 2
  DelayedArray-class, 3
  DelayedArray-utils, 5
  realize, 7
  RleArray-class, 9
  +,DelayedArray,missing-method (DelayedArray-utils), 5
  -,DelayedArray,missing-method (DelayedArray-utils), 5
  [,DelayedArray-method (DelayedArray-class), 3
  [[,DelayedArray-method (DelayedArray-class), 3
  **%,DelayedMatrix,DelayedMatrix-method (DelayedArray-utils), 5
  **%,DelayedMatrix,matrix-method (DelayedArray-utils), 5
  **%,matrix,DelayedMatrix-method (DelayedArray-utils), 5
  **%, 6
  acbind, 2
  acbind(cbind-methods), 2
  acbind,DelayedArray-method (cbind-methods), 2
  anyNA,DelayedArray-method (DelayedArray-utils), 5
  apply, 6
  apply(DelayedArray-utils), 5
  apply,DelayedArray-method (DelayedArray-utils), 5
  arbind(cbind-methods), 2
  arbind,DelayedArray-method (cbind-methods), 2
  array, 2, 4, 6, 8, 9
  arrayRealizationSink (realize), 7
  arrayRealizationSink-class (realize), 7
  as.array,DelayedArray-method (DelayedArray-class), 3
  as.array.DelayedArray (DelayedArray-class), 3
  as.character,DelayedArray-method (DelayedArray-class), 3
  as.character.DelayedArray (DelayedArray-class), 3
  as.complex,DelayedArray-method (DelayedArray-class), 3
  as.complex.DelayedArray (DelayedArray-class), 3
  as.data.frame,DelayedArray-method (DelayedArray-class), 3
  as.data.frame.DelayedArray (DelayedArray-class), 3
  as.integer,DelayedArray-method (DelayedArray-class), 3
  as.integer.DelayedArray (DelayedArray-class), 3
  as.logical,DelayedArray-method (DelayedArray-class), 3
  as.logical.DelayedArray (DelayedArray-class), 3
  as.numeric,DelayedArray-method (DelayedArray-class), 3
  as.numeric.DelayedArray (DelayedArray-class), 3
  as.raw,DelayedArray-method (DelayedArray-class), 3
  as.raw.DelayedArray (DelayedArray-class), 3
  as.vector,DelayedArray-method (DelayedArray-class), 3
  as.vector.DelayedArray (DelayedArray-class), 3
  c,DelayedArray-method (DelayedArray-class), 3
  cbind, 2, 4, 5
cbind (cbind-methods), 2

cbind, DelayedArray-method
  (cbind-methods), 2

cbind, DelayedMatrix-method
  (cbind-methods), 2

cbind-methods, 2

class: arrayRealizationSink (realize), 7

class: DelayedArray
  (DelayedArray-class), 3

class: DelayedMatrix
  (DelayedArray-class), 3
class: RleArray (RleArray-class), 9
class: RleMatrix (RleArray-class), 9
class: RleRealizationSink
  (RleArray-class), 9
close, RealizationSink-method (realize), 7
coerce, ANY, RleArray-method
  (RleArray-class), 9
coerce, ANY, RleMatrix-method
  (RleArray-class), 9
coerce, arrayRealizationSink, DelayedArray-method
  (realize), 7
coerce, DelayedArray, DelayedMatrix-method
  (DelayedArray-class), 3
coerce, DelayedArray, RleArray-method
  (RleArray-class), 9
coerce, RleArray, RleMatrix-method
  (RleArray-class), 9
class: RleRealizationSink, DelayedArray-method
  (RleArray-class), 9
class: RleRealizationSink, RleArray-method
  (RleArray-class), 9
class: RleRealizationSink, RleArraySeed-method
  (RleArray-class), 9
colMaxs (DelayedArray-utils), 5
colMaxs, DelayedMatrix-method
  (DelayedArray-utils), 5
colMeans (DelayedArray-utils), 5
colMeans, DelayedMatrix-method
  (DelayedArray-utils), 5
colMins (DelayedArray-utils), 5
colMins, DelayedMatrix-method
  (DelayedArray-utils), 5
colRanges (DelayedArray-utils), 5
colRanges, DelayedMatrix-method
  (DelayedArray-utils), 5
colSums (DelayedArray-utils), 5
colSums, DelayedMatrix-method
  (DelayedArray-utils), 5

drop, DelayedArray-method
  (DelayedArray-class), 3

dim, ConformableSeedCombiner-method
  (DelayedArray-utils), 5
dim, DelayedArray-method
  (DelayedArray-class), 3
dim, RleArraySeed-method
  (RleArray-class), 9

dim<-, DelayedArray-method
  (DelayedArray-class), 3
dimnames, ConformableSeedCombiner-method
  (DelayedArray-utils), 5
dimnames, DelayedArray-method
  (DelayedArray-class), 3
dimnames, RleArraySeed-method
  (RleArray-class), 9
dimnames, SeedBinder-method
  (cbind-methods), 2
dimnames<-, DelayedArray-method
  (DelayedArray-class), 3

drop, DelayedArray-method
  (DelayedArray-class), 3

getRealizationBackend (realize), 7

HDF5Array, 2–4, 6, 8, 9

is.finite, DelayedArray-method
  (DelayedArray-utils), 5
is.infinite, DelayedArray-method
  (DelayedArray-utils), 5
is.na, 6
is.na, DelayedArray-method
  (DelayedArray-utils), 5
is.nan, DelayedArray-method
  (DelayedArray-utils), 5
isEmpty, DelayedArray-method
  (DelayedArray-class), 3

length, DelayedArray-method
  (DelayedArray-class), 3
INDEX

length,RleArraySeed-method
(RleArray-class), 9

Math, 5, 6
Math2, 5, 6
matrixClass (DelayedArray-class), 3
matrixClass,DelayedArray-method (DelayedArray-class), 3
matrixClass,RleArray-method (RleArray-class), 9
mean, 6
mean,DelayedArray-method (DelayedArray-class), 3
mean,DelayedArray (DelayedArray-utils), 5
mean,DelayedArray (DelayedArray-utils), 5

Names,DelayedArray-method (DelayedArray-class), 3
names<-,DelayedArray-method (DelayedArray-class), 3
nchar,DelayedArray-method (DelayedArray-utils), 5

Ops, 5, 6

pmax2 (DelayedArray-utils), 5
pmax2, ANY, ANY-method (DelayedArray-utils), 5
pmax2,DelayedArray,DelayedArray-method (DelayedArray-utils), 5
pmax2,DelayedArray,vector-method (DelayedArray-utils), 5
pmax2,vector,DelayedArray-method (DelayedArray-utils), 5
pmin2 (DelayedArray-utils), 5
pmin2, ANY, ANY-method (DelayedArray-utils), 5
pmin2,DelayedArray,DelayedArray-method (DelayedArray-utils), 5
pmin2,DelayedArray,vector-method (DelayedArray-utils), 5
pmin2,vector,DelayedArray-method (DelayedArray-utils), 5

rbind (cbind-methods), 2
rbind,DelayedArray-method (cbind-methods), 2
rbind,DelayedMatrix-method (cbind-methods), 2
RealizationSink (realize), 7
RealizationSink-class (realize), 7
realize, 4, 7, 9
realize, ANY-method (realize), 7

Rle, 9
RleArray, 4, 8
RleArray (RleArray-class), 9
RleArray-class, 9
RleMatrix (RleArray-class), 9
RleMatrix-class (RleArray-class), 9
RleRealizationSink (RleArray-class), 9
RleRealizationSink-class (RleArray-class), 9
round,DelayedArray-method (DelayedArray-utils), 5
rowMaxs, 6
rowMaxs (DelayedArray-utils), 5
rowMaxs,DelayedMatrix-method (DelayedArray-utils), 5
rowMeans (DelayedArray-utils), 5
rowMeans,DelayedMatrix-method (DelayedArray-utils), 5
rowMeans,DelayedMatrix (DelayedArray-utils), 5
rowMins (DelayedArray-utils), 5
rowMins,DelayedMatrix-method (DelayedArray-utils), 5
rowRanges (DelayedArray-utils), 5
rowRanges,DelayedMatrix-method (DelayedArray-utils), 5
rowSums, 6
rowSums (DelayedArray-utils), 5
rowSums,DelayedMatrix-method (DelayedArray-utils), 5

S4groupGeneric, 6
setRealizationBackend, 6
setRealizationBackend (realize), 7
show,DelayedArray-method (DelayedArray-utils), 5
signif,DelayedArray-method (DelayedArray-utils), 5
split,DelayedArray, ANY-method (DelayedArray-utils), 5
split,DelayedArray (DelayedArray-utils), 3
splitAsList,DelayedArray-method (DelayedArray-utils), 3
subset_seed_as_array (DelayedArray-utils), 3
subset_seed_as_array, ANY-method (DelayedArray-utils), 3
subset_seed_as_array, array-method (DelayedArray-utils), 3
subset_seed_as_array, ConformableSeedCombiner-method (DelayedArray-utils), 5
subset_seed_as_array, data.frame-method (DelayedArray-utils), 3
subset_seed_as_array, DataFrame-method
   (DelayedArray-class), 3
subset_seed_as_array, RleArraySeed-method
   (RleArray-class), 9
subset_seed_as_array, SeedBinder-method
   (cbind-methods), 2
Summary, 6
supportedRealizationBackends (realize), 7
t, DelayedArray-method
   (DelayedArray-class), 3
tolower, DelayedArray-method
   (DelayedArray-utils), 5
toupper, DelayedArray-method
   (DelayedArray-utils), 5
type (DelayedArray-class), 3
type, array-method (DelayedArray-class), 3
type, DelayedArray-method
   (DelayedArray-class), 3
which, DelayedArray-method
   (DelayedArray-utils), 5
write_to_sink (realize), 7
write_to_sink, ANY, RealizationSink-method
   (realize), 7
write_to_sink, array, arrayRealizationSink-method
   (realize), 7
write_to_sink, array, RleRealizationSink-method
   (RleArray-class), 9
write_to_sink, DelayedArray, RealizationSink-method
   (realize), 7
writeHDF5Array, 3, 6