

Package ‘DelayedDataFrame’

July 1, 2025

Title Delayed operation on DataFrame using standard DataFrame metaphor

Version 1.25.0

Description Based on the standard DataFrame metaphor, we are trying to implement the feature of delayed operation on the DelayedDataFrame, with a slot of lazyIndex, which saves the mapping indexes for each column of DelayedDataFrame. Methods like show, validity check, [/[[subsetting, rbind/cbind are implemented for DelayedDataFrame to be operated around lazyIndex. The listData slot stays untouched until a realization call e.g., DataFrame constructor OR as.list() is invoked.

biocViews Infrastructure, DataRepresentation

Depends R (>= 3.6), S4Vectors (>= 0.23.19), DelayedArray (>= 0.7.5)

License GPL-3

Encoding UTF-8

URL <https://github.com/Bioconductor/DelayedDataFrame>

BugReports <https://github.com/Bioconductor/DelayedDataFrame/issues>

Imports methods, stats, BiocGenerics

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown, BiocStyle, SeqArray, GDSArray

Collate LazyIndex-class.R DelayedDataFrame-class.R
DelayedDataFrame-method.R

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/DelayedDataFrame>

git_branch devel

git_last_commit 38233f8

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-01

Author Qian Liu [aut, cre],
Hervé Pagès [aut],
Martin Morgan [aut]

Maintainer Qian Liu <Qian.Liu@roswellpark.org>

Contents

as.list,DelayedDataFrame-method	2
DelayedDataFrame	3
LazyIndex-class	4

Index	6
--------------	----------

as.list,DelayedDataFrame-method

DelayedDataFrame related methods.

Description

as.list, rbind would incur realization of the lazyIndex slot in DelayedDataFrame object.

cbind for DelayedDataFrame inherits the lazyIndex's if inputs have any DelayedDataFrame objects. Otherwise, return a new DelayedDataFrame with NULL lazyIndexes.

Usage

```
## S4 method for signature 'DelayedDataFrame'
as.list(x, use.names = TRUE)

## S4 method for signature 'DelayedDataFrame'
names(x)

## S4 method for signature 'DelayedDataFrame'
cbind(..., deparse.level = 1)

## S4 method for signature 'DelayedDataFrame'
bindROWS(
  x,
  objects = list(),
  use.names = TRUE,
  ignore.mcols = FALSE,
  check = TRUE
)

## S4 method for signature 'DelayedDataFrame,ANY'
extractROWS(x, i)

## S4 method for signature 'DelayedDataFrame'
extractCOLS(x, i)

## S4 method for signature 'DelayedDataFrame'
replaceCOLS(x, i, value)

## S4 method for signature 'DelayedDataFrame'
mergeROWS(x, i, value)

## S4 method for signature 'DelayedDataFrame,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

<code>x</code>	<code>as.list,DelayedDataFrame</code> : a <code>DelayedDataFrame</code> object. OR, <code>[,DelayedDataFrame</code> : <code>DelayedDataFrame</code> object to be subsetted.
<code>use.names</code>	<code>as.list,DelayedDataFrame</code> : whether to use the colnames of <code>DelayedDataFrame</code> as the names for the returned list. OR, <code>bindROWS,DelayedDataFrame</code> : whether to use rownames of the input arguments. Default is <code>TRUE</code> .
<code>...</code>	<code>cbind,DelayedDataFrame</code> : One or more vector-like or matrix-like objects. These can be given as named arguments. OR, <code>[,DelayedDataFrame</code> : other arguments to pass.
<code>deparse.level</code>	See ‘ <code>?base::cbind</code> ’ for a description of this argument.
<code>objects</code>	the <code>DelayedDataFrame</code> objects to be passed into <code>bindROWS</code> .
<code>ignore.mcols</code>	Logical. This argument is ignored for <code>bindROWS,DelayedDataFrame</code> .
<code>check</code>	Logical. This argument is ignored for <code>bindROWS,DelayedDataFrame</code> .
<code>i</code>	row subscript
<code>value</code>	the new values in the <code>i, j</code> subscripts of <code>DelayedDataFrame</code> object.
<code>j</code>	col subscript
<code>drop</code>	if drop with reduced dimension, default is <code>TRUE</code> .

Value

colnames of `DelayedDataFrame`

<code>DelayedDataFrame</code>	<i>DelayedDataFrame-class</i>
-------------------------------	-------------------------------

Description

The `DelayedDataFrame` class extends the `DataFrame` class and supports the storage of any type of object (with ‘`length`’ and ‘`[]`’ methods) as columns.

the `lazyIndex` slot getter and setter for `DelayedDataFrame` object.

the coercion method between `DataFrame` and `DelayedDataFrame` objects.

Usage

```
DelayedDataFrame(..., row.names = NULL, check.names = TRUE)

## S4 method for signature 'DelayedDataFrame'
lazyIndex(x)

.from_DataFrame_to_DelayedDataFrame(from)

.from_DelayedDataFrame_to_DFrame(from, to = "DFrame", strict = TRUE)

lazyIndex(x) <- value

## S4 replacement method for signature 'DelayedDataFrame'
lazyIndex(x) <- value
```

Arguments

<code>...</code>	the arguments to pass into construction of a new <code>DelayedDataFrame</code> .
<code>row.names</code>	the rownames for the newly constructed <code>DelayedDataFrame</code> object.
<code>check.names</code>	logical. If 'TRUE' then the names of the variables in the <code>DelayedDataFrame</code> are checked to ensure that they are syntactically valid variable names and are not duplicated. If necessary they are adjusted (by 'make.names') so that they are.
<code>x</code>	the <code>DelayedDataFrame</code> object.
<code>from</code>	the object to be converted.
<code>to</code>	the class of object to be returned by coercion.
<code>strict</code>	Logical. Whether to force return a <code>DataFrame</code> .
<code>value</code>	the new value of <code>lazyIndex</code> slot for <code>DelayedDataFrame</code> object.

Details

The `DelayedDataFrame` inherits from `DataFrame` and behaves very similarly in terms of construction, subsetting, splitting, combining, etc. The most notable exception is that The additional slot of `lazyIndex`, enables `DelayedArray` (with different back-ends) columns to share indexes when possible.

Please be very careful to use this replace method for `lazyIndex` slot. Because it only replace the `lazyIndex` slot, but not necessarily the `nrow` and `rownames` slots. If you want to have synchronized subsetting for all slots, the `[]` method should be used.

Value

`lazyIndex<-`: the `DelayedDataFrame` object with new value of `lazyIndex` slot.

Examples

```
DDF <- DelayedDataFrame(letters, LETTERS)
DDF1 <- DDF[1:10,]
DDF1
lazyIndex(DDF1)
as(DDF1, "DataFrame")
```

LazyIndex-class

The LazyIndex class and methods.

Description

The `LazyIndex` class is designed to carry mapping indexes for `DelayedDataFrame` columns. So that some operations (e.g., subsetting) on `DelayedDataFrame` are delayed until a realization call is incurred. (e.g., `as.list()`, `DataFrame()`, ...)

`LazyIndex` constructor.

the subsetting method for `LazyIndex` object.

Usage

```

LazyIndex(listData = list(), index = integer())

## S4 method for signature 'LazyIndex'
cbind(..., deparse.level = 1)

## S4 method for signature 'LazyIndex,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

```

Arguments

<code>listData</code>	the list data for all mapping indexes that are used in corresponding <code>DelayedDataFrame</code> object.
<code>index</code>	the position of mapping indexes in <code>listData</code> for each column of the corresponding <code>DelayedDataFrame</code> object.
<code>...</code>	<code>LazyIndex</code> objects.
<code>deparse.level</code>	See <code>?base::cbind</code> for a description of this argument.
<code>x</code>	<code>LazyIndex</code> object.
<code>i</code>	row subscript for <code>LazyIndex</code> , which will subset the <code>listData</code> slot.
<code>j</code>	column subscript for <code>LazyIndex</code> , which will subset the <code>index</code> slot.
<code>drop</code>	Logical. Whether to drop the dimension if any of the dimensions has length 1. Default is <code>TRUE</code> .

Details

the `cbind, LazyIndex` method is defined to bind the `LazyIndexes` column-wise when `cbind, DelayedDataFrame` function is called.

Value

a `LazyIndex` object.

Index

`.DelayedDataFrame (DelayedDataFrame)`, [3](#)
`.LazyIndex (LazyIndex-class)`, [4](#)
`.from_DataFrame_to_DelayedDataFrame`
 (`DelayedDataFrame`), [3](#)
`.from_DelayedDataFrame_to_DFrame`
 (`DelayedDataFrame`), [3](#)
`[, DelayedDataFrame, ANY, ANY, ANY-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)
`[, LazyIndex, ANY, ANY, ANY-method`
 (`LazyIndex-class`), [4](#)
`[, LazyIndex-method (LazyIndex-class)`, [4](#)

`as.list, DelayedDataFrame-method`, [2](#)

`bindROWS, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)

`cbind, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)
`cbind, LazyIndex-method`
 (`LazyIndex-class`), [4](#)
`coerce (DelayedDataFrame)`, [3](#)
`coerce, ANY, DelayedDataFrame-method`
 (`DelayedDataFrame`), [3](#)
`coerce, DataFrame, DelayedDataFrame-method`
 (`DelayedDataFrame`), [3](#)
`coerce, DelayedDataFrame, DataFrame-method`
 (`DelayedDataFrame`), [3](#)
`coerce, DelayedDataFrame, DFrame-method`
 (`DelayedDataFrame`), [3](#)
`coerce, DFrame, DelayedDataFrame-method`
 (`DelayedDataFrame`), [3](#)

`DelayedDataFrame`, [3](#)
`DelayedDataFrame-class`
 (`DelayedDataFrame`), [3](#)

`extractCOLS, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)

`extractROWS, DelayedDataFrame, ANY-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)
`extractROWS, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)

`LazyIndex (LazyIndex-class)`, [4](#)
`lazyIndex (DelayedDataFrame)`, [3](#)
`lazyIndex, DelayedDataFrame`
 (`DelayedDataFrame`), [3](#)
`lazyIndex, DelayedDataFrame-method`
 (`DelayedDataFrame`), [3](#)
`LazyIndex-class`, [4](#)
`lazyIndex<- (DelayedDataFrame)`, [3](#)
`lazyIndex<- , (DelayedDataFrame)`, [3](#)
`lazyIndex<- , DelayedDataFrame-method`
 (`DelayedDataFrame`), [3](#)

`mergeROWS, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)

`names, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)

`replaceCOLS, DelayedDataFrame-method`
 (`as.list`, `DelayedDataFrame-method`),
 [2](#)