Package: BPCellsArray (via r-universe)

March 21, 2025

Title Using BPCells as a DelayedArray Backend

Version 0.0.0.9000

Description Implements a DelayedArray backend for reading and writing arrays in the BPCells storage layout. The resulting BPCells*Arrays are compatible with all Bioconductor pipelines that can accept DelayedArray instances.

BugReports https://github.com/Yunuuuu/BPCellsArray

License MIT + file LICENSE

Imports methods, BPCells (>= 0.2.0), DelayedArray (>= 0.28.0), SparseArray, Matrix, MatrixGenerics, BiocGenerics, cli, rlang (>= 1.1.0), S4Vectors, BiocSingular

Remotes bnprks/BPCells/r

Suggests SingleCellExperiment, BiocNeighbors, bluster, knitr, rmarkdown, roxygen2, RSpectra, S4Arrays, scater, scran, scuttle, testthat (>= 3.0.0)

Encoding UTF-8

biocViews Software, DataImport, DataRepresentation, Infrastructure

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Collate 'BPCellsSeed.R' 'Class-Delayed.R' 'utils-BPCells.R' 'Class-BPCellsMatrix.R' 'Class-BPCellsSeed.R' 'Class-Transformed.R' 'IO-10xHDF5.R' 'IO-AnnHDF5.R' 'IO-Dir.R' 'IO-HDF5.R' 'IO-Mem.R' 'Method-Arith.R' 'Method-BindMatrix.R' 'Method-Compare.R' 'Method-Convert.R' 'Method-Mask.R' 'Method-Math.R' 'Method-Multiply.R' 'Method-RankTransform.R' 'Method-RenameDims.R' 'Method-Subassign.R' 'Method-Subset.R' 'Method-apply.R' 'Method-axis.R' 'Method-crossprod.R' 'Method-internal.R' 'Method-pmin.R' 'Method-summarization.R' 'Method-tcrossprod.R' 'import-standalone-assert.R' 'import-standalone-cli.R' 'import-standalone-obj-type.R' 'runSVD.R' 'showtree.R' 'utils-Rd.R' 'utils-check.R' 'utils.R' 'zzz.R'

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://yunuuuu.r-universe.dev

RemoteUrl https://github.com/Yunuuuu/BPCellsArray

RemoteRef HEAD

RemoteSha 914ae0ad8ca62459a8ce6cee0dad92ff63ca8aff

Contents

`apply,BPCellsMatrix-method` *Apply Functions Over matrix Margins*

Description

Apply Functions Over matrix Margins

Usage

```
## S4 method for signature 'BPCellsMatrix'
apply(X, MARGIN, FUN, ..., simplify = TRUE)
```

Arguments

`X`	A BPCellsMatrix object.
`MARGIN`	A single number giving the subscripts which the function will be applied over, `1` indicates rows, `2` indicates columns.
`FUN`	Function to be applied. `FUN` is found by a call to match.fun and typically is either a function or a symbol (e.g., a backquoted name) or a character string specifying a function to be searched for from the environment of the call to apply.
`...`	optional arguments to `FUN`.
`simplify`	a logical indicating whether results should be simplified if possible.

Value

If each call to `FUN` returns a vector of length `n`, and simplify is `TRUE`, then `apply` returns an array of dimension `c(n, dim(X)[MARGIN])` if `n > 1`. If `n` equals `1`, `apply` returns a vector if `MARGIN` has length 1 and an array of dimension `dim(X)[MARGIN]` otherwise. If `n` is `0`, the result has length 0 but not necessarily the 'correct' dimension.

If the calls to `FUN` return vectors of different lengths, or if `simplify` is `FALSE`, `apply` returns a list of length `dim(X)[MARGIN]`.

`BPCells-Arithmetic` *Arithmetic operators for BPCellsMatrix*

Description

Arithmetic operators for BPCellsMatrix

Usage

```
## S4 method for signature 'BPCellsMatrix,numeric'
Arith(e1, e2)

## S4 method for signature 'numeric,BPCellsMatrix'
Arith(e1, e2)
```

Arguments

e1, e2	One of e1 or e2 must be a BPCellsMatrix object, and the another can be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

Value

A BPCellsMatrix object.

Arithmetic operators

- BPCells: +, -, *, /, ^
- DelayedArray: %%, - %/%

See Also

BPCellsMatrix

BPCells-bind *Combine two Objects by Columns or Rows*

Description

Combine two Objects by Columns or Rows

Usage

```
## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
rbind2(x, y, mode = NULL, ..., threads = 0L)

## S4 method for signature 'BPCellsMatrix'
rbind(
  ...,
  mode = NULL,
  threads = 0L,
  use.first.dimnames = TRUE,
  deparse.level = 1L
)
```

```
## S4 method for signature 'BPCellsMatrix'
arbind(..., mode = NULL, threads = 0L)

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

## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
cbind2(x, y, mode = NULL, ..., threads = 0L)

## S4 method for signature 'BPCellsMatrix'
cbind(
  ...,
  mode = NULL,
  threads = 0L,
  use.first.dimnames = TRUE,
  deparse.level = 1L
)

## S4 method for signature 'BPCellsMatrix'
acbind(..., mode = NULL, threads = 0L, use.first.dimnames = TRUE)

## S4 method for signature 'BPCellsMatrix'
bindCOLS(
  x,
  objects = list(),
  use.names = TRUE,
  ignore.mcols = TRUE,
  check = TRUE
)
```

Arguments

x, y	A BPCellsMatrix object or BPCellsSeed object.
mode	Storage mode of BPCells matrix, one of `uint32_t` (unsigned 32-bit integer), `float` (32-bit real number), or `double` (64-bit real number). R cannot differentiate 32-bit and 64-bit real number, so type method always return "double" for both `float` and `double` mode.
...	• `rbind2` and `cbind2`: Not used currently. • `rbind`, `arbind`, `cbind`, and `acbind`: A list of BPCellsMatrix object.
threads	Set the number of threads to use for sparse-dense multiply and matrix_stats.
use.first.dimnames	Ignored, always be `TRUE` in BPCells.

deparse.level	Ignored, used by generic methods.
objects	A list of S4 objects to bind to x. They should typically (but not necessarily) have the same class as x.
use.names	Ignored, always be TRUE.
ignore.mcols	Ignored.
check	Ignored.

Value

If mode is specified, the mode of all specified object will be converted.

- cbind2, acbind, cbind, bindCOLS: A BPCellsMatrix object combined by columns.
- rbind2, arbind, rbind, bindROWS: A BPCellsMatrix object combined by rows.

See Also

convert_mode

BPCells-Compare *Convert matrix elements to zeros and ones*

Description

Binarize compares the matrix element values to the threshold value and sets the output elements to either zero or one. By default, element values greater than the threshold are set to one; otherwise, set to zero. When strict_inequality is set to FALSE, element values greater than or equal to the threshold are set to one. As an alternative, the <, <=, >, and >= operators are also supported.

Usage

```
binarize(object, ...)

## S4 method for signature 'BPCellsMatrix'
binarize(object, ...)

## S4 method for signature 'numeric,BPCellsMatrix'
e1 < e2

## S4 method for signature 'BPCellsMatrix,numeric'
e1 > e2

## S4 method for signature 'numeric,BPCellsMatrix'
e1 <= e2

## S4 method for signature 'BPCellsMatrix,numeric'
e1 >= e2
```

Arguments

`object`	A BPCellsMatrix object.
`...`	Arguments passed on to `BPCells::binarize` `threshold` A numeric value that determines whether the elements of x are set to zero or one. `strict_inequality` A logical value determining whether the comparison to the threshold is >= (strict_inequality=FALSE) or > (strict_inequality=TRUE).
`e1, e2`	One of `e1` or `e2` must be a BPCellsMatrix object, and the another can be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

Value

A BPCellsMatrix object.

Note

Methods listed here are supported by `BPCells`, other Compare operators will use the methods defined in DelayedArray.

`BPCells-crossprod` *Matrix Crossproduct*

Description

Given matrices x and y as arguments, return a matrix cross-product.

Usage

```
## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
crossprod(x, y)

## S4 method for signature 'BPCellsMatrix,ANY'
crossprod(x, y)

## S4 method for signature 'ANY,BPCellsMatrix'
crossprod(x, y)

## S4 method for signature 'BPCellsMatrix,matrix'
crossprod(x, y)

## S4 method for signature 'matrix,BPCellsMatrix'
crossprod(x, y)

## S4 method for signature 'BPCellsMatrix,numeric'
crossprod(x, y)
```

```
## S4 method for signature 'numeric,BPCellsMatrix'
crossprod(x, y)
```

Arguments

x, y	One of x or y must be a BPCellsMatrix object, and the another must be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

Value

A dense matrix if one of x or y is a regular matrix or atomic vector. Otherwise, a BPCellsMatrix object.

See Also

- %*%
- tcrossprod

BPCells-Math *Math operators for BPCellsMatrix*

Description

Math operators for BPCellsMatrix

Usage

```
expm1_slow(x)

## S4 method for signature 'BPCellsMatrix'
expm1_slow(x)

## S4 method for signature 'BPCellsMatrix'
expm1(x)

## S4 method for signature 'BPCellsMatrix'
log1p(x)

log1p_single(x)

## S4 method for signature 'BPCellsMatrix'
log1p_single(x)

## S4 method for signature 'BPCellsMatrix'
round(x, digits = 0)
```

Arguments

x	A BPCellsMatrix object.
digits	Integer indicating the number of decimal places

Value

- expm1 and expm1_slow: compute exp(x)-1 of matrix.
- log1p and log1p_single: compute log(1+x) of matrix. log1p_single use single-precision math in intermediate stages, which is 2x faster than log1p.
- round: Rounding of matrix Numbers.

Note

Methods listed here are supported by BPCells, other Math or Math2 operators will use the methods defined in DelayedArray.

BPCells-Multiplication

Matrix Multiplication

Description

Multiplies two matrices, if they are conformable. If one argument is a vector, it will be promoted to either a row or column matrix to make the two arguments conformable. If both are vectors of the same length, it will return the inner product (as a matrix).

Usage

```
## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
x %*% y

## S4 method for signature 'BPCellsMatrix,ANY'
x %*% y

## S4 method for signature 'ANY,BPCellsMatrix'
x %*% y

## S4 method for signature 'BPCellsMatrix,matrix'
x %*% y

## S4 method for signature 'matrix,BPCellsMatrix'
x %*% y

## S4 method for signature 'BPCellsMatrix,numeric'
x %*% y
```

```
## S4 method for signature 'numeric,BPCellsMatrix'
x %*% y
```

Arguments

x, y	One of x or y must be a BPCellsMatrix object, and the another must be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

Value

A dense matrix if one of x or y is a regular matrix or atomic vector. Otherwise, a BPCellsMatrix object.

See Also

- crossprod
- tcrossprod

BPCells-Summarization *BPCellsMatrix row/col summarization*

Description

Calculate matrix stats

Usage

```
matrix_stats(object, ...)

## S4 method for signature 'BPCellsMatrix'
matrix_stats(object, ...)

## S4 method for signature 'BPCellsMatrix'
rowSums(x)

## S4 method for signature 'BPCellsMatrix'
colSums(x)

## S4 method for signature 'BPCellsMatrix'
rowMeans(x)

## S4 method for signature 'BPCellsMatrix'
colMeans(x)

## S4 method for signature 'BPCellsMatrix'
```

```
rowVars(x)

## S4 method for signature 'BPCellsMatrix'
colVars(x)

## S4 method for signature 'BPCellsMatrix'
rowSds(x)

## S4 method for signature 'BPCellsMatrix'
colSds(x)

## S4 method for signature 'BPCellsMatrix'
rowMaxs(x)

## S4 method for signature 'BPCellsMatrix'
colMaxs(x)
```

Arguments

object	A BPCellsMatrix object.
...	Arguments passed on to BPCells::matrix_stats row_stats Which row statistics to compute col_stats Which col statistics to compute threads Number of threads to use during execution
x	A BPCellsMatrix object.

Details

The statistics will be calculated in a single pass over the matrix, so this method is desirable to use for efficiency purposes compared to the more standard rowMeans or colMeans if multiple statistics are needed. The stats are ordered by complexity: nonzero, mean, then variance. All less complex stats are calculated in the process of calculating a more complicated stat. So to calculate mean and variance simultaneously, just ask for variance, which will compute mean and nonzero counts as a side-effect

Value

- matrix_stats: A list of
  - row_stats: matrix of n_stats x n_rows
  - col_stats: matrix of n_stats x n_cols
- rowSums(): vector of row sums
- colSums(): vector of column sums
- rowMeans(): vector of row means
- colMeans(): vector of column means

- `rowVars()`: vector of row variance
- `colVars()`: vector of column variance
- `rowSds()`: vector of row Standard Deviation
- `colSds()`: vector of column Standard Deviation
- `rowMaxs()`: vector of row max values
- `colMaxs()`: vector of column max values

`BPCells-tcrossprod` *Matrix Products of Transpose*

Description

Given matrices x and y as arguments, return a matrix product of Transpose.

Usage

```
## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
tcrossprod(x, y)

## S4 method for signature 'BPCellsMatrix,ANY'
tcrossprod(x, y)

## S4 method for signature 'ANY,BPCellsMatrix'
tcrossprod(x, y)

## S4 method for signature 'BPCellsMatrix,matrix'
tcrossprod(x, y)

## S4 method for signature 'matrix,BPCellsMatrix'
tcrossprod(x, y)

## S4 method for signature 'BPCellsMatrix,numeric'
tcrossprod(x, y)

## S4 method for signature 'numeric,BPCellsMatrix'
tcrossprod(x, y)
```

Arguments

x, y	One of x or y must be a BPCellsMatrix object, and the another must be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

Value

A dense matrix if one of x or y is a regular matrix or atomic vector. Otherwise, a BPCellsMatrix object.

See Also

- %*%
- crossprod

BPCells10xHDF5-IO *Read/write sparse matrices from (or into) 10x feature matrix*

Description

- readBPCells10xHDF5Matrix: read a sparce matrices from a HDF5 file on disk
- writeBPCells10xHDF5Matrix: Write a sparce matrices into a HDF5 file on disk

Usage

```
readBPCells10xHDF5Matrix(path, feature_type = NULL, buffer_size = 16384L)

writeBPCells10xHDF5Matrix(x, ...)

## S4 method for signature 'ANY'
writeBPCells10xHDF5Matrix(
  x,
  path,
  barcodes = colnames(x),
  feature_ids = rownames(x),
  feature_names = rownames(x),
  feature_types = "Gene Expression",
  feature_metadata = list(),
  buffer_size = 16384L,
  chunk_size = 1024L,
  gzip = 0L
)
```

Arguments

path	A string path of the 10x HDF5 file to read or save data into.
feature_type	Optional selection of feature types to include in output matrix. For multiome data, the options are "Gene Expression" and "Peaks". This option is only compatible with files from cellranger 3.0 and newer.
buffer_size	For performance tuning only. The number of items to be buffered in memory before calling writes to disk.

x	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
...	Additional arguments passed into specific methods.
barcodes	Vector of names for the cells
feature_ids	Vector of IDs for the features
feature_names	Vector of names for the features
feature_types	String or vector of feature types
feature_metadata	Named list of additional metadata vectors to store for each feature
chunk_size	For performance tuning only. The chunk size used for the HDF5 array storage.
gzip	Gzip compression level. Default is 0 (no gzip compression).

Details

The 10x format makes use of gzip compression for the matrix data, which can slow down read performance. Consider writing into another format if the read performance is important to you.

Input matrices must be in column-major storage order, and if the rownames and colnames are not set, names must be provided for the relevant metadata parameters. Some of the metadata parameters are not read by default in BPCells, but it is possible to export them for use with other tools.

Value

A BPCellsMatrix object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

BPCellsAnnHDF5-IO *Read/write sparse matrices from (or into) AnnData HDF5 file*

Description

- readBPCellsAnnHDF5Matrix: read a sparce matrices from a AnnData HDF5 file on disk.
- writeBPCellsAnnHDF5Matrix: Write a sparce matrices into a AnnData HDF5 file on disk.

Usage

```
readBPCellsAnnHDF5Matrix(path, group = "X", buffer_size = 16384L)

writeBPCellsAnnHDF5Matrix(x, ...)

## S4 method for signature 'ANY'
writeBPCellsAnnHDF5Matrix(
  x,
  path,
  group = "X",
  buffer_size = 16384L,
  chunk_size = 1024L,
  gzip = 0L
)
```

Arguments

path	A string path of the AnnData HDF5 file to read or save data into.
group	The group within the hdf5 file to write the data to. If writing to an existing hdf5 file this group must not already be in use
buffer_size	For performance tuning only. The number of items to be buffered in memory before calling writes to disk.
x	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
...	Additional arguments passed into specific methods.
chunk_size	For performance tuning only. The chunk size used for the HDF5 array storage.
gzip	Gzip compression level. Default is 0 (no gzip compression).

Value

A BPCellsMatrix object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

BPCellsDelayedOp-class *BPCellsDelayedOp objects*

Description

Provide a parallel DelayedOp object

Usage

```
## S3 method for class 'BPCellsDelayedOp'
as.matrix(x)

## S4 method for signature 'BPCellsDelayedOp'
as.matrix(x)

## S3 method for class 'BPCellsDelayedOp'
as.array(x, drop = FALSE)

## S4 method for signature 'BPCellsDelayedOp'
as.array(x, drop = FALSE)

## S4 method for signature 'BPCellsDelayedOp'
type(x)

## S4 method for signature 'BPCellsDelayedOp'
is_sparse(x)

## S4 method for signature 'BPCellsDelayedOp'
extract_array(x, index)

## S4 method for signature 'BPCellsDelayedOp'
OLD_extract_sparse_array(x, index)

## S4 method for signature 'BPCellsDelayedOp'
extract_sparse_array(x, index)

## S4 method for signature 'BPCellsDelayedOp'
dim(x)

## S4 method for signature 'BPCellsDelayedOp'
dimnames(x)

## S4 method for signature 'BPCellsDelayedOp'
t(x)

## S4 method for signature 'BPCellsDelayedOp'
```

```
chunkdim(x)

## S4 method for signature 'BPCellsDelayedAbind'
is_noop(x)

## S4 method for signature 'BPCellsDelayedRenameDims'
is_noop(x)

## S4 method for signature 'BPCellsDelayedSubset'
is_noop(x)
```

Arguments

x	A BPCellsDelayedOp object.
drop	A bool, if TRUE, any extents of length one will be removed and return an atomic vector.
index	An unnamed list of integer vectors, one per dimension in x. Each vector is called a *subscript* and can only contain positive integers that are valid 1-based indices along the corresponding dimension in x. Empty or missing subscripts are allowed. They must be represented by list elements set to integer(0) or NULL, respectively. The subscripts cannot contain NAs or non-positive values. Individual subscripts are allowed to contain duplicated indices.

Value

- type: A string, indicates the storage type. For all BPCells matrix type of float and double, always return double since R cannot differentiate 32-bit and 64-bit real number. See storage_mode.
- extract_array: A dense matrix.
- OLD_extract_sparse_array: A SparseArraySeed object.
- extract_sparse_array: A SparseArray object.
- is_sparse: Always return TRUE.
- chunkdim: the chunk dimensions in an integer vector parallel to dim(x).

Note

Just like DelayedOp object, this is not intented used by users directly.

BPCellsDir-IO *Read/write sparse matrices from (or into) directory on disk*

Description

- `readBPCellsDirMatrix`: read a sparce matrices from a directory on disk
- `writeBPCellsDirMatrix`: Write a sparce matrices into a directory on disk

Usage

```
readBPCellsDirMatrix(path, buffer_size = 8192L)

writeBPCellsDirMatrix(x, ...)

## S4 method for signature 'ANY'
writeBPCellsDirMatrix(
  x,
  path = NULL,
  bitpacking = TRUE,
  buffer_size = 8192L,
  overwrite = FALSE
)
```

Arguments

`path`	A string path of directory to read or save the data into. For `writeBPCellsDirMatrix`, this can be `NULL` means using a temporary directory.
`buffer_size`	For performance tuning only. The number of items to be buffered in memory before calling writes to disk.
`x`	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
`...`	Additional arguments passed into specific methods.
`bitpacking`	A bool, whether or not to compress the data using Bitpacking Compression.
`overwrite`	A bool, If `TRUE`, write to a temp dir then overwrite existing data.

Details

Storage locations:

Matrices can be stored in a directory on disk, in memory, or in an HDF5 file. Saving in a directory on disk is a good default for local analysis, as it provides the best I/O performance and lowest memory usage. The HDF5 format allows saving within existing hdf5 files to group data together, and the in memory format provides the fastest performance in the event memory usage is unimportant.

Bitpacking Compression:

For typical RNA counts matrices holding integer counts, this bitpacking compression will result in 6-8x less space than an R dgCMatrix, and 4-6x smaller than a scipy csc_matrix. The compression will be more effective when the count values in the matrix are small, and when the rows of the matrix are sorted by rowMeans. In tests on RNA-seq data optimal ordering could save up to 40% of storage space. On non-integer data only the row indices are compressed, not the values themselves so space savings will be smaller.

For non-integer data matrices, bitpacking compression is much less effective, as it can only be applied to the indexes of each entry but not the values. There will still be some space savings, but far less than for counts matrices.

Value

A BPCellsMatrix object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

`BPCellsHDF5-IO` *Read/write sparse matrices from (or into) HDF5 file*

Description

- `readBPCellsHDF5Matrix`: read a sparce matrices from a HDF5 file on disk
- `writeBPCellsHDF5Matrix`: Write a sparce matrices into a HDF5 file on disk

Usage

```
readBPCellsHDF5Matrix(path, group, buffer_size = 8192L)

writeBPCellsHDF5Matrix(x, ...)

## S4 method for signature 'ANY'
writeBPCellsHDF5Matrix(
  x,
  path,
  group,
  bitpacking = TRUE,
  buffer_size = 8192L,
```

```
  chunk_size = 1024L,
  overwrite = FALSE,
  gzip = 0L
)
```

Arguments

path	A string path of the HDF5 file to read or save data into.
group	The group within the hdf5 file to write the data to. If writing to an existing hdf5 file this group must not already be in use
buffer_size	For performance tuning only. The number of items to be buffered in memory before calling writes to disk.
x	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
...	Additional arguments passed into specific methods.
bitpacking	A bool, whether or not to compress the data using Bitpacking Compression.
chunk_size	For performance tuning only. The chunk size used for the HDF5 array storage.
overwrite	A bool, If TRUE, write to a temp dir then overwrite existing data.
gzip	Gzip compression level. Default is 0 (no gzip compression). This is recommended when both compression and compatibility with outside programs is required. Using bitpacking=TRUE is recommended as it is >10x faster with often similar compression levels. So gzip will always be zero when bitpacking is TRUE.

Details

Storage locations:

Matrices can be stored in a directory on disk, in memory, or in an HDF5 file. Saving in a directory on disk is a good default for local analysis, as it provides the best I/O performance and lowest memory usage. The HDF5 format allows saving within existing hdf5 files to group data together, and the in memory format provides the fastest performance in the event memory usage is unimportant.

Bitpacking Compression:

For typical RNA counts matrices holding integer counts, this bitpacking compression will result in 6-8x less space than an R dgCMatrix, and 4-6x smaller than a scipy csc_matrix. The compression will be more effective when the count values in the matrix are small, and when the rows of the matrix are sorted by rowMeans. In tests on RNA-seq data optimal ordering could save up to 40% of storage space. On non-integer data only the row indices are compressed, not the values themselves so space savings will be smaller.

For non-integer data matrices, bitpacking compression is much less effective, as it can only be applied to the indexes of each entry but not the values. There will still be some space savings, but far less than for counts matrices.

Value

A BPCellsMatrix object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

`BPCellsMem-IO` *Write a sparce matrices into memory with BPCells format*

Description

Write a sparce matrices into memory with BPCells format

Usage

```
writeBPCellsMemMatrix(x, ...)

## S4 method for signature 'ANY'
writeBPCellsMemMatrix(x, compress = TRUE)
```

Arguments

`x`	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
`...`	Additional arguments passed into specific methods.
`compress`	Whether or not to compress the data.

Details

Storage locations:

Matrices can be stored in a directory on disk, in memory, or in an HDF5 file. Saving in a directory on disk is a good default for local analysis, as it provides the best I/O performance and lowest memory usage. The HDF5 format allows saving within existing hdf5 files to group data together, and the in memory format provides the fastest performance in the event memory usage is unimportant.

Bitpacking Compression:

For typical RNA counts matrices holding integer counts, this bitpacking compression will result in 6-8x less space than an R dgCMatrix, and 4-6x smaller than a scipy csc_matrix. The compression will be more effective when the count values in the matrix are small, and when the rows of the matrix are sorted by rowMeans. In tests on RNA-seq data optimal ordering could save up to 40% of storage space. On non-integer data only the row indices are compressed, not the values themselves so space savings will be smaller.

For non-integer data matrices, bitpacking compression is much less effective, as it can only be applied to the indexes of each entry but not the values. There will still be some space savings, but far less than for counts matrices.

Value

A BPCellsMatrix object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

`BPCellsSeed` *Transform into* `IterableMatrix`

Description

Transform into `IterableMatrix`

Usage

```
BPCellsSeed(x)

## S4 method for signature 'IterableMatrix'
BPCellsSeed(x)

## S4 method for signature 'matrix'
BPCellsSeed(x)

## S4 method for signature 'dgCMatrix'
BPCellsSeed(x)

## S4 method for signature 'ANY'
BPCellsSeed(x)
```

Arguments

`x`	A `IterableMatrix` object from `BPCells` or a matrix-like object which can be coerced into dgCMatrix object.

Value

A `IterableMatrix` object.

See Also

- BPCellsSeed
- BPCellsDir-IO
- BPCellsHDF5-IO
- BPCellsMem-IO
- BPCells10xHDF5-IO
- BPCellsAnnHDF5-IO

`BPCellsSeed-class` `IterableMatrix` *object methods*

Description

`IterableMatrix` object methods

Usage

```
## S4 method for signature 'IterableMatrix'
type(x)

## S3 method for class 'IterableMatrix'
as.array(x, drop = FALSE)

## S4 method for signature 'IterableMatrix'
as.array(x, drop = FALSE)

## S4 method for signature 'IterableMatrix'
extract_array(x, index)

## S4 method for signature 'IterableMatrix'
OLD_extract_sparse_array(x, index)

## S4 method for signature 'IterableMatrix'
extract_sparse_array(x, index)

## S4 method for signature 'IterableMatrix'
is_sparse(x)

## S4 method for signature 'IterableMatrix'
chunkdim(x)
```

Arguments

x	A IterableMatrix object.
drop	A bool, if TRUE, any extents of length one will be removed and return an atomic vector.
index	An unnamed list of integer vectors, one per dimension in x. Each vector is called a *subscript* and can only contain positive integers that are valid 1-based indices along the corresponding dimension in x. Empty or missing subscripts are allowed. They must be represented by list elements set to integer(0) or NULL, respectively. The subscripts cannot contain NAs or non-positive values. Individual subscripts are allowed to contain duplicated indices.

Value

- type: A string, indicates the storage type. For all BPCells matrix type of float and double, always return double since R cannot differentiate 32-bit and 64-bit real number. See storage_mode.
- extract_array: A dense matrix.
- OLD_extract_sparse_array: A SparseArraySeed object.
- extract_sparse_array: A SparseArray object.
- is_sparse: Always return TRUE.
- chunkdim: the chunk dimensions in an integer vector parallel to dim(x).

---

convert_mode *Convert the storage mode of a BPCellsArray object*

Description

Convert the storage mode of a BPCellsArray object

Usage

```
convert_mode(object, ...)

## S4 method for signature 'BPCellsMatrix'
convert_mode(object, mode)

storage_mode(object)

## S4 method for signature 'BPCellsMatrix'
storage_mode(object)
```

```
## S4 method for signature 'BPCellsDelayedOp'
storage_mode(object)

## S4 method for signature 'IterableMatrix'
storage_mode(object)

## S4 method for signature 'matrix'
storage_mode(object)
```

Arguments

object	A BPCellsMatrix object.
...	Additional parameters passed into specific methods.
mode	Storage mode of BPCells matrix, one of uint32_t (unsigned 32-bit integer), float (32-bit real number), or double (64-bit real number). R cannot differentiate 32-bit and 64-bit real number, so type method always return "double" for both float and double mode.

Value

- convert_mode: A BPCellsMatrix object with storage mode converted into the specified.
- storage_mode: A string indicates the storage mode.

See Also

convert_matrix_type

DelayedArray,BPCellsDelayedOp-method

House of internal methods

Description

Following methods are used by package internal, usually for messages purpose.

Usage

```
## S4 method for signature 'BPCellsDelayedOp'
DelayedArray(seed)

## S4 method for signature 'BPCellsMatrix'
BPCellsSeed(x)

## S4 method for signature 'numeric,BPCellsMatrix'
e1 / e2
```

```
## S4 method for signature 'numeric,BPCellsMatrix'
e1 %% e2

## S4 method for signature 'BPCellsMatrix,numeric'
e1 %% e2

## S4 method for signature 'numeric,BPCellsMatrix'
e1 %/% e2

## S4 method for signature 'BPCellsMatrix,numeric'
e1 %/% e2

## S4 method for signature 'BPCellsDelayedAbind'
set_threads(object, threads = 0L)

## S4 method for signature 'ColBindMatrices'
set_threads(object, threads = 0L)

## S4 method for signature 'RowBindMatrices'
set_threads(object, threads = 0L)

## S4 method for signature 'ANY'
set_threads(object, ...)

## S4 method for signature 'ANY,BPCellsMatrix'
rbind2(x, y, ...)

## S4 method for signature 'BPCellsMatrix,ANY'
rbind2(x, y, ...)

## S4 method for signature 'ANY,BPCellsMatrix'
cbind2(x, y, ...)

## S4 method for signature 'BPCellsMatrix,ANY'
cbind2(x, y, ...)

## S4 method for signature 'ANY'
binarize(object, ...)

## S4 method for signature 'BPCellsMatrix,ANY'
Compare(e1, e2)

## S4 method for signature 'ANY,BPCellsMatrix'
Compare(e1, e2)

## S4 method for signature 'ANY'
convert_mode(object, mode)
```

```
## S4 method for signature 'ANY,ANY'
mask_matrix(object, mask, invert = FALSE)

## S4 method for signature 'ANY'
expm1_slow(x)

## S4 method for signature 'BPCellsMatrix'
log(x)

## S4 method for signature 'BPCellsArray'
Math(x)

## S4 method for signature 'BPCellsArray'
Math2(x)

## S4 method for signature 'ANY'
rank_transform(object, axis = NULL, offset = TRUE, ...)

## S4 replacement method for signature 'BPCellsMatrix,ANY,ANY,dgCMatrix'
x[i, j] <- value

## S4 replacement method for signature 'BPCellsMatrix,ANY,ANY,matrix'
x[i, j] <- value

## S4 method for signature 'ANY'
transpose_axis(object, ...)

## S4 method for signature 'BPCellsMatrix,vector'
pmin2(e1, e2)

## S4 method for signature 'vector,BPCellsMatrix'
pmin2(e1, e2)

## S4 method for signature 'BPCellsMatrix,DelayedArray'
pmin2(e1, e2)

## S4 method for signature 'DelayedArray,BPCellsMatrix'
pmin2(e1, e2)

## S4 method for signature 'BPCellsMatrix,vector'
pmax2(e1, e2)

## S4 method for signature 'vector,BPCellsMatrix'
pmax2(e1, e2)

## S4 method for signature 'BPCellsMatrix,DelayedArray'
pmax2(e1, e2)
```

```
## S4 method for signature 'DelayedArray,BPCellsMatrix'
pmax2(e1, e2)
```

Arguments

seed	A BPCellsDelayedOp object.
x, y	See method signature.
e1, e2	One of e1 or e2 must be a BPCellsMatrix object, and the another can be a BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
object	A BPCellsMatrix object with a seed slot of BPCellsDelayedAbind object, usually derived from bind operators.
threads	Set the number of threads to use for sparse-dense multiply and matrix_stats.
...	Additional arguments to specific methods.
mode	Storage mode of BPCells matrix, one of uint32_t (unsigned 32-bit integer), float (32-bit real number), or double (64-bit real number). R cannot differentiate 32-bit and 64-bit real number, so type method always return "double" for both float and double mode.
mask	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object
invert	A bool, indicates whether revert the mask.
axis	Axis to rank values within. "col" to rank values within each column, and "row" to rank values within each row. If NULL, will use the storage axis of object (see storage_axis). If axis specified is different from the storage axis of object, transpose_axis will be used to transpose the underlying storage order.
offset	A bool, whether or not to add offset such that the rank of a 0 value is 0. Default: TRUE.
i, j	Row and Column index.
value	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.

---

mask_matrix *Mask matrix entries to zero*

Description

Set matrix entries to zero given a mask matrix of the same dimensions. Normally, non-zero values in the mask will set the matrix entry to zero. If inverted, zero values in the mask matrix will set the matrix entry to zero.

Usage

```
mask_matrix(object, mask, ...)

## S4 method for signature 'BPCellsMatrix,BPCellsMatrix'
mask_matrix(object, mask, invert = FALSE)

## S4 method for signature 'BPCellsMatrix,ANY'
mask_matrix(object, mask, invert = FALSE)
```

Arguments

object	A BPCellsMatrix object.
mask	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object
...	Additional parameters passed into specific methods.
invert	A bool, indicates whether revert the mask.

Value

A BPCellsMatrix object.

See Also

mask_matrix

---

pmin2 *Maxima and Minima*

Description

Maxima and Minima

Usage

```
pmin_by_col(object, values)

## S4 method for signature 'BPCellsMatrix'
pmin_by_col(object, values)

pmin_by_row(object, values)

## S4 method for signature 'BPCellsMatrix'
pmin_by_row(object, values)

pmin_scalar(object, value)
```

```
## S4 method for signature 'BPCellsMatrix'
pmin_scalar(object, value)

## S4 method for signature 'BPCellsMatrix,numeric'
pmin2(e1, e2)

## S4 method for signature 'numeric,BPCellsMatrix'
pmin2(e1, e2)
```

Arguments

object	A BPCellsMatrix object.
values	An atomic positive numeric.
value	A single positive numeric value
e1, e2	One of e1 or e2 must be a BPCellsMatrix object, and the another must be a scalar or of length nrow(e1)/nrow(e2).

Value

- pmin_by_col: Take the minimum with a per-col constant
- pmin_by_row: Take the minimum with a per-row constant
- pmin_scalar: Take minumum with a global constant

Note

For pmin2 Methods listed here are supported by BPCells, other pmin2 methods and pmax2 function will use the methods defined in DelayedArray.

---

rank_transform *Rank-transform a BPCells IterableMatrix matrix*

Description

Rank values are default offset such that the rank of a 0 value is 0. If you want to get the same result of regular matrix rowRanks(matrix, ties.method = "average")/colRanks(matrix, ties.method = "average"), set offset=FALSE.

Usage

```
rank_transform(object, ...)

## S4 method for signature 'BPCellsMatrix'
rank_transform(object, axis = NULL, offset = TRUE, ...)

## S4 method for signature 'BPCellsMatrix'
```

```
rowRanks(
  x,
  rows = NULL,
  cols = NULL,
  ties.method = "average",
  ...,
  useNames = TRUE
)

## S4 method for signature 'BPCellsMatrix'
colRanks(
  x,
  rows = NULL,
  cols = NULL,
  ties.method = "average",
  preserveShape = TRUE,
  ...,
  useNames = TRUE
)
```

Arguments

`...`	Arguments passed on to `BPCells::transpose_storage_order` `outdir` Directory to store the output `tmpdir` Temporary directory to use for intermediate storage `load_bytes` The minimum contiguous load size during the merge sort passes `sort_bytes` The amount of memory to allocate for re-sorting chunks of entries
`axis`	Axis to rank values within. "col" to rank values within each column, and "row" to rank values within each row. If `NULL`, will use the storage axis of `object` (see storage_axis). If `axis` specified is different from the storage axis of `object`, transpose_axis will be used to transpose the underlying storage order.
`offset`	A bool, whether or not to add offset such that the rank of a 0 value is 0. Default: `TRUE`.
`x, object`	A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object.
`rows, cols`	Ignored currently.
`ties.method`	Always be "average", cannot be changed.
`useNames`	Always be `TRUE`, cannot be changed.
`preserveShape`	Always be `TRUE`, cannot be changed.

Value

A BPCellsMatrix object.

See Also

rank_transform

set_threads *Set matrix op thread count*

Description

Set number of threads to use for sparse-dense multiply and matrix_stats.

Usage

```
set_threads(object, ...)

## S4 method for signature 'BPCellsMatrix'
set_threads(object, threads = 0L)
```

Arguments

object	A BPCellsMatrix object with a seed slot of BPCellsDelayedAbind object, usually derived from bind operators.
...	Additional arguments to specific methods.
threads	Set the number of threads to use for sparse-dense multiply and matrix_stats.

Value

A BPCellsMatrix object.

show,BPCellsArray-method *DelayedArray backend of BPCells matrix*

Description

The BPCellsMatrix class just inherits from the DelayedMatrix class.

Usage

```
## S4 method for signature 'BPCellsArray'
show(object)

## S4 method for signature 'BPCellsMatrix'
show(object)

BPCellsArray(x)

BPCellsMatrix(x)
```

```
## S4 method for signature 'BPCellsArray'
matrixClass(x)

## S4 method for signature 'IterableMatrix'
DelayedArray(seed)

## S3 method for class 'BPCellsMatrix'
aperm(a, perm, ...)

## S4 method for signature 'BPCellsMatrix'
aperm(a, perm, ...)

## S3 method for class 'BPCellsMatrix'
as.matrix(x)

## S4 method for signature 'BPCellsMatrix'
as.matrix(x)

## S3 method for class 'BPCellsMatrix'
as.array(x, drop = FALSE)

## S4 method for signature 'BPCellsMatrix'
as.array(x, drop = FALSE)

## S4 method for signature 'BPCellsMatrix'
t(x)

## S4 replacement method for signature 'BPCellsMatrix'
type(x) <- value

## S4 method for signature 'BPCellsMatrix'
is.na(x)

## S4 method for signature 'BPCellsMatrix'
is.finite(x)

## S4 method for signature 'BPCellsMatrix'
is.infinite(x)

## S4 method for signature 'BPCellsMatrix'
is.nan(x)

## S4 method for signature 'BPCellsArray,vector'
Ops(e1, e2)

## S4 method for signature 'vector,BPCellsArray'
Ops(e1, e2)
```

```
## S4 method for signature 'BPCellsArray,BPCellsArray'
Ops(e1, e2)

## S4 replacement method for signature 'BPCellsMatrix,ListOrNULL'
dimnames(x) <- value

## S4 replacement method for signature 'BPCellsMatrix'
rownames(x) <- value

## S4 replacement method for signature 'BPCellsMatrix'
colnames(x) <- value

## S4 replacement method for signature 'BPCellsMatrix,ANY,ANY,BPCellsMatrix'
x[i, j, ...] <- value

## S4 replacement method for signature 'BPCellsMatrix,ANY,ANY,IterableMatrix'
x[i, j, ...] <- value

## S4 replacement method for signature 'BPCellsMatrix,ANY,ANY,ANY'
x[i, j] <- value

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

Arguments

object	A BPCellsMatrix object.
x	A BPCellsMatrix object. For BPCellsArray and BPCellsMatrix function, a matrix-like object which can be coerced into dgCMatrix object would also be okay.
seed	A IterableMatrix object.
a	A BPCellsMatrix object.
perm	the subscript permutation vector, usually a permutation of the integers 1:n, where n is the number of dimensions of a. When a has named dimnames, it can be a character vector of length n giving a permutation of those names. The default (used whenever perm has zero length) is to reverse the order of the dimensions.
...	Additional arguments passed to specific methods • aperm: not used currently. • [<-: arguments passed to transpose_storage_order
drop	A bool, if TRUE, any extents of length one will be removed and return an atomic vector.
value	• type<-: See the mode argument in convert_mode. • dimnames<-: A list of dimnames or NULL. • [<-: A BPCellsMatrix object or a matrix-like object which can be coerced into dgCMatrix object

e1, e2	One of e1 or e2 must be a BPCellsMatrix object.
i, j	Row and Column index.

Value

- BPCellsArray and BPCellsMatrix: A BPCellsMatrix object, since BPCells can only support 2-dim array.
- matrixClass: A string, always be "BPCellsMatrix".
- as.matrix: A dense matrix.
- as.array: A dense matrix or an atomic vector.
- t: A BPCellsMatrix object
- type<-: A BPCellsMatrix object with storage mode converted into the specified.
- dimnames<-: A BPCellsMatrix object.
- rownames<-: A BPCellsMatrix object.
- colnames<-: A BPCellsMatrix object.
- [<-: A BPCellsMatrix object.
- [: A BPCellsMatrix object or an atomic vector.

See Also

- bind: Combine two Objects by Columns or Rows.
- %*%: Matrix Multiplication.
- crossprod: Matrix Crossproduct.
- summarization: row/col summarization.
- Arith: Binary Arithmetic operators.
- Math: Math operators.
- Compare: Compare matrix.
- pmin2/pmax2: Maxima and Minima.
- DelayedArray-utils: Common operations on DelayedArray objects

showtree *Visualize and access the leaves of a tree of delayed operations*

Description

showtree can be used to visualize the tree of delayed operations carried by a DelayedArray object.

Usage

```
showtree(x, show.node.dim = TRUE)

seedApply(x, FUN, ...)
```

Arguments

x	Typically a DelayedArray object but can also be a DelayedOp object or a list where each element is a DelayedArray or DelayedOp object.
show.node.dim	TRUE or FALSE. If TRUE (the default), the nodes dimensions and data type are displayed.
FUN	The function to be applied to each leaf in x.
...	Optional arguments to FUN for seedApply(). Additional arguments passed to methods for path().

Details

Use seedApply to apply a function to the seeds of a DelayedArray object.

Value

- showtree: return the input invisiblely
- seedApply: A list of length nseed(x) for seedApply.

SpectraParam *BiocSingularParam classes*

Description

Find the Largest k Singular Values/Vectors of a Matrix using RSpectra package.

Usage

```
SpectraParam()

## S4 method for signature 'SpectraParam'
runSVD(
  x,
  k,
  nu = k,
  nv = k,
  center = FALSE,
  scale = FALSE,
  ncv = NULL,
  tol = 1e-10,
  maxitr = 1000,
  threads = 0L,
  ...,
  BSPARAM
)
```

Arguments

x	A numeric matrix-like object to use in the SVD.
k	Number of singular values requested.
nu	Number of right singular vectors to be computed. This must be between 0 and 'k'. (Must be equal to 'k' for BPCells IterableMatrix)
nv	Number of right singular vectors to be computed. This must be between 0 and k.
center	Either a logical value (TRUE/FALSE), or a numeric vector of length n. If a vector c is supplied, then SVD is computed on the matrix $A - 1c'$, in an implicit way without actually forming this matrix. center = TRUE has the same effect as center = colMeans(A). Default is FALSE. Ignored if x is a IterableMatrix object.
scale	Either a logical value (TRUE/FALSE), or a numeric vector of length n. If a vector s is supplied, then SVD is computed on the matrix $(A - 1c')S$, where c is the centering vector and $S = diag(1/s)$.If scale = TRUE, then the vector s is computed as the column norm of $A - 1c'$.Default is FALSE. Ignored if x is a IterableMatrix object.
ncv	Number of Lanzcos basis vectors to use. More vectors will result in faster convergence, but with greater memory use. ncv must be satisfy $k < ncv \leq p$ where p = min(m, n). Default is min(p, max(2*k+1, 20)).
tol	Precision parameter. Default is 1e-10.
maxitr	Maximum number of iterations. Default is 1000.
threads	Control threads to use calculating mat-vec producs (BPCells specific)
...	Not used currently
BSPARAM	A BiocSingularParam object specifying the type of algorithm to run.

See Also

RSpectra and BPCells

`transpose_axis` *Transpose the storage axis for a* `BPCellsSeed` *or* `BPCellsMatrix` *object*

Description

Transpose the storage axis for a `BPCellsSeed` or `BPCellsMatrix` object

Usage

```
transpose_axis(object, ...)

## S4 method for signature 'BPCellsMatrix'
transpose_axis(object, ...)

storage_axis(object)

## S4 method for signature 'BPCellsMatrix'
storage_axis(object)

## S4 method for signature 'BPCellsDelayedOp'
storage_axis(object)

## S4 method for signature 'IterableMatrix'
storage_axis(object)
```

Arguments

`object`	A BPCellsMatrix object.
`...`	Arguments passed on to `BPCells::transpose_storage_order` `outdir` Directory to store the output `tmpdir` Temporary directory to use for intermediate storage `load_bytes` The minimum contiguous load size during the merge sort passes `sort_bytes` The amount of memory to allocate for re-sorting chunks of entries

Details

This re-sorts the entries of a matrix to change the storage order from row-major to col-major. For large matrices, this can be slow – around 2 minutes to transpose a 500k cell RNA-seq matrix The default load_bytes (4MiB) and sort_bytes (1GiB) parameters allow ~85GB of data to be sorted with two passes through the data, and ~7.3TB of data to be sorted in three passes through the data.

Value

- `transpose_axis`: A BPCellsMatrix object with storage axis flipped. Note: `identical(as.matrix(transpose_axis(object)), as.matrix(object))` is `TRUE`.
- `storage_axis`: A string indicates the storage axis, `"row"` or `"col"`.

Index