# Package 'SCArray'

May 16, 2026

**Type** Package

**Title** Large-scale single-cell omics data manipulation with GDS files

**Version** 1.21.0

**Date** 2026-04-13

**Depends** R (>= 3.5.0), gdsfmt (>= 1.36.0), methods, DelayedArray (>= 0.31.5)

**Imports** S4Vectors, utils, Matrix, SparseArray (>= 1.5.13), BiocParallel, DelayedMatrixStats, SummarizedExperiment, SingleCellExperiment, BiocSingular

**Suggests** BiocGenerics, scater, scuttle, uwot, RUnit, knitr, markdown, rmarkdown, rhdf5, HDF5Array

**Description** Provides large-scale single-cell omics data manipulation using Genomic Data Structure (GDS) files. It combines dense and sparse matrices stored in GDS files and the Bioconductor infrastructure framework (SingleCellExperiment and DelayedArray) to provide out-of-memory data storage and large-scale manipulation using the R programming language.

**License** GPL-3

**VignetteBuilder** knitr

**ByteCompile** TRUE

**URL** https://github.com/AbbVie-ComputationalGenomics/SCArray

**biocViews** Infrastructure, DataRepresentation, DataImport, SingleCell, RNASeq

**git_url** https://git.bioconductor.org/packages/SCArray

**git_branch** devel

**git_last_commit** 4fcd13e

**git_last_commit_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-15

**Author** Xiuwen Zheng [aut, cre] (ORCID: <https://orcid.org/0000-0002-1390-0708>)

**Maintainer** Xiuwen Zheng <xiuwen.zheng@abbvie.com>

# Contents



---

`SCArray-package` *Large-scale single-cell omics data manipulation with GDS files*

---

## Description

The package combines dense/sparse matrices stored in GDS files and the Bioconductor infrastructure framework to provide out-of-memory data storage and manipulation using the R programming language.

## Details

| | |
|---|---|
| Package: | SCArray |
| Type: | Package |
| License: | GPL version 3 |

## Author(s)

Xiuwen Zheng <xiuwen.zheng@abbvie.com>

## Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

sce <- scExperiment(fn)
sce

rm(sce)
```

---

`row_nnzero` *Numbers of Non-zeros*

---

## Description

Calculates the numbers of non-zeros for each row or column of a matrix-like object.

## Usage

```
row_nnzero(x, na.counted=NA, ...)
col_nnzero(x, na.counted=NA, ...)

## S4 method for signature 'matrix'
row_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'Matrix'
row_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'DelayedMatrix'
row_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'SC_GDSMatrix'
row_nnzero(x, na.counted=NA, ...)

## S4 method for signature 'matrix'
col_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'Matrix'
col_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'DelayedMatrix'
col_nnzero(x, na.counted=NA, ...)
## S4 method for signature 'SC_GDSMatrix'
col_nnzero(x, na.counted=NA, ...)
```

## Arguments

| | |
|---|---|
| `x` | a matrix-like object |
| `na.counted` | a logical: TRUE for counting NA/NaN as non-zero, FALSE for counting NA/NaN as zero, NA (default) for return NA when encountering NA/NaN |
| `...` | additional arguments passed to specific methods |

## Value

Return an integer vector object for the numbers of non-zeros.

### Author(s)

Xiuwen Zheng

### Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

cnt <- scArray(fn, "counts")
cnt

row_nnzero(cnt, na.counted=TRUE)
col_nnzero(cnt, na.counted=TRUE)


rm(cnt)
```

---

## scArray *Get an DelayedArray instance*

---

### Description

Gets an DelayedArray instance from a single-cell omics GDS file.

### Usage

```
scArray(gdsfile, varname)
```

### Arguments

| | |
|---|---|
| gdsfile | character for a file name, or a single-cell GDS object with class SCArrayFileClass |
| varname | character for the node name in the GDS file |

### Value

Return an object of class DelayedArray.

### Author(s)

Xiuwen Zheng

### See Also

scOpen, scExperiment

### Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

cnt <- scArray(fn, "counts")
cnt

rm(cnt)
```

## SCArray-classes *Class list defined in SCArray*

### Description

SCArrayFileClass is a class directly inheriting from gds.class. SC_GDSArray is a DelayedArray with a SCArraySeed. SC_GDSMatrix is 2-dim SC_GDSArray.

The package combines dense/sparse matrices stored in GDS files and the Bioconductor infrastructure framework to provide out-of-memory data storage and manipulation using the R programming language.

### Author(s)

Xiuwen Zheng <xiuwen.zheng@abbvie.com>

## SCArray-stats *SC_GDSMatrix row/column summarization*

### Description

The row/column summarization methods for the SC_GDSMatrix matrix, extending the S4 methods in the **DelayedArray** and **DelayedMatrixStats** packages.

### Usage

```
## S4 method for signature 'SC_GDSMatrix'
rowSums(x, na.rm=FALSE, dims=1)
## S4 method for signature 'SC_GDSMatrix'
colSums(x, na.rm=FALSE, dims=1)
## S4 method for signature 'SC_GDSMatrix'
rowSums2(x, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colSums2(x, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowLogSumExps(lx, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colLogSumExps(lx, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowProds(x, rows=NULL, cols=NULL, na.rm=FALSE,
    method=c("direct", "expSumLog"), ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colProds(x, rows=NULL, cols=NULL, na.rm=FALSE,
    method=c("direct", "expSumLog"), ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowMeans(x, na.rm=FALSE, dims=1)
## S4 method for signature 'SC_GDSMatrix'
```

```
colMeans(x, na.rm=FALSE, dims=1)
## S4 method for signature 'SC_GDSMatrix'
rowMeans2(x, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colMeans2(x, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
rowWeightedMeans(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colWeightedMeans(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowVars(x, rows=NULL, cols=NULL, na.rm=FALSE, center=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colVars(x, rows=NULL, cols=NULL, na.rm=FALSE, center=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
rowWeightedVars(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colWeightedVars(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowSds(x, rows=NULL, cols=NULL, na.rm=FALSE, center=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colSds(x, rows=NULL, cols=NULL, na.rm=FALSE, center=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
rowWeightedSds(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colWeightedSds(x, w=NULL, rows=NULL, cols=NULL, na.rm=FALSE, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowMins(x, rows=NULL, cols=NULL, na.rm=FALSE)
## S4 method for signature 'SC_GDSMatrix'
colMins(x, rows=NULL, cols=NULL, na.rm=FALSE)
## S4 method for signature 'SC_GDSMatrix'
rowMaxs(x, rows=NULL, cols=NULL, na.rm=FALSE)
## S4 method for signature 'SC_GDSMatrix'
colMaxs(x, rows=NULL, cols=NULL, na.rm=FALSE)
## S4 method for signature 'SC_GDSMatrix'
rowRanges(x, rows=NULL, cols=NULL, na.rm=FALSE)
## S4 method for signature 'SC_GDSMatrix'
colRanges(x, rows=NULL, cols=NULL, na.rm=FALSE)

# Get means and variances together for each row or column,
#     return a matrix with two columns for mean and variance
scRowMeanVar(x, na.rm=FALSE, useNames=FALSE, ...)
scColMeanVar(x, na.rm=FALSE, useNames=FALSE, ...)
## S4 method for signature 'SC_GDSMatrix'
scRowMeanVar(x, na.rm=FALSE, useNames=FALSE, ...)
## S4 method for signature 'SC_GDSMatrix'
scColMeanVar(x, na.rm=FALSE, useNames=FALSE, ...)

# Compute column sums across rows
## S4 method for signature 'SC_GDSMatrix'
```

```
rowsum(x, group, reorder=TRUE, na.rm=FALSE, ...)
# Compute row sums across columns
## S4 method for signature 'SC_GDSMatrix'
colsum(x, group, reorder=TRUE, na.rm=FALSE, ...)

## S4 method for signature 'SC_GDSMatrix'
rowAnyNAs(x, rows=NULL, cols=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colAnyNAs(x, rows=NULL, cols=NULL, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowCollapse(x, idxs, rows=NULL, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colCollapse(x, idxs, cols=NULL, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowDiffs(x, rows=NULL, cols=NULL, lag=1L,
    differences=1L, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colDiffs(x, rows=NULL, cols=NULL, lag=1L,
    differences=1L, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowSdDiffs(x, rows=NULL, cols=NULL, na.rm=FALSE,
    diff=1L, trim=0, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colSdDiffs(x, rows=NULL, cols=NULL, na.rm=FALSE,
    diff=1L, trim=0, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowVarDiffs(x, rows=NULL, cols=NULL, na.rm=FALSE,
    diff=1L, trim=0, ..., useNames=NA)
## S4 method for signature 'SC_GDSMatrix'
colVarDiffs(x, rows=NULL, cols=NULL, na.rm=FALSE,
    diff=1L, trim=0, ..., useNames=NA)

## S4 method for signature 'SC_GDSMatrix'
rowAvgsPerColSet(X, W=NULL, rows=NULL, S, FUN=rowMeans, ...,
    na.rm=NA, tFUN=FALSE)
## S4 method for signature 'SC_GDSMatrix'
colAvgsPerRowSet(X, W=NULL, cols=NULL, S, FUN=colMeans, ...,
    na.rm=NA, tFUN=FALSE)
```

## Arguments

| | |
|---|---|
| x, lx, X | A SC_GDSMatrix object (inherited from DelayedMatrix) |
| dims | not used, it should be 1 |
| rows, cols | specify the subset of rows (and/or columns) to operate over; if NULL, no subsetting |
| na.rm | if TRUE, missing values (NaN and NA) will be removed |
| w | NULL or a numeric vector for weights |

| | |
|---|---|
| W | NULL or a matrix for weights |
| center | NULL, or a vector of pre-calculated row (column) means |
| useNames | if TRUE, the name attributes of result are set |
| method | "direct" (by default) or "expSumLog" (calculates the product via the logarithmic transform) |
| group | a vector for grouping the rows or columns |
| reorder | if TRUE, order the resulting matrix as sort(unique(group)); otherwise, it will be in the order that groups were encountered |
| idxs | An index vector specifying the columns (rows) to be extracted; the vector will be reused if the length is less than the number of columns or rows |
| lag | the lag, an integer |
| differences, diff | the order of difference, an integer |
| trim | fraction of observations to be trimmed |
| S | an integer matrix specifying the subsets, see rowAvgsPerColSet |
| FUN | summary statistic function, see rowAvgsPerColSet |
| tFUN | If TRUE, X is transposed before it is passed to FUN, see rowAvgsPerColSet |
| ... | additional arguments passed to specific methods: BPPARAM can be specified (if not specified, getAutoBPPARAM() is used instead) |

## Details

All these operations are block-processed according to the data stored in the GDS file.

## Author(s)

Xiuwen Zheng

## See Also

- The **DelayedMatrixStats** package for more row/column summarization methods for DelayedMatrix objects.
- DelayedArray-utils for other common operations on DelayedMatrix objects.
- DelayedMatrix objects.
- matrix objects in base R.
- getAutoBPPARAM, BiocParallelParam for parallel processing,
- The **MatrixGenerics** package for more row/column summarization methods.

SCArray-utils *SC_GDSArray subsetting, Ops, Math*

## Description

Subsetting, Arith, Compare, Logic and Math operations on the SC_GDSArray object.

## Usage

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

## S4 method for signature 'SC_GDSArray'
Ops(e1, e2)
## S4 method for signature 'SC_GDSArray'
Math(x)

# names(x) <- value
# dimnames(x) <- value

# Centers and/or scales the columns of a matrix
## S4 method for signature 'SC_GDSMatrix'
scale(x, center=TRUE, scale=TRUE)

## S4 method for signature 'SC_GDSArray,SC_GDSArray'
pmin2(e1, e2)
## S4 method for signature 'SC_GDSArray,vector'
pmin2(e1, e2)
## S4 method for signature 'vector,SC_GDSArray'
pmin2(e1, e2)
## S4 method for signature 'SC_GDSArray,SC_GDSArray'
pmax2(e1, e2)
## S4 method for signature 'SC_GDSArray,vector'
pmax2(e1, e2)
## S4 method for signature 'vector,SC_GDSArray'
pmax2(e1, e2)
```

## Arguments

| | |
|---|---|
| x | A SC_GDSArray or SC_GDSMatrix object |
| i, j, ... | indices specifying elements to extract |
| drop | if TRUE the result will be coerced to the lowest possible dimension |
| e1, e2 | objects |
| value | NULL, a character vector for names<- or a list of character vectors for dimnames<- |
| center | either a logical value or a numeric vector (e.g., FALSE or 0 for no centering) |
| scale | either a logical value or a numeric vector (e.g., TRUE or 1 for no scaling) |

## Details

All these operations return a SC_GDSArray or SC_GDSMatrix object.

`Arith`: "+", "-", "*", "^", "%%", "%/%", "/"

`Compare`: "==", ">", "<", "!=", "<=", ">="

`Logic`: "&", "|".

`Ops`: "Arith", "Compare", "Logic"

`Math`: "abs", "sign", "sqrt", "ceiling", "floor", "trunc", "cummax", "cummin", "cumprod", "cumsum", "log", "log10", "log2", "log1p", "acos", "acosh", "asin", "asinh", "atan", "atanh", "exp", "expm1", "cos", "cosh", "cospi", "sin", "sinh", "sinpi", "tan", "tanh", "tanpi", "gamma", "lgamma", "digamma", "trigamma"

## Value

All these operations return a SC_GDSArray or SC_GDSMatrix object.

## Author(s)

Xiuwen Zheng

## See Also

Ops, Math, SCArray-stats

## Examples

```
fn <- system.file("extdata", "example.gds", package="SCArray")

x <- scArray(fn, "counts")

x[1:8, 1:32]
x > 0
pmin2(x, 1)
log1p(x)
scale(x)

rm(x)
```

---

# scConvGDS *Create a GDS file*

---

## Description

Creates a single-cell GDS file from an R object.

## Usage

```
scConvGDS(obj, outfn, assay.name=NULL, save.sp=TRUE,
   type=c("float32", "float64", "int32"), compress="LZMA_RA", clean=TRUE, verbose=TRUE)
```

### Arguments

| | |
|---|---|
| obj | a dense/sparse matrix, DelayedMatrix, SummarizedExperiment or SingleCellExperiment |
| outfn | the output file name in GDS format |
| assay.name | a character vector for assay names or NULL; if NULL, to include all available assays, otherwise only include the assays in assay.name |
| save.sp | if TRUE, save it to a sparse matrix in GDS; otherwise, store dense matrix |
| type | numeric data type in the output file |
| compress | the compression method, see add.gdsn; or "" for no data compression |
| clean | TRUE |
| verbose | if TRUE, show information |

### Value

Return the path of the output file.

### Author(s)

Xiuwen Zheng

### See Also

scOpen, scClose, scMEX2GDS, scHDF2GDS

### Examples

```
# load a SingleCellExperiment object
fn <- system.file("extdata", "example.rds", package="SCArray")
sce <- readRDS(fn)
sce

scConvGDS(sce, "test.gds")

# remove the temporary file
unlink("test.gds")
```

---

## scExperiment *Get a SummarizedExperiment*

---

### Description

Gets an instance of SingleCellExperiment or SummarizedExperiment.

### Usage

```
scExperiment(gdsfile, sce=TRUE, use.names=TRUE, load.row=TRUE, load.col=TRUE)
```

### Arguments

| | |
|---|---|
| gdsfile | character for a file name, or a single-cell GDS object with class SCArrayFileClass |
| sce | if TRUE, return an instance of SingleCellExperiment, otherwise an instance of SummarizedExperiment |
| use.names | if TRUE, load dimnames from 'feature.id' and 'sample.id' |
| load.row | TRUE for loading rowData from the gds node "feature.data" in gdsfile |
| load.col | TRUE for loading colData from the gds node "sample.data" in gdsfile |

### Value

Return an instance of SingleCellExperiment or SummarizedExperiment.

### Author(s)

Xiuwen Zheng

### See Also

scOpen, scClose

### Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

sce <- scExperiment(fn)
sce

remove(sce)
```

---

## scGetFiles *File names for on-disk backend*

---

### Description

Get a list of file names for DelayedArray with an on-disk backend.

### Usage

```
scGetFiles(object, ...)
## S4 method for signature 'SC_GDSArray'
scGetFiles(object, ...)
## S4 method for signature 'SummarizedExperiment'
scGetFiles(object, ...)
```

### Arguments

| | |
|---|---|
| object | input R object (e.g., a GDS-specific DelayedArray) |
| ... | additional arguments passed to specific methods |

## Value

Return a character vector storing file names.

## Author(s)

Xiuwen Zheng

## See Also

path

---

scHDF2GDS *Convert HDF5 files to GDS*

---

## Description

Creates a single-cell GDS file from Cell Ranger HDF5 files.

## Usage

```
scHDF2GDS(h5_fn, outfn, group=c("matrix", "mm10"), feature_path=character(),
    type=c("float32", "float64", "int32"), compress="LZMA_RA", clean=TRUE,
    verbose=TRUE)
```

## Arguments

| | |
|---|---|
| h5_fn | the input HDF5 file name |
| outfn | the output file name in GDS format |
| group | the name of the group in the HDF5 file where the sparse matrix is stored; if there are more than one group names, the first existing group in the HDF5 file is used; "mm10" is usually used for 10x Genomics datasets |
| feature_path | a character vector for feature variables, otherwise detecting automatically using "genes", "gene_names" and "features/*" when available |
| type | numeric data type in the output file |
| compress | the compression method, see add.gdsn |
| clean | TRUE |
| verbose | if TRUE, show information |

## Details

The packages **rhdf5** and **HDF5Array** should be installed.

## Value

Return the path of the output file.

## Author(s)

Xiuwen Zheng

## See Also

scConvGDS, scMEX2GDS

---

scMemory *Load Data to Memory*

---

## Description

Loads the internal data to memory for any on-disk object.

## Usage

```
scMemory(x, ...)
## S4 method for signature 'DelayedArray'
scMemory(x, ...)
## S4 method for signature 'SummarizedExperiment'
scMemory(x, ...)
```

## Arguments

| | |
|---|---|
| x | input R object (e.g., a DelayedArray) |
| ... | additional arguments passed to specific methods |

## Value

Return an object (it maybe a different type compared with x).

## Author(s)

Xiuwen Zheng

## Examples

```
suppressPackageStartupMessages(library(DelayedArray))

m <- matrix(1:12, nrow=3)
(mat <- DelayedArray(m))

str(scMemory(mat))
```

scMEX2GDS *Convert MEX files to GDS*

## Description

Creates a single-cell GDS file from Cell Ranger MEX files.

## Usage

```
scMEX2GDS(feature_fn, barcode_fn, mtx_fn, outfn,
    feature_colnm=c("id", "gene", "feature_type"),
    type=c("float32", "float64", "int32"), compress="LZMA_RA", clean=TRUE,
    verbose=TRUE)
```

## Arguments

| | |
|---|---|
| feature_fn | the input file name for features |
| barcode_fn | the input file name for barcodes |
| mtx_fn | the input count matrix in MEX format |
| outfn | the output file name in GDS format |
| feature_colnm | the column names used in feature_fn |
| type | numeric data type in the output file |
| compress | the compression method, see add.gdsn |
| clean | TRUE |
| verbose | if TRUE, show information |

## Value

Return the path of the output file.

## Author(s)

Xiuwen Zheng

## See Also

scConvGDS, scHDF2GDS

scNumSplit *Split a number*

## Description

Splits a number into multiple groups with equal size.

## Usage

```
scNumSplit(num, BPPARAM=getAutoBPPARAM())
```

## Arguments

| | |
|---|---|
| num | a length-one number (the total count) for splitting (must be >= 0) |
| BPPARAM | NULL, a number for the number of groups, or a BiocParallelParam object; if not specified, call getAutoBPPARAM() |

## Value

Return a list of length-two numeric vectors for the start and end positions. BPPARAM=NULL is as the same as BPPARAM=1, if it is a BiocParallelParam object, call bpnworkers() to get the number of groups.

## Author(s)

Xiuwen Zheng

## See Also

getAutoBPPARAM, BiocParallelParam-class, bpnworkers

## Examples

```
scNumSplit(100, NULL)
scNumSplit(100, 0)
scNumSplit(100, 1)
scNumSplit(100, 3)
scNumSplit(100)

scNumSplit(0)  # zero-length
```

## scObj *DelayedArray Object in GDS*

### Description

Convert to SC_GDSArray/SC_GDSMatrix for utilizing GDS specific functions.

### Usage

```
scObj(obj, verbose=FALSE)
```

### Arguments

| | |
|---|---|
| obj | a SummarizedExperiment, SingleCellExperiment or DelayedArray object |
| verbose | if TRUE, show information |

### Value

Return the object obj with the object class DelayedArray replaced by the class SC_GDSMatrix or SC_GDSArray.

### Author(s)

Xiuwen Zheng

### See Also

scArray, scExperiment

## scOpen *Open/Close a Single-cell GDS File*

### Description

Opens or closes a single-cell GDS file.

### Usage

```
scOpen(gdsfn, readonly=TRUE, allow.duplicate=TRUE)
scClose(gdsfile)
```

### Arguments

| | |
|---|---|
| gdsfn | the input file name |
| readonly | whether read-only or not |
| allow.duplicate | if TRUE, it is allowed to open a GDS file with read-only mode when it has been opened in the same R session |
| gdsfile | a single-cell GDS object with class SCArrayFileClass |

## Value

Return an object of class SCArrayFileClass inherited from gds.class.

## Author(s)

Xiuwen Zheng

## See Also

scArray

## Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

# open the GDS file
(f <- scOpen(fn))

# read a GDS file
cell.id <- read.gdsn(index.gdsn(f, "feature.id"))
samp.id <- read.gdsn(index.gdsn(f, "sample.id"))

# get a DelayedArray object
(cnt <- scArray(f, "counts"))

scClose(f)
```

---

# scReplaceNA *Replacement*

---

## Description

Replace NA/NaN in a GDS-specific DelayedArray by a specified value.

## Usage

```
scReplaceNA(x, v=0L)
```

## Arguments

| | |
|---|---|
| x | a SC_GDSArray object |
| v | a length-one double or integer value |

## Value

Return an object with the class SC_GDSMatrix or SC_GDSArray.

## Author(s)

Xiuwen Zheng

## See Also

scSetMin, scSetMax, scSetBounds

## Examples

```
suppressPackageStartupMessages(library(DelayedArray))

m <- matrix(1:12, nrow=3)
m[2, c(1,3)] <- NA
(mat <- DelayedArray(m))

new_m <-  scObj(mat)  # wrap a in-memory DelayedMatrix
class(new_m)  # SC_GDSMatrix

scReplaceNA(new_m,  999)
```

---

# scRowAutoGrid *Automatic grids for matrix-like objects*

---

## Description

Create automatic grids (RegularArrayGrid or ArbitraryArrayGrid for sparse matrices) to use for block processing of matrix-like objects, where the blocks are made of full rows or full columns.

## Usage

```
scRowAutoGrid(x, force=FALSE, nnzero=NULL)
scColAutoGrid(x, force=FALSE, nnzero=NULL)
```

## Arguments

| | |
|---|---|
| x | a matrix-like object (e.g., a SC_GDSMatrix object) |
| force | a logical, only applicable when x is a sparse in-memory matrix or a sparse SC_GDSMatrix object, see details |
| nnzero | a numeric vector for the numbers of non-zeros for rows or columns, NULL (default) for calling row_nnzero() or col_nnzero() when needed |

## Details

The functions return regular RegularArrayGrid (calling rowAutoGrid() or colAutoGrid), when x is neither a sparse in-memory matrix nor a sparse SC_GDSMatrix object; otherwise, make use of the information of the numbers of non-zeros to create ArbitraryArrayGrid for more efficient grids. When force is applicable and force=TRUE, the functions return ArbitraryArrayGrid which needs the nnzero values. For force=FALSE, scRowAutoGrid() returns ArbitraryArrayGrid when x is not transposed, and scColAutoGrid() returns ArbitraryArrayGrid when x is transposed. If nnzero=NULL and it is needed, the numbers of non-zeros for rows or columns will be calculated internally. For a large matrix, it is more efficient when nnzero is pre-defined. The internal block size can be controlled by setAutoBlockSize(). If the number of blocks in ArbitraryArrayGrid is more than RegularArrayGrid, the functions return RegularArrayGrid instead when force is not TRUE.

Usually, gd <- scRowAutoGrid() or gd <- scColAutoGrid() is used together with blockApply(, grid=gd, as.sparse=attr(gd, "as.sparse")) or blockReduce(, grid=gd, as.sparse=attr(gd, "as.sparse")) to take advantage of sparse matrices.

## Value

Return an object of RegularArrayGrid or ArbitraryArrayGrid. attr(, "as.sparse") is a suggested logical value for as.sparse in blockApply() or blockReduce().

## Author(s)

Xiuwen Zheng

## See Also

rowAutoGrid, colAutoGrid, setAutoBlockSize, blockApply, blockReduce

## Examples

```
# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

cnt <- scArray(fn, "counts")
cnt

setAutoBlockSize(1048576)  # use 1MB

scRowAutoGrid(cnt)  # it returns RegularArrayGrid since cnt is not very sparse
rowAutoGrid(cnt)
scRowAutoGrid(cnt, force=TRUE)  # ArbitraryArrayGrid


library(Matrix)
cnt2 <- Diagonal(1e5)  # a very sparse matrix

scRowAutoGrid(cnt2)    # 5 blocks
length(rowAutoGrid(cnt2))    # 100000

scColAutoGrid(cnt2)          # 5 blocks
length(colAutoGrid(cnt2))    # 100000 blocks


setAutoBlockSize()  # reset

rm(cnt)
```

---

scRunPCA *Perform PCA on SC_GDSMatrix and expression data*

---

## Description

Perform a Principal Components Analysis (PCA) on cells in the SingleCellExperiment object.

## Usage

```
scRunPCA(sce, ncomponents=50, ntop=500, subset_row=NULL, scale=FALSE,
    altexp=NULL, name="PCA", exprs_values="logcounts", dimred=NULL,
    n_dimred=NULL, BSPARAM=NULL, BPPARAM=SerialParam(), verbose=TRUE)

## S4 method for signature 'SC_GDSMatrix'
runPCA(x, rank, center=TRUE, scale=FALSE, get.rotation=TRUE,
    get.pcs=TRUE, ...)
```

## Arguments

| | |
|---|---|
| sce | a SingleCellExperiment or SummarizedExperiment object |
| x | a SC_GDSMatrix object |
| ncomponents, rank | # of calculated principal components |
| ntop | # of features with the highest variances to use for PCA |
| subset_row | specifying the subset of features to use |
| center | if TRUE, expression values will be be centered |
| scale | if TRUE, expression values will be be standardized |
| altexp | String or integer scalar specifying an alternative experiment containing the input data |
| name | the name to be used to store the result in reducedDims |
| exprs_values | the assay name containing the expression values |
| dimred | String or integer scalar specifying the existing dimensionality reduction results to use |
| n_dimred | Integer scalar or vector specifying the dimensions to use if dimred is specified |
| BSPARAM | A BiocSingularParam object specifying which algorithm to be used in runPCA in the BiocSingular package |
| BPPARAM | A BiocParallelParam object for parallelized calculation |
| get.rotation | if TRUE, return rotation vectors |
| get.pcs | if TRUE, return principal component scores |
| verbose | if TRUE, show information |
| ... | For runPCA, this contains further arguments to pass to runSVD, including BSPARAM to specify the algorithm that should be used, and BPPARAM to control parallelization. |

## Details

The function runPCA() simply calls runSVD and converts the results into a format similar to that returned by prcomp.

BSPARAM can be one of

ExactParam(): exact SVD with runExactSVD.

IrlbaParam(): approximate SVD with **irlba** via runIrlbaSVD.

RandomParam(): approximate SVD with **rsvd** via runRandomSVD.

FastAutoParam(): fast approximate SVD, chosen based on the matrix representation.

`fold=1` in BiocSingularParam is used for the situation that the covariance matrix is relatively small, and running SVD on the small covariance matrix can be more effecient. When `fold=Inf`, running SVD on the matrix directly and will read the matrix multiple times. If it is a file-based matrix, `fold=Inf` could be slow.

## Value

Returns a SingleCellExperiment object containing the PC coordinate matrix in `reducedDims(..., name)`. The attributes of the PC coordinate matrix have "percentVar", "varExplained" and "rotation" (see `scater::runPCA` for more details).

## Author(s)

Xiuwen Zheng

## See Also

`runSVD` for the underlying SVD function.

?`BiocSingularParam` for the SVD algorithm choices.

`runPCA`.

## Examples

```
library(BiocSingular)

# a GDS file for SingleCellExperiment
fn <- system.file("extdata", "example.gds", package="SCArray")

x <- scArray(fn, "counts")
x <- x[1:200, ]
x

pc <- runPCA(x, BSPARAM=ExactParam(fold=1))  # using covariance matrix
str(pc)

rm(x)
```

---

# scSetBounds *Set the bounds*

---

## Description

Set the maximum and/or minimum on a GDS-specific DelayedArray.

## Usage

```
scSetMax(x, vmax)
scSetMin(x, vmin)
scSetBounds(x, vmin=NaN, vmax=NaN)
```

## Arguments

| | |
|---|---|
| x | a SC_GDSArray object |
| vmax | maximum, length-one |
| vmin | minimum, length-one |

## Value

Return an object with the class SC_GDSMatrix or SC_GDSArray.

## Author(s)

Xiuwen Zheng

## See Also

scReplaceNA

## Examples

```
suppressPackageStartupMessages(library(DelayedArray))

m <- matrix(1:12, nrow=3)
(mat <- DelayedArray(m))

new_m <-  scObj(mat)  # wrap a in-memory DelayedMatrix
class(new_m)  # SC_GDSMatrix

scSetMax(new_m, 5)
scSetMin(new_m, 5)
scSetBounds(new_m, 4, 9)
```

# Index