

# Package ‘SCArray’

April 1, 2025

**Type** Package

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

**Version** 1.14.0

**Date** 2023-08-02

**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** RELEASE\_3\_20

**git\_last\_commit** 7b7fe44

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2025-03-31

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

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

## Contents

SCArray-package . . . . .	2
row_nnzzero . . . . .	3
scArray . . . . .	4
SCArray-classes . . . . .	5
SCArray-stats . . . . .	5
SCArray-utils . . . . .	9
scConvGDS . . . . .	10
scExperiment . . . . .	11
scGetFiles . . . . .	12
scHDF2GDS . . . . .	13
scMemory . . . . .	14
scMEX2GDS . . . . .	15
scNumSplit . . . . .	16
scObj . . . . .	17
scOpen . . . . .	17
scReplaceNA . . . . .	18
scRowAutoGrid . . . . .	19
scRunPCA . . . . .	20
scSetBounds . . . . .	22
<b>Index</b>	<b>24</b>

---

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\_nnzzero

*Numbers of Non-zeros***Description**

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

**Usage**

```
row_nnzzero(x, na.counted=NA, ...)
col_nnzzero(x, na.counted=NA, ...)

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

## S4 method for signature 'matrix'
col_nnzzero(x, na.counted=NA, ...)
## S4 method for signature 'Matrix'
col_nnzzero(x, na.counted=NA, ...)
## S4 method for signature 'DelayedMatrix'
col_nnzzero(x, na.counted=NA, ...)
## S4 method for signature 'SC_GDSMatrix'
col_nnzzero(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'
rowAvsPerColSet(X, W=NULL, rows=NULL, S, FUN=rowMeans, ...,
  na.rm=NA, tFUN=FALSE)
## S4 method for signature 'SC_GDSMatrix'
colAvsPerRowSet(X, W=NULL, cols=NULL, S, FUN=colMeans, ...,
  na.rm=NA, tFUN=FALSE)

```

### Arguments

<code>x, lx, X</code>	A <a href="#">SC_GDSMatrix</a> object (inherited from <code>DelayedMatrix</code> )
<code>dims</code>	not used, it should be 1
<code>rows, cols</code>	specify the subset of rows (and/or columns) to operate over; if <code>NULL</code> , no subsetting
<code>na.rm</code>	if <code>TRUE</code> , missing values ( <code>NaN</code> and <code>NA</code> ) will be removed
<code>w</code>	<code>NULL</code> 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 <code>sort(unique(group))</code> ; 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 <a href="#">rowAvsPerColSet</a>
FUN	summary statistic function, see <a href="#">rowAvsPerColSet</a>
tFUN	If TRUE, X is transposed before it is passed to FUN, see <a href="#">rowAvsPerColSet</a>
...	additional arguments passed to specific methods: BPPARAM can be specified (if not specified, <code>getAutoBPPARAM()</code> 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.



**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 <a href="#">SC_GDSArray</a> or <a href="#">SC_GDSMatrix</a> 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", "dgamma", "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 <a href="#">add.gdsn</a> ; 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 <code>SCArrayFileClass</code>
sce	if TRUE, return an instance of <code>SingleCellExperiment</code> , otherwise an instance of <code>SummarizedExperiment</code>
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 <code>DelayedArray</code> )
...	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 <a href="#">add.gdsn</a>
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**

<code>feature_fn</code>	the input file name for features
<code>barcode_fn</code>	the input file name for barcodes
<code>mtx_fn</code>	the input count matrix in MEX format
<code>outfn</code>	the output file name in GDS format
<code>feature_colnm</code>	the column names used in <code>feature_fn</code>
<code>type</code>	numeric data type in the output file
<code>compress</code>	the compression method, see <a href="#">add.gdsn</a>
<code>clean</code>	TRUE
<code>verbose</code>	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 $\geq 0$ )
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](#), [bpnworkers](#)

### Examples

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

scNumSplit(0) # zero-length
```



---

scObj	<i>DelayedArray Object in GDS</i>
-------	-----------------------------------

---

**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	<i>Open/Close a Single-cell GDS File</i>
--------	--

---

**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**

<code>x</code>	a matrix-like object (e.g., a <code>SC_GDSMatrix</code> object)
<code>force</code>	a logical, only applicable when <code>x</code> is a sparse in-memory matrix or a sparse <code>SC_GDSMatrix</code> object, see details
<code>nnzero</code>	a numeric vector for the numbers of non-zeros for rows or columns, <code>NULL</code> (default) for calling <code>row_nnzero()</code> or <code>col_nnzero()</code> 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 centered
scale	if TRUE, expression values will 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`.

`Ir1baParam()`: approximate SVD with `irlba` via `runIr1baSVD`.

`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 efficient. 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 scatter::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

- \* **CellRanger**
  - scHDF2GDS, [13](#)
  - scMEX2GDS, [15](#)
- \* **GDS**
  - row\_nzero, [3](#)
  - scArray, [4](#)
  - SCArray-classes, [5](#)
  - SCArray-package, [2](#)
  - SCArray-stats, [5](#)
  - SCArray-utils, [9](#)
  - scConvGDS, [10](#)
  - scExperiment, [11](#)
  - scGetFiles, [12](#)
  - scHDF2GDS, [13](#)
  - scMemory, [14](#)
  - scMEX2GDS, [15](#)
  - scNumSplit, [16](#)
  - scObj, [17](#)
  - scOpen, [17](#)
  - scReplaceNA, [18](#)
  - scRowAutoGrid, [19](#)
  - scRunPCA, [20](#)
  - scSetBounds, [22](#)
- \* **PCA**
  - scRunPCA, [20](#)
- \* **SingleCell**
  - scArray, [4](#)
  - SCArray-classes, [5](#)
  - SCArray-package, [2](#)
  - scConvGDS, [10](#)
  - scExperiment, [11](#)
  - scHDF2GDS, [13](#)
  - scMEX2GDS, [15](#)
  - scObj, [17](#)
  - scOpen, [17](#)
  - scRunPCA, [20](#)
- \* **methods**
  - row\_nzero, [3](#)
  - SCArray-stats, [5](#)
  - SCArray-utils, [9](#)
  - scGetFiles, [12](#)
  - scMemory, [14](#)
  - + (SCArray-utils), [9](#)
  - + , SC\_GDSArray, missing-method (SCArray-utils), [9](#)
  - (SCArray-utils), [9](#)
  - , SC\_GDSArray, missing-method (SCArray-utils), [9](#)
  - [ (SCArray-utils), [9](#)
  - [ , SC\_GDSArray, ANY, ANY, ANY-method (SCArray-utils), [9](#)
  - [ , SC\_GDSArray-method (SCArray-utils), [9](#)
  - [[ (SCArray-utils), [9](#)
  - [[ , SC\_GDSArray, ANY, ANY-method (SCArray-utils), [9](#)
  - [[ , SC\_GDSArray-method (SCArray-utils), [9](#)
  - %% (SCArray-utils), [9](#)
  - %% , ANY, SC\_GDSMatrix-method (SCArray-utils), [9](#)
  - %% , SC\_GDSMatrix, ANY-method (SCArray-utils), [9](#)
- add.gdsn, [11](#), [13](#), [15](#)
- BiocParallelParam, [8](#), [16](#)
- BiocSingularParam, [22](#)
- blockApply, [20](#)
- blockReduce, [20](#)
- bpnworkers, [16](#)
- col\_nzero (row\_nzero), [3](#)
- col\_nzero, DelayedMatrix-method (row\_nzero), [3](#)
- col\_nzero, Matrix-method (row\_nzero), [3](#)
- col\_nzero, matrix-method (row\_nzero), [3](#)
- col\_nzero, SC\_GDSMatrix-method (row\_nzero), [3](#)
- colAnyNAs (SCArray-stats), [5](#)
- colAnyNAs, SC\_GDSMatrix-method (SCArray-stats), [5](#)
- colAutoGrid, [20](#)
- colAvgPerRowSet (SCArray-stats), [5](#)
- colAvgPerRowSet, SC\_GDSMatrix-method (SCArray-stats), [5](#)
- colCollapse (SCArray-stats), [5](#)
- colCollapse, SC\_GDSMatrix-method (SCArray-stats), [5](#)



- colDiffs (SCArray-stats), 5
- colDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- colLogSumExps (SCArray-stats), 5
- colLogSumExps, SC\_GDSMatrix-method (SCArray-stats), 5
- colMaxs (SCArray-stats), 5
- colMaxs, SC\_GDSMatrix-method (SCArray-stats), 5
- colMeans (SCArray-stats), 5
- colMeans, SC\_GDSMatrix-method (SCArray-stats), 5
- colMeans2 (SCArray-stats), 5
- colMeans2, SC\_GDSMatrix-method (SCArray-stats), 5
- colMins (SCArray-stats), 5
- colMins, SC\_GDSMatrix-method (SCArray-stats), 5
- colProds (SCArray-stats), 5
- colProds, SC\_GDSMatrix-method (SCArray-stats), 5
- colRanges (SCArray-stats), 5
- colRanges, SC\_GDSMatrix-method (SCArray-stats), 5
- colSdDiffs (SCArray-stats), 5
- colSdDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- colSds (SCArray-stats), 5
- colSds, SC\_GDSMatrix-method (SCArray-stats), 5
- colsum (SCArray-stats), 5
- colsum, SC\_GDSMatrix-method (SCArray-stats), 5
- colSums (SCArray-stats), 5
- colSums, SC\_GDSMatrix-method (SCArray-stats), 5
- colSums2 (SCArray-stats), 5
- colSums2, SC\_GDSMatrix-method (SCArray-stats), 5
- colVarDiffs (SCArray-stats), 5
- colVarDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- colVars (SCArray-stats), 5
- colVars, SC\_GDSMatrix-method (SCArray-stats), 5
- colWeightedMeans (SCArray-stats), 5
- colWeightedMeans, SC\_GDSMatrix-method (SCArray-stats), 5
- colWeightedSds (SCArray-stats), 5
- colWeightedSds, SC\_GDSMatrix-method (SCArray-stats), 5
- colWeightedVars (SCArray-stats), 5
- colWeightedVars, SC\_GDSMatrix-method (SCArray-stats), 5
- crossprod (SCArray-utils), 9
- crossprod, ANY, SC\_GDSMatrix-method (SCArray-utils), 9
- crossprod, SC\_GDSMatrix, ANY-method (SCArray-utils), 9
- crossprod, SC\_GDSMatrix, missing-method (SCArray-utils), 9
- DelayedArray, 4
- DelayedArray-utils, 8
- DelayedMatrix, 8
- dimnames<- (SCArray-utils), 9
- dimnames<- , SC\_GDSArray, ANY-method (SCArray-utils), 9
- gds.class, 18
- getAutoBPPARAM, 8, 16
- Math, 10
- Math (SCArray-utils), 9
- Math, SC\_GDSArray-method (SCArray-utils), 9
- matrix, 8
- names<- (SCArray-utils), 9
- names<- , SC\_GDSArray-method (SCArray-utils), 9
- Ops, 10
- Ops (SCArray-utils), 9
- Ops, SC\_GDSArray-method (SCArray-utils), 9
- path, 13
- pmax2 (SCArray-utils), 9
- pmax2, SC\_GDSArray, SC\_GDSArray-method (SCArray-utils), 9
- pmax2, SC\_GDSArray, vector-method (SCArray-utils), 9
- pmax2, vector, SC\_GDSArray-method (SCArray-utils), 9
- pmin2 (SCArray-utils), 9
- pmin2, SC\_GDSArray, SC\_GDSArray-method (SCArray-utils), 9
- pmin2, SC\_GDSArray, vector-method (SCArray-utils), 9
- pmin2, vector, SC\_GDSArray-method (SCArray-utils), 9
- prcomp, 21
- row\_nzero, 3

- row\_nnzero, DelayedMatrix-method (row\_nnzero), 3
- row\_nnzero, Matrix-method (row\_nnzero), 3
- row\_nnzero, matrix-method (row\_nnzero), 3
- row\_nnzero, SC\_GDSMatrix-method (row\_nnzero), 3
- rowAnyNAs (SCArray-stats), 5
- rowAnyNAs, SC\_GDSMatrix-method (SCArray-stats), 5
- rowAutoGrid, 20
- rowAvgPerColSet, 8
- rowAvgPerColSet (SCArray-stats), 5
- rowAvgPerColSet, SC\_GDSMatrix-method (SCArray-stats), 5
- rowCollapse (SCArray-stats), 5
- rowCollapse, SC\_GDSMatrix-method (SCArray-stats), 5
- rowDiffs (SCArray-stats), 5
- rowDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- rowLogSumExps (SCArray-stats), 5
- rowLogSumExps, SC\_GDSMatrix-method (SCArray-stats), 5
- rowMaxs (SCArray-stats), 5
- rowMaxs, SC\_GDSMatrix-method (SCArray-stats), 5
- rowMeans (SCArray-stats), 5
- rowMeans, SC\_GDSMatrix-method (SCArray-stats), 5
- rowMeans2 (SCArray-stats), 5
- rowMeans2, SC\_GDSMatrix-method (SCArray-stats), 5
- rowMins (SCArray-stats), 5
- rowMins, SC\_GDSMatrix-method (SCArray-stats), 5
- rowProds (SCArray-stats), 5
- rowProds, SC\_GDSMatrix-method (SCArray-stats), 5
- rowRanges (SCArray-stats), 5
- rowRanges, SC\_GDSMatrix-method (SCArray-stats), 5
- rowSdDiffs (SCArray-stats), 5
- rowSdDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- rowSds (SCArray-stats), 5
- rowSds, SC\_GDSMatrix-method (SCArray-stats), 5
- rowsum (SCArray-stats), 5
- rowsum, SC\_GDSMatrix-method (SCArray-stats), 5
- rowSums (SCArray-stats), 5
- rowSums, SC\_GDSMatrix-method (SCArray-stats), 5
- rowSums2 (SCArray-stats), 5
- rowSums2, SC\_GDSMatrix-method (SCArray-stats), 5
- rowVarDiffs (SCArray-stats), 5
- rowVarDiffs, SC\_GDSMatrix-method (SCArray-stats), 5
- rowVars (SCArray-stats), 5
- rowVars, SC\_GDSMatrix-method (SCArray-stats), 5
- rowWeightedMeans (SCArray-stats), 5
- rowWeightedMeans, SC\_GDSMatrix-method (SCArray-stats), 5
- rowWeightedSds (SCArray-stats), 5
- rowWeightedSds, SC\_GDSMatrix-method (SCArray-stats), 5
- rowWeightedVars (SCArray-stats), 5
- rowWeightedVars, SC\_GDSMatrix-method (SCArray-stats), 5
- runExactSVD, 21
- runIrlbaSVD, 21
- runPCA, 22
- runPCA (scRunPCA), 20
- runPCA, SC\_GDSMatrix-method (scRunPCA), 20
- runRandomSVD, 21
- runSVD, 21, 22
- SC\_GDSArray, 9
- SC\_GDSArray (SCArray-classes), 5
- SC\_GDSArray-class (SCArray-classes), 5
- SC\_GDSMatrix, 7, 9
- SC\_GDSMatrix (SCArray-classes), 5
- SC\_GDSMatrix-class (SCArray-classes), 5
- scale (SCArray-utils), 9
- scale, SC\_GDSMatrix-method (SCArray-utils), 9
- SCArray (SCArray-package), 2
- scArray, 4, 17, 18
- SCArray-classes, 5
- SCArray-package, 2
- SCArray-stats, 5, 10
- SCArray-utils, 9
- SCArrayFileClass (SCArray-classes), 5
- SCArrayFileClass-class (SCArray-classes), 5
- scClose, 11, 12
- scClose (scOpen), 17
- scColAutoGrid (scRowAutoGrid), 19
- scColMeanVar (SCArray-stats), 5
- scColMeanVar, Matrix-method (SCArray-stats), 5

scColMeanVar, matrix-method  
    (SCArray-stats), 5  
scColMeanVar, SC\_GDSMatrix-method  
    (SCArray-stats), 5  
scConvGDS, 10, 14, 15  
scExperiment, 4, 11, 17  
scGetFiles, 12  
scGetFiles, SC\_GDSArray-method  
    (scGetFiles), 12  
scGetFiles, SummarizedExperiment-method  
    (scGetFiles), 12  
scHDF2GDS, 11, 13, 15  
scMemory, 14  
scMemory, DelayedArray-method  
    (scMemory), 14  
scMemory, SummarizedExperiment-method  
    (scMemory), 14  
scMEX2GDS, 11, 14, 15  
scNumSplit, 16  
scObj, 17  
scOpen, 4, 11, 12, 17  
scReplaceNA, 18, 23  
scRowAutoGrid, 19  
scRowMeanVar (SCArray-stats), 5  
scRowMeanVar, Matrix-method  
    (SCArray-stats), 5  
scRowMeanVar, matrix-method  
    (SCArray-stats), 5  
scRowMeanVar, SC\_GDSMatrix-method  
    (SCArray-stats), 5  
scRunPCA, 20  
scSetBounds, 19, 22  
scSetMax, 19  
scSetMax (scSetBounds), 22  
scSetMin, 19  
scSetMin (scSetBounds), 22  
setAutoBlockSize, 20  
SingleCellExperiment, 12  
SummarizedExperiment, 12  
  
tcrossprod (SCArray-utils), 9  
tcrossprod, ANY, SC\_GDSMatrix-method  
    (SCArray-utils), 9  
tcrossprod, SC\_GDSMatrix, ANY-method  
    (SCArray-utils), 9  
tcrossprod, SC\_GDSMatrix, missing-method  
    (SCArray-utils), 9