

# Package ‘SCArray.sat’

May 26, 2026

**Type** Package

**Title** Large-scale single-cell RNA-seq data analysis using GDS files and Seurat

**Version** 1.13.0

**Date** 2026-04-21

**Depends** methods, SCArray (>= 1.13.1), SeuratObject (>= 5.0), Seurat (>= 5.0)

**Imports** S4Vectors, utils, stats, BiocGenerics, BiocParallel, gdsfmt, DelayedArray, BiocSingular, SummarizedExperiment, Matrix

**Suggests** future, RUnit, knitr, markdown, rmarkdown, BiocStyle

**Description** Extends the Seurat classes and functions to support Genomic Data Structure (GDS) files as a DelayedArray backend for data representation. It relies on the implementation of GDS-based DelayedMatrix in the SCArray package to represent single cell RNA-seq data. The common optimized algorithms leveraging GDS-based and single cell-specific DelayedMatrix (SC\_GDSMatrix) are implemented in the SCArray package. SCArray.sat introduces a new SCArrayAssay class (derived from the Seurat Assay), which wraps raw counts, normalized expressions and scaled data matrix based on GDS-specific DelayedMatrix. It is designed to integrate seamlessly with the Seurat package to provide common data analysis in the SeuratObject-based workflow. Compared with Seurat, SCArray.sat significantly reduces the memory usage without downsampling and can be applied to very large datasets.

**License** GPL-3

**VignetteBuilder** knitr

**BugReports** <https://github.com/AbbVie-ComputationalGenomics/SCArray/issues>

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

**git\_url** <https://git.bioconductor.org/packages/SCArray.sat>

**git\_branch** devel

**git\_last\_commit** 7d1739a

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-25

**Author** Xiuwen Zheng [aut, cre] (ORCID:  
<https://orcid.org/0000-0002-1390-0708>),  
 Seurat contributors [ctb] (for the classes and methods defined in  
 Seurat)

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

## Contents

SCArray.sat-package . . . . .	2
CreateAssayObject2 . . . . .	3
NormalizeData . . . . .	4
RunPCA . . . . .	5
ScaleData . . . . .	6
SCArrayAssay-class . . . . .	8
SCArrayAssay-methods . . . . .	9
scGetFiles . . . . .	10
scMemory . . . . .	11
scNewAssayGDS . . . . .	12
scNewSeuratGDS . . . . .	13
Seurat_g-class . . . . .	14

**Index** **15**

---

SCArray.sat-package	<i>Large-scale single-cell RNA-seq data analysis using GDS files and Seurat</i>
---------------------	---

---

## Description

The package extends the Seurat classes and functions to support GDS files as a DelayedArray backend for data representation. It introduces a new SCArrayAssay class (derived from the Seurat Assay), which wraps raw counts, normalized expressions and scaled data matrix based on DelayedMatrix. It is designed to integrate seamlessly with the SeuratObject and Seurat packages to provide common data analysis, with the optimized algorithms for GDS data files.

## Details

Package: SCArray.sat  
 Type: Package  
 License: GPL version 3

## Author(s)

Xiuwen Zheng

---

CreateAssayObject2     *Create an Assay object*

---

### Description

Create an `SCArrayAssay` (inherited from `Assay`) object from counts or prenormalized data.

### Usage

```
CreateAssayObject2(counts, data, min.cells=0, min.features=0,  
  key=NULL, check.matrix=FALSE, ...)
```

### Arguments

<code>counts</code>	Unnormalized raw counts (matrix, <code>dgCMatrix</code> or <code>DelayedMatrix</code> )
<code>data</code>	Prenormalized data (matrix, <code>dgCMatrix</code> or <code>DelayedMatrix</code> )
<code>min.cells</code>	if > 0, a lower cutoff for filtering cells
<code>min.features</code>	if > 0, a lower cutoff for filtering features
<code>check.matrix</code>	Check counts matrix for NA, NaN, Inf, and non-integer values
<code>key</code>	Key name for the assay
<code>...</code>	Arguments passed to <a href="#">as.sparse</a> when counts or data is matrix or <code>dgCMatrix</code>

### Details

Similar to `SeuratObject::CreateAssayObject()`, except allowing `DelayedMatrix` counts or data, and returning a `SCArrayAssay` object. `counts` and `data` should not be provided at the same time.

### Value

Return an instance of `SCArrayAssay`.

### Author(s)

Xiuwen Zheng

### See Also

[CreateAssayObject](#), [scGetFiles](#)

### Examples

```
fn <- system.file("extdata", "example.gds", package="SCArray")  
  
x <- scArray(fn, "counts")  
colnames(x) <- paste0("c", 1:ncol(x))  
rownames(x) <- paste0("g", 1:nrow(x))  
x  
  
a <- CreateAssayObject2(x)  
a
```

```
scGetFiles(x)
scGetFiles(a)

remove(x, a)
```

---

NormalizeData

*Normalize Count Data*

---

### Description

Normalizes the count data in the Seurat assay.

### Usage

```
# NormalizeData(object, ...)
## S3 method for class 'SC_GDSMatrix'
NormalizeData(object,
  normalization.method="LogNormalize", scale.factor=1e4, margin=1,
  verbose=TRUE, ...)
```

### Arguments

object	input R object (e.g., a SC_GDSMatrix object)
normalization.method	"LogNormalize", "CLR" or "RC"; see NormalizeData.Seurat for more details
scale.factor	the scale factor for cell-level normalization
margin	only applicable when normalization.method="CLR", normalize across features (margin=1) or cells (margin=2)
verbose	if TRUE, show information
...	additional arguments passed to specific methods

### Details

NormalizeData() does not store the normalized data in a GDS file, since the calculation is "delayed" until it is needed.

### Value

Returns a SC\_GDSMatrix matrix.

### Author(s)

Xiuwen Zheng

### See Also

[NormalizeData](#)

**Examples**

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

d <- scNewSeuratGDS(fn)
d
d <- NormalizeData(d)

remove(a, d)
```

RunPCA

*Run PCA***Description**

Performs PCA on a Seurat SCArrayAssay or a DelayedMatrix object.

**Usage**

```
# RunPCA(object, ...)
## S3 method for class 'SCArrayAssay'
RunPCA(object, assay=NULL, features=NULL, npcs=50,
        rev.pca=FALSE, weight.by.var=TRUE, verbose=TRUE, ndims.print=1:5,
        nfeatures.print=30, reduction.key="PC_", seed.use=42, ...)
## S3 method for class 'SC_GDSMatrix'
RunPCA(object, assay=NULL, npcs=50, rev.pca=FALSE,
        weight.by.var=TRUE, verbose=TRUE, ndims.print=1:5,
        nfeatures.print=30, reduction.key="PC_", seed.use=42, approx=TRUE,
        BPPARAM, ...)
```

**Arguments**

object	input R object (e.g., a SCArrayAssay object)
assay	NULL for using the active assay, or an assay name
features	if NULL, PCA will be run on the scaled data; otherwise, features to compute PCA on
npcs	# of top PCs to be calculated
rev.pca	By default (FALSE), perform PCA on the cell x gene matrix; otherwise, compute it on gene x cell matrix
weight.by.var	if TRUE, weight the cell embeddings (when rev.pca=FALSE) or the the gene loadings (when rev.pca=TRUE) by the variance of each PC
verbose	if TRUE, show information
ndims.print	which PCs to print genes for
nfeatures.print	# of genes to print for each PC
reduction.key	dimensional reduction key
seed.use	a random seed; or NULL for not setting a seed internally
approx	if TRUE, use IrlbaSVD; otherwise use ExactSVD
BPPARAM	NULL for non-parallel execution, or a BiocParallelParam object for parallelization; if it is missing, getAutoBPPARAM() will be used
...	additional arguments passed to specific methods

**Details**

RunPCA() computes the covariance matrix of genes (if # of genes <= # of cells) or the cell covariance matrix for the PCA calculation, which can reduce the times of accessing on-disk data.

**Value**

Return a data frame for reduction data (via CreateDimReducObject).

**Author(s)**

Xiuwen Zheng

**See Also**

[RunPCA](#), [CreateDimReducObject](#), [BiocParallelParam](#), [getAutoBPPARAM](#)

**Examples**

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

d <- scNewSeuratGDS(fn)

d <- NormalizeData(d)
d <- FindVariableFeatures(d, nfeatures=250)
d <- ScaleData(d)

d <- RunPCA(d, ndims.print=1:2)
DimPlot(d, reduction="pca")

remove(a, d)
```

---

ScaleData

*Scale and Center the Data*

---

**Description**

Scales and centers features or residuals in the dataset.

**Usage**

```
# ScaleData(object, ...)
## S3 method for class 'SC_GDSMatrix'
ScaleData(object, features=NULL, vars.to.regress=NULL,
  latent.data=NULL, split.by=NULL, model.use='linear', use.umi=FALSE,
  do.scale=TRUE, do.center=TRUE, scale.max=10, block.size=1000,
  min.cells.to.block=3000, verbose=TRUE, use_gds=TRUE, rm_tmpfile=TRUE, ...)
```

**Arguments**

<code>object</code>	input R object (e.g., a <code>SC_GDSMatrix</code> object)
<code>features</code>	if <code>NULL</code> , to use the variable features (found via <code>FindVariableFeatures()</code> ); or features names to scale/center
<code>vars.to.regress</code>	<code>NULL</code> or variable names to regress out
<code>latent.data</code>	<code>NULL</code> or a <code>data.frame</code> to regress out the covariates
<code>split.by</code>	variable name in the metadata, or a vector or factor defining grouping of cells
<code>model.use</code>	regression model: "linear" (default), "poisson" or "negbinom"
<code>use.umi</code>	only applicable when the covariates are given in <code>vars.to.regress</code> for regression; default is <code>FALSE</code> for linear regression, <code>TRUE</code> for negbinom and poisson models
<code>do.scale</code>	if <code>TRUE</code> , scale the data
<code>do.center</code>	if <code>TRUE</code> , center the data
<code>scale.max</code>	max value in the resulting scaled data; see <code>ScaleData.Seurat</code> for more details
<code>block.size</code>	not used
<code>min.cells.to.block</code>	not used
<code>verbose</code>	if <code>TRUE</code> , show information
<code>use_gds</code>	if <code>TRUE</code> , use <code>SC_GDSMatrix</code> for the scaled data; if <code>FALSE</code> , to use a dense in-memory scaled matrix; or a GDS file name for storing the resulting data matrix; see details
<code>rm_tmpfile</code>	if <code>TRUE</code> , remove any temporary GDS file after the calculation; the temporary file will be created when calculating the residuals
<code>...</code>	additional arguments passed to specific methods

**Details**

`ScaleData()` stores the scaled data in a GDS file when `use_gds=TRUE` or an output GDS file name is given via `use_gds`. When `vars.to.regress` and `split.by` are both `NULL`, an output GDS file is not needed, since the resulting `DelayedMatrix` can be represented as common operations on the count matrix. If `use_gds=TRUE`, an output file name "\_scale\_data.gds" will be used if it does not exist, or "\_scale\_data2.gds" (if not exists), "\_scale\_data3.gds" and so on. If `use_gds` is an output file name, the resulting data matrix will be saved to a GDS file. When `vars.to.regress` are given, a temporary GDS file (e.g., "\_temp\_scale\_data.gds", `use_gds` with a prefix "\_temp") will be created to store the residuals before scaling. This temporary file will be deleted after the calculation when `rm_tmpfile=TRUE`.

**Value**

Returns a `SC_GDSMatrix` matrix if `use_gds=TRUE` or `use_gds` is an output file name, otherwise returns an in-memory matrix.

**Author(s)**

Xiuwen Zheng

**See Also**

[ScaleData](#)

## Examples

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

d <- scNewSeuratGDS(fn)

d <- NormalizeData(d)
d <- FindVariableFeatures(d, nfeatures=50)
d <- ScaleData(d)

GetAssayData(d, layer="scale.data") # DelayedMatrix

# scale with split.by
ss <- rep(c(TRUE, FALSE), length.out=ncol(d))
d <- ScaleData(d, split.by=ss)

fn <- scGetFiles(d)
fn[2L] # the file name storing scaled data

remove(a, d)
unlink(fn[grepl("^_scale", fn)], force=TRUE)
```

---

SCArrayAssay-class      *GDS-specific Assay Class*

---

## Description

The SCArrayAssay class extends the Assay class of Seurat with the new slots counts2, data2 and scale.data2 replacing counts, data and scale.data.

## Slots

counts2 Unnormalized raw counts (dgCMatrx or SC\_GDSMatrix), replacing Assay@counts

data2 Normalized expression data (dgCMatrx or SC\_GDSMatrix), replacing Assay@data

scale.data2 Scaled expression data (NULL, matrix or SC\_GDSMatrix), replacing Assay@scale.data

## Author(s)

Xiuwen Zheng

## See Also

[Assay-class](#), [Seurat\\_g-class](#), [GetAssayData](#), [SetAssayData](#)

**Description**

Gets and sets data in the Seurat Assay object.

**Usage**

```
## S3 method for class 'SCArrayAssay'  
GetAssayData(object,  
  slot=c("data", "scale.data", "counts"), ...)  
## S3 method for class 'SCArrayAssay'  
SetAssayData(object, layer, new.data,  
  slot=c('data', 'scale.data', 'counts'), ...)  
  
## S3 method for class 'SCArrayAssay'  
subset(x, cells=NULL, features=NULL, ...)
```

**Arguments**

object, x	a SCArrayAssay object (inherited from SeuratObject::Assay)
layer	layer
new.data	a new data matrix (dgCMatrix or SC_GDSMatrix)
slot	data matrix in the Assay object, "data" is used by default
cells	names or indices for selected cells
features	names or indices for selected features
...	further arguments to be passed to or from other methods

**Value**

Return a data matrix or an instance of [SCArrayAssay](#).

**Author(s)**

Xiuwen Zheng

**See Also**

[Assay](#)

---

`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 'Assay'  
scGetFiles(object, ...)  
## S4 method for signature 'Assay5'  
scGetFiles(object, ...)  
## S4 method for signature 'SCArrayAssay'  
scGetFiles(object, ...)  
## S4 method for signature 'Seurat'  
scGetFiles(object, ...)
```

**Arguments**

<code>object</code>	input R object (e.g., a Seurat object)
<code>...</code>	additional arguments passed to specific methods

**Value**

Return a character vector storing file names.

**Author(s)**

Xiuwen Zheng

**See Also**

[scGetFiles](#)

**Examples**

```
fn <- system.file("extdata", "example.gds", package="SCArray")  
  
a <- scNewAssayGDS(fn)  
d <- Seurat::CreateSeuratObject(a)  
  
scGetFiles(a)  
scGetFiles(d)  
  
remove(a, d)
```

---

`scMemory`*Load Data to Memory*

---

## Description

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

## Usage

```
scMemory(x, ...)
## S4 method for signature 'SCArrayAssay'
scMemory(x, slot=NULL, ...)
## S4 method for signature 'Seurat'
scMemory(x, assay=NULL, slot=NULL, ...)
```

## Arguments

<code>x</code>	input R object (e.g., a Seurat object)
<code>assay</code>	NULL for using the active assay, or a list of assay names
<code>slot</code>	NULL for all "counts", "data" and "scale.data"; or a character vector including "counts", "data" or "scale.data"; see details
<code>...</code>	additional arguments passed to specific methods

## Details

If `slot=NULL`, return a Assay object instead of SCArrayAssay object, so it can downgrade a SCArrayAssay object to a Assay object.

## Value

Return an object (it maybe a different type from `class(x)`).

## Author(s)

Xiuwen Zheng

## See Also

[scMemory](#)

## Examples

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

d1 <- scNewSeuratGDS(fn)
is(GetAssay(d1))

d2 <- scMemory(d1)
is(GetAssay(d2))

remove(a, d1, d2)
```

---

`scNewAssayGDS`*Create Assay Object*

---

**Description**

Creates a new Seurat Assay object (`SCArrayAssay`) from a GDS file.

**Usage**

```
scNewAssayGDS(gdsfile, name="counts", key="rna_", row_data=TRUE, check=TRUE,
              verbose=TRUE)
```

**Arguments**

<code>gdsfile</code>	a file name for the GDS file, or a <code>SCArrayFileClass</code> object
<code>name</code>	characters for the name of data matrix in the GDS file; if <code>NA_character_</code> , to use the first assay
<code>key</code>	characters for the Assay key
<code>row_data</code>	if TRUE, add <code>rowData()</code> to the feature-level meta data of the Seurat Assay
<code>check</code>	if TRUE, check the feature names
<code>verbose</code>	if TRUE, show information

**Value**

Return an instance of [SCArrayAssay](#).

**Author(s)**

Xiuwen Zheng

**See Also**

[SCArrayAssay](#), [SCArrayFileClass](#), [scExperiment](#), [scNewSeuratGDS](#)

**Examples**

```
# raw count data in a GDS file
fn <- system.file("extdata", "example.gds", package="SCArray")

a <- scNewAssayGDS(fn)
a
class(a)

d <- Seurat::CreateSeuratObject(a)
d

rm(a, d)
```

---

scNewSeuratGDS                      *Create Seurat Object*

---

## Description

Creates a new Seurat object from a GDS file.

## Usage

```
scNewSeuratGDS(gdsfile, assay.name=NULL, key=c(counts="rna_"), row_data=TRUE,
               col_data=TRUE, check=TRUE, verbose=TRUE)
```

## Arguments

gdsfile	a file name for the GDS file, or a <code>SCArrayFileClass</code> object
assay.name	characters for the name of data matrix in the GDS file; if <code>NULL</code> , to use all of the assays
key	a character vector for an assay key map, where its names are the GDS node names
row_data	if <code>TRUE</code> , add <code>rowData()</code> to the feature-level meta data of the first Seurat Assay
col_data	if <code>TRUE</code> , add <code>colData()</code> to the cell-level meta data of the Seurat object
check	if <code>TRUE</code> , check the feature names
verbose	if <code>TRUE</code> , show information

## Details

"counts" must be in the input GDS file and it is used as the raw count data in the active Seurat assay. If "logcounts" exists, it is used as normalized data associated with "counts". If there are other data matrices in the GDS file, they will be added to the assay list.

## Value

Return an instance of [Seurat](#).

## Author(s)

Xiuwen Zheng

## See Also

[SCArrayAssay](#), [SCArrayFileClass](#), [scExperiment](#), [scNewAssayGDS](#)

## Examples

```
# raw count data in a GDS file
fn <- system.file("extdata", "example.gds", package="SCArray")

d <- scNewSeuratGDS(fn)
d
class(d)

rm(d)
```

---

Seurat\_g-class

*GDS-based Seurat Class*

---

**Description**

The Seurat\_g class inherits directly from "Seurat".

**Slots**

```
setClass("Seurat_g", contains="Seurat")
```

**Author(s)**

Xiuwen Zheng

**See Also**

[SCArrayAssay-class](#)

# Index

## \* GDS

- CreateAssayObject2, 3
- NormalizedData, 4
- RunPCA, 5
- ScaleData, 6
- SCArray.sat-package, 2
- SCArrayAssay-class, 8
- SCArrayAssay-methods, 9
- scGetFiles, 10
- scMemory, 11
- scNewAssayGDS, 12
- scNewSeuratGDS, 13
- Seurat\_g-class, 14

## \* SingleCell

- CreateAssayObject2, 3
- SCArray.sat-package, 2
- SCArrayAssay-class, 8
- SCArrayAssay-methods, 9
- scNewAssayGDS, 12
- scNewSeuratGDS, 13
- Seurat\_g-class, 14

## \* methods

- NormalizedData, 4
- RunPCA, 5
- ScaleData, 6
- scGetFiles, 10
- scMemory, 11

as.sparse, 3

Assay, 9

BiocParallelParam, 6

CreateAssayObject, 3

CreateAssayObject2, 3

CreateDimReducObject, 6

GetAssayData, 8

GetAssayData (SCArrayAssay-methods), 9

getAutoBPPARAM, 6

NormalizeData, 4, 4

RunPCA, 5, 6

ScaleData, 6, 7

SCArray.sat (SCArray.sat-package), 2

SCArray.sat-package, 2

SCArrayAssay, 3, 9, 12, 13

SCArrayAssay (SCArrayAssay-class), 8

SCArrayAssay-class, 8

SCArrayAssay-methods, 9

SCArrayFileClass, 12, 13

scExperiment, 12, 13

scGetFiles, 3, 10, 10

scGetFiles, Assay-method (scGetFiles), 10

scGetFiles, Assay5-method (scGetFiles), 10

scGetFiles, SCArrayAssay-method (scGetFiles), 10

scGetFiles, Seurat-method (scGetFiles), 10

scMemory, 11, 11

scMemory, SCArrayAssay-method (scMemory), 11

scMemory, Seurat-method (scMemory), 11

scMemory, Seurat\_g-method (scMemory), 11

scNewAssayGDS, 12, 13

scNewSeuratGDS, 12, 13

SetAssayData, 8

SetAssayData (SCArrayAssay-methods), 9

Seurat, 13

Seurat\_g (Seurat\_g-class), 14

Seurat\_g-class, 14

subset (SCArrayAssay-methods), 9