Package 'SeuratObject'

April 22, 2025

Title Data Structures for Single Cell Data

Version 5.1.0

Description Defines S4 classes for single-cell genomic data and associated information, such as dimensionality reduction embeddings, nearest-neighbor graphs, and spatially-resolved coordinates. Provides data access methods and R-native hooks to ensure the Seurat object is familiar to other R users. See Satija R, Farrell J, Gennert D, et al (2015) <doi:10.1038/nbt.3192>, Macosko E, Basu A, Satija R, et al (2015) <doi:10.1016/j.cell.2015.05.002>, and Stuart T, Butler A, et al (2019) <doi:10.1016/j.cell.2019.05.031>, Hao Y, Hao S, et al (2021) <doi:10.1016/j.cell.2021.04.048> and Hao Y, et al (2023) <doi:10.1101/2022.02.24.481684> for more details.

License MIT + file LICENSE

URL https://satijalab.github.io/seurat-object/, https://github.com/satijalab/seurat-object

BugReports https://github.com/satijalab/seurat-object/issues

Additional_repositories https://bnprks.r-universe.dev

Depends R (>= 4.1.0), sp (>= 1.5.0)

Imports future, future.apply, generics, grDevices, grid, lifecycle, Matrix (>= 1.6.4), methods, progressr, Rcpp (>= 1.0.5), rlang (>= 0.4.7), spam, stats,

tools,
utils

Suggests BPCells,
DelayedArray,
fs (>= 1.5.2),
ggplot2,
HDF5Array,
rmarkdown,
sf (>= 1.0.0),
testthat

LinkingTo Rcpp, RcppEigen

Config/Needs/website pkgdown

BuildManual true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Collate 'RcppExports.R'
'zzz.R'
'generics.R'
'keymixin.R'
'graph.R'
'default.R'
'assay.R'
'logmap.R'
'layers.R'
'assay5.R'
'centroids.R'
'command.R'
'compliance.R'
'data.R'
'jackstraw.R'
'dimreduc.R'
'segmentation.R'
'molecules.R'
'spatial.R'
'fov.R'
'neighbor.R'
'seurat.R'
'sparse.R'
'utils.R'

Contents

SeuratObject-package *SeuratObject: Data Structures for Single Cell Data*

Description

Defines S4 classes for single-cell genomic data and associated information, such as dimensionality reduction embeddings, nearest-neighbor graphs, and spatially-resolved coordinates. Provides data access methods and R-native hooks to ensure the Seurat object is familiar to other R users. See Satija R, Farrell J, Gennert D, et al (2015) doi:10.1038/nbt.3192, Macosko E, Basu A, Satija R, et al (2015) doi:10.1016/j.cell.2015.05.002, and Stuart T, Butler A, et al (2019) doi:10.1016/j.cell.2019.05.031, Hao Y, Hao S, et al (2021) doi:10.1016/j.cell.2021.04.048 and Hao Y, et al (2023) doi:10.1101/2022.02.24.481684 for more details.

Author(s)

Maintainer: Rahul Satija <seurat@nygenome.org> (ORCID)

Authors:

- Paul Hoffman <hoff0792@alumni.umn.edu> (ORCID)
- David Collins <dcollins@nygenome.org> (ORCID)
- Yuhan Hao <yhao@nygenome.org> (ORCID)
- Austin Hartman <ahartman@nygenome.org> (ORCID)
- Gesmira Molla <gmolla@nygenome.org> (ORCID)
- Andrew Butler <abutler@nygenome.org> (ORCID)
- Tim Stuart <tstuart@nygenome.org> (ORCID)

Other contributors:

- Madeline Kowalski <mkowalski@nygenome.org> (ORCID) [contributor]
- Saket Choudhary <schoudhary@nygenome.org> (ORCID) [contributor]
- Skylar Li <sli@nygenome.org> [contributor]
- Longda Jiang <ljiang@nygenome.org> (ORCID) [contributor]
- Jeff Farrell <jfarrell@g.harvard.edu> [contributor]
- Shiwei Zheng <szheng@nygenome.org> (ORCID) [contributor]
- Christoph Hafemeister <chafemeister@nygenome.org> (ORCID) [contributor]
- Patrick Roelli <proelli@nygenome.org> [contributor]

See Also

Useful links:

- https://satijalab.github.io/seurat-object/
- https://github.com/satijalab/seurat-object
- Report bugs at https://github.com/satijalab/seurat-object/issues

.DollarNames.SeuratCommand *Dollar-sign Autocompletion*

Description

Autocompletion for `$` access on a `SeuratCommand` object

Usage

```
## S3 method for class 'SeuratCommand'
.DollarNames(x, pattern = "")
```

Arguments

`x`	A `SeuratCommand` object
`pattern`	A regular expression. Only matching names are returned.

Value

The parameter name matches for `pattern`

See Also

Command log object and interaction methods `$.SeuratCommand()`, `LogSeuratCommand()`, `SeuratCommand-class`, `[.SeuratCommand()`, `as.list.SeuratCommand()`

[.Assay *Layer Data*

Description

Get and set layer data

Usage

```
## S3 method for class 'Assay'
x[i = missing_arg(), j = missing_arg(), ...]

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

Arguments

x	An Assay object
i	Name of layer data to get or set
j	Ignored
...	Arguments passed to LayerData
value	A matrix-like object to add as a new layer

Value

[: The layer data for layer i

[<-: x with layer data value saved as i

See Also

LayerData

v3 Assay object, validity, and interaction methods: $.Assay(), Assay-class, Assay-validity, CreateAssayObject(), [[.Assay(), dim.Assay(), dimnames.Assay(), merge.Assay(), split.Assay(), subset.Assay()

Examples

```
rna <- pbmc_small[["RNA"]]

# Get a vector of layer names in this assay
rna[]

# Fetch layer data
rna["data"][1:10, 1:4]

# Set layer data
rna["data"] <- rna["counts"]
rna["data"][1:10, 1:4]
```

[.Assay5 *Layer Data*

Description

Get and set layer data

Usage

```
## S3 method for class 'Assay5'
x[i = missing_arg(), j = missing_arg(), ...]

## S4 replacement method for signature 'Assay5,character,ANY,ANY'
x[i, j, ...] <- value
```

Arguments

x	An Assay5 object
i	Name of layer data to get or set
j	Ignored
...	Arguments passed to LayerData
value	A matrix-like object to add as a new layer

Value

[: The layer data for layer i

[<-: x with layer data value saved as i

See Also

LayerData

v5 Assay object, validity, and interaction methods: $.Assay5(), Assay5-class, Assay5-validity, [[.Assay5(), dim.Assay5(), dimnames.Assay5(), merge.Assay5(), split.Assay5(), subset.Assay5()

[.DimReduc *Get Feature Loadings*

Description

Pull feature loadings from a dimensional reduction

Usage

```
## S3 method for class 'DimReduc'
x[i, j, drop = FALSE, ...]
```

Arguments

x	A DimReduc object
i	Feature identifiers or indices
j	Dimension identifiers or indices
drop	Coerce the result to the lowest possible dimension; see drop for further details
...	Arguments passed to other methods

Details

[does not distinguish between projected and unprojected feature loadings; to select whether projected or unprojected loadings should be pulled, please use Loadings

Value

Feature loadings for features `i` and dimensions `j`

See Also

`Loadings`

Dimensional reduction object, validity, and interaction methods `CreateDimReducObject()`, `DimReduc-class`, `DimReduc-validity`, `[[.DimReduc()`, `dim.DimReduc()`, `merge.DimReduc()`, `print.DimReduc()`, `subset.DimReduc()`

Examples

```
pca <- pbmc_small[["pca"]]
pca[1:10, 1:5]
```

`[.SeuratCommand` *Command Log Data Access*

Description

Access data from a `SeuratCommand` object

Usage

```
## S3 method for class 'SeuratCommand'
x[i, ...]
```

Arguments

`x`	A `SeuratCommand` object
`i`	The name of a command log slot
`...`	Ignored

Value

`[`: Slot `i` from `x`

See Also

Command log object and interaction methods `$.SeuratCommand()`, `.DollarNames.SeuratCommand()`, `LogSeuratCommand()`, `SeuratCommand-class`, `as.list.SeuratCommand()`

Examples

```
cmd <- pbmc_small[["NormalizeData.RNA"]]
cmd["call.string"]
```

`[[.Assay` *Feature-Level Meta Data*

Description

Get and set feature-level meta data

Usage

```
## S3 method for class 'Assay'
x[[i, ..., drop = FALSE]]

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

## S3 method for class 'Assay'
head(x, n = 10L, ...)

## S3 method for class 'Assay'
tail(x, n = 10L, ...)

## S4 replacement method for signature 'Assay,missing,missing,data.frame'
x[[i, j, ...]] <- value
```

Arguments

`x`	An `Assay` object
`i`	Name of feature-level meta data to fetch or add
`...`	Ignored
`drop`	See `drop`
`j`	Ignored
`value`	Feature-level meta data to add
`n`	Number of meta data rows to show

Value

`[[`: The feature-level meta data for `i`

`[[<-`: `x` with `value` added as `i` in feature-level meta data

`head`: The first `n` rows of feature-level meta data

`tail`: the last `n` rows of feature-level meta data

See Also

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-class`, `Assay-validity`, `CreateAssayObject()`, `[.Assay()`, `dim.Assay()`, `dimnames.Assay()`, `merge.Assay()`, `split.Assay()`, `subset.Assay()`

Examples

```
rna <- pbmc_small[["RNA"]]

# Pull the entire feature-level meta data data frame
head(rna[[]])

# Pull a specific column of feature-level meta data
head(rna[["vst.mean"]])
head(rna[["vst.mean", drop = TRUE]])

# `head` and `tail` can be used to quickly view feature-level meta data
head(rna)

tail(rna)
```

[[.Assay5 *Feature-Level Meta Data*

Description

Get and set feature-level meta data

Usage

```
## S3 method for class 'Assay5'
x[[i, j, ..., drop = FALSE]]

## S4 replacement method for signature 'Assay5,ANY,ANY,ANY'
x[[i, j, ...]] <- value

## S3 method for class 'Assay5'
head(x, n = 10L, ...)

## S3 method for class 'Assay5'
tail(x, n = 10L, ...)
```

Arguments

x	An Assay5 object
i	Name of feature-level meta data to fetch or add
j	Ignored
...	Ignored
drop	See drop
value	Feature-level meta data to add
n	Number of meta data rows to show

Value

[[: The feature-level meta data for i

[[<-: x with value added as i in feature-level meta data

head: The first n rows of feature-level meta data

tail: the last n rows of feature-level meta data

See Also

v5 Assay object, validity, and interaction methods: $.Assay5(), Assay5-class, Assay5-validity, [.Assay5(), dim.Assay5(), dimnames.Assay5(), merge.Assay5(), split.Assay5(), subset.Assay5()

[[.DimReduc *Get Cell Embeddings*

Description

Pull cell embeddings from a dimensional reduction

Usage

```
## S3 method for class 'DimReduc'
x[[i, j, drop = FALSE, ...]]
```

Arguments

x	A DimReduc object
i	Cell names or indices
j	Dimension identifiers or indices
drop	Coerce the result to the lowest possible dimension; see drop for further details
...	Arguments passed to other methods

Value

Cell embeddings for cells i and dimensions j

See Also

Embeddings

Dimensional reduction object, validity, and interaction methods CreateDimReducObject(), DimReduc-class, DimReduc-validity, [.DimReduc(), dim.DimReduc(), merge.DimReduc(), print.DimReduc(), subset.DimReduc()

Examples

```
pca <- pbmc_small[["pca"]]
pca[[1:10, 1:5]]
```

`[[.Seurat` *Subobjects and Cell-Level Meta Data*

Description

The `[[` operator pulls either subobjects (eg. v3 or v5 assays, dimensional reduction information, or nearest-neighbor graphs) or cell-level meta data from a `Seurat` object

Usage

```
## S3 method for class 'Seurat'
x[[i = missing_arg(), ..., drop = FALSE, na.rm = FALSE]]

## S3 method for class 'Seurat'
head(x, n = 10L, ...)

## S3 method for class 'Seurat'
tail(x, n = 10L, ...)
```

Arguments

`x`	A `Seurat` object
`i`	Name of cell-level meta data
`...`	Ignored
`drop`	See `drop`
`na.rm`	Remove cells where meta data is all `NA`
`n`	Number of meta data rows to show

Value

Varies based on the value of `i`:

- If `i` is missing, a data frame with cell-level meta data
- If `i` is a vector with cell-level meta data names, a data frame (or vector of `drop = TRUE`) with cell-level meta data requested
- If `i` is a one-length character with the name of a subobject, the subobject specified by `i`

`head`: The first `n` rows of cell-level metadata

`tail`: The last `n` rows of cell-level metadata

See Also

See here for adding meta data with `[[<-`, here for adding subobjects with `[[<-`, and here for removing subobjects and cell-level meta data with `[[<-`

Seurat object, validity, and interaction methods `$.Seurat()`, `Seurat-class`, `Seurat-validity`, `[[<-,Seurat`, `[[<-,Seurat,NULL`, `dim.Seurat()`, `dimnames.Seurat()`, `merge.Seurat()`, `names.Seurat()`, `subset.Seurat()`

Examples

```
# Get the cell-level metadata data frame
head(pbmc_small[[]])

# Pull specific metadata information
head(pbmc_small[[c("letter.idents", "groups")]])
head(pbmc_small[["groups", drop = TRUE]])

# Get a sub-object (eg. an `Assay` or `DimReduc`)
pbmc_small[["RNA"]]
pbmc_small[["pca"]]

# Get the first 10 rows of cell-level metadata
head(pbmc_small)

# Get the last 10 rows of cell-level metadata
tail(pbmc_small)
```

[[<-,Seurat *Add Subobjects*

Description

Add subobjects containing expression, dimensional reduction, or other containerized data to a Seurat object. Subobjects can be accessed with [[and manipulated directly within the Seurat object or used independently

Usage

```
## S4 replacement method for signature 'Seurat,character,missing,Assay'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,Assay5'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,DimReduc'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,Graph'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,Neighbor'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,SeuratCommand'
x[[i, j, ...]] <- value
```

```
## S4 replacement method for signature 'Seurat,character,missing,SpatialImage'
x[[i, j, ...]] <- value
```

Arguments

x	A Seurat object
i	Name to add subobject as
j	Ignored
...	Ignored
value	A valid subobject (eg. a v3 or v5 assay, or a dimensional reduction)

Value

x with value added as i

See Also

See here for pulling subobjects using [[, here for adding metadata with [[<-, and here for removing subobjects and cell-level meta data with [[<-

Seurat object, validity, and interaction methods $.Seurat(), Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat,NULL, dim.Seurat(), dimnames.Seurat(), merge.Seurat(), names.Seurat(), subset.Seurat()

[[<-,Seurat,NULL *Remove Subobjects and Cell-Level Meta Data*

Description

Remove Subobjects and Cell-Level Meta Data

Usage

```
## S4 replacement method for signature 'Seurat,character,missing,NULL'
x[[i, j, ...]] <- value
```

Arguments

x	A Seurat object
i	Name(s) of subobject(s) or cell-level meta data to remove
j	Ignored
...	Ignored
value	NULL

Value

x with i removed from the object

See Also

See here for pulling subobjects using [[, here for adding metadata with [[<-, and here for adding subobjects with [[<-

Seurat object, validity, and interaction methods $.Seurat(), Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat, dim.Seurat(), dimnames.Seurat(), merge.Seurat(), names.Seurat(), subset.Seurat()

`$.Assay` *Layer Data*

Description

Get and set layer data

Usage

```
## S3 method for class 'Assay'
x$i

## S3 replacement method for class 'Assay'
x$i <- value
```

Arguments

x	An Assay object
i	Name of layer data to get or set
value	A matrix-like object to add as a new layer

Value

$: Layer data for layer i

$<-: x with layer data value saved as i

See Also

v3 Assay object, validity, and interaction methods: Assay-class, Assay-validity, CreateAssayObject(), [.Assay(), [[.Assay(), dim.Assay(), dimnames.Assay(), merge.Assay(), split.Assay(), subset.Assay()

Examples

```
rna <- pbmc_small[["RNA"]]

# Fetch a layer with `$`
rna$data[1:10, 1:4]

# Add a layer with `$`
```

```
rna$data <- rna$counts
rna$data[1:10, 1:4]
```

`$.Assay5` *Layer Data*

Description

Get and set layer data

Usage

```
## S3 method for class 'Assay5'
x$i

## S3 replacement method for class 'Assay5'
x$i <- value
```

Arguments

`x`	An `Assay5` object
`i`	Name of layer data to get or set
`value`	A matrix-like object to add as a new layer

Value

`$`: Layer data for layer `i`

`$<-`: `x` with layer data `value` saved as `i`

See Also

v5 Assay object, validity, and interaction methods: `Assay5-class`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `split.Assay5()`, `subset.Assay5()`

`$.Seurat` *Cell-Level Meta Data*

Description

Get and set cell-level meta data

Usage

```
## S3 method for class 'Seurat'
x$i

## S3 replacement method for class 'Seurat'
x$i, ... <- value

## S4 replacement method for signature 'Seurat,character,missing,data.frame'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,missing,missing,data.frame'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,factor'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,list'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,missing,missing,list'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'Seurat,character,missing,vector'
x[[i, j, ...]] <- value
```

Arguments

`x`	A `Seurat` object
`i`	Name of cell-level meta data
`...`	Ignored
`value`	A vector to add as cell-level meta data
`j`	Ignored

Value

$: Metadata column `i` for object `x`; **note**: unlike `[[`, `$` drops the shape of the metadata to return a vector instead of a data frame

`$<-`: `x` with metadata `value` saved as `i`

See Also

Seurat object, validity, and interaction methods Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat, [[<-,Seurat,NULL, dim.Seurat(), dimnames.Seurat(), merge.Seurat(), names.Seurat(), subset.Seurat()

Examples

```
# Get metadata using `$'
head(pbmc_small$groups)

# Add metadata using the `$' operator
set.seed(42)
pbmc_small$value <- sample(1:3, size = ncol(pbmc_small), replace = TRUE)
head(pbmc_small[["value"]])
```

$.SeuratCommand *Command Log Parameter Access*

Description

Pull parameter values from a SeuratCommand object

Usage

```
## S3 method for class 'SeuratCommand'
x$i
```

Arguments

x	A SeuratCommand object
i	A parameter name

Value

The value for parameter i

See Also

Command log object and interaction methods .DollarNames.SeuratCommand(), LogSeuratCommand(), SeuratCommand-class, [.SeuratCommand(), as.list.SeuratCommand()

Examples

```
cmd <- pbmc_small[["NormalizeData.RNA"]]
cmd$normalization.method
```

AddMetaData *Add in metadata associated with either cells or features.*

Description

Adds additional data to the object. Can be any piece of information associated with a cell (examples include read depth, alignment rate, experimental batch, or subpopulation identity) or feature (ENSG name, variance). To add cell level information, add to the Seurat object. If adding feature-level metadata, add to the Assay object (e.g. `object[["RNA"]]`)

Usage

```
AddMetaData(object, metadata, col.name = NULL)

## S3 method for class 'Assay'
AddMetaData(object, metadata, col.name = NULL)

## S3 method for class 'Assay5'
AddMetaData(object, metadata, col.name = NULL)

## S3 method for class 'Seurat'
AddMetaData(object, metadata, col.name = NULL)
```

Arguments

`object`	An object
`metadata`	A vector, list, or data.frame with metadata to add
`col.name`	A name for meta data if not a named list or data.frame

Value

`object` with metadata added

Examples

```
cluster_letters <- LETTERS[Idents(object = pbmc_small)]
names(cluster_letters) <- colnames(x = pbmc_small)
pbmc_small <- AddMetaData(
  object = pbmc_small,
  metadata = cluster_letters,
  col.name = 'letter.idents'
)
head(x = pbmc_small[[]])
```

as.Centroids *Convert Segmentation Layers*

Description

Convert Segmentation Layers

Usage

```
as.Centroids(x, nsides = NULL, radius = NULL, theta = NULL, ...)

as.Segmentation(x, ...)

## S3 method for class 'Segmentation'
as.Centroids(x, nsides = NULL, radius = NULL, theta = NULL, ...)

## S3 method for class 'Centroids'
as.Segmentation(x, ...)
```

Arguments

x	An object
nsides	The number of sides to represent cells/spots; pass Inf to plot as circles
radius	Radius of shapes when plotting
theta	Angle to adjust shapes when plotting
...	Arguments passed to other methods

Value

as.Centroids: A Centroids object

as.Segmentation: A Segmentation object

as.Graph *Coerce to a* Graph *Object*

Description

Convert a matrix (or Matrix) to a Graph object

Usage

```
as.Graph(x, ...)

## S3 method for class 'Matrix'
as.Graph(x, ...)

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

## S3 method for class 'Neighbor'
as.Graph(x, weighted = TRUE, ...)
```

Arguments

x	The matrix to convert
...	Ignored
weighted	If TRUE, fill entries in Graph matrix with value from the nn.dist slot of the Neighbor object

Value

A Graph object

See Also

Other graph: Graph-class

Examples

```
# converting sparse matrix
mat <- Matrix::rsparsematrix(nrow = 10, ncol = 10, density = 0.1)
rownames(x = mat) <- paste0("feature_", 1:10)
colnames(x = mat) <- paste0("cell_", 1:10)
g <- as.Graph(x = mat)

# converting dense matrix
mat <- matrix(data = 1:16, nrow = 4)
rownames(x = mat) <- paste0("feature_", 1:4)
colnames(x = mat) <- paste0("cell_", 1:4)
g <- as.Graph(x = mat)
```

---

as.list.SeuratCommand *Coerce a* SeuratCommand *to a list*

Description

Coerce a SeuratCommand to a list

Usage

```
## S3 method for class 'SeuratCommand'
as.list(x, complete = FALSE, ...)
```

Arguments

x	A SeuratCommand object
complete	Include slots besides just parameters (eg. call string, name, timestamp)
...	Ignored

Value

A list with the parameters and, if complete = TRUE, the call string, name, and timestamp

See Also

Command log object and interaction methods $.SeuratCommand(), .DollarNames.SeuratCommand(), LogSeuratCommand(), SeuratCommand-class, [.SeuratCommand()

Examples

```
cmd <- pbmc_small[["NormalizeData.RNA"]]
as.list(cmd)
as.list(cmd, complete = TRUE)
```

---

as.matrix.LogMap *Coerce Logical Maps to Matrices*

Description

Coerce a logical map to a matrix; this removes all logical map class capabilities from the object and returns a base-R matrix object

Usage

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

Arguments

x	A LogMap object
...	Ignored

Value

A base-R matrix created from x

See Also

Logical map objects, validity, and interaction methods: LogMap, LogMap-validity, droplevels.LogMap(), intersect.LogMap(), labels.LogMap()

Examples

```
map <- LogMap(letters[1:10])
map[['obs']] <- c(1, 3, 7)
mat <- as.matrix(map)
mat
class(mat)
```

---

as.Neighbor *Coerce to a* Neighbor *Object*

Description

Convert objects to Neighbor objects

Usage

```
as.Neighbor(x, ...)

## S3 method for class 'Graph'
as.Neighbor(x, ...)
```

Arguments

x	An object to convert to Neighbor
...	Arguments passed to other methods

Value

A Neighbor object

as.Seurat *Coerce to a* Seurat *Object*

Description

Convert objects to Seurat objects

Usage

```
as.Seurat(x, ...)
```

Arguments

x	An object to convert to class Seurat
...	Arguments passed to other methods

Value

A Seurat object generated from x

as.sparse *Cast to Sparse*

Description

Convert dense objects to sparse representations

Usage

```
as.sparse(x, ...)

## S3 method for class 'data.frame'
as.sparse(x, row.names = NULL, ...)

## S3 method for class 'Matrix'
as.sparse(x, ...)

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

## S3 method for class 'ngCMatrix'
as.sparse(x, ...)
```

Arguments

`x`	An object
`...`	Arguments passed to other methods
`row.names`	`NULL` or a character vector giving the row names for the data; missing values are not allowed

Value

A sparse representation of the input data

`Assay-class` *The Assay Class*

Description

The Assay object is the basic unit of Seurat; each Assay stores raw, normalized, and scaled data as well as cluster information, variable features, and any other assay-specific metadata. Assays should contain single cell expression data such as RNA-seq, protein, or imputed expression data.

Slots

`counts` Unnormalized data such as raw counts or TPMs

`data` Normalized expression data

`scale.data` Scaled expression data

`assay.orig` Original assay that this assay is based off of. Used to track assay provenance

`var.features` Vector of features exhibiting high variance across single cells

`meta.features` Feature-level metadata

`misc` A named list of unstructured miscellaneous data

`key` A one-length character vector with the object's key; keys must be one or more alphanumeric characters followed by an underscore "_" (regex pattern "`^[a-zA-Z][a-zA-Z0-9]*_$`")

See Also

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-validity`, `CreateAssayObject()`, `[.Assay()`, `[[.Assay()`, `dim.Assay()`, `dimnames.Assay()`, `merge.Assay()`, `split.Assay()`, `subset.Assay()`

Assay-validity *V3 Assay Validity*

Description

Validation of `Assay` objects is handled by `validObject`

`data` Validation

blah

`counts` Validation

blah

`scale.data` Validation

blah

Feature-Level Meta Data Validation

blah

Variable Feature Validation

blah

Key Validation

Keys must be a one-length character vector; a key must be composed of one of the following:

- An empty string (eg. "`''`") where `nchar() == 0`
- An string composed of one or more alphanumeric values (both lower- and upper-case) that ends with an underscore ("_"); the first character must be a letter

Keys that are not empty strings are validated with the regex "`^[a-zA-Z][a-zA-Z0-9]*_$`"

See Also

`validObject`

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-class`, `CreateAssayObject()`, `[.Assay()`, `[[.Assay()`, `dim.Assay()`, `dimnames.Assay()`, `merge.Assay()`, `split.Assay()`, `subset.Assay()`

Examples

```
rna <- pbmc_small[["RNA"]]
validObject(rna)
```

`Assay5-class` *The v5* `Assay` *Object*

Description

The v5 `Assay` is the typical `Assay` class used in **Seurat** v5; ...

Slots

`layers` A named list containing expression matrices; each matrix should be a two-dimensional object containing some subset of cells and features defined in the `cells` and `features` slots. Cell and feature membership is recorded in the `cells` and `features` slots, respectively

`cells` A logical mapping of cell names and layer membership; this map contains all the possible cells that this assay can contain. New layers must have some subset of cells present in this map

`features` A logical mapping of feature names and layer membership; this map contains all the possible features that this assay can contain. New layers must have some subset of features present in this map

`default` A one-length integer with the end index of the default layer; the default layer be all layers up to and including the layer at index `default`

`assay.orig` Original assay that this assay is based off of; used to track assay provenance

`meta.data` A data frame with feature-level meta data; should have the same number of rows as `features`

`misc` A named list of unstructured miscellaneous data

`key` A one-length character vector with the object's key; keys must be one or more alphanumeric characters followed by an underscore "_" (regex pattern "`^[a-zA-Z][a-zA-Z0-9]*_$`")

See Also

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `split.Assay5()`, `subset.Assay5()`

`Assay5-validity` *V5 Assay Validity*

Description

Validation of `Assay5` objects is handled by `validObject`

Layer Validation

blah

Key Validation

Keys must be a one-length character vector; a key must be composed of one of the following:

- An empty string (eg. "''") where `nchar() == 0`
- An string composed of one or more alphanumeric values (both lower- and upper-case) that ends with an underscore ("_"); the first character must be a letter

Keys that are not empty strings are validated with the regex "`^[a-zA-Z][a-zA-Z0-9]*_$`"

See Also

`validObject`

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-class`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `split.Assay5()`, `subset.Assay5()`

AssayData *Get and Set Assay Data*

Description

General accessor and setter functions for `Assay` objects. `GetAssayData` can be used to pull information from any of the expression matrices (eg. "counts", "data", or "scale.data"). `SetAssayData` can be used to replace one of these expression matrices

Usage

```
GetAssayData(object, ...)

SetAssayData(object, layer, new.data, slot = deprecated(), ...)

## S3 method for class 'Seurat'
GetAssayData(object, assay = NULL, layer = NULL, slot = deprecated(), ...)

## S3 method for class 'Seurat'
SetAssayData(
  object,
  layer = "data",
  new.data,
  slot = deprecated(),
  assay = NULL,
  ...
)

## S3 method for class 'Assay'
GetAssayData(
  object,
  layer = c("data", "scale.data", "counts"),
```

```
  slot = deprecated(),
  ...
)

## S3 method for class 'Assay'
SetAssayData(
  object,
  layer = c("data", "scale.data", "counts"),
  new.data,
  slot = deprecated(),
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
layer	Name of layer to get or set
new.data	New assay data to add
slot	**[Deprecated]** Specific assay data to get or set
assay	Specific assay to get data from or set data for; defaults to the default assay

Value

GetAssayData: returns the specified assay data

SetAssayData: object with the assay data set

Lifecycle

[Superseded]

GetAssayData and SetAssayData have been superseded. To fetch expression matrices, use LayerData; to set expression data, use LayerData<-

Examples

```
# Get assay data from the default assay in a Seurat object
GetAssayData(object = pbmc_small, layer = "data")[1:5,1:5]

# Set an Assay layer through the Seurat object
count.data <- GetAssayData(object = pbmc_small[["RNA"]], layer = "counts")
count.data <- as.matrix(x = count.data + 1)
new.seurat.object <- SetAssayData(
    object = pbmc_small,
    layer = "counts",
    new.data = count.data,
    assay = "RNA"
)
```

```
# Get the data directly from an Assay object
GetAssayData(pbmc_small[["RNA"]], layer = "data")[1:5,1:5]

# Set an Assay layer directly
count.data <- GetAssayData(pbmc_small[["RNA"]], layer = "counts")
count.data <- as.matrix(x = count.data + 1)
new.assay <- SetAssayData(pbmc_small[["RNA"]], layer = "counts", new.data = count.data)
```

Assays *Query Specific Object Types*

Description

List the names of Assay, DimReduc, Graph, Neighbor objects

Usage

```
Assays(object, ...)

Graphs(object, slot = NULL)

Neighbors(object, slot = NULL)

Reductions(object, slot = NULL)

## S3 method for class 'Seurat'
Assays(object, slot = deprecated(), ...)
```

Arguments

object	A Seurat object
...	Ignored
slot	Name of component object to return

Value

If slot is NULL, the names of all component objects in this Seurat object. Otherwise, the specific object specified

Examples

```
Assays(pbmc_small)

Graphs(pbmc_small)

Reductions(object = pbmc_small)
```

`AttachDeps` *Attach Required Packages*

Description

Helper function to attach required packages. Detects if a package is already attached and if so, skips it. Should be called in `.onAttach`

Usage

```
AttachDeps(deps)
```

Arguments

`deps`	A character vector of packages to attach

Value

Invisibly returns `NULL`

Lifecycle

[Superseded]

`AttachDeps` has been superseded as of **SeuratObject** v5.0.0; as an alternative, list dependencies in the `Depends` section of `DESCRIPTION`

Examples

```
# Use in your .onAttach hook
if (FALSE) {
  .onAttach <- function(libname, pkgname) {
    AttachDeps(c("SeuratObject", "rlang"))
  }
}
```

`Boundaries` *Get, Set, and Query Segmentation Boundaries*

Description

Get, Set, and Query Segmentation Boundaries

Usage

```
Boundaries(object, ...)

DefaultBoundary(object)

DefaultBoundary(object, ...) <- value

Molecules(object, ...)

## S3 method for class 'FOV'
Boundaries(object, ...)

## S3 method for class 'FOV'
DefaultBoundary(object)

## S3 replacement method for class 'FOV'
DefaultBoundary(object, ...) <- value

## S3 method for class 'FOV'
Molecules(object, ...)
```

Arguments

object	An object
...	Arguments passed to other methods
value	The name of a segmentation boundary to set as default

Value

`Boundaries`: The names of all segmentation boundaries present within `object`

`DefaultBoundary`: The name of the default segmentation boundary

`DefaultBoundary<-`: `object` with the default segmentation boundary set to `value`

`Molecules`: The names of all molecule sets present within `object`

CastAssay *Cast Assay Layers*

Description

Cast layers in v5 assays to other classes

Usage

```
CastAssay(object, to, ...)

## S3 method for class 'Assay5'
CastAssay(object, to, layers = NA, verbose = TRUE, ...)
```

Arguments

`object`	An object
`to`	Either a class name or a function that takes a layer and returns the same layer as a new class
`...`	If `to` is a function, arguments passed to `to`
`layers`	A vector of layers to cast; defaults to all layers
`verbose`	Show progress updates

Value

`object` with the layers cast to class specified by `to`

`Cells` *Cell and Feature Names*

Description

Get the cell and feature names of an object

Usage

```
Cells(x, ...)

Features(x, ...)

## Default S3 method:
Cells(x, ...)

## S3 method for class 'Assay5'
Cells(x, layer = NULL, simplify = TRUE, ...)

## S3 method for class 'Assay5'
Features(x, layer = NULL, simplify = TRUE, ...)

## S3 method for class 'DimReduc'
Cells(x, ...)

## S3 method for class 'Neighbor'
Cells(x, ...)
```

Arguments

`x`	An object
`...`	Arguments passed to other methods
`layer`	Layer to pull cells/features for; defaults to default layer; if `NA`, returns all cells for the assay

simplify	Simplify the cell/feature names into a single vector; if FALSE, separates each cell/feature names by layer

Value

Cell: A vector of cell names

Features: A vector of feature names

See Also

dimnames.Assay(), dimnames.Assay5(), dimnames.Seurat()

Examples

```
Cells(x = pbmc_small)
```

CellsByIdentities *Get cell names grouped by identity class*

Description

Get cell names grouped by identity class

Usage

```
CellsByIdentities(object, idents = NULL, cells = NULL, return.null = FALSE)
```

Arguments

object	A Seurat object
idents	A vector of identity class levels to limit resulting list to; defaults to all identity class levels
cells	A vector of cells to grouping to
return.null	If no cells are requested, return a NULL; by default, throws an error

Value

A named list where names are identity classes and values are vectors of cells belonging to that class

Examples

```
CellsByIdentities(object = pbmc_small)
```

`CellsByImage` *Get a vector of cell names associated with an image (or set of images)*

Description

Get a vector of cell names associated with an image (or set of images)

Usage

```
CellsByImage(object, images = NULL, unlist = FALSE)
```

Arguments

`object`	Seurat object
`images`	Vector of image names
`unlist`	Return as a single vector of cell names as opposed to a list, named by image name.

Value

A vector of cell names

Examples

```
## Not run:
CellsByImage(object = object, images = "slice1")

## End(Not run)
```

`Centroids-class` *The* `Centroids` *Class*

Description

The `Centroids` Class

Slots

`cells` (`character [n]`) A vector of cell names; there should be as many cell names as there are points and no duplicate names

`nsides` (`integer [1L]`) The number of sides to draw when plotting centroids; must be either `0L` for circles or greater than 3

`radius` (`numeric [1L]`) The radius of the shape when plotting the centroids

`theta` (`numeric [1L]`) The angle in degrees to adjust the shape when plotting the centroids

See Also

Centroids methods: Centroids-methods

Segmentation layer classes: Centroids-methods, Molecules-class, Molecules-methods, Segmentation-class, Segmentation-methods

---

Centroids-methods Centroids *Methods*

Description

Methods for Centroids objects

Usage

```
## S3 method for class 'Centroids'
Cells(x, ...)

## S3 method for class 'Centroids'
GetTissueCoordinates(object, full = TRUE, ...)

## S3 method for class 'Centroids'
Radius(object, ...)

## S3 method for class 'Centroids'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'Centroids'
Theta(object)

## S3 method for class 'Centroids'
is.finite(x)

## S3 method for class 'Centroids'
is.infinite(...)

## S3 method for class 'Centroids'
length(x)

## S3 method for class 'Centroids'
lengths(x, use.names = TRUE)

## S3 method for class 'Centroids'
subset(x, cells = NULL, ...)

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

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

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

Arguments

x, object	A Centroids object
...	Arguments passed to other methods
full	Expand the coordinates to the full polygon
new.names	vector of new cell names
use.names	Ignored
i, cells	A vector of cells to keep; if NULL, defaults to all cells
j, drop	Ignored

Details

GetTissueCoordinates: Get cell spatial coordinates

Radius: Get the centroid radius

RenameCells: Update cell names

Theta: Get the offset angle

is.finite, is.infinite: Test to see if the centroids are circular or polygonal

length: Get the number of sides for the polygonal centroid

lengths: Generate a run-length encoding of the cells present

subset, [: Subset a Centroids object to certain cells

show: Display an object summary to stdout

Value

GetTissueCoordinates: A data frame with three columns:

- "x": the x-coordinate
- "y": the y-coordinate
- "cell": the cell name

If full is TRUE, then each coordinate will indicate a vertex for the cell polygon; otherwise, each coordinate will indicate a centroid for the cell

Radius The radius of the centroids

RenameCells: object with the cells renamed to new.names

Theta: The offset angle in degrees

is.finite: TRUE if the centroids are polygonal, FALSE if circular

`is.infinite`: The opposite of `is.finite`

`length`: `0` if the centroids are circular, otherwise the number of sides of the polygonal centroid

`lengths`: An `rle` object for the cells

`subset`, `[`: `x` subsetted to the cells specified by `cells/i`

`show`: Invisibly returns `NULL`

See Also

`Centroids-class`

Segmentation layer classes: `Centroids-class`, `Molecules-class`, `Molecules-methods`, `Segmentation-class`, `Segmentation-methods`

`CheckGC` *Conditional Garbage Collection*

Description

Call gc only when desired

Usage

```
CheckGC(option = "SeuratObject.memsafe")
```

Arguments

`option`	...

Value

Invisibly returns `NULL`

`CheckLayersName` *Check layers names for the input list*

Description

Check layers names for the input list

Usage

```
CheckLayersName(matrix.list, layers.type = c("counts", "data"))
```

Arguments

`matrix.list`	A list of matrices
`layers.type`	layers type, such as counts or data

Command *Get SeuratCommands*

Description

Pull information on previously run commands in the Seurat object.

Usage

```
Command(object, ...)

## S3 method for class 'Seurat'
Command(object, command = NULL, value = NULL, ...)
```

Arguments

object	An object
...	Arguments passed to other methods
command	Name of the command to pull, pass NULL to get the names of all commands run
value	Name of the parameter to pull the value for

Value

Either a SeuratCommand object or the requested parameter value

CreateAssay5Object *Create a v5 Assay object*

Description

Create an Assay5 object from a feature expression matrix; the expected format of the matrix is features x cells

Usage

```
CreateAssay5Object(
  counts = NULL,
  data = NULL,
  min.cells = 0,
  min.features = 0,
  csum = NULL,
  fsum = NULL,
  ...
)
```

Arguments

counts	A two-dimensional expression matrix
data	Optional prenormalized data matrix
min.cells	Include features detected in at least this many cells; will subset the counts matrix as well. To reintroduce excluded features, create a new object with a lower cutoff
min.features	Include cells where at least this many features are detected
csum	Function for calculating cell sums
fsum	Function for calculating feature sums
...	Arguments passed to other methods

Value

An Assay5 object

CreateAssayObject *Create an Assay object*

Description

Create an Assay object from a feature (e.g. gene) expression matrix. The expected format of the input matrix is features x cells.

Usage

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

Arguments

counts	Unnormalized data such as raw counts or TPMs
data	Prenormalized data; if provided, do not pass counts
min.cells	Include features detected in at least this many cells. Will subset the counts matrix as well. To reintroduce excluded features, create a new object with a lower cutoff
min.features	Include cells where at least this many features are detected
key	Optional key to initialize assay with
check.matrix	Check counts matrix for NA, NaN, Inf, and non-integer values
...	Arguments passed to as.sparse

Details

Non-unique cell or feature names are not allowed. Please make unique before calling this function.

Value

A Assay object

See Also

v3 Assay object, validity, and interaction methods: $.Assay(), Assay-class, Assay-validity, [.Assay(), [[.Assay(), dim.Assay(), dimnames.Assay(), merge.Assay(), split.Assay(), subset.Assay()

Examples

```
## Not run:
pbmc_raw <- read.table(
  file = system.file('extdata', 'pbmc_raw.txt', package = 'Seurat'),
  as.is = TRUE
)
pbmc_rna <- CreateAssayObject(counts = pbmc_raw)
pbmc_rna

## End(Not run)
```

CreateCentroids *Create a Centroids Objects*

Description

Create a Centroids Objects

Usage

```
CreateCentroids(coords, nsides, radius, theta)
```

Arguments

coords	The coordinates of cell/spot centroids
nsides	The number of sides to represent cells/spots; pass Inf to plot as circles
radius	Radius of shapes when plotting
theta	Angle to adjust shapes when plotting

Value

A Centroids object

`CreateDimReducObject` *Create a DimReduc object*

Description

Create a DimReduc object

Usage

```
CreateDimReducObject(
  embeddings = new(Class = "matrix"),
  loadings = new(Class = "matrix"),
  projected = new(Class = "matrix"),
  assay = NULL,
  stdev = numeric(),
  key = NULL,
  global = FALSE,
  jackstraw = NULL,
  misc = list()
)
```

Arguments

`embeddings`	A matrix with the cell embeddings
`loadings`	A matrix with the feature loadings
`projected`	A matrix with the projected feature loadings
`assay`	Assay used to calculate this dimensional reduction
`stdev`	Standard deviation (if applicable) for the dimensional reduction
`key`	A character string to facilitate looking up features from a specific DimReduc
`global`	Specify this as a global reduction (useful for visualizations)
`jackstraw`	Results from the JackStraw function
`misc`	list for the user to store any additional information associated with the dimensional reduction

Value

A `DimReduc` object

See Also

Dimensional reduction object, validity, and interaction methods `DimReduc-class`, `DimReduc-validity`, `[.DimReduc()`, `[[.DimReduc()`, `dim.DimReduc()`, `merge.DimReduc()`, `print.DimReduc()`, `subset.DimReduc()`

Examples

```
data <- GetAssayData(pbmc_small[["RNA"]], slot = "scale.data")
pcs <- prcomp(x = data)
pca.dr <- CreateDimReducObject(
  embeddings = pcs$rotation,
  loadings = pcs$x,
  stdev = pcs$sdev,
  key = "PC",
  assay = "RNA"
)
```

CreateFOV *Create Spatial Coordinates*

Description

Create Spatial Coordinates

Usage

```
CreateFOV(coords, ...)

## S3 method for class 'Centroids'
CreateFOV(
  coords,
  molecules = NULL,
  assay = "Spatial",
  key = NULL,
  name = NULL,
  ...
)

## S3 method for class 'data.frame'
CreateFOV(
  coords,
  type = c("segmentation", "centroids"),
  nsides = Inf,
  radius = NULL,
  theta = 0L,
  molecules = NULL,
  assay = "Spatial",
  key = NULL,
  name = NULL,
  ...
)
```

```
## S3 method for class 'list'
CreateFOV(coords, molecules = NULL, assay = "Spatial", key = NULL, ...)

## S3 method for class 'Segmentation'
CreateFOV(
  coords,
  molecules = NULL,
  assay = "Spatial",
  key = NULL,
  name = NULL,
  ...
)
```

Arguments

coords	Spatial coordinates
...	Arguments passed to other methods
molecules	A data.frame with spatially-resolved molecule information or a Molecules object
assay	Name of associated assay
key	Key for these spatial coordinates
name	When coords is a data.frame, Centroids, or Segmentation, name to store coordinates as
type	When providing a data.frame, specify if the coordinates represent a cell segmentation or voxel centroids
nsides	The number of sides to represent cells/spots; pass Inf to plot as circles
radius	Radius of shapes when plotting
theta	Angle to adjust shapes when plotting

Value

A FOV object

See Also

FOV-class

CreateMolecules *Create a Molecules Object*

Description

Create a Molecules Object

Usage

```
CreateMolecules(coords, ...)

## S3 method for class 'data.frame'
CreateMolecules(coords, key = "", ...)

## S3 method for class 'Molecules'
CreateMolecules(coords, ...)

## S3 method for class '`NULL`'
CreateMolecules(coords, ...)
```

Arguments

coords	Spatial coordinates for molecules; should be a data frame with three columns: • "x": x-coordinates for each molecule • "y": y-coordinates for each molecule • "gene": gene name for each molecule
...	Arguments passed to other methods
key	A key to set for the molecules

Value

A Molecules object

CreateSegmentation *Create a* Segmentation *Objects*

Description

Create a Segmentation Objects

Usage

```
CreateSegmentation(coords)

## S3 method for class 'data.frame'
CreateSegmentation(coords)

## S3 method for class 'Segmentation'
CreateSegmentation(coords)
```

Arguments

coords	The coordinates of cell segmentations

Value

A Segmentation object

CreateSeuratObject *Create a* Seurat *object*

Description

Create a Seurat object from raw data

Usage

```
CreateSeuratObject(
  counts,
  assay = "RNA",
  names.field = 1,
  names.delim = "_",
  meta.data = NULL,
  project = "CreateSeuratObject",
  ...
)

## Default S3 method:
CreateSeuratObject(
  counts,
  assay = "RNA",
  names.field = 1L,
  names.delim = "_",
  meta.data = NULL,
  project = "SeuratProject",
  min.cells = 0,
  min.features = 0,
  ...
)

## S3 method for class 'Assay'
CreateSeuratObject(
  counts,
  assay = "RNA",
  names.field = 1L,
  names.delim = "_",
  meta.data = NULL,
  project = "SeuratProject",
  ...
)

## S3 method for class 'Assay5'
```

```
CreateSeuratObject(
  counts,
  assay = "RNA",
  names.field = 1L,
  names.delim = "_",
  meta.data = NULL,
  project = "SeuratProject",
  ...
)
```

Arguments

counts	Either a matrix-like object with unnormalized data with cells as columns and features as rows or an Assay-derived object
assay	Name of the initial assay
names.field	For the initial identity class for each cell, choose this field from the cell's name. E.g. If your cells are named as BARCODE_CLUSTER_CELLTYPE in the input matrix, set names.field to 3 to set the initial identities to CELLTYPE.
names.delim	For the initial identity class for each cell, choose this delimiter from the cell's column name. E.g. If your cells are named as BARCODE-CLUSTER-CELLTYPE, set this to "-" to separate the cell name into its component parts for picking the relevant field.
meta.data	Additional cell-level metadata to add to the Seurat object. Should be a data.frame where the rows are cell names and the columns are additional metadata fields. Row names in the metadata need to match the column names of the counts matrix.
project	Project name for the Seurat object
...	Arguments passed to other methods
min.cells	Include features detected in at least this many cells. Will subset the counts matrix as well. To reintroduce excluded features, create a new object with a lower cutoff
min.features	Include cells where at least this many features are detected

Value

A Seurat object

Note

In previous versions (<3.0), this function also accepted a parameter to set the expression threshold for a 'detected' feature (gene). This functionality has been removed to simplify the initialization process/assumptions. If you would still like to impose this threshold for your particular dataset, simply filter the input expression matrix before calling this function.

Examples

```
## Not run:
pbmc_raw <- read.table(
  file = system.file('extdata', 'pbmc_raw.txt', package = 'Seurat'),
  as.is = TRUE
)
pbmc_small <- CreateSeuratObject(counts = pbmc_raw)
pbmc_small

## End(Not run)
```

Crop *Crop Coordinates*

Description

Crop Coordinates

Usage

```
Crop(object, x = NULL, y = NULL, coords = c("plot", "tissue"), ...)

## S3 method for class 'FOV'
Crop(object, x = NULL, y = NULL, coords = c("plot", "tissue"), ...)
```

Arguments

`object`	An object
`x, y`	Range to crop x/y limits to; if `NULL`, uses full range of `x/y`
`coords`	Coordinate system to execute crop; choose from: • "`plot`": Coordinates as shown when plotting • "`tissue`": Coordinates from `GetTissueCoordinates`
`...`	Arguments passed to other methods

Value

`object` cropped to the region specified by `x` and `y`

DefaultAssay *Default Assay*

Description

Get and set the default assay

Usage

```
DefaultAssay(object, ...)

DefaultAssay(object, ...) <- value

## S3 method for class 'Graph'
DefaultAssay(object, ...)

## S3 replacement method for class 'Graph'
DefaultAssay(object, ...) <- value

## S3 method for class 'Assay'
DefaultAssay(object, ...)

## S3 replacement method for class 'Assay'
DefaultAssay(object, ...) <- value

## S3 method for class 'Assay5'
DefaultAssay(object, ...)

## S3 replacement method for class 'Assay5'
DefaultAssay(object, ...) <- value

## S3 method for class 'SeuratCommand'
DefaultAssay(object, ...)

## S3 method for class 'DimReduc'
DefaultAssay(object, ...)

## S3 replacement method for class 'DimReduc'
DefaultAssay(object, ...) <- value

## S3 method for class 'Seurat'
DefaultAssay(object, ...)

## S3 replacement method for class 'Seurat'
DefaultAssay(object, ...) <- value
```

Arguments

object	An object
...	Arguments passed to other methods
value	Name of assay to set as default

Value

DefaultAssay: The name of the default assay

DefaultAssay<-: An object with the default assay updated

Examples

```
# Get current default assay
DefaultAssay(object = pbmc_small)

# Create dummy new assay to demo switching default assays
new.assay <- pbmc_small[["RNA"]]
Key(object = new.assay) <- "RNA2_"
pbmc_small[["RNA2"]] <- new.assay
# switch default assay to RNA2
DefaultAssay(object = pbmc_small) <- "RNA2"
DefaultAssay(object = pbmc_small)
```

DefaultDimReduc *Find the default* DimReduc

Description

Searches for DimReducs matching "umap", "tsne", or "pca", case-insensitive, and in that order. Priority given to DimReducs matching the DefaultAssay or assay specified (eg. "pca" for the default assay weights higher than "umap" for a non-default assay)

Usage

```
DefaultDimReduc(object, assay = NULL)
```

Arguments

object	A Seurat object
assay	Name of assay to use; defaults to the default assay of the object

Value

The default DimReduc, if possible

Examples

```
DefaultDimReduc(pbmc_small)
```

DefaultFOV *Get and Set the Default FOV*

Description

Get and Set the Default FOV

Usage

```
DefaultFOV(object, ...)

DefaultFOV(object, ...) <- value

## S3 method for class 'Seurat'
DefaultFOV(object, assay = NULL, ...)

## S3 replacement method for class 'Seurat'
DefaultFOV(object, assay = NA, ...) <- value
```

Arguments

object	A Seurat Object
...	Arguments passed to other methods
value	The name of the FOV to set as the default
assay	Name of assay to get or set default FOV for; pass NA to get or set the global default FOV

Value

DefaultFOV: The name of the default FOV

DefaultFOV<-: object with the default FOV set to value

DefaultLayer *Default Layer*

Description

Get and set the default layer

Usage

```
DefaultLayer(object, ...)

DefaultLayer(object, ...) <- value

## S3 method for class 'Assay'
DefaultLayer(object, ...)

## S3 method for class 'Assay5'
DefaultLayer(object, ...)

## S3 replacement method for class 'Assay5'
DefaultLayer(object, ...) <- value
```

Arguments

object	An object
...	Arguments passed to other methods
value	Name of layer to set as default

Value

`DefaultLayer`: The name of the default layer

`DefaultLayer<-`: An object with the default layer updated

---

dim.Assay *Feature and Cell Numbers*

Description

Feature and Cell Numbers

Usage

```
## S3 method for class 'Assay'
dim(x)
```

Arguments

x	An Assay object

Value

A two-length numeric vector with the total number of features and cells in x

See Also

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-class`, `Assay-validity`, `CreateAssayObject()`, `[.Assay()`, `[[.Assay()`, `dimnames.Assay()`, `merge.Assay()`, `split.Assay()`, `subset.Assay()`

Examples

```
rna <- pbmc_small[["RNA"]]
dim(rna)
```

---

dim.Assay5 *Feature and Cell Numbers*

Description

Feature and Cell Numbers

Usage

```
## S3 method for class 'Assay5'
dim(x)
```

Arguments

x	An `Assay5` object

Value

A two-length numeric vector with the total number of features and cells in `x`

See Also

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-class`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `split.Assay5()`, `subset.Assay5()`

dim.DimReduc *Dimensional Reduction Meta-Information*

Description

Pull meta-information about cells and dimensions for a given dimensional reduction; cell meta-information is stored as row meta-information (eg. `nrow`, `rownames`) and dimension meta-information is stored as column meta-information (eg. `ncol`, `colnames`)

Usage

```
## S3 method for class 'DimReduc'
dim(x)

## S3 method for class 'DimReduc'
dimnames(x)

## S3 method for class 'DimReduc'
length(x)

## S3 method for class 'DimReduc'
names(x)
```

Arguments

`x`	A `DimReduc` object

Value

`dim`: The number of cells (`nrow`) and dimensions (`ncol`)

`dimnames`: The cell (row) and dimension (column) names

`length`: The number of dimensions

`names`: The dimension identifiers

See Also

`Cells`

Dimensional reduction object, validity, and interaction methods `CreateDimReducObject()`, `DimReduc-class`, `DimReduc-validity`, `[.DimReduc()`, `[[.DimReduc()`, `merge.DimReduc()`, `print.DimReduc()`, `subset.DimReduc()`

Examples

```
pca <- pbmc_small[["pca"]]
pca
dim(pca)
```

```
# nrow is number of cells
nrow(pca)

# rownames pulls cell names
head(rownames(pca))

# ncol and length are number of dimensions
ncol(pca)
length(pca)

# colnames and names pull dimension identifiers
head(colnames(pca))
head(names(pca))
```

`dim.Seurat` *Feature and Cell Numbers*

Description

Feature and Cell Numbers

Usage

```
## S3 method for class 'Seurat'
dim(x)
```

Arguments

`x`	A `Seurat` object

Value

A two-length numeric vector with the total number of features and cells in `x`

See Also

Seurat object, validity, and interaction methods `$.Seurat()`, `Seurat-class`, `Seurat-validity`, `[[.Seurat()`, `[[<-,Seurat`, `[[<-,Seurat,NULL`, `dimnames.Seurat()`, `merge.Seurat()`, `names.Seurat()`, `subset.Seurat()`

Examples

```
# Get the number of features in an object
nrow(pbmc_small)

# Get the number of cells in an object
ncol(pbmc_small)
```

dimnames.Assay *Assay-Level Feature and Cell Names*

Description

Get and set feature and cell names in v5 Assays

Usage

```
## S3 method for class 'Assay'
dimnames(x)

## S3 replacement method for class 'Assay'
dimnames(x) <- value
```

Arguments

x	An Assay object
value	A two-length list where the first entry is the existing feature names for x and the second entry is the *updated* cell names for x

Value

dimnames: A two-length list with the following values:

- A character vector will all features in x
- A character vector will all cells in x

dimnames<-: x with the cell names updated to those in value[[2L]]

See Also

v3 Assay object, validity, and interaction methods: $.Assay(), Assay-class, Assay-validity, CreateAssayObject(), [.Assay(), [[.Assay(), dim.Assay(), merge.Assay(), split.Assay(), subset.Assay()

Cells(), dimnames.Assay5(), dimnames.Seurat()

Examples

```
rna <- pbmc_small[["RNA"]]

# Feature and cell names can be acquired with `rownames` and `colnames`
head(rownames(rna))
head(colnames(rna))

# Cell names can be updated with `colnames<-`
colnames(rna)[1] <- "newcell"
head(colnames(rna))
```

dimnames.Assay5 *Assay-Level Feature and Cell Names*

Description

Get and set feature and cell names in v5 Assays

Usage

```
## S3 method for class 'Assay5'
dimnames(x)

## S3 replacement method for class 'Assay5'
dimnames(x) <- value
```

Arguments

x	An Assay5 object
value	A two-length list with updated feature and/or cells names

Value

dimnames: A two-length list with the following values:

- A character vector with all features in x
- A character vector with all cells in x

dimnames<-: x with the feature and/or cell names updated to value

See Also

v5 Assay object, validity, and interaction methods: $.Assay5(), Assay5-class, Assay5-validity, [.Assay5(), [[.Assay5(), dim.Assay5(), merge.Assay5(), split.Assay5(), subset.Assay5()

Cells(), dimnames.Assay(), dimnames.Seurat()

dimnames.Seurat *Feature and Cell Names*

Description

Get and set feature and cell inames in Seurat objects

Usage

```
## S3 method for class 'Seurat'
dimnames(x)

## S3 replacement method for class 'Seurat'
dimnames(x) <- value
```

Arguments

x	A Seurat object
value	A two-length list with updated feature and/or cells names

Value

dimnames: A two-length list with the following values:

- A character vector with all features in the default assay
- A character vector with all cells in x

dimnames<-: x with the feature and/or cell names updated to value

See Also

Seurat object, validity, and interaction methods $.Seurat(), Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat, [[<-,Seurat,NULL, dim.Seurat(), merge.Seurat(), names.Seurat(), subset.Seurat()

Cells(), dimnames.Assay(), dimnames.Assay5()

Examples

```
# Get the feature names of an object
head(rownames(pbmc_small))

# Get the cell names of an object
head(colnames(pbmc_small))

colnames(pbmc_small)[1] <- "newcell"
head(colnames(pbmc_small))
```

DimReduc-class *The Dimensional Reduction Class*

Description

The DimReduc object stores a dimensionality reduction taken out in Seurat; each DimReduc consists of a cell embeddings matrix, a feature loadings matrix, and a projected feature loadings matrix.

Slots

`cell.embeddings` Cell embeddings matrix (required)

`feature.loadings` Feature loadings matrix (optional)

`feature.loadings.projected` Projected feature loadings matrix (optional)

`assay.used` Name of assay used to generate `DimReduc` object

`global` Is this `DimReduc` global/persistent? If so, it will not be removed when removing its associated assay

`stdev` A vector of standard deviations

`jackstraw` A `JackStrawData-class` object associated with this `DimReduc`

`misc` A named list of unstructured miscellaneous data

`key` A one-length character vector with the object's key; keys must be one or more alphanumeric characters followed by an underscore "_" (regex pattern "`^[a-zA-Z][a-zA-Z0-9]*_$`")

See Also

Dimensional reduction object, validity, and interaction methods `CreateDimReducObject()`, `DimReduc-validity`, `[.DimReduc()`, `[[.DimReduc()`, `dim.DimReduc()`, `merge.DimReduc()`, `print.DimReduc()`, `subset.DimReduc()`

`DimReduc-validity` *Dimensional Reduction Validity*

Description

Validation of `DimReduc` objects is handled by `validObject`

Cell Embeddings Validation

The cell embeddings matrix must be a numeric matrix of dimensions n_{cells} by $d_{dimensions}$; row names must be the cell names and column names must be the dimension identifier. The dimension identifier must be "`key_dimension`" (eg. "`PC_1`"). Dimension identifiers must be in order and cannot be skipped

Feature and Projected Feature Loadings Validation

blah

Standard Deviations Validation

blah

Key Validation

Keys must be a one-length character vector; a key must be composed of one of the following:

- An empty string (eg. "''") where `nchar() == 0`
- An string composed of one or more alphanumeric values (both lower- and upper-case) that ends with an underscore ("_"); the first character must be a letter

Keys that are not empty strings are validated with the regex "`^[a-zA-Z][a-zA-Z0-9]*_$`"

See Also

Dimensional reduction object, validity, and interaction methods `CreateDimReducObject()`, `DimReduc-class`, `[.DimReduc()`, `[[.DimReduc()`, `dim.DimReduc()`, `merge.DimReduc()`, `print.DimReduc()`, `subset.DimReduc()`

`Distances` *Get the Neighbor nearest neighbors distance matrix*

Description

Get the Neighbor nearest neighbors distance matrix

Usage

```
Distances(object, ...)

## S3 method for class 'Neighbor'
Distances(object, ...)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods

Value

The distance matrix

droplevels.LogMap *Drop Unused Logical Map Values*

Description

Remove any unused values from a logical map

Usage

```
## S3 method for class 'LogMap'
droplevels(x, ...)
```

Arguments

x	A LogMap object
...	Ignored

Value

x with values not present in any observation removed

See Also

Logical map objects, validity, and interaction methods: LogMap, LogMap-validity, as.matrix.LogMap(), intersect.LogMap(), labels.LogMap()

Examples

```
map <- LogMap(letters[1:10])
map[['obs']] <- c(1, 3, 7)
map[['entry']] <- c(2, 7, 10)

# Remove unused values
map <- droplevels(map)
map
map[[]]
```

Embeddings *Get Cell Embeddings*

Description

Get Cell Embeddings

Usage

```
Embeddings(object, ...)

## S3 method for class 'DimReduc'
Embeddings(object, ...)

## S3 method for class 'Seurat'
Embeddings(object, reduction = "pca", ...)
```

Arguments

object	An object
...	Arguments passed to other methods
reduction	Name of reduction to pull cell embeddings for

Value

The embeddings matrix

Examples

```
# Get the embeddings directly from a DimReduc object
Embeddings(object = pbmc_small[["pca"]])[1:5, 1:5]

# Get the embeddings from a specific DimReduc in a Seurat object
Embeddings(object = pbmc_small, reduction = "pca")[1:5, 1:5]
```

EmptyMatrix *Empty Matrices*

Description

Create empty 0x0 matrices of varying types

Usage

```
EmptyMatrix(repr = "C", type = "d")
```

Arguments

repr	Representation of empty matrix; choose from: • "C" for a CsparseMatrix • "T" for a TsparseMatrix • "R" for an RsparseMatrix • "e" for an unpackedMatrix • "d" for a dense S3 matrix

- "spam" for a spam matrix

type — Type of resulting matrix to return, choose from:

- "d" for numeric matrices
- "l" for logical matrices
- "n" for pattern matrices

Note, when repr is "spam", type must be "d"; when repr is "d", setting type to "n" returns a logical matrix

Value

A 0x0 matrix of the specified representation and type

See Also

IsMatrixEmpty()

Examples

```
EmptyMatrix()
EmptyMatrix("spam")
```

FetchData *Access cellular data*

Description

Retrieves data (feature expression, PCA scores, metrics, etc.) for a set of cells in a Seurat object

Usage

```
FetchData(object, ...)

## S3 method for class 'DimReduc'
FetchData(object, vars, cells = NULL, ...)

## S3 method for class 'Seurat'
FetchData(
  object,
  vars,
  cells = NULL,
  layer = NULL,
  clean = TRUE,
  slot = deprecated(),
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
vars	List of all variables to fetch, use keyword "ident" to pull identity classes
cells	Cells to collect data for (default is all cells)
layer	Layer to pull feature data for
clean	Remove cells that are missing data; choose from: • "all": consider all columns for cleaning • "ident": consider all columns except the identity class for cleaning • "project": consider all columns except the identity class for cleaning; fill missing identity values with the object's project • "none": do not clean Passing TRUE is a shortcut for "ident"; passing FALSE is a shortcut for "none"
slot	Deprecated in favor of layer

Value

A data frame with cells as rows and cellular data as columns

Examples

```
pc1 <- FetchData(object = pbmc_small, vars = 'PC_1')
head(x = pc1)
head(x = FetchData(object = pbmc_small, vars = c('groups', 'ident')))
```

FilterObjects *Find Sub-objects of a Certain Class*

Description

Get the names of objects within a Seurat object that are of a certain class

Usage

```
FilterObjects(object, classes.keep = c("Assay", "StdAssay", "DimReduc"))
```

Arguments

object	A Seurat object
classes.keep	A vector of names of classes to get

Value

A vector with the names of objects within the Seurat object that are of class classes.keep

Lifecycle

[Deprecated]

`FilterObjects` was deprecated in version 5.0.0; use `.FilterObjects` instead

Examples

```
FilterObjects(pbmc_small)
```

FOV-class *The Field of View Object*

Description

A modern container for storing coordinates of spatially-resolved single cells. Capable of storing multiple cell segmentation boundary masks. Supports coordinates for spatially-resolved molecule (FISH) data. Compatible with `SpatialImage`

Slots

`molecules` A named list of `Molecules` objects defining spatially-resolved molecular coordinates

`boundaries` A named list of `Segmentation` and `Centroids` objects defining spatially-resolved boundaries

`assay` A character naming the associated assay of the spatial coordinates

`key` A one-length character vector with the object's key; keys must be one or more alphanumeric characters followed by an underscore "_" (regex pattern "`^[a-zA-Z][a-zA-Z0-9]*_$`")

See Also

`FOV-methods`

FOV-methods *FOV Methods*

Description

Methods for `FOV` objects

Usage

```
## S3 method for class 'FOV'
Cells(x, boundary = NULL, ...)

## S3 method for class 'FOV'
Features(x, set = NULL, ...)

## S3 method for class 'FOV'
FetchData(object, vars, cells = NULL, simplify = TRUE, ...)

## S3 method for class 'FOV'
GetTissueCoordinates(object, which = NULL, ...)

## S3 method for class 'FOV'
Keys(object, ...)

## S3 method for class 'FOV'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'FOV'
x$i, ...

## S3 method for class 'FOV'
x[i, j, ...]

## S3 method for class 'FOV'
x[[i, ...]]

## S3 method for class 'FOV'
length(x)

## S3 method for class 'FOV'
names(x)

## S3 method for class 'FOV'
subset(x, cells = NULL, features = NULL, ...)

## S4 replacement method for signature 'FOV,character,missing,Centroids'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'FOV,character,missing,Molecules'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'FOV,character,missing,NULL'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'FOV,character,missing,Segmentation'
x[[i, j, ...]] <- value
```

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

Arguments

x, object	A FOV object
boundary, set	Name of segmentation boundary or molecule set to extract cell or feature names for; pass NA to return all cells or feature names
...	Arguments passed to other methods
vars	A vector of variables to fetch; can be the name of a segmentation boundary, to get tissue coordinates, or molecule names, to get molecule coordinates
simplify	If only returning either boundary or molecule coordinates, return a single data frame instead of a list
which	Name of segmentation boundary or molecule set
new.names	vector of new cell names
i, cells	For [[and [[<-, the name of a segmentation or "molecules"; for FetchData, subset. and [, a vector of cells to keep
j, features	For subset and [, a vector of features to keep; for [[<-, not used
value	For [[<-, a replacement Molecules, Centroids, or Segmentation object; otherwise NULL to remove the boundary stored at i

Details

The following methods are defined for interacting with a FOV object:

Cells: Get cell names

Features: Get spatially-resolved molecule names

FetchData: Fetch boundary and/or molecule coordinates from a FOV object

GetTissueCoordinates: Get boundary or molecule coordinates from a FOV object

Keys: Get the keys of molecule sets contained within a FOV object

RenameCells: Update cell names

$, [[: Extract a segmentation boundary

length: Get the number of segmentation layers in a FOV object

names: Get the names of segmentation layers and molecule sets

subset, [: Subset a FOV object

[[<-: Add or remove segmentation layers and molecule information to/from a FOV object

show: Display an object summary to stdout

Value

`Cells`: A vector of cell names

`Features`: A vector of spatially-resolved molecule names; if no molecular information present, returns `NULL`

`FetchData`: If both molecule and boundary coordinates are requested, then a two-length list:

- "`molecules`": A data frame with the molecule coordinates requested. If molecules requested are keyed, the keys are preserved in the data frame
- "`coordinates`": A data frame with coordinates from the segmentation boundaries requested

If `simplify` is `TRUE` and only one data frame is generated, then only the data frame is returned. Otherwise, a one-length list is returned with the single data frame generated

`GetTissueCoordinates`: ...

`Keys`: A named vector of molecule set keys; names are the names of the molecule sets and values are the keys for the respective molecule set

`RenameCells`: `object` with the cells renamed to `new.names`

`$`, `[[`: The segmentation boundary or spatially-resolved molecule information stored at `i`

`length`: The number of segmentation layers (`Segmentation` or `Centroids` objects)

`names`: A vector of segmentation boundary and molecule set names

`subset`: `x` with just the cells and features specified

`[[<-`: Varies depending on the class of `value`:

- If `value` is `NULL`, returns `x` with the boundary `i` removed; also allows removing `molecules`; does not allow removing the default segmentation
- If `value` is a `Molecules`, returns `x` with `value` stored in `molecules`; requires that `i` is "molecules"
- Otherwise, stores `value` as a segmentation boundary named `i`

`show`: Invisibly returns `NULL`

See Also

`FOV-class`

`FOV-validity` *FOV Validity*

Description

Validation of `FOV` objects is handled by `validObject`

Boundary Validation

blah

Molecule Validation

blah

See Also

`validObject`

`GetImage` *Get image data*

Description

Get image data

Usage

```
GetImage(object, mode = c("grob", "raster", "plotly", "raw"), ...)

## S3 method for class 'Seurat'
GetImage(
  object,
  mode = c("grob", "raster", "plotly", "raw"),
  image = NULL,
  ...
)
```

Arguments

`object`	An object
`mode`	How to return the image; should accept one of "grob", "raster", "plotly", or "raw"
`...`	Arguments passed to other methods
`image`	Name of `SpatialImage` object to pull image data for; if `NULL`, will attempt to select an image automatically

Value

Image data, varying depending on the value of `mode`:

"grob" An object representing image data inheriting from `grob` objects (eg. `rastergrob`)

"raster" An object of class `raster`

"plotly" A list with image data suitable for Plotly rendering, see `plotly::layout` for more details

"raw" The raw image data as stored in the object

See Also

`layout`

`GetTissueCoordinates` *Get tissue coordinates*

Description

Get tissue coordinates

Usage

```
GetTissueCoordinates(object, ...)

## S3 method for class 'Seurat'
GetTissueCoordinates(object, image = NULL, ...)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`image`	Name of `SpatialImage` object to get coordinates for; if `NULL`, will attempt to select an image automatically

Value

A data frame with tissue coordinates

`Graph-class` *The Graph Class*

Description

The Graph class inherits from `dgCMatrix`. We do this to enable future expandability of graphs.

Slots

`assay.used` Optional name of assay used to generate `Graph` object

See Also

`dgCMatrix-class`

Other graph: `as.Graph()`

`HVFInfo` *Highly Variable Features*

Description

Get and set variable feature information for an `Assay` object. `HVFInfo` and `VariableFeatures` utilize generally variable features, while `SVFInfo` and `SpatiallyVariableFeatures` are restricted to spatially variable features

Usage

```
HVFInfo(object, method, status = FALSE, ...)

VariableFeatures(object, method = NULL, ...)

VariableFeatures(object, ...) <- value

SVFInfo(object, method, status, ...)

SpatiallyVariableFeatures(object, method, ...)

## S3 method for class 'Seurat'
HVFInfo(
  object,
  method = NULL,
  status = FALSE,
  assay = NULL,
  selection.method = deprecated(),
```

```
  ...
)

## S3 method for class 'Seurat'
VariableFeatures(
  object,
  method = NULL,
  assay = NULL,
  nfeatures = NULL,
  layer = NA,
  simplify = TRUE,
  selection.method = deprecated(),
  ...
)

## S3 replacement method for class 'Seurat'
VariableFeatures(object, assay = NULL, ...) <- value

## S3 method for class 'Seurat'
SVFInfo(
  object,
  method = c("markvariogram", "moransi"),
  status = FALSE,
  assay = NULL,
  selection.method = deprecated(),
  ...
)

## S3 method for class 'Seurat'
SpatiallyVariableFeatures(
  object,
  method = "moransi",
  assay = NULL,
  decreasing = TRUE,
  selection.method = deprecated(),
  ...
)

## S3 method for class 'Assay'
HVFInfo(object, method, status = FALSE, selection.method = deprecated(), ...)

## S3 method for class 'Assay'
SpatiallyVariableFeatures(
  object,
  method = "moransi",
  decreasing = TRUE,
  selection.method = deprecated(),
  ...
```

```
)

## S3 method for class 'Assay'
SVFInfo(
  object,
  method = c("markvariogram", "moransi"),
  status = FALSE,
  selection.method = deprecated(),
  ...
)

## S3 method for class 'Assay'
VariableFeatures(object, method = NULL, selection.method = deprecated(), ...)

## S3 replacement method for class 'Assay'
VariableFeatures(object, ...) <- value

## S3 method for class 'Assay5'
HVFInfo(object, method = NULL, status = FALSE, layer = NA, strip = TRUE, ...)

## S3 method for class 'Assay5'
VariableFeatures(
  object,
  method = NULL,
  layer = NA,
  simplify = TRUE,
  nfeatures = NULL,
  selection.method = deprecated(),
  ...
)

## S3 method for class 'StdAssay'
SVFInfo(
  object,
  method = c("markvariogram", "moransi"),
  status = FALSE,
  selection.method = deprecated(),
  ...
)

## S3 method for class 'Assay5'
SVFInfo(
  object,
  method = c("markvariogram", "moransi"),
  status = FALSE,
  selection.method = deprecated(),
  ...
)
```

```
## S3 method for class 'Assay5'
SpatiallyVariableFeatures(
  object,
  method = "moransi",
  decreasing = TRUE,
  selection.method = deprecated(),
  ...
)
```

Arguments

object	An object
method	Which method to pull. For HVFInfo and VariableFeatures, choose one from one of the following: • "vst" • "sctransform" or "sct" • "mean.var.plot", "dispersion", "mvp", or "disp" For SVFInfo and SpatiallyVariableFeatures, choose from: • "markvariogram" • "moransi"
status	Add variable status to the resulting data frame
...	Arguments passed to other methods
value	A character vector of variable features
assay	Name of assay to pull highly variable feature information for
selection.method	**[Deprecated]**
nfeatures	Maximum number of features to select when simplifying
layer	Layer to pull variable features for
simplify	When pulling for multiple layers, combine into a single vector and select a common set of variable features for all layers
decreasing	Return features in decreasing order (most spatially variable first).
strip	Remove method/layer identifiers from highly variable data frame

Value

HVFInfo: A data frame with feature means, dispersion, and scaled dispersion

VariableFeatures: a vector of the variable features

SVFInfo: a data frame with the spatially variable features

SpatiallyVariableFeatures: a character vector of the spatially variable features

Examples

```
# Get the HVF info from a specific Assay in a Seurat object
HVFInfo(object = pbmc_small, assay = "RNA")[1:5, ]

# Get the HVF info directly from an Assay object
HVFInfo(pbmc_small[["RNA"]], method = 'vst')[1:5, ]
```

Idents *Get, set, and manipulate an object's identity classes*

Description

Get, set, and manipulate an object's identity classes

Usage

```
Idents(object, ...)

Idents(object, ...) <- value

RenameIdents(object, ...)

ReorderIdent(object, var, ...)

SetIdent(object, ...)

StashIdent(object, save.name, ...)

## S3 method for class 'Seurat'
Idents(object, ...)

## S3 replacement method for class 'Seurat'
Idents(object, cells = NULL, drop = FALSE, replace = FALSE, ...) <- value

## S3 method for class 'Seurat'
ReorderIdent(
  object,
  var,
  reverse = FALSE,
  afxn = mean,
  reorder.numeric = FALSE,
  ...
)

## S3 method for class 'Seurat'
RenameIdents(object, ...)
```

```
## S3 method for class 'Seurat'
SetIdent(object, cells = NULL, value, ...)

## S3 method for class 'Seurat'
StashIdent(object, save.name = "orig.ident", ...)

## S3 method for class 'Seurat'
droplevels(x, ...)

## S3 method for class 'Seurat'
levels(x)

## S3 replacement method for class 'Seurat'
levels(x) <- value
```

Arguments

...	Arguments passed to other methods; for RenameIdents: named arguments as old.ident = new.ident; for ReorderIdent: arguments passed on to FetchData
value	The name of the identities to pull from object metadata or the identities themselves
var	Feature or variable to order on
save.name	Store current identity information under this name
cells	Set cell identities for specific cells
drop	Drop unused levels
replace	Replace identities for unset cells with NA
reverse	Reverse ordering
afxn	Function to evaluate each identity class based on; default is mean
reorder.numeric	Rename all identity classes to be increasing numbers starting from 1 (default is FALSE)
x, object	An object

Value

Idents: The cell identities

Idents<-: object with the cell identities changed

RenameIdents: An object with selected identity classes renamed

ReorderIdent: An object with

SetIdent: An object with new identity classes set

StashIdent: An object with the identities stashed

Examples

```
# Get cell identity classes
Idents(pbmc_small)

# Set cell identity classes
# Can be used to set identities for specific cells to a new level
Idents(pbmc_small, cells = 1:4) <- 'a'
head(Idents(pbmc_small))

# Can also set idents from a value in object metadata
colnames(pbmc_small[[]])
Idents(pbmc_small) <- 'RNA_snn_res.1'
levels(pbmc_small)

# Rename cell identity classes
# Can provide an arbitrary amount of idents to rename
levels(pbmc_small)
pbmc_small <- RenameIdents(pbmc_small, '0' = 'A', '2' = 'C')
levels(pbmc_small)

## Not run:
head(Idents(pbmc_small))
pbmc_small <- ReorderIdent(pbmc_small, var = 'PC_1')
head(Idents(pbmc_small))

## End(Not run)

# Set cell identity classes using SetIdent
cells.use <- WhichCells(pbmc_small, idents = '1')
pbmc_small <- SetIdent(pbmc_small, cells = cells.use, value = 'B')

head(pbmc_small[[]])
pbmc_small <- StashIdent(pbmc_small, save.name = 'idents')
head(pbmc_small[[]])

# Get the levels of identity classes of a Seurat object
levels(x = pbmc_small)

# Reorder identity classes
levels(x = pbmc_small)
levels(x = pbmc_small) <- c('C', 'A', 'B')
levels(x = pbmc_small)
```

Images *Pull spatial image names*

Description

List the names of `SpatialImage` objects present in a `Seurat` object. If `assay` is provided, limits search to images associated with that assay

Usage

```
Images(object, assay = NULL)
```

Arguments

object	A Seurat object
assay	Name of assay to limit search to

Value

A list of image names

Examples

```
## Not run:
Images(object)

## End(Not run)
```

Index *Get Neighbor algorithm index*

Description

Get Neighbor algorithm index

Usage

```
Index(object, ...)

Index(object, ...) <- value

## S3 method for class 'Neighbor'
Index(object, ...)

## S3 replacement method for class 'Neighbor'
Index(object, ...) <- value
```

Arguments

object	An object
...	Arguments passed to other methods
value	The index to store

Value

Returns the value in the alg.idx slot of the Neighbor object

Idents<-: A Neighbor object with the index stored

`Indices` *Get Neighbor nearest neighbor index matrices*

Description

Get Neighbor nearest neighbor index matrices

Usage

```
Indices(object, ...)

## S3 method for class 'Neighbor'
Indices(object, ...)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods

Value

A matrix with the nearest neighbor indices

`intersect.LogMap` *Find Common Logical Map Values*

Description

Identify values in a logical map that are common to every observation

Usage

```
## S3 method for class 'LogMap'
intersect(x, y = missing_arg(), ...)
```

Arguments

`x`	A `LogMap` object
`y`	Ignored
`...`	Ignored

Value

The values of `x` that are present in **every** observation

See Also

Logical map objects, validity, and interaction methods: LogMap, LogMap-validity, as.matrix.LogMap(), droplevels.LogMap(), labels.LogMap()

Examples

```
map <- LogMap(letters[1:10])
map[['obs']] <- c(1, 3, 7)
map[['entry']] <- c(2, 7, 10)

# Identify values that are present in every observation
intersect(map)
```

IsGlobal *Is an object global/persistent?*

Description

Typically, when removing Assay objects from an Seurat object, all associated objects (eg. DimReduc, Graph, and SeuratCommand objects) are removed as well. If an associated object is marked as global/persistent, the associated object will remain even if its original assay was deleted

Usage

```
IsGlobal(object, ...)

## Default S3 method:
IsGlobal(object, ...)

## S3 method for class 'DimReduc'
IsGlobal(object, ...)
```

Arguments

object	An object
...	Arguments passed to other methods

Value

TRUE if the object is global/persistent otherwise FALSE

Examples

```
IsGlobal(pbmc_small[['pca']])
```

IsMatrixEmpty *Check if a matrix is empty*

Description

Takes a matrix and asks if it's empty (either 0x0 or 1x1 with a value of NA)

Usage

```
IsMatrixEmpty(x)

## Default S3 method:
IsMatrixEmpty(x)
```

Arguments

x	A matrix

Value

Whether or not x is empty

See Also

EmptyMatrix()

Examples

```
IsMatrixEmpty(new("matrix"))
IsMatrixEmpty(matrix())
IsMatrixEmpty(matrix(1:3))
```

IsNamedList *Check List Names*

Description

Check to see if a list has names; also check to enforce that all names are present and unique

Usage

```
IsNamedList(x, all.unique = TRUE, allow.empty = FALSE, pass.zero = FALSE)
```

Arguments

x	A list
all.unique	Require that all names are unique from one another
allow.empty	Allow empty (nchar = 0) names
pass.zero	Pass on zero-length lists

Value

TRUE if ..., otherwise FALSE

Examples

```
IsNamedList(list())
IsNamedList(list(), pass.zero = TRUE)
IsNamedList(list(1, 2, 3))
IsNamedList(list(a = 1, b = 2, c = 3))
IsNamedList(list(a = 1, 2, c = 3))
IsNamedList(list(a = 1, 2, c = 3), allow.empty = TRUE)
IsNamedList(list(a = 1, a = 2, a = 3))
IsNamedList(list(a = 1, a = 2, a = 3), all.unique = FALSE)
```

JackStrawData-class *The JackStrawData Class*

Description

The JackStrawData is used to store the results of a JackStraw computation.

Slots

empirical.p.values Empirical p-values

fake.reduction.scores Fake reduction scores

empirical.p.values.full Empirical p-values on full

overall.p.values Overall p-values from ScoreJackStraw

JackStrawData-methods JackStrawData *Methods*

Description

Methods for JackStrawData objects for generics defined in other packages

Usage

```
## S3 method for class 'JackStrawData'
.DollarNames(x, pattern = "")

## S3 method for class 'JackStrawData'
x$i, ...

## S3 method for class 'JackStrawData'
as.logical(x, ...)

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

Arguments

x, object	A JackStrawData object
pattern	A regular expression. Only matching names are returned.
i	A JackStrawData slot name
...	Ignored

Value

$: Slot i from x

as.logical: TRUE if empirical p-values have been calculated otherwise FALSE

show: Prints summary to stdout and invisibly returns NULL

Functions

- .DollarNames(JackStrawData): Autocompletion for $ access on a JackStrawData object
- $: Access data from a JackStrawData object
- as.logical(JackStrawData): Have empirical p-values for a JackStrawData object been calculated
- show(JackStrawData): Overview of a JackStrawData object

JoinLayers *Split and Join Layers Together*

Description

Split and Join Layers Together

Usage

```
JoinLayers(object, ...)

## S3 method for class 'Assay5'
JoinLayers(object, layers = NULL, new = NULL, ...)

## S3 method for class 'Seurat'
JoinLayers(object, assay = NULL, layers = NULL, new = NULL, ...)
```

Arguments

object	An object
...	Arguments passed to other methods
layers	Names of layers to split or join
new	Name of new layers
assay	Name of assay to split layers

Value

object with the layers specified joined

JS *Get and set JackStraw information*

Description

Get and set JackStraw information

Usage

```
JS(object, ...)

JS(object, ...) <- value

## S3 method for class 'JackStrawData'
JS(object, slot, ...)
```

```
## S3 replacement method for class 'JackStrawData'
JS(object, slot, ...) <- value

## S3 method for class 'DimReduc'
JS(object, slot = NULL, ...)

## S3 replacement method for class 'DimReduc'
JS(object, slot = NULL, ...) <- value
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`value`	JackStraw information
`slot`	Name of slot to store JackStraw scores to Can shorten to 'empirical', 'fake', 'full', or 'overall'

Value

`JS`: either a `JackStrawData` object or the specified jackstraw data

`JS<-`: `object` with the update jackstraw information

`Key` *Get and set object keys*

Description

Get and set object keys

Usage

```
Key(object, ...)

Keys(object, ...)

Key(object, ...) <- value

## S3 method for class 'Assay'
Key(object, ...)

## S3 replacement method for class 'Assay'
Key(object, ...) <- value

## S3 method for class 'Assay5'
Key(object, ...)
```

```
## S3 replacement method for class 'Assay5'
Key(object, ...) <- value

## S3 method for class 'DimReduc'
Key(object, ...)

## S3 replacement method for class 'DimReduc'
Key(object, ...) <- value

## S3 method for class 'Seurat'
Key(object, ...)

## S3 method for class 'Seurat'
Keys(object, ...)
```

Arguments

object	An object
...	Arguments passed to other methods
value	Key value

Value

Key: the object key

Keys: a named vector of keys of sub-objects

Key<-: object with an updated key

Examples

```
# Get an Assay key
Key(pbmc_small[["RNA"]])

# Set the key for an Assay
Key(pbmc_small[["RNA"]]) <- "newkey_"
Key(pbmc_small[["RNA"]])

# Get a DimReduc key
Key(object = pbmc_small[["pca"]])

# Set the key for DimReduc
Key(object = pbmc_small[["pca"]]) <- "newkey2_"
Key(object = pbmc_small[["pca"]])

# Show all keys associated with a Seurat object
Key(object = pbmc_small)
Keys(object = pbmc_small)
```

`labels.LogMap` *Find Observations by Value*

Description

Identify the observations that contain a specific value in a logical map

Usage

```
## S3 method for class 'LogMap'
labels(
  object,
  values,
  select = c("first", "last", "common", "all"),
  simplify = TRUE,
  ...
)
```

Arguments

`object`	A `LogMap` object
`values`	A vector of values to find observations for
`select`	Observation selection method; choose from: • "`first`": the first observation the value is found in • "`last`": the last observation the value is found in • "`common`": the first most-common observation the value is found in; most-common is determined by the observation that contains the most of the values requested • "`all`": all observations the value is found in
`simplify`	Simplify the resulting list to a vector
`...`	Ignored

Value

`labels`: A list, or vector if `simplify` is `TRUE`, of all values and the observations they're found in, according to the value of `select`

See Also

Logical map objects, validity, and interaction methods: `LogMap`, `LogMap-validity`, `as.matrix.LogMap()`, `droplevels.LogMap()`, `intersect.LogMap()`

Examples

```
map <- LogMap(letters[1:10])
map[['obs']] <- c(1, 3, 7)
map[['entry']] <- c(2, 7, 10)

# Find observations for a set of values
labels(map, c('a', 'b', 'g'))
```

LayerData *Query and Manipulate Assay Layers*

Description

Query and Manipulate Assay Layers

Usage

```
LayerData(object, layer, ...)

LayerData(object, layer, ...) <- value

Layers(object, ...)

## S3 method for class 'Assay'
LayerData(
  object,
  layer = NULL,
  cells = NULL,
  features = NULL,
  slot = deprecated(),
  ...
)

## S3 replacement method for class 'Assay'
LayerData(object, layer, ...) <- value

## S3 method for class 'Assay'
Layers(object, search = NA, ...)

## S3 method for class 'Assay5'
LayerData(
  object,
  layer = NULL,
  cells = NULL,
  features = NULL,
  fast = FALSE,
```

```
  slot = deprecated(),
  ...
)

## S3 replacement method for class 'Assay5'
LayerData(object, layer, features = NULL, cells = NULL, ...) <- value

## S3 method for class 'Assay5'
Layers(object, search = NA, ...)

## S3 method for class 'Seurat'
LayerData(object, layer = NULL, assay = NULL, slot = deprecated(), ...)

## S3 replacement method for class 'Seurat'
LayerData(object, layer, assay = NULL, ...) <- value

## S3 method for class 'Seurat'
Layers(object, search = NA, assay = NULL, ...)
```

Arguments

`object`	An object
`layer`	Name of layer to fetch or set
`...`	Arguments passed to other methods
`value`	New two-dimensional data to be added as a layer
`features`, `cells`	Vectors of features/cells to include
`slot`	**[Deprecated]**
`search`	A pattern to search layer names for; pass one of: • "`NA`" to pull all layers • "`NULL`" to pull the default layer(s) • a regular expression that matches layer names
`fast`	Determine how to return the layer data; choose from: `FALSE` Apply any transpositions and attempt to add feature/cell names (if supported) back to the layer data `NA` Attempt to add feature/cell names back to the layer data, skip any transpositions `TRUE` Do not apply any transpositions or add feature/cell names to the layer data
`assay`	Name of assay to fetch layer data from or assign layer data to

Value

`LayerData`: the layer data for `layer` from `object`

`Layer<-`: `object` with `value` added as a layer named `layer`

`Layers`: the names of the layers present in `object`

Loadings *Get and set feature loadings*

Description

Get and set feature loadings

Usage

```
Loadings(object, ...)

Loadings(object, ...) <- value

## S3 method for class 'DimReduc'
Loadings(object, projected = FALSE, ...)

## S3 replacement method for class 'DimReduc'
Loadings(object, projected = TRUE, ...) <- value

## S3 method for class 'Seurat'
Loadings(object, reduction = "pca", projected = FALSE, ...)
```

Arguments

object	An object
...	Arguments passed to other methods
value	Feature loadings to add
projected	Pull the projected feature loadings?
reduction	Name of reduction to pull feature loadings for

Value

Loadings: the feature loadings for object

Loadings<-: object with the updated loadings

Examples

```
# Get the feature loadings for a given DimReduc
Loadings(object = pbmc_small[["pca"]])[1:5,1:5]

# Set the feature loadings for a given DimReduc
new.loadings <- Loadings(object = pbmc_small[["pca"]])
new.loadings <- new.loadings + 0.01
Loadings(object = pbmc_small[["pca"]]) <- new.loadings

# Get the feature loadings for a specified DimReduc in a Seurat object
Loadings(object = pbmc_small, reduction = "pca")[1:5,1:5]
```

LogMap *A Logical Map*

Description

A simple container for storing mappings of values using logical matrices. Keeps track of which values (rows) are present in which observations (columns). `LogMap` objects can be created with `LogMap()`; queries can be performed with `[[` and observations can be added or removed with `[[<-`

Usage

```
LogMap(y)

## S4 method for signature 'LogMap,character,missing'
x[[i, j, ...]]

## S4 method for signature 'LogMap,missing,missing'
x[[i, j, ...]]

## S4 method for signature 'LogMap,NULL,missing'
x[[i, j, ...]]

## S4 replacement method for signature 'LogMap,character,missing,character'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'LogMap,character,missing,integer'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'LogMap,character,missing,NULL'
x[[i, j, ...]] <- value

## S4 replacement method for signature 'LogMap,character,missing,numeric'
x[[i, j, ...]] <- value
```

Arguments

`y`	A character vector
`x`	A `LogMap` object
`i`	A character vector of length 1, or `NULL`
`j`	Not used
`...`	Ignored
`value`	A character or integer vector of values to record in the map for `i`, or `NULL` to remove the record for `i`

Value

`LogMap`: A new `LogMap` object with zero columns and `length(x = x)` rows; rownames are set to `x`

`[[`: if `i` is a character vector, the rownames that are mapped to `i`; otherwise the rownames of `x`

`[[<-`: If `value` is `NULL`, then `x` without the observations for `i`; otherwise, `x` with a new column for `i` recording a `TRUE` for all values present in `value`

Slots

`.Data` A logical matrix with at least one row

See Also

Logical map objects, validity, and interaction methods: `LogMap-validity`, `as.matrix.LogMap()`, `droplevels.LogMap()`, `intersect.LogMap()`, `labels.LogMap()`

Examples

```
# Create a LogMap
map <- LogMap(letters[1:10])
map

# Get the names of values in the LogMap
map[[NULL]]
rownames(map)

# Add an observation to the LogMap
map[['obs']] <- c(1, 3, 7)
map[['entry']] <- c(2, 7, 10)
map

# Get the names of observations in the LogMap
colnames(map)

# Fetch an observation from the LogMap
map[['obs']]

# Get the full logical matrix
map[[]]

# Remove an observation from the LogMap
map[['obs']] <- NULL
map[['entry']] <- NULL
map
```

`LogMap-validity` *Logical Map Validity*

Description

Validation of `LogMap` objects is handled by `validObject`

Data Validation

Logical maps must be a logical matrix containing only TRUE or FALSE values

Value Validation

All values must be named within the rownames of the object. Duplicate or empty ("") values are not allowed

Observation Validation

All observations must be named within the column names of the object. Duplicate or empty ("") observations are not allowed

See Also

`validObject`

Logical map objects, validity, and interaction methods: `LogMap`, `as.matrix.LogMap()`, `droplevels.LogMap()`, `intersect.LogMap()`, `labels.LogMap()`

Examples

```
map <- LogMap(letters[1:10])
map[['obs']] <- c(1, 3, 7)
map[['entry']] <- c(2, 7, 10)
validObject(map)
```

`LogSeuratCommand` *Log a command*

Description

Logs command run, storing the name, timestamp, and argument list. Stores in the Seurat object

Usage

```
LogSeuratCommand(object, return.command = FALSE)
```

Arguments

object	Name of Seurat object
return.command	Return a SeuratCommand object instead

Value

If return.command, returns a SeuratCommand object; otherwise, returns the Seurat object with command stored

See Also

Command

Command log object and interaction methods $.SeuratCommand(), .DollarNames.SeuratCommand(), SeuratCommand-class, [.SeuratCommand(), as.list.SeuratCommand()

---

merge.Assay *Merge Assays*

Description

Merge one or more v3 assays together

Usage

```
## S3 method for class 'Assay'
merge(
  x = NULL,
  y = NULL,
  add.cell.ids = NULL,
  merge.data = TRUE,
  labels = NULL,
  collapse = TRUE,
  ...
)
```

Arguments

x	An Assay object
y	One or more Assay objects
add.cell.ids	A character vector of length(x = c(x, y)); appends the corresponding values to the start of each objects' cell names
merge.data	Merge the data slots instead of just merging the counts (which requires renormalization); this is recommended if the same normalization approach was applied to all objects
labels, collapse	Currently unused
...	Ignored

Value

A new assay with data merged from `c(x, y)`

See Also

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-class`, `Assay-validity`, `CreateAssayObject()`, `[.Assay()`, `[[.Assay()`, `dim.Assay()`, `dimnames.Assay()`, `split.Assay()`, `subset.Assay()`

---

merge.Assay5 *Merge Assays*

Description

Merge one or more v5 assays together

Usage

```
## S3 method for class 'Assay5'
merge(x, y, labels = NULL, add.cell.ids = NULL, collapse = FALSE, ...)
```

Arguments

`x`	An `Assay5` object
`y`	One or more `Assay5` objects
`labels`	A character vector equal to the number of objects; defaults to `as.character(seq_along(c(x, y)))`
`add.cell.ids`	A character vector equal to the number of objects provided to append to all cell names; if `TRUE`, uses `labels` as `add.cell.ids`
`collapse`	If `TRUE`, merge layers of the same name together; if `FALSE`, appends `labels` to the layer name
`...`	Ignored

Details

Note: collapsing layers is currently not supported

Value

A new v5 assay with data merged from `c(x, y)`

See Also

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-class`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `split.Assay5()`, `subset.Assay5()`

merge.DimReduc *Merge Dimensional Reductions*

Description

Merge two or more dimensional reduction together

Usage

```
## S3 method for class 'DimReduc'
merge(x = NULL, y = NULL, add.cell.ids = NULL, ...)
```

Arguments

x	A DimReduc object
y	One or more DimReduc objects
add.cell.ids	A character vector equal to the number of objects provided to append to all cell names; if TRUE, uses labels as add.cell.ids
...	Ignored

Value

A new DimReduc object with data merged from c(x, y)

See Also

Dimensional reduction object, validity, and interaction methods CreateDimReducObject(), DimReduc-class, DimReduc-validity, [.DimReduc(), [[.DimReduc(), dim.DimReduc(), print.DimReduc(), subset.DimReduc()

merge.Seurat *Merge Seurat Objects*

Description

Merge Seurat Objects

Usage

```
## S3 method for class 'Seurat'
merge(
  x = NULL,
  y = NULL,
  add.cell.ids = NULL,
  collapse = FALSE,
```

```
  merge.data = TRUE,
  merge.dr = FALSE,
  project = getOption(x = "Seurat.object.project", default = "SeuratProject"),
  ...
)
```

Arguments

x	A Seurat object
y	A single Seurat object or a list of Seurat objects
add.cell.ids	A character vector of length(x = c(x, y)); appends the corresponding values to the start of each objects' cell names
collapse	If TRUE, merge layers of the same name together; if FALSE, appends labels to the layer name
merge.data	Merge the data slots instead of just merging the counts (which requires renormalization); this is recommended if the same normalization approach was applied to all objects
merge.dr	Choose how to handle merging dimensional reductions: • "TRUE": merge dimensional reductions with the same name across objects; dimensional reductions with different names are added as-is • "NA": keep dimensional reductions from separate objects separate; will append the project name for duplicate reduction names • "FALSE": do not add dimensional reductions
project	Project name for the Seurat object
...	Arguments passed to other methods

Value

merge: Merged object

Merge Details

When merging Seurat objects, the merge procedure will merge the Assay level counts and potentially the data slots (depending on the merge.data parameter). It will also merge the cell-level meta data that was stored with each object and preserve the cell identities that were active in the objects pre-merge. The merge will optionally merge reductions depending on the values passed to merge.dr if they have the same name across objects. Here the embeddings slots will be merged and if there are differing numbers of dimensions across objects, only the first N shared dimensions will be merged. The feature loadings slots will be filled by the values present in the first object.The merge will not preserve graphs, logged commands, or feature-level metadata that were present in the original objects. If add.cell.ids isn't specified and any cell names are duplicated, cell names will be appended with _X, where X is the numeric index of the object in c(x, y).

See Also

Seurat object, validity, and interaction methods `$.Seurat()`, `Seurat-class`, `Seurat-validity`, `[[.Seurat()`, `[[<-,Seurat`, `[[<-,Seurat,NULL`, `dim.Seurat()`, `dimnames.Seurat()`, `names.Seurat()`, `subset.Seurat()`

Examples

```
# `merge' examples
# merge two objects
merge(pbmc_small, y = pbmc_small)
# to merge more than two objects, pass one to x and a list of objects to y
merge(pbmc_small, y = c(pbmc_small, pbmc_small))
```

Misc *Get and set miscellaneous data*

Description

Get and set miscellaneous data

Usage

```
Misc(object, ...)

Misc(object, ...) <- value

## S3 method for class 'Assay'
Misc(object, slot = NULL, ...)

## S3 replacement method for class 'Assay'
Misc(object, slot, ...) <- value

## S3 method for class 'Assay5'
Misc(object, slot = NULL, ...)

## S3 replacement method for class 'Assay5'
Misc(object, slot, ...) <- value

## S3 method for class 'DimReduc'
Misc(object, slot = NULL, ...)

## S3 replacement method for class 'DimReduc'
Misc(object, slot, ...) <- value

## S3 method for class 'Seurat'
Misc(object, slot = NULL, ...)
```

```
## S3 replacement method for class 'Seurat'
Misc(object, slot, ...) <- value
```

Arguments

object	An object
...	Arguments passed to other methods
value	Data to add
slot	Name of specific bit of meta data to pull

Value

Miscellaneous data

An object with miscellaneous data added

Examples

```
# Get the misc info
Misc(object = pbmc_small, slot = "example")

# Add misc info
Misc(object = pbmc_small, slot = "example") <- "testing_misc"
```

Molecules-class *The Spatial Molecules Class*

Description

The Spatial Molecules Class

Slots

`.Data` A list of `SpatialPoints` objects

`key` The key for the `Molecules`

See Also

`Molecules` methods: `Molecules-methods`

Segmentation layer classes: `Centroids-class`, `Centroids-methods`, `Molecules-methods`, `Segmentation-class`, `Segmentation-methods`

Molecules-methods Molecules *Methods*

Description

Methods for Molecules objects

Usage

```
## S3 method for class 'Molecules'
Features(x, ...)

## S3 method for class 'Molecules'
GetTissueCoordinates(object, features = NULL, ...)

## S3 method for class 'Molecules'
subset(x, features = NULL, ...)

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

Arguments

x, object	A Molecules object
...	Arguments passed to other methods
features	A vector of molecule names to keep; if NULL, defaults to all molecules

Details

Features: Get spatially-resolved molecule names

GetTissueCoordinates: Get spatially-resolved molecule coordinates

subset: Subset a Molecules object to certain molecules

show: Display an object summary to stdout

Value

Features: A vector of spatially-resolved molecule names; if no molecular information present, returns NULL

GetTissueCoordinates: A data frame with three columns:

- "x": the x-coordinate of a molecule
- "y": the y-coordinate of a molecule
- "molecule": the molecule name

subset: x subsetted to the features specified by features

show: Invisibly returns NULL

See Also

Molecules-class

Segmentation layer classes: Centroids-class, Centroids-methods, Molecules-class, Segmentation-class, Segmentation-methods

---

`names.Seurat` *Subobject Names*

Description

Get the names of subobjects within a Seurat object

Usage

```
## S3 method for class 'Seurat'
names(x)
```

Arguments

x	A Seurat object

Value

The names of all of the following subobjects within x:

- v3 and v5 assays
- dimensional reductions
- images and FOVs
- nearest-neighbor graphs

See Also

Seurat object, validity, and interaction methods $.Seurat(), Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat, [[<-,Seurat,NULL, dim.Seurat(), dimnames.Seurat(), merge.Seurat(), subset.Seurat()

Examples

```
names(pbmc_small)
```

Neighbor-class *The Neighbor class*

Description

The Neighbor class is used to store the results of neighbor finding algorithms

Slots

`nn.idx` Matrix containing the nearest neighbor indices

`nn.dist` Matrix containing the nearest neighbor distances

`alg.idx` The neighbor finding index (if applicable). E.g. the annoy index

`alg.info` Any information associated with the algorithm that may be needed downstream (e.g. distance metric used with annoy is needed when reading in from stored file).

`cell.names` Names of the cells for which the neighbors have been computed.

Neighbor-methods Neighbor *Methods*

Description

Methods for `Neighbor` objects for generics defined in other packages

Usage

```
## S3 method for class 'Neighbor'
dim(x)

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

Arguments

`x, object`	A `Neighbor` object

Value

`dim` Dimensions of the indices matrix

`show`: Prints summary to `stdout` and invisibly returns `NULL`

Functions

- `dim(Neighbor)`: Dimensions of the neighbor indices
- `show(Neighbor)`: Overview of a `Neighbor` object

`Overlay` *Overlay* `Spatial` *Objects Over One Another*

Description

Create an overlay of some query spatial object (`x`) against some target object (`y`). Basically, find all components of a query that fall within the bounds of a target spatial region

Usage

```
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'Centroids,SpatialPolygons'
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'Segmentation,SpatialPolygons'
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'Molecules,SpatialPolygons'
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'FOV,Spatial'
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'FOV,SpatialPolygons'
Overlay(x, y, invert = FALSE, ...)

## S4 method for signature 'FOV,FOV'
Overlay(x, y, invert = FALSE, ...)
```

Arguments

`x`	Query `Spatial` object
`y`	Target `Spatial` object
`invert`	Invert the overlay and return only the components of `x` that fall *outside* the bounds of `y`
`...`	Ignored

Value

`x` with only the components that fall within the bounds of `y`

Note

This function requires the **sf** package to be installed

PackageCheck *Check the existence of a package*

Description

Check the existence of a package

Usage

```
PackageCheck(..., error = TRUE)
```

Arguments

...	Package names
error	If true, throw an error if the package doesn't exist

Value

Invisibly returns boolean denoting if the package is installed

Lifecycle

[Deprecated]

PackageCheck was deprecated in version 5.0.0; please use rlang::check_installed() instead

Examples

```
PackageCheck("SeuratObject", error = FALSE)
```

pbmc_small *A small example version of the PBMC dataset*

Description

A subsetted version of 10X Genomics' 3k PBMC dataset

Usage

```
pbmc_small
```

Format

A Seurat object with the following slots filled

assays Currently only contains one assay ("RNA" - scRNA-seq expression data)

counts - Raw expression data

- data - Normalized expression data
- scale.data - Scaled expression data
- var.features - names of the current features selected as variable
- meta.features - Assay level metadata such as mean and variance

meta.data Cell level metadata

active.assay Current default assay

active.ident Current default idents

graphs Neighbor graphs computed, currently stores the SNN

reductions Dimensional reductions: currently PCA and tSNE

version Seurat version used to create the object

commands Command history

Source

https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k

---

print.DimReduc *Print Top Feature Loadings*

Description

Prints a set of features that most strongly define a set of components; **note**: requires feature loadings to be present in order to work

Usage

```
## S3 method for class 'DimReduc'
print(x, dims = 1:5, nfeatures = 20, projected = FALSE, ...)
```

Arguments

x	A DimReduc object
dims	Number of dimensions to display
nfeatures	Number of genes to display
projected	Use projected slot
...	Ignored

Value

Displays set of features defining the components and invisibly returns `x`

See Also

`cat`

Dimensional reduction object, validity, and interaction methods `CreateDimReducObject()`, `DimReduc-class`, `DimReduc-validity`, `[.DimReduc()`, `[[.DimReduc()`, `dim.DimReduc()`, `merge.DimReduc()`, `subset.DimReduc()`

Examples

```
pca <- pbmc_small[["pca"]]
print(pca)
```

`Project` *Get and set project information*

Description

Get and set project information

Usage

```
Project(object, ...)

Project(object, ...) <- value

## S3 method for class 'Seurat'
Project(object, ...)

## S3 replacement method for class 'Seurat'
Project(object, ...) <- value
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`value`	Project information to set

Value

Project information

An object with project information added

`Radius` *Get the spot radius from an image*

Description

Get the spot radius from an image

Usage

```
Radius(object, ...)
```

Arguments

`object`	An image object
`...`	Arguments passed to other methods

Value

The radius size

`RandomName` *Generate a random name*

Description

Make a name from randomly sampled characters, pasted together with no spaces

Usage

```
RandomName(length = 5L, chars = letters, ...)
```

Arguments

`length`	How long should the name be
`chars`	A vector of 1-length characters to use to generate the name
`...`	Extra parameters passed to `sample`

Value

A character with `nchar == length` of randomly sampled letters

See Also

`sample`

Examples

```
set.seed(42L)
RandomName()
RandomName(7L, replace = TRUE)
```

RenameAssays *Rename assays in a* Seurat *object*

Description

Rename assays in a Seurat object

Usage

```
RenameAssays(
  object,
  assay.name = NULL,
  new.assay.name = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
assay.name	original name of assay
new.assay.name	new name of assay
verbose	Whether to print messages
...	Named arguments as old.assay = new.assay

Value

object with assays renamed

Examples

```
RenameAssays(object = pbmc_small, RNA = 'rna')
```

`RenameCells` *Rename cells*

Description

Change the cell names in all the different parts of an object. Can be useful before combining multiple objects.

Usage

```
RenameCells(object, ...)

## S3 method for class 'Assay'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'Assay5'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'DimReduc'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'Neighbor'
RenameCells(object, old.names = NULL, new.names = NULL, ...)

## S3 method for class 'Seurat'
RenameCells(
  object,
  add.cell.id = missing_arg(),
  new.names = missing_arg(),
  for.merge = deprecated(),
  ...
)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`new.names`	vector of new cell names
`old.names`	vector of old cell names
`add.cell.id`	prefix to add cell names
`for.merge`	Deprecated

Details

If `add.cell.id` is set a prefix is added to existing cell names. If `new.names` is set these will be used to replace existing names.

Value

An object with new cell names

Examples

```
# Rename cells in an Assay
head(x = colnames(x = pbmc_small[["RNA"]]))
renamed.assay <- RenameCells(
    pbmc_small[["RNA"]],
    new.names = paste0("A_", colnames(x = pbmc_small[["RNA"]]))
)
head(x = colnames(x = renamed.assay))

# Rename cells in a DimReduc
head(x = Cells(x = pbmc_small[["pca"]]))
renamed.dimreduc <- RenameCells(
    object = pbmc_small[["pca"]],
    new.names = paste0("A_", Cells(x = pbmc_small[["pca"]]))
)
head(x = Cells(x = renamed.dimreduc))

# Rename cells in a Seurat object
head(x = colnames(x = pbmc_small))
pbmc_small <- RenameCells(object = pbmc_small, add.cell.id = "A")
head(x = colnames(x = pbmc_small))
```

RowMergeSparseMatrices

Merge Sparse Matrices by Row

Description

Merge two or more sparse matrices by rowname.

Usage

```
RowMergeSparseMatrices(mat1, mat2)
```

Arguments

mat1	First matrix
mat2	Second matrix or list of matrices

Details

Shared matrix rows (with the same row name) will be merged, and unshared rows (with different names) will be filled with zeros in the matrix not containing the row.

Value

Returns a sparse matrix

`SaveSeuratRds` *Save and Load* `Seurat` *Objects from Rds files*

Description

Save and Load `Seurat` Objects from Rds files

Usage

```
SaveSeuratRds(
  object,
  file = NULL,
  move = TRUE,
  destdir = deprecated(),
  relative = FALSE,
  ...
)

LoadSeuratRds(file, ...)
```

Arguments

`object`	A `Seurat` object
`file`	Path to save `object` to; defaults to `file.path(getwd(), paste0(Project(object), ".Rds"))`
`move`	Move on-disk layers into `dirname(file)`
`destdir`	**[Deprecated]**
`relative`	Save relative paths instead of absolute ones
`...`	Arguments passed on to `base::saveRDS`, `base::readRDS` `ascii` a logical. If `TRUE` or `NA`, an ASCII representation is written; otherwise (default), a binary one is used. See the comments in the help for `save`. `version` the workspace format version to use. `NULL` specifies the current default version (3). The only other supported value is 2, the default from R 1.4.0 to R 3.5.0. `compress` a logical specifying whether saving to a named file is to use `"gzip"` compression, or one of `"gzip"`, `"bzip2"` or `"xz"` to indicate the type of compression to be used. Ignored if `file` is a connection. `refhook` a hook function for handling reference objects.

Value

Invisibly returns `file`

Progress Updates with progressr

This function uses **progressr** to render status updates and progress bars. To enable progress updates, wrap the function call in `with_progress` or run `handlers(global = TRUE)` before running this function. For more details about **progressr**, please read `vignette("progressr-intro")`

Note

This function requires the **fs** package to be installed

See Also

`saveRDS()` `readRDS()`

Examples

```
if (requireNamespace("fs", quietly = TRUE)) {
  # Write out with DelayedArray
  if (requireNamespace("HDF5Array", quietly = TRUE)) {
    pbmc <- pbmc_small

    pbmc[["disk"]] <- CreateAssay5Object(list(
      mem = LayerData(pbmc, "counts"),
      disk = as(LayerData(pbmc, "counts"), "HDF5Array")
    ))

    # Save `pbmc` to an Rds file
    out <- tempfile(fileext = ".Rds")
    SaveSeuratRds(pbmc, file = out)

    # Object cache
    obj <- readRDS(out)
    Tool(obj, "SaveSeuratRds")

    # Load the saved object with on-disk layers back into memory
    pbmc2 <- LoadSeuratRds(out)
    pbmc2
    pbmc2[["disk"]]
  }

  # Write out with BPCells
  if (requireNamespace("BPCells", quietly = TRUE)) {
    pbmc <- pbmc_small

    bpm <- BPCells::write_matrix_dir(LayerData(pbmc, "counts"), dir = tempfile())
    bph <- BPCells::write_matrix_hdf5(
      LayerData(pbmc, "counts"),
      path = tempfile(fileext = ".h5"),
      group = "counts"
    )
    pbmc[["disk"]] <- CreateAssay5Object(list(dir = bpm, h5 = bph))

    # Save `pbmc` to an Rds file
```

```
    out <- tempfile(fileext = ".Rds")
    SaveSeuratRds(pbmc, file = out)

    # Object cache
    obj <- readRDS(out)
    Tool(obj, "SaveSeuratRds")

    # Load the saved object with on-disk layers back into memory
    pbmc2 <- LoadSeuratRds(out)
    pbmc2
    pbmc2[["disk"]]
  }
}
```

Segmentation-class *The* Segmentation *Class*

Description

The Segmentation Class

See Also

Segmentation methods: Segmentation-methods

Segmentation layer classes: Centroids-class, Centroids-methods, Molecules-class, Molecules-methods, Segmentation-methods

---

Segmentation-methods Segmentation *Methods*

Description

Methods for Segmentation objects

Usage

```
## S3 method for class 'Segmentation'
Cells(x, ...)

## S3 method for class 'Segmentation'
GetTissueCoordinates(object, full = TRUE, ...)

## S3 method for class 'Segmentation'
RenameCells(object, new.names = NULL, ...)
```

```
## S3 method for class 'Segmentation'
lengths(x, use.names = TRUE)

## S3 method for class 'Segmentation'
subset(x, cells = NULL, ...)

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

## S4 method for signature 'Segmentation'
coordinates(obj, full = TRUE, ...)

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

Arguments

x, object, obj	A Segmentation object
...	Arguments passed to other methods
full	Expand the coordinates to the full polygon
new.names	vector of new cell names
use.names	Ignored
i, cells	A vector of cells to keep; if NULL, defaults to all cells
j, drop	Ignored

Details

Cells: Get cell names

GetTissueCoordinates, coordinates: Get tissue coordinates

RenameCells: Update cell names

lengths: Generate a run-length encoding of the cells present

subset, [: Subset a Segmentation object to certain cells

show: Display an object summary to stdout

Value

Cells: A vector of cell names

GetTissueCoordinates, coordinates: A data frame with three columns:

- "x": the x-coordinate
- "y": the y-coordinate
- "cell" or "ID": the cell name

If `full` is `TRUE`, then each coordinate will indicate a vertex for the cell polygon; otherwise, each coordinate will indicate a centroid for the cell. Note: `GetTissueCoordinates` ....

`RenameCells`: `object` with the cells renamed to `new.names`

`lengths`: An `rle` object for the cells

`subset`, `[`: `x` subsetted to the cells specified by `cells`/`i`

`show`: Invisibly returns `NULL`

Progress Updates with progressr

The following methods use **progressr** to render status updates and progress bars:

- `RenameCells`

To enable progress updates, wrap the function call in `with_progress` or run `handlers(global = TRUE)` before running this function. For more details about **progressr**, please read `vignette("progressr-intro")`

Parallelization with future

The following methods use **future** to enable parallelization:

- `RenameCells`

Parallelization strategies can be set using `plan`. Common plans include "`sequential`" for non-parallelized processing or "`multisession`" for parallel evaluation using multiple `R` sessions; for other plans, see the "Implemented evaluation strategies" section of `?future::plan`. For a more thorough introduction to **future**, see `vignette("future-1-overview")`

See Also

`Segmentation-class`

Segmentation layer classes: `Centroids-class`, `Centroids-methods`, `Molecules-class`, `Molecules-methods`, `Segmentation-class`

`set-if-null` *Set If or If Not* `NULL`

Description

Set a default value depending on if an object is `NULL`

Usage

```
x %||% y

x %iff% y
```

Arguments

x	An object to test
y	A default value

Value

For %||%: y if x is NULL; otherwise x

For %iff%: y if x is **not** NULL; otherwise x

Author(s)

For %||%: **rlang** developers

See Also

rlang::%||%

Examples

```
# Set if NULL
1 %||% 2
NULL %||% 2

# Set if *not* NULL
1 %iff% 2
NULL %iff% 2
```

Seurat-class *The Seurat Class*

Description

The Seurat object is a representation of single-cell expression data for R; each Seurat object revolves around a set of cells and consists of one or more Assay objects, or individual representations of expression data (eg. RNA-seq, ATAC-seq, etc). These assays can be reduced from their high-dimensional state to a lower-dimension state and stored as DimReduc objects. Seurat objects also store additional metadata, both at the cell and feature level (contained within individual assays). The object was designed to be as self-contained as possible, and easily extendable to new methods.

Slots

assays A list of assays for this project

meta.data Contains meta-information about each cell, starting with number of features detected (nFeature) and the original identity class (orig.ident); more information is added using AddMetaData

`active.assay` Name of the active, or default, assay; settable using `DefaultAssay`

`active.ident` The active cluster identity for this Seurat object; settable using `Idents`

`graphs` A list of `Graph` objects

`neighbors` ...

`reductions` A list of dimensional reduction objects for this object

`images` A list of spatial image objects

`project.name` Name of the project

`misc` A list of miscellaneous information

`version` Version of Seurat this object was built under

`commands` A list of logged commands run on this `Seurat` object

`tools` A list of miscellaneous data generated by other tools, should be filled by developers only using `Tool<-`

See Also

Seurat object, validity, and interaction methods `$.Seurat()`, `Seurat-validity`, `[[.Seurat()`, `[[<-,Seurat`, `[[<-,Seurat,NULL`, `dim.Seurat()`, `dimnames.Seurat()`, `merge.Seurat()`, `names.Seurat()`, `subset.Seurat()`

`Seurat-validity` *Seurat Object Validity*

Description

Validation of `Seurat` objects is handled by `validObject`

See Also

`validObject`

Seurat object, validity, and interaction methods `$.Seurat()`, `Seurat-class`, `[[.Seurat()`, `[[<-,Seurat`, `[[<-,Seurat,NULL`, `dim.Seurat()`, `dimnames.Seurat()`, `merge.Seurat()`, `names.Seurat()`, `subset.Seurat()`

SeuratCommand-class *The* SeuratCommand *Class*

Description

The SeuratCommand is used for logging commands that are run on a Seurat object; it stores parameters and timestamps

Slots

`name` Command name

`time.stamp` Timestamp of when command was tun

`assay.used` Optional name of assay used to generate SeuratCommand object

`call.string` String of the command call

`params` List of parameters used in the command call

See Also

Command log object and interaction methods `$.SeuratCommand()`, `.DollarNames.SeuratCommand()`, `LogSeuratCommand()`, `[.SeuratCommand()`, `as.list.SeuratCommand()`

show,LogMap-method `LogMap` *Object Overview*

Description

Overview of a `LogMap` object

Usage

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

Arguments

`object`	A `LogMap` object

Value

Prints summary to `stdout` and invisibly returns `NULL`

`Simplify` *Simplify Geometry*

Description

Simplify Geometry

Simplify segmentations by reducing the number of vertices

Usage

```
Simplify(coords, tol, topologyPreserve = TRUE)

## S3 method for class 'Spatial'
Simplify(coords, tol, topologyPreserve = TRUE)
```

Arguments

`coords`	A 'Segmentation' object
`tol`	Numerical tolerance value to be used by the Douglas-Peuker algorithm
`topologyPreserve`	Logical determining if the algorithm should attempt to preserve the topology of the original geometry

Value

A simplified version of `coords`

A 'Segmentation' object with simplified segmentation vertices

---

`SpatialImage-class` *The SpatialImage class*

Description

The `SpatialImage` class is a virtual class representing spatial information for Seurat. All spatial image information must inherit from this class for use with `Seurat` objects

Slots

`assay` Name of assay to associate image data with; will give this image priority for visualization when the assay is set as the active/default assay in a `Seurat` object

`key` A one-length character vector with the object's key; keys must be one or more alphanumeric characters followed by an underscore "_" (regex pattern "`^[a-zA-Z][a-zA-Z0-9]*_$`")

See Also

`SpatialImage-methods` for a list of required and provided methods

SpatialImage-methods SpatialImage *methods*

Description

Methods defined on the SpatialImage class. Some of these methods must be overridden in order to ensure proper functionality of the derived classes (see **Required methods** below). Other methods are designed to work across all SpatialImage-derived subclasses, and should only be overridden if necessary

Usage

```
## S3 method for class 'SpatialImage'
Cells(x, ...)

## S3 method for class 'SpatialImage'
DefaultAssay(object, ...)

## S3 replacement method for class 'SpatialImage'
DefaultAssay(object, ...) <- value

## S3 method for class 'SpatialImage'
GetImage(object, mode = c("grob", "raster", "plotly", "raw"), ...)

## S3 method for class 'SpatialImage'
GetTissueCoordinates(object, ...)

## S3 method for class 'SpatialImage'
IsGlobal(object, ...)

## S3 method for class 'SpatialImage'
Key(object, ...)

## S3 replacement method for class 'SpatialImage'
Key(object, ...) <- value

## S3 method for class 'SpatialImage'
Radius(object, ...)

## S3 method for class 'SpatialImage'
RenameCells(object, new.names = NULL, ...)

## S3 method for class 'SpatialImage'
x[i, ...]

## S3 method for class 'SpatialImage'
dim(x)
```

```
## S3 method for class 'SpatialImage'
subset(x, cells, ...)

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

Arguments

x, object	A SpatialImage-derived object
...	Arguments passed to other methods
value	Depends on the method:
	DefaultAssay<- Assay that the image should be associated with
	Key<- New key for the image
mode	How to return the image; should accept one of "grob", "raster", "plotly", or "raw"
new.names	vector of new cell names
i, cells	A vector of cells to keep

Value

[Override] Cells: should return cell names

DefaultAssay: The associated assay of a SpatialImage-derived object

DefaultAssay<-: object with the associated assay updated

[Override] GetImage: The image data from a SpatialImage-derived object

[Override] GetTissueCoordinates: ...

IsGlobal: returns TRUE as images are, by default, global

Key: The key for a SpatialImage-derived object

Key<-: object with the key set to value

Radius: The spot radius size; by default, returns NULL

[Override] RenameCells: object with the new cell names

[, subset: x/object for only the cells requested

[Override] dim: The dimensions of the image data in (Y, X) format

show: Prints summary to stdout and invisibly returns NULL

Functions

- Cells(SpatialImage): Get the cell names from an image (**[Override]**)
- DefaultAssay(SpatialImage): Get the associated assay of a SpatialImage-derived object
- DefaultAssay(SpatialImage) <- value: Set the associated assay of a SpatialImage-derived object
- GetImage(SpatialImage): Get the image data from a SpatialImage-derived object

- GetTissueCoordinates(SpatialImage): Get tissue coordinates for a SpatialImage-derived object (**[Override]**)
- IsGlobal(SpatialImage): Globality test for SpatialImage-derived object
- Key(SpatialImage): Get the key for a SpatialImage-derived object
- Key(SpatialImage) <- value: Set the key for a SpatialImage-derived object
- Radius(SpatialImage): Get the spot radius size
- RenameCells(SpatialImage): Rename cells in a SpatialImage-derived object (**[Override]**)
- [: Subset a SpatialImage-derived object
- dim(SpatialImage): Get the plotting dimensions of an image (**[Override]**)
- subset(SpatialImage): Subset a SpatialImage-derived object (**[Override]**)
- show(SpatialImage): Overview of a SpatialImage-derived object

Provided methods

These methods are defined on the SpatialImage object and should not be overridden without careful thought

- DefaultAssay and DefaultAssay<-
- Key and Key<-
- GetImage; this method *can* be overridden to provide image data, normally returns empty image data. If overridden, should default to returning a grob object
- IsGlobal
- Radius; this method *can* be overridden to provide a spot radius for image objects
- [; this method *can* be overridden to change default subset behavior, normally returns subset(x = x, cells = i). If overridden, should only accept i

Required methods

All subclasses of the SpatialImage class must define the following methods; simply relying on the SpatialImage method will result in errors. For required parameters and their values, see the Usage and Arguments sections

Cells Return the cell/spot barcodes associated with each position

dim Return the dimensions of the image for plotting in (Y, X) format

GetTissueCoordinates Return tissue coordinates; by default, must return a two-column data.frame with x-coordinates in the first column and y-coordinates in the second

Radius Return the spot radius; returns NULL by default for use with non-spot image technologies

RenameCells Rename the cell/spot barcodes for this image

subset Subset the image data by cells/spots

These methods are used throughout Seurat, so defining them and setting the proper defaults will allow subclasses of SpatialImage to work seamlessly

See Also

DefaultAssay

GetImage

GetTissueCoordinates

IsGlobal

Key

RenameCells

---

split.Assay *Split an Assay*

Description

Split an Assay

Usage

```
## S3 method for class 'Assay'
split(x, f, drop = FALSE, layers = NA, ...)
```

Arguments

x	An Assay object
f	a 'factor' in the sense that as.factor(f) defines the grouping, or a list of such factors in which case their interaction is used for the grouping. If x is a data frame, f can also be a formula of the form ~ g to split by the variable g, or more generally of the form ~ g1 + ... + gk to split by the interaction of the variables g1, ..., gk, where these variables are evaluated in the data frame x using the usual non-standard evaluation rules.
drop	logical indicating if levels that do not occur should be dropped (if f is a factor or a list).
layers	Names of layers to include in the split; pass NA for all layers; pass NULL for the default layer
...	Ignored

Value

Returns a v5 assay with splitted layers

See Also

v3 Assay object, validity, and interaction methods: $.Assay(), Assay-class, Assay-validity, CreateAssayObject(), [.Assay(), [[.Assay(), dim.Assay(), dimnames.Assay(), merge.Assay(), subset.Assay()

`split.Assay5` *Split an Assay*

Description

Split an Assay

Usage

```
## S3 method for class 'Assay5'
split(
  x,
  f,
  drop = FALSE,
  layers = c("counts", "data"),
  ret = c("assay", "multiassays", "layers"),
  ...
)
```

Arguments

`x`	An `Assay5` object
`f`	a 'factor' in the sense that `as.factor(f)` defines the grouping, or a list of such factors in which case their interaction is used for the grouping. If `x` is a data frame, `f` can also be a formula of the form `~ g` to split by the variable `g`, or more generally of the form `~ g1 + ... + gk` to split by the interaction of the variables `g1`, ..., `gk`, where these variables are evaluated in the data frame `x` using the usual non-standard evaluation rules.
`drop`	logical indicating if levels that do not occur should be dropped (if `f` is a `factor` or a list).
`layers`	Names of layers to include in the split; pass `NA` for all layers; pass `NULL` for the default layer
`ret`	Type of return value; choose from: • "`assay`": a single `Assay5` object • "`multiassay`": a list of `Assay5` objects • "`layers`": a list of layer matrices
`...`	Ignored

Value

Depends on the value of `ret`:

- "`assay`": `x` with the layers requested in `layers` split based on `f`; all other layers are left as-is
- "`multiassay`": a list of `Assay5` objects; the list contains one value per split and each assay contains only the layers requested in `layers` with the key set to the split

- "layers": a list of matrices of length `length(assays) * length(unique(f))`; the list is named as "`layer.split`"

Progress Updates with progressr

This function uses **progressr** to render status updates and progress bars. To enable progress updates, wrap the function call in `with_progress` or run `handlers(global = TRUE)` before running this function. For more details about **progressr**, please read `vignette("progressr-intro")`

See Also

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-class`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `subset.Assay5()`

Stdev *Get the standard deviations for an object*

Description

Get the standard deviations for an object

Usage

```
Stdev(object, ...)

## S3 method for class 'DimReduc'
Stdev(object, ...)

## S3 method for class 'Seurat'
Stdev(object, reduction = "pca", ...)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`reduction`	Name of reduction to use

Value

The standard deviations

Examples

```
# Get the standard deviations for each PC from the DimReduc object
Stdev(object = pbmc_small[["pca"]])

# Get the standard deviations for each PC from the Seurat object
Stdev(object = pbmc_small, reduction = "pca")
```

StitchMatrix *Stitch Matrices Together*

Description

Stitch Matrices Together

Usage

```
StitchMatrix(x, y, rowmap, colmap, ...)
```

Arguments

x	A matrix
y	One or more matrices of the same class or coercible to the same class as x
rowmap, colmap	LogMaps describing the row and cell membership of each matrix; the LogMap entries are assumed to be in the order of c(x, y)
...	Arguments passed to other methods

Value

A single matrix of type class(x) consisting of all values in component matrices

subset.Assay *Subset an Assay*

Description

Subset an Assay

Usage

```
## S3 method for class 'Assay'
subset(x, cells = NULL, features = NULL, ...)
```

Arguments

x	An Assay object
cells	Cell names
features	Feature names
...	Ignored

Value

x with just the cells and features specified by `cells` and `features`

See Also

v3 Assay object, validity, and interaction methods: `$.Assay()`, `Assay-class`, `Assay-validity`, `CreateAssayObject()`, `[.Assay()`, `[[.Assay()`, `dim.Assay()`, `dimnames.Assay()`, `merge.Assay()`, `split.Assay()`

Examples

```
rna <- pbmc_small[["RNA"]]
rna2 <- subset(rna, features = VariableFeatures(rna))
rna2
```

---

subset.Assay5 *Subset an Assay*

Description

Subset an Assay

Usage

```
## S3 method for class 'Assay5'
subset(x, cells = NULL, features = NULL, layers = NULL, ...)
```

Arguments

x	An `Assay5` object
cells	Cell names
features	Feature names
layers	Layer to keep; defaults to all layers
...	Ignored

Value

x with just the cells and features specified by `cells` and `features` for the layers specified by `layers`

See Also

v5 Assay object, validity, and interaction methods: `$.Assay5()`, `Assay5-class`, `Assay5-validity`, `[.Assay5()`, `[[.Assay5()`, `dim.Assay5()`, `dimnames.Assay5()`, `merge.Assay5()`, `split.Assay5()`

subset.DimReduc *Subset a Dimensional Reduction*

Description

Subset a DimReduc object

Usage

```
## S3 method for class 'DimReduc'
subset(x, cells = NULL, features = NULL, ...)
```

Arguments

x	A DimReduc object
cells, features	Cells and features to keep during the subset
...	Ignored

Value

x for cells cells and features features

See Also

Dimensional reduction object, validity, and interaction methods CreateDimReducObject(), DimReduc-class, DimReduc-validity, [.DimReduc(), [[.DimReduc(), dim.DimReduc(), merge.DimReduc(), print.DimReduc()

subset.Seurat *Subset* Seurat *Objects*

Description

Subset Seurat Objects

Usage

```
## S3 method for class 'Seurat'
subset(
  x,
  subset,
  cells = NULL,
  features = NULL,
  idents = NULL,
  return.null = FALSE,
  ...
```

```
)

## S3 method for class 'Seurat'
x[i, j, ...]
```

Arguments

x	A Seurat object
subset	Logical expression indicating features/variables to keep
cells, j	A vector of cell names or indices to keep
features, i	A vector of feature names or indices to keep
idents	A vector of identity classes to keep
return.null	If no cells are requested, return a NULL; by default, throws an error
...	Arguments passed to WhichCells

Value

subset: A subsetted Seurat object

[: object x with features i and cells j

See Also

WhichCells

Seurat object, validity, and interaction methods $.Seurat(), Seurat-class, Seurat-validity, [[.Seurat(), [[<-,Seurat, [[<-,Seurat,NULL, dim.Seurat(), dimnames.Seurat(), merge.Seurat(), names.Seurat()

Examples

```
# `subset` examples
subset(pbmc_small, subset = MS4A1 > 4)
subset(pbmc_small, subset = `DLGAP1-AS1` > 2)
subset(pbmc_small, idents = '0', invert = TRUE)
subset(pbmc_small, subset = MS4A1 > 3, slot = 'counts')
subset(pbmc_small, features = VariableFeatures(object = pbmc_small))

# `[` examples
pbmc_small[VariableFeatures(object = pbmc_small), ]
pbmc_small[, 1:10]
```

Theta *Get the offset angle*

Description

Get the offset angle

Usage

```
Theta(object)
```

Arguments

object	An object

Tool *Get and Set Additional Tool Data*

Description

Use `Tool` to get tool data. If no additional arguments are provided, will return a vector with the names of tools in the object.

Usage

```
Tool(object, ...)

Tool(object, ...) <- value

## S3 method for class 'Seurat'
Tool(object, slot = NULL, ...)

## S3 replacement method for class 'Seurat'
Tool(object, ...) <- value
```

Arguments

object	An object
...	Arguments passed to other methods
value	Information to be added to tool list
slot	Name of tool to pull

Value

If no additional arguments, returns the names of the tools in the object; otherwise returns the data placed by the tool requested

Note

For developers: set tool data using `Tool<-`. `Tool<-` will automatically set the name of the tool to the function that called `Tool<-`, so each function gets one entry in the tools list and cannot overwrite another function's entry. The automatic naming will also remove any method identifiers (eg. `RunPCA.Seurat` will become `RunPCA`); please plan accordingly

Examples

```
# Example function that adds unstructured data to tools
MyTool <- function(object) {
  sample.tool.output <- matrix(rnorm(n = 16), nrow = 4)
  # Note: `Tool<-` must be called from within a function
  # and the name of the tool will be generated from the function name
  Tool(object) <- sample.tool.output
  return(object)
}

# Run our tool
set.seed(42L)
pbmc_small <- MyTool(pbmc_small)

# Get a list of tools run
Tool(pbmc_small)

# Access specific tool data
Tool(pbmc_small, slot = "MyTool")
```

`UpdateSeuratObject` *Update old Seurat object to accommodate new features*

Description

Updates Seurat objects to new structure for storing data/calculations. For Seurat v3 objects, will validate object structure ensuring all keys and feature names are formed properly.

Usage

```
UpdateSeuratObject(object)
```

Arguments

`object`	Seurat object

Value

Returns a Seurat object compatible with latest changes

Examples

```
## Not run:
updated_seurat_object = UpdateSeuratObject(object = old_seurat_object)

## End(Not run)
```

UpdateSlots *Update slots in an object*

Description

Update slots in an object

Usage

```
UpdateSlots(object)
```

Arguments

object	An object to update

Value

object with the latest slot definitions

Version *Get Version Information*

Description

Get Version Information

Usage

```
Version(object, ...)

## S3 method for class 'Seurat'
Version(object, ...)
```

Arguments

object	An object
...	Arguments passed to other methods

Examples

```
Version(pbmc_small)
```

`WhichCells` *Identify cells matching certain criteria*

Description

Returns a list of cells that match a particular set of criteria such as identity class, high/low values for particular PCs, etc.

Usage

```
WhichCells(object, ...)

## S3 method for class 'Assay'
WhichCells(object, cells = NULL, expression, invert = FALSE, ...)

## S3 method for class 'Seurat'
WhichCells(
  object,
  cells = NULL,
  idents = NULL,
  expression,
  slot = "data",
  invert = FALSE,
  downsample = Inf,
  seed = 1,
  ...
)
```

Arguments

`object`	An object
`...`	Arguments passed on to `CellsByIdentities`
	`return.null` If no cells are requested, return a `NULL`; by default, throws an error
`cells`	Subset of cell names
`expression`	A predicate expression for feature/variable expression, can evaluate anything that can be pulled by `FetchData`; please note, you may need to wrap feature names in backticks (``) if dashes between numbers are present in the feature name
`invert`	Invert the selection of cells
`idents`	A vector of identity classes to keep
`slot`	Slot to pull feature data for
`downsample`	Maximum number of cells per identity class, default is `Inf`; downsampling will happen after all other operations, including inverting the cell selection
`seed`	Random seed for downsampling. If NULL, does not set a seed

Value

A vector of cell names

See Also

FetchData

Examples

```
WhichCells(pbmc_small, idents = 2)
WhichCells(pbmc_small, expression = MS4A1 > 3)
levels(pbmc_small)
WhichCells(pbmc_small, idents = c(1, 2), invert = TRUE)
```

Index