Package 'SeuratObject'

November 7, 2022

Type Package

Title Data Structures for Single Cell Data

Version 4.1.3

Date 2022-11-07

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> for more details.

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

BugReports `https://github.com/mojaveazure/seurat-object/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Depends R (>= 4.0.0), sp (>= 1.5.0)

Imports future, future.apply, grDevices, grid, Matrix (>= 1.5.0), methods, progressr, Rcpp (>= 1.0.5), rlang (>= 0.4.7), stats, tools, utils

Suggests ggplot2, rgeos, testthat

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

LinkingTo Rcpp, RcppEigen

NeedsCompilation yes

Author Rahul Satija [aut] (<https://orcid.org/0000-0001-9448-8833>),
Andrew Butler [aut] (<https://orcid.org/0000-0003-3608-0463>),
Paul Hoffman [aut, cre] (<https://orcid.org/0000-0002-7693-8957>),
Tim Stuart [aut] (<https://orcid.org/0000-0002-3044-0897>),
Jeff Farrell [ctb],
Shiwei Zheng [ctb] (<https://orcid.org/0000-0001-6682-6743>),
Christoph Hafemeister [ctb] (<https://orcid.org/0000-0001-6365-8254>),
Patrick Roelli [ctb],
Yuhan Hao [ctb] (<https://orcid.org/0000-0002-1810-0822>)

Maintainer Paul Hoffman <seurat@nygenome.org>

Repository CRAN

Date/Publication 2022-11-07 18:50:02 UTC

R topics documented:

`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 for more details.

Author(s)

Maintainer: Paul Hoffman `<seurat@nygenome.org>` (ORCID)

Authors:

- Rahul Satija `<rsatija@nygenome.org>` (ORCID)
- Andrew Butler `<abutler@nygenome.org>` (ORCID)
- Tim Stuart `<tstuart@nygenome.org>` (ORCID)

Other contributors:

- 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]
- Yuhan Hao `<yhao@nygenome.org>` (ORCID) [contributor]

See Also

Useful links:

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

---

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 '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[[]])
```

`aggregate` *Aggregate Molecules into an Expression Matrix*

Description

Aggregate Molecules into an Expression Matrix

Usage

```
## S3 method for class 'FOV'
aggregate(x, by = NULL, set = NULL, drop = TRUE, ...)

## S3 method for class 'Molecules'
aggregate(x, by, drop = TRUE, ...)
```

Arguments

`x`	An object with spatially-resolved molecule information
`by`	Name of a `Segmentation` within `object` or a `Segmentation` object
`set`	Name of molecule set to aggregate
`drop`	Drop molecules not present in a segmentation; if `FALSE`, adds a column called “`boundless`” consisting of molecule counts not in a segmentation
`...`	Arguments passed to other methods

Value

An expression matrix

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")`

Parallelization with future

This function uses **future** to enable parallelization. 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")`

`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
`...`	Arguments passed to other methods (ignored for now)
`weighted`	If TRUE, fill entries in Graph matrix with value from the nn.dist slot of the Neighbor object

Value

A `Graph` object

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.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
- `key` Key for the Assay
- `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` Utility slot for storing additional data associated with the assay

See Also

`Assay-methods`

`Assay-methods` `Assay` *Methods*

Description

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

Usage

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

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

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

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

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

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

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

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

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

## S4 method for signature 'Assay'
colMeans(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Assay'
colSums(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Assay'
rowMeans(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Assay'
rowSums(x, na.rm = FALSE, dims = 1, ..., slot = "data")

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

Arguments

`x, object`	An `Assay` object
`i, features`	For `[[`: metadata names; for all other methods, feature names or indices
`j, cells`	Cell names or indices
`...`	Arguments passed to other methods
`drop`	See `drop`
`n`	an integer vector of length up to `dim(x)` (or 1, for non-dimensioned objects). Values specify the indices to be selected in the corresponding dimension (or along the length) of the object. A positive value of `n[i]` includes the first/last `n[i]` indices in that dimension, while a negative value excludes the last/first `abs(n[i])`, including all remaining indices. `NA` or non-specified values (when `length(n)` < `length(dim(x))`) select all indices in that dimension. Must contain at least one non-missing value.
`y`	A vector or list of one or more objects to merge
`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
`value`	Additional metadata to add
`na.rm`	logical. Should missing values (including `NaN`) be omitted from the calculations?
`dims`	completely ignored by the `Matrix` methods.
`slot`	Name of assay expression matrix to calculate column/row means/sums on

Value

`[`: The `data` slot for features `i` and cells `j`

`[[`: The feature-level metadata for `i`

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

`dimnames`: Feature (row) and cell (column) names

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

`merge`: Merged object

`subset`: A subsetted `Assay`

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

`[[<-`: `x` with metadata `value` added as `i`

`colMeans`: The column (cell-wise) means of `slot`

`colSums`: The column (cell-wise) sums of `slot`

`rowMeans`: The row (feature-wise) means of `slot`

`rowSums`: The row (feature-wise) sums of `slot`

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

Functions

- `[`: Get expression data from an `Assay`
- `[[`: Get feature-level metadata
- `dim(Assay)`: Number of cells and features for an `Assay`
- `dimnames(Assay)`: Cell- and feature-names for an `Assay`
- `head(Assay)`: Get the first rows of feature-level metadata
- `merge(Assay)`: Merge `Assay` objects
- `subset(Assay)`: Subset an `Assay`
- `tail(Assay)`: Get the last rows of feature-level metadata
- `` `[[`(x = Assay, i = ANY, j = ANY) <- value ``: Add feature-level metadata
- `colMeans(Assay)`: Calculate `colMeans` on an `Assay`
- `colSums(Assay)`: Calculate `colSums` on an `Assay`
- `rowMeans(Assay)`: Calculate `rowMeans` on an `Assay`
- `rowSums(Assay)`: Calculate `rowSums` on an `Assay`
- `show(Assay)`: Overview of an `Assay` object

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, slot, ...)

SetAssayData(object, slot, new.data, ...)

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

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

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

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

Arguments

object	An object
slot	Specific assay data to get or set
...	Arguments passed to other methods
new.data	New assay data to add
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

Examples

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

# Set an Assay slot through the Seurat object
```

```
count.data <- GetAssayData(object = pbmc_small[["RNA"]], slot = "counts")
count.data <- as.matrix(x = count.data + 1)
new.seurat.object <- SetAssayData(
    object = pbmc_small,
    slot = "counts",
    new.data = count.data,
    assay = "RNA"
)

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

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

Assays *Query Specific Object Types*

Description

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

Usage

```
Assays(object, slot = NULL)

Graphs(object, slot = NULL)

Neighbors(object, slot = NULL)

Reductions(object, slot = NULL)
```

Arguments

`object`	A `Seurat` object
`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(object = 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`

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`

`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 'DimReduc'
Cells(x, ...)

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

Arguments

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

Value

`Cell`: A vector of cell names

`Features`: A vector of feature names

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 request, 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: Molecules-class, Segmentation-class

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`

`CheckGC` *Conditional Garbage Collection*

Description

Call gc only when desired

Usage

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

Arguments

`option`	...

Value

Invisibly returns `NULL`

`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

`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,
  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.
`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

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

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

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

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

Arguments

`counts`	Either a `matrix`-like object with unnormalized data with cells as columns and features as rows or an `Assay`-derived object
`project`	Project name for the `Seurat` 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.
`...`	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.
`row.names`	When `counts` is a `data.frame` or `data.frame`-derived object: an optional vector of feature names to be used

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`
`...`	...

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 '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`

`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

`key` Key for the `DimReduc`, must be alphanumeric characters followed by an underscore

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

`misc` Utility slot for storing additional data associated with the `DimReduc` (e.g. the total variance of the PCA)

DimReduc-methods — DimReduc *Methods*

Description

Methods for DimReduc objects for generics defined in other packages

Usage

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

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

## 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'
merge(x = NULL, y = NULL, add.cell.ids = NULL, ...)

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

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

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

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

Arguments

`x, object`	A DimReduc object
`i`	For `[`: feature names or indices; for `[[`: cell names or indices
`j`	Dimensions to pull for
`drop`	See drop
`...`	Arguments passed to other methods

`y`	A vector or list of one or more objects to merge
`add.cell.ids`	A character vector of `length(x = c(x, y))`; appends the corresponding values to the start of each objects' cell names
`dims`	Number of dimensions to display
`nfeatures`	Number of genes to display
`projected`	Use projected slot
`cells, features`	Cells and features to keep during the subset

Value

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

`[[`: Cell embeddings for cells `i` and dimensions `j`

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

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

`length`: The number of dimensions

`names`: The names for the dimensions (eg. "PC_1")

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

`subset`: `x` for cells `cells` and features `features`

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

Functions

- `[`: Pull feature loadings
- `[[`: Pull cell embeddings
- `dim(DimReduc)`: The number of cells and dimensions for a `DimReduc`
- `dimnames(DimReduc)`: The cell and dimension names for a `DimReduc` object
- `length(DimReduc)`: The number of dimensions for a `DimReduc` object
- `merge(DimReduc)`: Merge two or more `DimReduc` objects together
- `names(DimReduc)`: The dimension names for a `DimReduc` object
- `print(DimReduc)`: Prints a set of features that most strongly define a set of components; **note**: requires feature loadings to be present in order to work
- `subset(DimReduc)`: Subset a `DimReduc` object
- `show(DimReduc)`: Show basic summary of a `DimReduc` object

See Also

`cat`

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

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]
```

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,
  slot = c("embeddings", "loadings", "projected"),
  ...
)

## S3 method for class 'Seurat'
FetchData(object, vars, cells = NULL, slot = "data", ...)
```

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)
`slot`	Slot to pull feature data for

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", "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`

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` (`list`) A named list of `Molecules` objects defining spatially-resolved molecular coordinates
- `boundaries` (`[named]list` {`Segmentation`, `Centroids`}) A named list of `Segmentation` and `Centroids` objects defining spatially-resolved boundaries
- `assay` (`character [1L]`) A character naming the associated assay of the spatial coordinates
- `key` (`character [1L]`) The key for the spatial coordinates

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`

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`

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, selection.method, status = FALSE, ...)

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

VariableFeatures(object, ...) <- value

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

SpatiallyVariableFeatures(object, selection.method, ...)

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

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

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

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

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

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

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

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

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

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

Arguments

`object`	An object
`selection.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
`decreasing`	Return features in decreasing order (most spatially variable first).

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"]], selection.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, ...) <- 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
`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

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)
```

Arguments

`x`	A matrix

Value

Whether or not x is empty

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`

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

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, ...)

Key(object, ...) <- value

Keys(object, ...)

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

## S3 replacement method for class 'Assay'
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

Key<-: `object` with an updated key

Keys: a named vector of keys of sub-objects

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)
```

`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-class` *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

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

Arguments

`y`	A character vector
`x, object`	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

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

# 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
```

`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`

`MatchCells` *Match Cells*

Description

Match Cells

Usage

```
MatchCells(new, orig, ordered = FALSE)

## S3 method for class 'character'
MatchCells(new, orig, ordered = FALSE)

## S3 method for class '`NULL`'
MatchCells(new, orig, ordered = FALSE)

## S3 method for class 'numeric'
MatchCells(new, orig, ordered = FALSE)
```

Arguments

`new`	A vector of new cells
`orig`	A vector of existing cells
`ordered`	Sort the result to the same order as `orig`

Value

A numeric vector with new cells in order of the original cells; if no match can be found, returns `NULL`

`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 '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`, `Segmentation-class`

`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

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`

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

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

---

`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

Value

The radius size

`RandomName` *Generate a random name*

Description

Make a name from randomly sampled lowercase letters, pasted together with no spaces or other characters

Usage

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

Arguments

`length`	How long should the name be
`...`	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, ...)
```

Arguments

`object`	A `Seurat` object
`...`	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 '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 = NULL,
  new.names = NULL,
  for.merge = FALSE,
  ...
)
```

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`	Only rename slots needed for merging Seurat objects. Currently only renames the raw.data and meta.data slots.

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

s4list *S4/List Conversion*

Description

Convert S4 objects to lists and vice versa. Useful for declassing an S4 object while keeping track of it's class using attributes (see section **S4 Class Definition Attributes** below for more details). Both `ListToS4` and `S4ToList` are recursive functions, affecting all lists/S4 objects contained as sub-lists/sub-objects.

Usage

```
S4ToList(object)

IsS4List(x)

ListToS4(x)

## Default S3 method:
S4ToList(object)

## S3 method for class 'list'
S4ToList(object)
```

Arguments

`object`	An S4 object
`x`	A list with an S4 class definition attribute

Value

`S4ToList`: A list with an S4 class definition attribute

`IsS4List`: `TRUE` if `x` is a list with an S4 class definition attribute

`ListToS4`: An S4 object as defined by the S4 class definition attribute

S4 Class Definition Attributes

S4 classes are scoped to the package and class name. In order to properly track which class a list is generated from in order to build a new one, these function use an `attribute` to denote the class name and package of origin. This attribute is stored as "classDef" and takes the form of "`package:class`".

`Segmentation-class` *The* Segmentation *Class*

Description

The Segmentation Class

See Also

Segmentation methods: `Segmentation-methods`

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

`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`

`set-if-na` *Set if* `NA`

Description

Set a default value depending on if an object is `NA`

Usage

```
x %NA% y

x %na% y

x %!NA% y

x %!na% y
```

Arguments

`x`	An object to test
`y`	A default value

Value

For `%NA%`: `y` if `x` is `NA`; otherwise `x`

For `%!NA%`: `y` if `x` is **not** `NA`; otherwise `x`

Examples

```
1 %NA% 2
NA %NA% 2

1 %!NA% 2
NA %!NA% 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`<-

`Seurat-methods` *Seurat Methods*

Description

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

Usage

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

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

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

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

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

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

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

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

## S3 method for class 'Seurat'
merge(
  x = NULL,
  y = NULL,
  add.cell.ids = NULL,
  merge.data = TRUE,
  merge.dr = NULL,
  project = "SeuratProject",
  ...
)

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

## S3 method for class 'Seurat'
subset(
  x,
  subset,
  cells = NULL,
  features = NULL,
  idents = NULL,
  return.null = FALSE,
  ...
)

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

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

## S4 method for signature 'Seurat'
colMeans(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Seurat'
colSums(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Seurat'
rowMeans(x, na.rm = FALSE, dims = 1, ..., slot = "data")

## S4 method for signature 'Seurat'
rowSums(x, na.rm = FALSE, dims = 1, ..., slot = "data")

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

Arguments

`x, object`	A `Seurat` object
`pattern`	A regular expression. Only matching names are returned.
`i, features`	Depends on the method `[, subset` Feature names or indices `$, $<-` Name of a single metadata column `[[, [[<-` Name of one or more metadata columns or an associated object; associated objects include `Assay`, `DimReduc`, `Graph`, `SeuratCommand`, or `SpatialImage` objects
`...`	Arguments passed to other methods
`value`	Additional metadata or associated objects to add; **note**: can pass `NULL` to remove metadata or an associated object
`j, cells`	Cell names or indices
`drop`	See `drop`
`n`	The number of rows of metadata to return
`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
`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`	Merge specified DimReducs that are present in all objects; will only merge the embeddings slots for the first `N` dimensions that are shared across all objects.
`project`	Project name for the `Seurat` object
`subset`	Logical expression indicating features/variables to keep
`idents`	A vector of identity classes to keep
`return.null`	If no cells are request, return a `NULL`; by default, throws an error
`na.rm`	logical. Should missing values (including `NaN`) be omitted from the calculations?
`dims`	completely ignored by the `Matrix` methods.
`slot`	Name of assay expression matrix to calculate column/row means/sums on

Value

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

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

`[`: object `x` with features `i` and cells `j`

`[[`: If `i` is missing, the metadata data frame; if `i` is a vector of metadata names, a data frame with the requested metadata, otherwise, the requested associated object

`dim`: The number of features (`nrow`) and cells (`ncol`) for the default assay; **note**: while the number of features changes depending on the active assay, the number of cells remains the same across all assays

`dimnames`: The feature (row) and cell (column) names; **note**: while the features change depending on the active assay, the cell names remain the same across all assays

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

`merge`: Merged object

`names`: The names of all `Assay`, `DimReduc`, `Graph`, and `SpatialImage` objects in the `Seurat` object

`subset`: A subsetted `Seurat` object

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

`[[<-`: `x` with the metadata or associated objects added as `i`; if `value` is `NULL`, removes metadata or associated object `i` from object `x`

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

Functions

- `.DollarNames(Seurat)`: Autocompletion for `$` access on a `Seurat` object
- `$`: Metadata access for `Seurat` objects
- `` `$`(Seurat) <- value ``: Metadata setter for `Seurat` objects
- `[`: Simple subsetter for `Seurat` objects
- `[[`: Metadata and associated object accessor
- `dim(Seurat)`: Number of cells and features for the active assay
- `dimnames(Seurat)`: The cell and feature names for the active assay

- `head(Seurat)`: Get the first rows of cell-level metadata
- `merge(Seurat)`: Merge two or more `Seurat` objects together
- `names(Seurat)`: Common associated objects
- `subset(Seurat)`: Subset a `Seurat` object
- `tail(Seurat)`: Get the last rows of cell-level metadata
- `` `[[`(x = Seurat, i = ANY, j = ANY) <- value ``: Add cell-level metadata or associated objects
- `colMeans(Seurat)`: Calculate `colMeans` on a `Seurat` object
- `colSums(Seurat)`: Calculate `colSums` on a `Seurat` object
- `rowMeans(Seurat)`: Calculate `rowMeans` on a `rowMeans` object
- `rowSums(Seurat)`: Calculate `rowSums` on a `Seurat` object
- `show(Seurat)`: Overview of a `Seurat` 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

`subset` `WhichCells`

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"]])

# `[' examples
pbmc_small[VariableFeatures(object = pbmc_small), ]
pbmc_small[, 1:10]

# 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 number of features in an object
nrow(pbmc_small)

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

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

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

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

# `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))

names(pbmc_small)

# `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))

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

head(colMeans(pbmc_small))

head(colSums(pbmc_small))

head(rowMeans(pbmc_small))

head(rowSums(pbmc_small))
```

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

`SeuratCommand-methods` `SeuratCommand` *Methods*

Description

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

Usage

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

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

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

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

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

Arguments

`x, object`	A `SeuratCommand` object
`pattern`	A regular expression. Only matching names are returned.
`i`	For a `$`, a parameter name; for `[`, a `SeuratCommand` slot name
`...`	Arguments passed to other methods
`complete`	Include slots besides just parameters (eg. call string, name, timestamp)

Value

`$`: The value for parameter `i`

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

`as.list`: A list with the parameters and, if `complete = TRUE`, the call string, name, and timestamp

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

Functions

- `.DollarNames(SeuratCommand)`: Autocompletion for `$` access on a `SeuratCommand` object
- `$`: Access a parameter from a `SeuratCommand` object
- `[`: Access data from a `SeuratCommand` object
- `as.list(SeuratCommand)`: Coerce a `SeuratCommand` to a list
- `show(SeuratCommand)`: Overview of a `SeuratCommand` object

`Simplify` *Simplify Geometry*

Description

Simplify Geometry

Usage

```
Simplify(coords, tol, topologyPreserve = TRUE)

## S3 method for class 'Spatial'
Simplify(coords, tol, topologyPreserve = TRUE)
```

Arguments

`coords`	...
`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

...

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` Key for the image

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`

`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")
```

`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

```
Tool(object = pbmc_small)

## Not run:
sample.tool.output <- matrix(data = rnorm(n = 16), nrow = 4)
# must be run from within a function
Tool(object = pbmc_small) <- sample.tool.output

## End(Not run)
```

`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 request, 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