Package 'Seurat'

April 16, 2020

Version 3.1.5

Date 2020-04-14

Title Tools for Single Cell Genomics

Description A toolkit for quality control, analysis, and exploration of single cell RNA sequencing data. 'Seurat' aims to enable users to identify and interpret sources of heterogeneity from single cell transcriptomic measurements, and to integrate diverse types of single cell data. 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. Please note: SDMTools is available is available from the CRAN archives with install.packages(``https://cran.rstudio.com//src/contrib/Archive/SDMTools/SDMTools_1.1-221.2.tar.gz", repos = NULL); it is not in the standard repositories.

URL http://www.satijalab.org/seurat, https://github.com/satijalab/seurat

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

Additional_repositories https://mojaveazure.github.io/loomR

Depends R (>= 3.4.0), methods,

Imports ape, cluster, cowplot, fitdistrplus, future, future.apply, ggplot2 (>= 3.0.0), ggrepel, ggridges, graphics, grDevices, grid, httr, ica, igraph, irlba, KernSmooth, leiden (>= 0.3.1), lmtest, MASS, Matrix (>= 1.2-14), patchwork, pbapply, plotly, png, RANN, RColorBrewer, Rcpp, RcppAnnoy, reticulate, rlang, ROCR, rsvd, Rtsne, scales, sctransform (>= 0.2.0), stats, tools, tsne, utils, uwot (>= 0.1.5)

LinkingTo Rcpp (>= 0.11.0), RcppEigen, RcppProgress

License GPL-3 | file LICENSE

LazyData true

Collate 'RcppExports.R' 'generics.R' 'clustering.R' 'visualization.R' 'convenience.R' 'data.R' 'differential_expression.R' 'dimensional_reduction.R' 'integration.R' 'objects.R' 'preprocessing.R' 'tree.R' 'utilities.R' 'zzz.R'

RoxygenNote 7.1.0

Encoding UTF-8

biocViews

Suggests loomR, SDMTools, testthat, hdf5r, S4Vectors, SummarizedExperiment, SingleCellExperiment, MAST, DESeq2, BiocGenerics, GenomicRanges, GenomeInfoDb, IRanges, rtracklayer, monocle, Biobase, VGAM, limma, metap

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 <nygcSatijalab@nygenome.org>

Repository CRAN

Date/Publication 2020-04-16 11:50:08 UTC

R topics documented:

Seurat-package *Seurat package*

Description

Tools for single-cell genomics

Details

Tools for single-cell genomics

Package options

Seurat uses the following [options()] to configure behaviour:

`Seurat.memsafe` global option to call gc() after many operations. This can be helpful in cleaning up the memory status of the R session and prevent use of swap space. However, it does add to the computational overhead and setting to FALSE can speed things up if you're working in an environment where RAM availabiliy is not a concern.

`Seurat.warn.umap.uwot` Show warning about the default backend for `RunUMAP` changing from Python UMAP via reticulate to UWOT

`Seurat.checkdots` For functions that have ... as a parameter, this controls the behavior when an item isn't used. Can be one of warn, stop, or silent.

`Seurat.limma.wilcox.msg` Show message about more efficient Wilcoxon Rank Sum test available via the limma package

`Seurat.warn.vlnplot.split` Show message about changes to default behavior of split/multi violin plots

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)

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

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

Arguments

x, object	An object
i, col.name	Name to store metadata or object as
j	Ignored
...	Arguments passed to other methods
value, metadata	Metadata or object to add

Value

An object with metadata or and object 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[[]])
```

AddModuleScore *Calculate module scores for feature expression programs in single cells*

Description

Calculate the average expression levels of each program (cluster) on single cell level, subtracted by the aggregated expression of control feature sets. All analyzed features are binned based on averaged expression, and the control features are randomly selected from each bin.

Usage

```
AddModuleScore(
  object,
  features,
  pool = NULL,
  nbin = 24,
  ctrl = 100,
  k = FALSE,
  assay = NULL,
  name = "Cluster",
  seed = 1,
  search = FALSE,
  ...
)
```

Arguments

`object`	Seurat object
`features`	Feature expression programs in list
`pool`	List of features to check expression levels agains, defaults to `rownames(x = object)`
`nbin`	Number of bins of aggregate expression levels for all analyzed features
`ctrl`	Number of control features selected from the same bin per analyzed feature
`k`	Use feature clusters returned from DoKMeans
`assay`	Name of assay to use
`name`	Name for the expression programs
`seed`	Set a random seed. If NULL, seed is not set.
`search`	Search for symbol synonyms for features in `features` that don't match features in `object`? Searches the HGNC's gene names database; see `UpdateSymbolList` for more details
`...`	Extra parameters passed to `UpdateSymbolList`

Value

Returns a Seurat object with module scores added to object meta data

References

Tirosh et al, Science (2016)

Examples

```
## Not run:
cd_features <- list(c(
  'CD79B',
  'CD79A',
  'CD19',
  'CD180',
  'CD200',
  'CD3D',
  'CD2',
  'CD3E',
  'CD7',
  'CD8A',
  'CD14',
  'CD1C',
  'CD68',
  'CD9',
  'CD247'
))
pbmc_small <- AddModuleScore(
  object = pbmc_small,
  features = cd_features,
  ctrl = 5,
  name = 'CD_Features'
)
head(x = pbmc_small[])

## End(Not run)
```

`ALRAChooseKPlot` *ALRA Approximate Rank Selection Plot*

Description

Plots the results of the approximate rank selection process for ALRA.

Usage

```
ALRAChooseKPlot(object, start = 0, combine = TRUE)
```

Arguments

`object`	Seurat object
`start`	Index to start plotting singular value spacings from. The transition from "signal" to "noise" in the is hard to see because the first singular value spacings are so large. Nicer visualizations result from skipping the first few. If set to 0 (default) starts from k/2.
`combine`	Combine plots into a single `patchwork`ed ggplot object. If `FALSE`, return a list of ggplot objects

Value

A list of 3 patchworked ggplot objects splotting the singular values, the spacings of the singular values, and the p-values of the singular values.

Note

ALRAChooseKPlot and associated functions are being moved to SeuratWrappers; for more information on SeuratWrappers, please see https://github.com/satijalab/seurat-wrappers

Author(s)

Jun Zhao, George Linderman

See Also

RunALRA

AnchorSet-class *The AnchorSet Class*

Description

The AnchorSet class is an intermediate data storage class that stores the anchors and other related information needed for performing downstream analyses - namely data integration (IntegrateData) and data transfer (TransferData).

Slots

`object.list` List of objects used to create anchors

`reference.cells` List of cell names in the reference dataset - needed when performing data transfer.

`reference.objects` Position of reference object/s in object.list

`query.cells` List of cell names in the query dataset - needed when performing data transfer

`anchors` The anchor matrix. This contains the cell indices of both anchor pair cells, the anchor score, and the index of the original dataset in the object.list for cell1 and cell2 of the anchor.

`offsets` The offsets used to enable cell look up in downstream functions

`anchor.features` The features used when performing anchor finding.

`command` Store log of parameters that were used

as.CellDataSet *Convert objects to CellDataSet objects*

Description

Convert objects to CellDataSet objects

Usage

```
as.CellDataSet(x, ...)

## S3 method for class 'Seurat'
as.CellDataSet(x, assay = NULL, reduction = NULL, ...)
```

Arguments

x	An object to convert to class CellDataSet
...	Arguments passed to other methods
assay	Assay to convert
reduction	Name of DimReduc to set to main reducedDim in cds

as.Graph *Convert a matrix (or Matrix) to the Graph class.*

Description

Convert a matrix (or Matrix) to the Graph class.

Usage

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

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

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

Arguments

x	The matrix to convert
...	Arguments passed to other methods (ignored for now)

Examples

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

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

---

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

Description

Coerce a SeuratCommand to a list

Usage

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

Arguments

x	object to be coerced or tested.
complete	Include slots besides just parameters (eg. call string, name, timestamp)
...	objects, possibly named.

Value

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

---

as.loom *Convert objects to loom objects*

Description

Convert objects to loom objects

Usage

```
as.loom(x, ...)

## S3 method for class 'Seurat'
as.loom(
  x,
  assay = NULL,
  filename = file.path(getwd(), paste0(Project(object = x), ".loom")),
  max.size = "400mb",
  chunk.dims = NULL,
  chunk.size = NULL,
  overwrite = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	An object to convert to class loom
...	Ignored for now
assay	Assay to store in loom file
filename	The name of the new loom file
max.size	Set maximum chunk size in terms of memory usage, unused if chunk.dims is set; may pass a character string (eg. 3gb, 1200mb) or exact value in bytes
chunk.dims	Matrix chunk dimensions; auto-determined by default
chunk.size	Maximum number of cells read/written to disk at once; auto-determined by default
overwrite	Overwrite an already existing loom file?
verbose	Display a progress bar

Details

The Seurat method for as.loom will try to automatically fill in datasets based on data presence. For example, if an assay's scaled data slot isn't filled, then dimensional reduction and graph information will not be filled, since those depend on scaled data. The following is a list of how datasets will be filled

- counts will be stored in matrix
- Cell names will be stored in col_attrs/CellID; feature names will be stored in row_attrs/Gene
- data will be stored in layers/norm_data
- scale.data will be stored in layers/scale_data
- Cell-level metadata will be stored in col_attrs; all periods '.' in metadata will be replaced with underscores '_'
- Clustering information from Idents(object = x) will be stored in col_attrs/ClusterID and col_attrs/ClusterName for the numeric and string representation of the factor, respectively

- Feature-level metadata will be stored in `Feature_attrs`; all periods '.' in metadata will be replaced with underscores '_'
- Variable features, if set, will be stored in `row_attrs/Selected`; features declared as variable will be stored as '1', others will be stored as '0'
- Dimensional reduction information for the assay provided will be stored in `col_attrs` for cell embeddings and `row_attrs` for feature loadings; datasets will be named as `name_type` where `name` is the name within the Seurat object and `type` is `cell_embeddings` or `feature_loadings`; if feature loadings have been projected for all features, then projected loadings will be stored instead and `type` will be `feature_loadings_projected`
- Nearest-neighbor graphs that start with the name of the assay will be stored in `col_graphs`
- Assay information will be stored as an HDF5 attribute called `assay` at the root level

See Also

create

Examples

```
## Not run:
lfile <- as.loom(x = pbmc_small)

## End(Not run)
```

`as.Seurat` *Convert objects to Seurat objects*

Description

Convert objects to Seurat objects

Usage

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

## S3 method for class 'CellDataSet'
as.Seurat(x, slot = "counts", assay = "RNA", verbose = TRUE, ...)

## S3 method for class 'loom'
as.Seurat(
  x,
  cells = "CellID",
  features = "Gene",
  normalized = NULL,
  scaled = NULL,
  assay = NULL,
  verbose = TRUE,
```

```
  ...
)

## S3 method for class 'SingleCellExperiment'
as.Seurat(
  x,
  counts = "counts",
  data = "logcounts",
  assay = "RNA",
  project = "SingleCellExperiment",
  ...
)
```

Arguments

x	An object to convert to class Seurat
...	Arguments passed to other methods
slot	Slot to store expression data as
assay	Name to store expression matrices as
verbose	Display progress updates
cells	The name of the dataset within col_attrs containing cell names
features	The name of the dataset within row_attrs containing feature names
normalized	The name of the dataset within layers containing the normalized expression matrix; pass /matrix (with preceeding forward slash) to store /matrix as normalized data
scaled	The name of the dataset within layers containing the scaled expression matrix
counts	name of the SingleCellExperiment assay to store as counts; set to NULL if only normalized data are present
data	name of the SingleCellExperiment assay to slot as data. Set to NULL if only counts are present
project	Project name for new Seurat object

Details

The loom method for as.Seurat will try to automatically fill in a Seurat object based on data presence. For example, if no normalized data is present, then scaled data, dimensional reduction informan, and neighbor graphs will not be pulled as these depend on normalized data. The following is a list of how the Seurat object will be constructed

- If no assay information is provided, will default to an assay name in a root-level HDF5 attribute called assay; if no attribute is present, will default to "RNA"
- Cell-level metadata will consist of all one-dimensional datasets in col_attrs **except** datasets named "ClusterID", "ClusterName", and whatever is passed to cells
- Identity classes will be set if either col_attrs/ClusterID or col_attrs/ClusterName are present; if both are present, then the values in col_attrs/ClusterID will set the order (numeric value of a factor) for values in col_attrs/ClusterName (charater value of a factor)

- Feature-level metadata will consist of all one-dimensional datasets in `row_attrs` **except** datasets named "Selected" and whatever is passed to `features`; any feature-level metadata named "variance_standardized", "variance_expected", or "dispersion_scaled" will have underscores "_" replaced with a period "."
- Variable features will be set if `row_attrs/Selected` exists and it is a numeric type
- If a dataset is passed to `normalized`, stored as a sparse matrix in `data`; if no dataset provided, `scaled` will be set to `NULL`
- If a dataset is passed to `scaled`, stored as a dense matrix in `scale.data`; all rows entirely consisting of `NA`s will be removed
- If a dataset is passed to `scaled`, dimensional reduction information will assembled from cell embedding information stored in `col_attrs`; cell embeddings will be pulled from two-dimensional datasets ending with "_cell_embeddings"; priority will be given to cell embeddings that have the name of `assay` in their name; feature loadings will be added from two-dimensional datasets in `row_attrs` that start with the name of the dimensional reduction and end with either "feature_loadings" or "feature_loadings_projected" (priority given to the latter)
- If a dataset is passed to `scaled`, neighbor graphs will be pulled from `col_graphs`, provided the name starts with the value of `assay`

Examples

```
## Not run:
lfile <- as.loom(x = pbmc_small)
pbmc <- as.Seurat(x = lfile)

## End(Not run)
```

---

as.SingleCellExperiment

Convert objects to SingleCellExperiment objects

Description

Convert objects to SingleCellExperiment objects

Usage

```
as.SingleCellExperiment(x, ...)

## S3 method for class 'Seurat'
as.SingleCellExperiment(x, assay = NULL, ...)
```

Arguments

x	An object to convert to class SingleCellExperiment
...	Arguments passed to other methods
assay	Assay to convert

---

as.sparse *Convert between data frames and sparse matrices*

Description

Convert between data frames and sparse matrices

Usage

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

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

## S3 method for class 'H5Group'
as.sparse(x, ...)

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

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

## S3 method for class 'Matrix'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...,
  stringsAsFactors = default.stringsAsFactors()
)
```

Arguments

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

optional — logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's **base** package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*,check.names = !optional). See also the make.names argument of the matrix method.

stringsAsFactors — logical: should the character vector be converted to a factor?

Value

as.sparse: A sparse representation of the input data

as.data.frame.Matrix: A data frame representation of the S4 Matrix

---

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 provenence

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

Assays *Pull Assays or assay names*

Description

Lists the names of Assay objects present in a Seurat object. If slot is provided, pulls specified Assay object.

Usage

```
Assays(object, slot = NULL)
```

Arguments

object	A Seurat object
slot	Name of Assay to return

Value

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

Examples

```
Assays(object = pbmc_small)
```

AugmentPlot *Augments ggplot2-based plot with a PNG image.*

Description

Creates "vector-friendly" plots. Does this by saving a copy of the plot as a PNG file, then adding the PNG image with annotation_raster to a blank plot of the same dimensions as plot. Please note: original legends and axes will be lost during augmentation.

Usage

```
AugmentPlot(plot, width = 10, height = 10, dpi = 100)
```

Arguments

plot	A ggplot object
width, height	Width and height of PNG version of plot
dpi	Plot resolution

Value

A ggplot object

Examples

```
## Not run:
plot <- DimPlot(object = pbmc_small)
AugmentPlot(plot = plot)

## End(Not run)
```

`AverageExpression` *Averaged feature expression by identity class*

Description

Returns expression for an 'average' single cell in each identity class

Usage

```
AverageExpression(
  object,
  assays = NULL,
  features = NULL,
  return.seurat = FALSE,
  add.ident = NULL,
  slot = "data",
  use.scale = FALSE,
  use.counts = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	Seurat object
`assays`	Which assays to use. Default is all assays
`features`	Features to analyze. Default is all features in the assay
`return.seurat`	Whether to return the data as a Seurat object. Default is FALSE
`add.ident`	Place an additional label on each cell prior to averaging (very useful if you want to observe cluster averages, separated by replicate, for example)
`slot`	Slot to use; will be overriden by `use.scale` and `use.counts`
`use.scale`	Use scaled values for feature expression
`use.counts`	Use count values for feature expression
`verbose`	Print messages and show progress bar
`...`	Arguments to be passed to methods such as `CreateSeuratObject`

Details

Output is in log-space when `return.seurat = TRUE`, otherwise it's in non-log space. Averaging is done in non-log space.

Value

Returns a matrix with genes as rows, identity classes as columns. If return.seurat is TRUE, returns an object of class Seurat.

Examples

```
head(AverageExpression(object = pbmc_small))
```

BarcodeInflectionsPlot

Plot the Barcode Distribution and Calculated Inflection Points

---

Description

This function plots the calculated inflection points derived from the barcode-rank distribution.

Usage

```
BarcodeInflectionsPlot(object)
```

Arguments

`object`	Seurat object

Details

See [CalculateBarcodeInflections()] to calculate inflection points and [SubsetByBarcodeInflections()] to subsequently subset the Seurat object.

Value

Returns a 'ggplot2' object showing the by-group inflection points and provided (or default) rank threshold values in grey.

Author(s)

Robert A. Amezquita, <robert.amezquita@fredhutch.org>

See Also

CalculateBarcodeInflections SubsetByBarcodeInflections

Examples

```
pbmc_small <- CalculateBarcodeInflections(pbmc_small, group.column = 'groups')
BarcodeInflectionsPlot(pbmc_small)
```

`BlackAndWhite` *Create a custom color palette*

Description

Creates a custom color palette based on low, middle, and high color values

Usage

```
BlackAndWhite(mid = NULL, k = 50)

BlueAndRed(k = 50)

CustomPalette(low = "white", high = "red", mid = NULL, k = 50)

PurpleAndYellow(k = 50)
```

Arguments

`mid`	middle color. Optional.
`k`	number of steps (colors levels) to include between low and high values
`low`	low color
`high`	high color

Value

A color palette for plotting

Examples

```
df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
plot(df, col = BlackAndWhite())

df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
plot(df, col = BlueAndRed())

myPalette <- CustomPalette()
myPalette

df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
plot(df, col = PurpleAndYellow())
```

`BuildClusterTree` *Phylogenetic Analysis of Identity Classes*

Description

Constructs a phylogenetic tree relating the 'average' cell from each identity class. Tree is estimated based on a distance matrix constructed in either gene expression space or PCA space.

Usage

```
BuildClusterTree(
  object,
  assay = NULL,
  features = NULL,
  dims = NULL,
  graph = NULL,
  slot = "data",
  reorder = FALSE,
  reorder.numeric = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	Seurat object
`assay`	Assay to use for the analysis.
`features`	Genes to use for the analysis. Default is the set of variable genes (`VariableFeatures(object = object)`)
`dims`	If set, tree is calculated in PCA space; overrides `features`
`graph`	If graph is passed, build tree based on graph connectivity between clusters; overrides `dims` and `features`
`slot`	Slot to use; will be overriden by `use.scale` and `use.counts`
`reorder`	Re-order identity classes (factor ordering), according to position on the tree. This groups similar classes together which can be helpful, for example, when drawing violin plots.
`reorder.numeric`	Re-order identity classes according to position on the tree, assigning a numeric value ('1' is the leftmost node)
`verbose`	Show progress updates

Details

Note that the tree is calculated for an 'average' cell, so gene expression or PC scores are averaged across all cells in an identity class before the tree is constructed.

Value

A Seurat object where the cluster tree can be accessed with Tool

Examples

```
pbmc_small
pbmc_small <- BuildClusterTree(object = pbmc_small)
Tool(object = pbmc_small, slot = 'BuildClusterTree')
```

CalculateBarcodeInflections

Calculate the Barcode Distribution Inflection

Description

This function calculates an adaptive inflection point ("knee") of the barcode distribution for each sample group. This is useful for determining a threshold for removing low-quality samples.

Usage

```
CalculateBarcodeInflections(
  object,
  barcode.column = "nCount_RNA",
  group.column = "orig.ident",
  threshold.low = NULL,
  threshold.high = NULL
)
```

Arguments

object	Seurat object
barcode.column	Column to use as proxy for barcodes ("nCount_RNA" by default)
group.column	Column to group by ("orig.ident" by default)
threshold.low	Ignore barcodes of rank below this threshold in inflection calculation
threshold.high	Ignore barcodes of rank above thisf threshold in inflection calculation

Details

The function operates by calculating the slope of the barcode number vs. rank distribution, and then finding the point at which the distribution changes most steeply (the "knee"). Of note, this calculation often must be restricted as to the range at which it performs, so ‘threshold‘ parameters are provided to restrict the range of the calculation based on the rank of the barcodes. [BarcodeInflectionsPlot()] is provided as a convenience function to visualize and test different thresholds and thus provide more sensical end results.

See [BarcodeInflectionsPlot()] to visualize the calculated inflection points and [SubsetByBarcodeInflections()] to subsequently subset the Seurat object.

Value

Returns Seurat object with a new list in the ‘tools‘ slot, ‘CalculateBarcodeInflections‘ with values:

* ‘barcode_distribution‘ - contains the full barcode distribution across the entire dataset * ‘inflection_points‘ - the calculated inflection points within the thresholds * ‘threshold_values‘ - the provided (or default) threshold values to search within for inflections * ‘cells_pass‘ - the cells that pass the inflection point calculation

Author(s)

Robert A. Amezquita, <robert.amezquita@fredhutch.org>

See Also

BarcodeInflectionsPlot SubsetByBarcodeInflections

Examples

```
CalculateBarcodeInflections(pbmc_small, group.column = 'groups')
```

CaseMatch *Match the case of character vectors*

Description

Match the case of character vectors

Usage

```
CaseMatch(search, match)
```

Arguments

search	A vector of search terms
match	A vector of characters whose case should be matched

Value

Values from search present in match with the case of match

Examples

```
cd_genes <- c('Cd79b', 'Cd19', 'Cd200')
CaseMatch(search = cd_genes, match = rownames(x = pbmc_small))
```

cc.genes *Cell cycle genes*

Description

A list of genes used in cell-cycle regression

Usage

```
cc.genes
```

Format

A list of two vectors

s.genes Genes associated with S-phase

g2m.genes Genes associated with G2M-phase

Source

http://science.sciencemag.org/content/352/6282/189

cc.genes.updated.2019 *Cell cycle genes: 2019 update*

Description

A list of genes used in cell-cycle regression, updated with 2019 symbols

Usage

```
cc.genes.updated.2019
```

Format

A list of two vectors

s.genes Genes associated with S-phase

g2m.genes Genes associated with G2M-phase

Updated symbols

The following symbols were updated from cc.genes

s.genes
- *MCM2*: *MCM7*
- *MLF1IP*: *CENPU*
- *RPA2*: *POLR1B*
- *BRIP1*: *MRPL36*

g2m.genes
- *FAM64A*: *PIMREG*
- *HN1*: *JPT1*

Source

http://science.sciencemag.org/content/352/6282/189

See Also

cc.genes

Examples

```
## Not run:
cc.genes.updated.2019 <- cc.genes
cc.genes.updated.2019$s.genes <- UpdateSymbolList(symbols = cc.genes.updated.2019$s.genes)
cc.genes.updated.2019$g2m.genes <- UpdateSymbolList(symbols = cc.genes.updated.2019$g2m.genes)

## End(Not run)
```

CellCycleScoring *Score cell cycle phases*

Description

Score cell cycle phases

Usage

```
CellCycleScoring(object, s.features, g2m.features, set.ident = FALSE, ...)
```

Arguments

object	A Seurat object
s.features	A vector of features associated with S phase
g2m.features	A vector of features associated with G2M phase
set.ident	If true, sets identity to phase assignments
...	Arguments to be passed to AddModuleScore Stashes old identities in 'old.ident'

Value

A Seurat object with the following columns added to object meta data: S.Score, G2M.Score, and Phase

See Also

AddModuleScore

Examples

```
## Not run:
# pbmc_small doesn't have any cell-cycle genes
# To run CellCycleScoring, please use a dataset with cell-cycle genes
# An example is available at http://satijalab.org/seurat/cell_cycle_vignette.html
pbmc_small <- CellCycleScoring(
  object = pbmc_small,
  g2m.features = cc.genes$g2m.genes,
  s.features = cc.genes$s.genes
)
head(x = pbmc_small@meta.data)

## End(Not run)
```

Cells *Get cells present in an object*

Description

Get cells present in an object

Usage

```
Cells(x)

## Default S3 method:
Cells(x)

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

Arguments

x	An object

Value

A vector of cell 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)
```

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

Value

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

Examples

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

`CellScatter` *Cell-cell scatter plot*

Description

Creates a plot of scatter plot of features across two single cells. Pearson correlation between the two cells is displayed above the plot.

Usage

```
CellScatter(
  object,
  cell1,
  cell2,
  features = NULL,
  highlight = NULL,
  cols = NULL,
  pt.size = 1,
  smooth = FALSE
)
```

Arguments

object	Seurat object
cell1	Cell 1 name
cell2	Cell 2 name
features	Features to plot (default, all features)
highlight	Features to highlight
cols	Colors to use for identity class plotting.
pt.size	Size of the points on the plot
smooth	Smooth the graph (similar to smoothScatter)

Value

A ggplot object

Examples

```
CellScatter(object = pbmc_small, cell1 = 'ATAGGAGAAACAGA', cell2 = 'CATCAGGATGCACA')
```

CellSelector *Cell selector*

Description

Select points on a scatterplot and get information about them

Usage

```
CellSelector(plot, object = NULL, ident = "SelectedCells", ...)

FeatureLocator(plot, ...)
```

Arguments

plot	A ggplot2 plot
object	An optional Seurat object; if passes, will return an object with the identities of selected cells set to ident
ident	An optional new identity class to assign the selected cells
...	Extra parameters, such as dark.theme, recolor, or smooth for using a dark theme, recoloring based on selected cells, or using a smooth scatterplot, respectively

Value

If object is NULL, the names of the points selected; otherwise, a Seurat object with the selected cells identity classes set to ident

See Also

locator ggplot_build pnt.in.poly DimPlot FeaturePlot

Examples

```
## Not run:
plot <- DimPlot(object = pbmc_small)
# Follow instructions in the terminal to select points
cells.located <- CellSelector(plot = plot)
cells.located
# Automatically set the identity class of selected cells and return a new Seurat object
pbmc_small <- CellSelector(plot = plot, object = pbmc_small, ident = 'SelectedCells')

## End(Not run)
```

CollapseEmbeddingOutliers

Move outliers towards center on dimension reduction plot

Description

Move outliers towards center on dimension reduction plot

Usage

```
CollapseEmbeddingOutliers(
  object,
  reduction = "umap",
  dims = 1:2,
  group.by = "ident",
  outlier.sd = 2,
  reduction.key = "UMAP_"
)
```

Arguments

object	Seurat object
reduction	Name of DimReduc to adjust
dims	Dimensions to visualize
group.by	Group (color) cells in different ways (for example, orig.ident)
outlier.sd	Controls the outlier distance
reduction.key	Key for DimReduc that is returned

Value

Returns a DimReduc object with the modified embeddings

Examples

```
## Not run:
pbmc_small <- FindClusters(pbmc_small, resolution = 1.1)
pbmc_small <- RunUMAP(pbmc_small, dims = 1:5)
DimPlot(pbmc_small, reduction = "umap")
pbmc_small[["umap_new"]] <- CollapseEmbeddingOutliers(pbmc_small,
    reduction = "umap", reduction.key = 'umap_', outlier.sd = 0.5)
DimPlot(pbmc_small, reduction = "umap_new")

## End(Not run)
```

CollapseSpeciesExpressionMatrix

Slim down a multi-species expression matrix, when only one species is primarily of interenst.

Description

Valuable for CITE-seq analyses, where we typically spike in rare populations of 'negative control' cells from a different species.

Usage

```
CollapseSpeciesExpressionMatrix(
  object,
  prefix = "HUMAN_",
  controls = "MOUSE_",
  ncontrols = 100
)
```

Arguments

object	A UMI count matrix. Should contain rownames that start with the ensuing arguments prefix.1 or prefix.2
prefix	The prefix denoting rownames for the species of interest. Default is "HUMAN_". These rownames will have this prefix removed in the returned matrix.
controls	The prefix denoting rownames for the species of 'negative control' cells. Default is "MOUSE_".
ncontrols	How many of the most highly expressed (average) negative control features (by default, 100 mouse genes), should be kept? All other rownames starting with prefix.2 are discarded.

Value

A UMI count matrix. Rownames that started with `prefix` have this prefix discarded. For rownames starting with `controls`, only the `ncontrols` most highly expressed features are kept, and the prefix is kept. All other rows are retained.

Examples

```
## Not run:
cbmc.rna.collapsed <- CollapseSpeciesExpressionMatrix(cbmc.rna)

## End(Not run)
```

ColorDimSplit *Color dimensional reduction plot by tree split*

Description

Returns a DimPlot colored based on whether the cells fall in clusters to the left or to the right of a node split in the cluster tree.

Usage

```
ColorDimSplit(
  object,
  node,
  left.color = "red",
  right.color = "blue",
  other.color = "grey50",
  ...
)
```

Arguments

object	Seurat object
node	Node in cluster tree on which to base the split
left.color	Color for the left side of the split
right.color	Color for the right side of the split
other.color	Color for all other cells
...	Arguments passed on to DimPlot

dims Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions

cells Vector of cells to plot (default is all cells)

cols Vector of colors, each color corresponds to an identity class. This may also be a single character or numeric value corresponding to a palette as specified by brewer.pal.info. By default, ggplot2 assigns colors. We also include a number of palettes from the pals package. See DiscretePalette for details.

pt.size Adjust point size for plotting

reduction Which dimensionality reduction to use. If not specified, first searches for umap, then tsne, then pca

group.by Name of one or more metadata columns to group (color) cells by (for example, orig.ident); pass 'ident' to group by identity class

split.by Name of a metadata column to split plot by; see FetchData for more details

shape.by If NULL, all points are circles (default). You can specify any cell attribute (that can be pulled with FetchData) allowing for both different colors and different shapes on cells

order Specify the order of plotting for the idents. This can be useful for crowded plots if points of interest are being buried. Provide either a full list of valid idents or a subset to be plotted last (on top)

label Whether to label the clusters

label.size Sets size of labels

repel Repel labels

cells.highlight A list of character or numeric vectors of cells to highlight. If only one group of cells desired, can simply pass a vector instead of a list. If set, colors selected cells to the color(s) in cols.highlight and other cells black (white if dark.theme = TRUE); will also resize to the size(s) passed to sizes.highlight

cols.highlight A vector of colors to highlight the cells as; will repeat to the length groups in cells.highlight

sizes.highlight Size of highlighted cells; will repeat to the length groups in cells.highlight

na.value Color value for NA points when using custom scale

ncol Number of columns for display when combining plots

combine Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects

Value

Returns a DimPlot

See Also

DimPlot

Examples

```
pbmc_small
pbmc_small <- BuildClusterTree(object = pbmc_small, verbose = FALSE)
PlotClusterTree(pbmc_small)
ColorDimSplit(pbmc_small, node = 5)
```

CombinePlots *Combine ggplot2-based plots into a single plot*

Description

Combine ggplot2-based plots into a single plot

Usage

```
CombinePlots(plots, ncol = NULL, legend = NULL, ...)
```

Arguments

plots	A list of gg objects
ncol	Number of columns
legend	Combine legends into a single legend choose from 'right' or 'bottom'; pass 'none' to remove legends, or NULL to leave legends as they are
...	Extra parameters passed to plot_grid

Value

A combined plot

Examples

```
pbmc_small[['group']] <- sample(
  x = c('g1', 'g2'),
  size = ncol(x = pbmc_small),
  replace = TRUE
)
plot1 <- FeaturePlot(
  object = pbmc_small,
  features = 'MS4A1',
```

```
  split.by = 'group'
)
plot2 <- FeaturePlot(
  object = pbmc_small,
  features = 'FCN1',
  split.by = 'group'
)
CombinePlots(
  plots = list(plot1, plot2),
  legend = 'none',
  nrow = length(x = unique(x = pbmc_small[['group', drop = TRUE]]))
)
```

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 paramter 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)
```

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.

Details

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

Examples

```
pbmc_raw <- read.table(
  file = system.file('extdata', 'pbmc_raw.txt', package = 'Seurat'),
  as.is = TRUE
)
pbmc_rna <- CreateAssayObject(counts = pbmc_raw)
pbmc_rna
```

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

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"
)
```

CreateGeneActivityMatrix

Convert a peak matrix to a gene activity matrix

Description

This function will take in a peak matrix and an annotation file (gtf) and collapse the peak matrix to a gene activity matrix. It makes the simplifying assumption that all counts in the gene body plus X kb up and or downstream should be attributed to that gene.

Usage

```
CreateGeneActivityMatrix(
  peak.matrix,
  annotation.file,
  seq.levels = c(1:22, "X", "Y"),
  include.body = TRUE,
  upstream = 2000,
  downstream = 0,
  keep.sparse = FALSE,
  verbose = TRUE
)
```

Arguments

peak.matrix	Matrix of peak counts
annotation.file	Path to GTF annotation file
seq.levels	Which seqlevels to keep (corresponds to chromosomes usually)
include.body	Include the gene body?
upstream	Number of bases upstream to consider
downstream	Number of bases downstream to consider
keep.sparse	Leave the matrix as a sparse matrix. Setting this option to TRUE will take much longer but will use less memory. This can be useful if you have a very large matrix that cannot fit into memory when converted to a dense form.
verbose	Print progress/messages

CreateSeuratObject *Create a Seurat object*

Description

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

Usage

```
CreateSeuratObject(
  counts,
  project = "SeuratProject",
  assay = "RNA",
  min.cells = 0,
  min.features = 0,
  names.field = 1,
  names.delim = "_",
  meta.data = NULL
)
```

Arguments

counts	Unnormalized data such as raw counts or TPMs
project	Sets the project name for the Seurat object.
assay	Name of the assay corresponding to the initial input data.
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.
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.

Details

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

```
pbmc_raw <- read.table(
  file = system.file('extdata', 'pbmc_raw.txt', package = 'Seurat'),
  as.is = TRUE
)
pbmc_small <- CreateSeuratObject(counts = pbmc_raw)
pbmc_small
```

CustomDistance *Run a custom distance function on an input data matrix*

Description

Run a custom distance function on an input data matrix

Usage

```
CustomDistance(my.mat, my.function, ...)
```

Arguments

my.mat	A matrix to calculate distance on
my.function	A function to calculate distance
...	Extra parameters to my.function

Value

A distance matrix

Author(s)

Jean Fan

Examples

```
# Define custom distance matrix
manhattan.distance <- function(x, y) return(sum(abs(x-y)))

input.data <- GetAssayData(pbmc_small, assay.type = "RNA", slot = "scale.data")
cell.manhattan.dist <- CustomDistance(input.data, manhattan.distance)
```

DefaultAssay *Get and set the default assay*

Description

Get and set the default assay

Usage

```
DefaultAssay(object, ...)

DefaultAssay(object, ...) <- value

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

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

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

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

## S3 method for class 'SeuratCommand'
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

The name of the default assay

An object with the new default assay

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)
```

DietSeurat *Slim down a Seurat object*

Description

Keep only certain aspects of the Seurat object. Can be useful in functions that utilize merge as it reduces the amount of data in the merge.

Usage

```
DietSeurat(
  object,
  counts = TRUE,
  data = TRUE,
  scale.data = FALSE,
  features = NULL,
  assays = NULL,
  dimreducs = NULL,
  graphs = NULL
)
```

Arguments

object	Seurat object
counts	Preserve the count matrices for the assays specified
data	Preserve the data slot for the assays specified
scale.data	Preserve the scale.data slot for the assays specified
features	Only keep a subset of features, defaults to all features
assays	Only keep a subset of assays specified here
dimreducs	Only keep a subset of DimReducs specified here (if NULL, remove all DimReducs)
graphs	Only keep a subset of Graphs specified here (if NULL, remove all Graphs)

DimHeatmap *Dimensional reduction heatmap*

Description

Draws a heatmap focusing on a principal component. Both cells and genes are sorted by their principal component scores. Allows for nice visualization of sources of heterogeneity in the dataset.

Usage

```
DimHeatmap(
  object,
  dims = 1,
  nfeatures = 30,
  cells = NULL,
  reduction = "pca",
  disp.min = -2.5,
  disp.max = NULL,
  balanced = TRUE,
  projected = FALSE,
  ncol = NULL,
  fast = TRUE,
  raster = TRUE,
  slot = "scale.data",
  assays = NULL,
  combine = TRUE
)

PCHeatmap(object, ...)
```

Arguments

object	Seurat object
dims	Dimensions to plot
nfeatures	Number of genes to plot
cells	A list of cells to plot. If numeric, just plots the top cells.
reduction	Which dimmensional reduction to use
disp.min	Minimum display value (all values below are clipped)
disp.max	Maximum display value (all values above are clipped); defaults to 2.5 if slot is 'scale.data', 6 otherwise
balanced	Plot an equal number of genes with both + and - scores.
projected	Use the full projected dimensional reduction
ncol	Number of columns to plot
fast	If true, use image to generate plots; faster than using ggplot2, but not customizable
raster	If true, plot with geom_raster, else use geom_tile. geom_raster may look blurry on some viewing applications such as Preview due to how the raster is interpolated. Set this to FALSE if you are encountering that issue (note that plots may take longer to produce/render).
slot	Data slot to use, choose from 'raw.data', 'data', or 'scale.data'
assays	A vector of assays to pull data from
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects
...	Extra parameters passed to DimHeatmap

Value

No return value by default. If using fast = FALSE, will return a patchworked ggplot object if combine = TRUE, otherwise returns a list of ggplot objects

See Also

image geom_raster

Examples

```
DimHeatmap(object = pbmc_small)
```

DimPlot *Dimensional reduction plot*

Description

Graphs the output of a dimensional reduction technique on a 2D scatter plot where each point is a cell and it's positioned based on the cell embeddings determined by the reduction technique. By default, cells are colored by their identity class (can be changed with the group.by parameter).

Usage

```
DimPlot(
  object,
  dims = c(1, 2),
  cells = NULL,
  cols = NULL,
  pt.size = NULL,
  reduction = NULL,
  group.by = NULL,
  split.by = NULL,
  shape.by = NULL,
  order = NULL,
  label = FALSE,
  label.size = 4,
  repel = FALSE,
  cells.highlight = NULL,
  cols.highlight = "#DE2D26",
  sizes.highlight = 1,
  na.value = "grey50",
  ncol = NULL,
  combine = TRUE
)

PCAPlot(object, ...)
```

```
TSNEPlot(object, ...)

UMAPPlot(object, ...)
```

Arguments

object	Seurat object
dims	Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions
cells	Vector of cells to plot (default is all cells)
cols	Vector of colors, each color corresponds to an identity class. This may also be a single character or numeric value corresponding to a palette as specified by brewer.pal.info. By default, ggplot2 assigns colors. We also include a number of palettes from the pals package. See DiscretePalette for details.
pt.size	Adjust point size for plotting
reduction	Which dimensionality reduction to use. If not specified, first searches for umap, then tsne, then pca
group.by	Name of one or more metadata columns to group (color) cells by (for example, orig.ident); pass 'ident' to group by identity class
split.by	Name of a metadata column to split plot by; see FetchData for more details
shape.by	If NULL, all points are circles (default). You can specify any cell attribute (that can be pulled with FetchData) allowing for both different colors and different shapes on cells
order	Specify the order of plotting for the idents. This can be useful for crowded plots if points of interest are being buried. Provide either a full list of valid idents or a subset to be plotted last (on top)
label	Whether to label the clusters
label.size	Sets size of labels
repel	Repel labels
cells.highlight	A list of character or numeric vectors of cells to highlight. If only one group of cells desired, can simply pass a vector instead of a list. If set, colors selected cells to the color(s) in cols.highlight and other cells black (white if dark.theme = TRUE); will also resize to the size(s) passed to sizes.highlight
cols.highlight	A vector of colors to highlight the cells as; will repeat to the length groups in cells.highlight
sizes.highlight	Size of highlighted cells; will repeat to the length groups in cells.highlight
na.value	Color value for NA points when using custom scale
ncol	Number of columns for display when combining plots
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects
...	Extra parameters passed to DimPlot

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

Note

For the old do.hover and do.identify functionality, please see HoverLocator and CellSelector, respectively.

See Also

FeaturePlot HoverLocator CellSelector FetchData

Examples

```
DimPlot(object = pbmc_small)
DimPlot(object = pbmc_small, split.by = 'ident')
```

DimReduc-class *The Dimmensional 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 alphanumerics 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)

DiscretePalette *Discrete colour palettes from the pals package*

Description

These are included here because pals depends on a number of compiled packages, and this can lead to increases in run time for Travis, and generally should be avoided when possible.

Usage

```
DiscretePalette(n, palette = NULL)
```

Arguments

n	Number of colours to be generated.
palette	Options are "alphabet", "alphabet2", "glasbey", "polychrome", and "stepped". Can be omitted and the function will use the one based on the requested n.

Details

These palettes are a much better default for data with many classes than the default ggplot2 palette.

Many thanks to Kevin Wright for writing the pals package.

Taken from the pals package (Licence: GPL-3). https://cran.r-project.org/package=pals
Credit: Kevin Wright

Value

A vector of colors

DoHeatmap *Feature expression heatmap*

Description

Draws a heatmap of single cell feature expression.

Usage

```
DoHeatmap(
  object,
  features = NULL,
  cells = NULL,
  group.by = "ident",
  group.bar = TRUE,
  group.colors = NULL,
```

```
  disp.min = -2.5,
  disp.max = NULL,
  slot = "scale.data",
  assay = NULL,
  label = TRUE,
  size = 5.5,
  hjust = 0,
  angle = 45,
  raster = TRUE,
  draw.lines = TRUE,
  lines.width = NULL,
  group.bar.height = 0.02,
  combine = TRUE
)
```

Arguments

object	Seurat object
features	A vector of features to plot, defaults to VariableFeatures(object = object)
cells	A vector of cells to plot
group.by	A vector of variables to group cells by; pass 'ident' to group by cell identity classes
group.bar	Add a color bar showing group status for cells
group.colors	Colors to use for the color bar
disp.min	Minimum display value (all values below are clipped)
disp.max	Maximum display value (all values above are clipped); defaults to 2.5 if slot is 'scale.data', 6 otherwise
slot	Data slot to use, choose from 'raw.data', 'data', or 'scale.data'
assay	Assay to pull from
label	Label the cell identies above the color bar
size	Size of text above color bar
hjust	Horizontal justification of text above color bar
angle	Angle of text above color bar
raster	If true, plot with geom_raster, else use geom_tile. geom_raster may look blurry on some viewing applications such as Preview due to how the raster is interpolated. Set this to FALSE if you are encountering that issue (note that plots may take longer to produce/render).
draw.lines	Include white lines to separate the groups
lines.width	Integer number to adjust the width of the separating white lines. Corresponds to the number of "cells" between each group.
group.bar.height	Scale the height of the color bar
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

Examples

```
DoHeatmap(object = pbmc_small)
```

DotPlot *Dot plot visualization*

Description

Intuitive way of visualizing how feature expression changes across different identity classes (clusters). The size of the dot encodes the percentage of cells within a class, while the color encodes the AverageExpression level across all cells within a class (blue is high).

Usage

```
DotPlot(
  object,
  assay = NULL,
  features,
  cols = c("lightgrey", "blue"),
  col.min = -2.5,
  col.max = 2.5,
  dot.min = 0,
  dot.scale = 6,
  group.by = NULL,
  split.by = NULL,
  scale = TRUE,
  scale.by = "radius",
  scale.min = NA,
  scale.max = NA
)
```

Arguments

object	Seurat object
assay	Name of assay to use, defaults to the active assay
features	Input vector of features
cols	Colors to plot, can pass a single character giving the name of a palette from RColorBrewer::brewer.pal.info
col.min	Minimum scaled average expression threshold (everything smaller will be set to this)

col.max	Maximum scaled average expression threshold (everything larger will be set to this)
dot.min	The fraction of cells at which to draw the smallest dot (default is 0). All cell groups with less than this expressing the given gene will have no dot drawn.
dot.scale	Scale the size of the points, similar to cex
group.by	Factor to group the cells by
split.by	Factor to split the groups by (replicates the functionality of the old SplitDotPlotGG); see FetchData for more details
scale	Determine whether the data is scaled, TRUE for default
scale.by	Scale the size of the points by 'size' or by 'radius'
scale.min	Set lower limit for scaling, use NA for default
scale.max	Set upper limit for scaling, use NA for default

Value

A ggplot object

See Also

RColorBrewer::brewer.pal.info

Examples

```
cd_genes <- c("CD247", "CD3E", "CD9")
DotPlot(object = pbmc_small, features = cd_genes)
pbmc_small[['groups']] <- sample(x = c('g1', 'g2'), size = ncol(x = pbmc_small), replace = TRUE)
DotPlot(object = pbmc_small, features = cd_genes, split.by = 'groups')
```

ElbowPlot *Quickly Pick Relevant Dimensions*

Description

Plots the standard deviations (or approximate singular values if running PCAFast) of the principle components for easy identification of an elbow in the graph. This elbow often corresponds well with the significant dims and is much faster to run than Jackstraw

Usage

```
ElbowPlot(object, ndims = 20, reduction = "pca")
```

Arguments

object	Seurat object
ndims	Number of dimensions to plot standard deviation for
reduction	Reduction technique to plot standard deviation for

Value

A ggplot object

Examples

```
ElbowPlot(object = pbmc_small)
```

`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

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]
```

ExpMean *Calculate the mean of logged values*

Description

Calculate mean of logged values in non-log space (return answer in log-space)

Usage

```
ExpMean(x, ...)
```

Arguments

x	A vector of values
...	Other arguments (not used)

Value

Returns the mean in log-space

Examples

```
ExpMean(x = c(1, 2, 3))
```

ExportToCellbrowser *Export Seurat object for UCSC cell browser*

Description

Export Seurat object for UCSC cell browser

Usage

```
ExportToCellbrowser(
  object,
  dir,
  dataset.name = Project(object = object),
  reductions = "tsne",
  markers.file = NULL,
  cluster.field = "Cluster",
  cb.dir = NULL,
  port = NULL,
  skip.expr.matrix = FALSE,
  skip.metadata = FALSE,
  skip.reductions = FALSE,
  ...
)
```

Arguments

object	Seurat object
dir	path to directory where to save exported files. These are: exprMatrix.tsv, tsne.coords.tsv, meta.tsv, markers.tsv and a default cellbrowser.conf
dataset.name	name of the dataset. Defaults to Seurat project name
reductions	vector of reduction names to export
markers.file	path to file with marker genes
cluster.field	name of the metadata field containing cell cluster
cb.dir	path to directory where to create UCSC cellbrowser static website content root, e.g. an index.html, .json files, etc. These files can be copied to any webserver. If this is specified, the cellbrowser package has to be accessible from R via reticulate.
port	on which port to run UCSC cellbrowser webserver after export
skip.expr.matrix	whether to skip exporting expression matrix
skip.metadata	whether to skip exporting metadata
skip.reductions	whether to skip exporting reductions
...	specifies the metadata fields to export. To supply field with human readable name, pass name as field="name" parameter.

Value

This function exports Seurat object as a set of tsv files to dir directory, copying the markers.file if it is passed. It also creates the default cellbrowser.conf in the directory. This directory could be read by cbBuild to create a static website viewer for the dataset. If cb.dir parameter is passed, the function runs cbBuild (if it is installed) to create this static website in cb.dir directory. If port parameter is passed, it also runs the webserver for that directory and opens a browser.

Author(s)

Maximilian Haeussler, Nikolay Markov

Examples

```
## Not run:
ExportToCellbrowser(object = pbmc_small, dataset.name = "PBMC", dir = "out")

## End(Not run)
```

ExpSD — *Calculate the standard deviation of logged values*

Description

Calculate SD of logged values in non-log space (return answer in log-space)

Usage

```
ExpSD(x)
```

Arguments

x	A vector of values

Value

Returns the standard deviation in log-space

Examples

```
ExpSD(x = c(1, 2, 3))
```

ExpVar — *Calculate the variance of logged values*

Description

Calculate variance of logged values in non-log space (return answer in log-space)

Usage

```
ExpVar(x)
```

Arguments

x	A vector of values

Value

Returns the variance in log-space

Examples

```
ExpVar(x = c(1, 2, 3))
```

FeaturePlot *Visualize 'features' on a dimensional reduction plot*

Description

Colors single cells on a dimensional reduction plot according to a 'feature' (i.e. gene expression, PC scores, number of genes detected, etc.)

Usage

```
FeaturePlot(
  object,
  features,
  dims = c(1, 2),
  cells = NULL,
  cols = if (blend) {     c("lightgrey", "#ff0000", "#00ff00") } else {
    c("lightgrey", "blue") },
  pt.size = NULL,
  order = FALSE,
  min.cutoff = NA,
  max.cutoff = NA,
  reduction = NULL,
  split.by = NULL,
  shape.by = NULL,
  slot = "data",
  blend = FALSE,
  blend.threshold = 0.5,
  label = FALSE,
  label.size = 4,
  repel = FALSE,
  ncol = NULL,
  coord.fixed = FALSE,
  by.col = TRUE,
  sort.cell = NULL,
  combine = TRUE
)
```

Arguments

object	Seurat object
features	Vector of features to plot. Features can come from: • An Assay feature (e.g. a gene name - "MS4A1") • A column name from meta.data (e.g. mitochondrial percentage - "percent.mito") • A column name from a DimReduc object corresponding to the cell embedding values (e.g. the PC 1 scores - "PC_1")

`dims` Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions

`cells` Vector of cells to plot (default is all cells)

`cols` The two colors to form the gradient over. Provide as string vector with the first color corresponding to low values, the second to high. Also accepts a Brewer color scale or vector of colors. Note: this will bin the data into number of colors provided. When blend is `TRUE`, takes anywhere from 1-3 colors:

1 color: Treated as color for double-negatives, will use default colors 2 and 3 for per-feature expression

2 colors: Treated as colors for per-feature expression, will use default color 1 for double-negatives

3+ colors: First color used for double-negatives, colors 2 and 3 used for per-feature expression, all others ignored

`pt.size` Adjust point size for plotting

`order` Boolean determining whether to plot cells in order of expression. Can be useful if cells expressing given feature are getting buried.

`min.cutoff, max.cutoff`
Vector of minimum and maximum cutoff values for each feature, may specify quantile in the form of 'q##' where '##' is the quantile (eg, 'q1', 'q10')

`reduction` Which dimensionality reduction to use. If not specified, first searches for umap, then tsne, then pca

`split.by` A factor in object metadata to split the feature plot by, pass 'ident' to split by cell identity'; similar to the old `FeatureHeatmap`

`shape.by` If NULL, all points are circles (default). You can specify any cell attribute (that can be pulled with FetchData) allowing for both different colors and different shapes on cells

`slot` Which slot to pull expression data from?

`blend` Scale and blend expression values to visualize coexpression of two features

`blend.threshold`
The color cutoff from weak signal to strong signal; ranges from 0 to 1.

`label` Whether to label the clusters

`label.size` Sets size of labels

`repel` Repel labels

`ncol` Number of columns to combine multiple feature plots to, ignored if `split.by` is not `NULL`

`coord.fixed` Plot cartesian coordinates with fixed aspect ratio

`by.col` If splitting by a factor, plot the splits per column with the features as rows; ignored if `blend = TRUE`

`sort.cell` Redundant with `order`. This argument is being deprecated. Please use `order` instead.

`combine` Combine plots into a single `patchwork`ed ggplot object. If `FALSE`, return a list of ggplot objects

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

Note

For the old do.hover and do.identify functionality, please see HoverLocator and CellSelector, respectively.

See Also

DimPlot HoverLocator CellSelector

Examples

```
FeaturePlot(object = pbmc_small, features = 'PC_1')
```

FeatureScatter *Scatter plot of single cell data*

Description

Creates a scatter plot of two features (typically feature expression), across a set of single cells. Cells are colored by their identity class. Pearson correlation between the two features is displayed above the plot.

Usage

```
FeatureScatter(
  object,
  feature1,
  feature2,
  cells = NULL,
  group.by = NULL,
  cols = NULL,
  pt.size = 1,
  shape.by = NULL,
  span = NULL,
  smooth = FALSE,
  combine = TRUE,
  slot = "data"
)
```

Arguments

object	Seurat object
feature1	First feature to plot. Typically feature expression but can also be metrics, PC scores, etc. - anything that can be retreived with FetchData
feature2	Second feature to plot.
cells	Cells to include on the scatter plot.
group.by	Name of one or more metadata columns to group (color) cells by (for example, orig.ident); pass 'ident' to group by identity class
cols	Colors to use for identity class plotting.
pt.size	Size of the points on the plot
shape.by	Ignored for now
span	Spline span in loess function call, if NULL, no spline added
smooth	Smooth the graph (similar to smoothScatter)
combine	Combine plots into a single patchworked
slot	Slot to pull data from, should be one of 'counts', 'data', or 'scale.data'

Value

A ggplot object

Examples

```
FeatureScatter(object = pbmc_small, feature1 = 'CD9', feature2 = 'CD3E')
```

FetchData *Access cellular data*

Description

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

Usage

```
FetchData(object, vars, cells = NULL, slot = "data")
```

Arguments

object	Seurat object
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')))
```

FindAllMarkers *Gene expression markers for all identity classes*

Description

Finds markers (differentially expressed genes) for each of the identity classes in a dataset

Usage

```
FindAllMarkers(
  object,
  assay = NULL,
  features = NULL,
  logfc.threshold = 0.25,
  test.use = "wilcox",
  slot = "data",
  min.pct = 0.1,
  min.diff.pct = -Inf,
  node = NULL,
  verbose = TRUE,
  only.pos = FALSE,
  max.cells.per.ident = Inf,
  random.seed = 1,
  latent.vars = NULL,
  min.cells.feature = 3,
  min.cells.group = 3,
  pseudocount.use = 1,
  return.thresh = 0.01,
  ...
)
```

Arguments

object	An object
assay	Assay to use in differential expression testing
features	Genes to test. Default is to use all genes

`logfc.threshold`

Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells. Default is 0.25 Increasing logfc.threshold speeds up the function, but can miss weaker signals.

`test.use` Denotes which test to use. Available options are:

- "wilcox" : Identifies differentially expressed genes between two groups of cells using a Wilcoxon Rank Sum test (default)
- "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc" : Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
- "t" : Identify differentially expressed genes between two groups of cells using the Student's t-test.
- "negbinom" : Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson" : Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR" : Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST" : Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2" : Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014).This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/D

`slot` Slot to pull data from; note that if `test.use` is "negbinom", "poisson", or "DESeq2", `slot` will be set to "counts"

`min.pct` only test genes that are detected in a minimum fraction of min.pct cells in either of the two populations. Meant to speed up the function by not testing genes that are very infrequently expressed. Default is 0.1

`min.diff.pct` only test genes that show a minimum difference in the fraction of detection between the two groups. Set to -Inf by default

node	A node to find markers for and all its children; requires BuildClusterTree to have been run previously; replaces FindAllMarkersNode
verbose	Print a progress bar once expression testing begins
only.pos	Only return positive markers (FALSE by default)
max.cells.per.ident	Down sample each identity class to a max number. Default is no downsampling. Not activated by default (set to Inf)
random.seed	Random seed for downsampling
latent.vars	Variables to test, used only when test.use is one of 'LR', 'negbinom', 'poisson', or 'MAST'
min.cells.feature	Minimum number of cells expressing the feature in at least one of the two groups, currently only used for poisson and negative binomial tests
min.cells.group	Minimum number of cells in one of the groups
pseudocount.use	Pseudocount to add to averaged expression values when calculating logFC. 1 by default.
return.thresh	Only return markers that have a p-value < return.thresh, or a power > return.thresh (if the test is ROC)
...	Arguments passed to other methods and to specific DE methods

Value

Matrix containing a ranked list of putative markers, and associated statistics (p-values, ROC score, etc.)

Examples

```
# Find markers for all clusters
all.markers <- FindAllMarkers(object = pbmc_small)
head(x = all.markers)
## Not run:
# Pass a value to node as a replacement for FindAllMarkersNode
pbmc_small <- BuildClusterTree(object = pbmc_small)
all.markers <- FindAllMarkers(object = pbmc_small, node = 4)
head(x = all.markers)

## End(Not run)
```

FindClusters *Cluster Determination*

Description

Identify clusters of cells by a shared nearest neighbor (SNN) modularity optimization based clustering algorithm. First calculate k-nearest neighbors and construct the SNN graph. Then optimize the modularity function to determine clusters. For a full description of the algorithms, see Waltman and van Eck (2013) *The European Physical Journal B*. Thanks to Nigel Delaney (evolvedmicrobe@github) for the rewrite of the Java modularity optimizer code in Rcpp!

Usage

```
FindClusters(object, ...)

## Default S3 method:
FindClusters(
  object,
  modularity.fxn = 1,
  initial.membership = NULL,
  weights = NULL,
  node.sizes = NULL,
  resolution = 0.8,
  method = "matrix",
  algorithm = 1,
  n.start = 10,
  n.iter = 10,
  random.seed = 0,
  group.singletons = TRUE,
  temp.file.location = NULL,
  edge.file.name = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
FindClusters(
  object,
  graph.name = NULL,
  modularity.fxn = 1,
  initial.membership = NULL,
  weights = NULL,
  node.sizes = NULL,
  resolution = 0.8,
  method = "matrix",
  algorithm = 1,
  n.start = 10,
```

```
  n.iter = 10,
  random.seed = 0,
  group.singletons = TRUE,
  temp.file.location = NULL,
  edge.file.name = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
modularity.fxn	Modularity function (1 = standard; 2 = alternative).
initial.membership, weights, node.sizes	Parameters to pass to the Python leidenalg function.
resolution	Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.
method	Method for running leiden (defaults to matrix which is fast for small datasets). Enable method = "igraph" to avoid casting large data to a dense matrix.
algorithm	Algorithm for modularity optimization (1 = original Louvain algorithm; 2 = Louvain algorithm with multilevel refinement; 3 = SLM algorithm; 4 = Leiden algorithm). Leiden requires the leidenalg python.
n.start	Number of random starts.
n.iter	Maximal number of iterations per random start.
random.seed	Seed of the random number generator.
group.singletons	Group singletons into nearest cluster. If FALSE, assign all singletons to a "singleton" group
temp.file.location	Directory where intermediate files will be written. Specify the ABSOLUTE path.
edge.file.name	Edge file to use as input for modularity optimizer jar.
verbose	Print output
graph.name	Name of graph to use for the clustering algorithm

Details

To run Leiden algorithm, you must first install the leidenalg python package (e.g. via pip install leidenalg), see Traag et al (2018).

Value

Returns a Seurat object where the idents have been updated with new cluster info; latest clustering results will be stored in object metadata under 'seurat_clusters'. Note that 'seurat_clusters' will be overwritten everytime FindClusters is run

`FindConservedMarkers` *Finds markers that are conserved between the groups*

Description

Finds markers that are conserved between the groups

Usage

```
FindConservedMarkers(
  object,
  ident.1,
  ident.2 = NULL,
  grouping.var,
  assay = "RNA",
  slot = "data",
  meta.method = metap::minimump,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	An object
`ident.1`	Identity class to define markers for
`ident.2`	A second identity class for comparison. If NULL (default) - use all other cells for comparison.
`grouping.var`	grouping variable
`assay`	of assay to fetch data for (default is RNA)
`slot`	Slot to pull data from; note that if `test.use` is "negbinom", "poisson", or "DESeq2", `slot` will be set to "counts"
`meta.method`	method for combining p-values. Should be a function from the metap package (NOTE: pass the function, not a string)
`verbose`	Print a progress bar once expression testing begins
`...`	parameters to pass to FindMarkers

Value

data.frame containing a ranked list of putative conserved markers, and associated statistics (p-values within each group and a combined p-value (such as Fishers combined p-value or others from the metap package), percentage of cells expressing the marker, average differences). Name of group is appended to each associated output column (e.g. CTRL_p_val). If only one group is tested in the grouping.var, max and combined p-values are not returned.

Examples

```
## Not run:
pbmc_small
# Create a simulated grouping variable
pbmc_small[['groups']] <- sample(x = c('g1', 'g2'), size = ncol(x = pbmc_small), replace = TRUE)
FindConservedMarkers(pbmc_small, ident.1 = 0, ident.2 = 1, grouping.var = "groups")

## End(Not run)
```

`FindIntegrationAnchors` *Find integration anchors*

Description

Find a set of anchors between a list of `Seurat` objects. These anchors can later be used to integrate the objects using the `IntegrateData` function.

Usage

```
FindIntegrationAnchors(
  object.list = NULL,
  assay = NULL,
  reference = NULL,
  anchor.features = 2000,
  scale = TRUE,
  normalization.method = c("LogNormalize", "SCT"),
  sct.clip.range = NULL,
  reduction = c("cca", "rpca"),
  l2.norm = TRUE,
  dims = 1:30,
  k.anchor = 5,
  k.filter = 200,
  k.score = 30,
  max.features = 200,
  nn.method = "rann",
  eps = 0,
  verbose = TRUE
)
```

Arguments

`object.list`	A list of `Seurat` objects between which to find anchors for downstream integration.
`assay`	A vector of assay names specifying which assay to use when constructing anchors. If NULL, the current default assay for each object is used.

reference	A vector specifying the object/s to be used as a reference during integration. If NULL (default), all pairwise anchors are found (no reference/s). If not NULL, the corresponding objects in object.list will be used as references. When using a set of specified references, anchors are first found between each query and each reference. The references are then integrated through pairwise integration. Each query is then mapped to the integrated reference.
anchor.features	Can be either: • A numeric value. This will call SelectIntegrationFeatures to select the provided number of features to be used in anchor finding • A vector of features to be used as input to the anchor finding process
scale	Whether or not to scale the features provided. Only set to FALSE if you have previously scaled the features you want to use for each object in the object.list
normalization.method	Name of normalization method used: LogNormalize or SCT
sct.clip.range	Numeric of length two specifying the min and max values the Pearson residual will be clipped to
reduction	Dimensional reduction to perform when finding anchors. Can be one of: • cca: Canonical correlation analysis • rpca: Reciprocal PCA
l2.norm	Perform L2 normalization on the CCA cell embeddings after dimensional reduction
dims	Which dimensions to use from the CCA to specify the neighbor search space
k.anchor	How many neighbors (k) to use when picking anchors
k.filter	How many neighbors (k) to use when filtering anchors
k.score	How many neighbors (k) to use when scoring anchors
max.features	The maximum number of features to use when specifying the neighborhood search space in the anchor filtering
nn.method	Method for nearest neighbor finding. Options include: rann, annoy
eps	Error bound on the neighbor finding algorithm (from RANN)
verbose	Print progress bars and output

Details

The main steps of this procedure are outlined below. For a more detailed description of the methodology, please see Stuart, Butler, et al Cell 2019: https://doi.org/10.1016/j.cell.2019.05.031; https://doi.org/10.1101/460147

First, determine anchor.features if not explicitly specified using SelectIntegrationFeatures. Then for all pairwise combinations of reference and query datasets:

- Perform dimensional reduction on the dataset pair as specified via the reduction parameter. If l2.norm is set to TRUE, perform L2 normalization of the embedding vectors.
- Identify anchors - pairs of cells from each dataset that are contained within each other's neighborhoods (also known as mutual nearest neighbors).

- Filter low confidence anchors to ensure anchors in the low dimension space are in broad agreement with the high dimensional measurements. This is done by looking at the neighbors of each query cell in the reference dataset using `max.features` to define this space. If the reference cell isn't found within the first `k.filter` neighbors, remove the anchor.
- Assign each remaining anchor a score. For each anchor cell, determine the nearest `k.score` anchors within its own dataset and within its pair's dataset. Based on these neighborhoods, construct an overall neighbor graph and then compute the shared neighbor overlap between anchor and query cells (analogous to an SNN graph). We use the 0.01 and 0.90 quantiles on these scores to dampen outlier effects and rescale to range between 0-1.

Value

Returns an `AnchorSet` object that can be used as input to `IntegrateData`.

References

Stuart T, Butler A, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177:1888-1902
https://doi.org/10.1016/j.cell.2019.05.031

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("panc8")

# panc8 is a merged Seurat object containing 8 separate pancreas datasets
# split the object by dataset
pancreas.list <- SplitObject(panc8, split.by = "tech")

# perform standard preprocessing on each object
for (i in 1:length(pancreas.list)) {
  pancreas.list[[i]] <- NormalizeData(pancreas.list[[i]], verbose = FALSE)
  pancreas.list[[i]] <- FindVariableFeatures(
    pancreas.list[[i]], selection.method = "vst",
    nfeatures = 2000, verbose = FALSE
  )
}

# find anchors
anchors <- FindIntegrationAnchors(object.list = pancreas.list)

# integrate data
integrated <- IntegrateData(anchorset = anchors)

## End(Not run)
```

FindMarkers *Gene expression markers of identity classes*

Description

Finds markers (differentially expressed genes) for identity classes

Usage

```
FindMarkers(object, ...)

## Default S3 method:
FindMarkers(
  object,
  slot = "data",
  counts = numeric(),
  cells.1 = NULL,
  cells.2 = NULL,
  features = NULL,
  reduction = NULL,
  logfc.threshold = 0.25,
  test.use = "wilcox",
  min.pct = 0.1,
  min.diff.pct = -Inf,
  verbose = TRUE,
  only.pos = FALSE,
  max.cells.per.ident = Inf,
  random.seed = 1,
  latent.vars = NULL,
  min.cells.feature = 3,
  min.cells.group = 3,
  pseudocount.use = 1,
  ...
)

## S3 method for class 'Seurat'
FindMarkers(
  object,
  ident.1 = NULL,
  ident.2 = NULL,
  group.by = NULL,
  subset.ident = NULL,
  assay = NULL,
  slot = "data",
  reduction = NULL,
  features = NULL,
  logfc.threshold = 0.25,
```

```
  test.use = "wilcox",
  min.pct = 0.1,
  min.diff.pct = -Inf,
  verbose = TRUE,
  only.pos = FALSE,
  max.cells.per.ident = Inf,
  random.seed = 1,
  latent.vars = NULL,
  min.cells.feature = 3,
  min.cells.group = 3,
  pseudocount.use = 1,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods and to specific DE methods
slot	Slot to pull data from; note that if test.use is "negbinom", "poisson", or "DESeq2", slot will be set to "counts"
counts	Count matrix if using scale.data for DE tests. This is used for computing pct.1 and pct.2 and for filtering features based on fraction expressing
cells.1	Vector of cell names belonging to group 1
cells.2	Vector of cell names belonging to group 2
features	Genes to test. Default is to use all genes
reduction	Reduction to use in differential expression testing - will test for DE on cell embeddings
logfc.threshold	Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells. Default is 0.25 Increasing logfc.threshold speeds up the function, but can miss weaker signals.
test.use	Denotes which test to use. Available options are: • "wilcox" : Identifies differentially expressed genes between two groups of cells using a Wilcoxon Rank Sum test (default) • "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013) • "roc" : Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.

- "t" : Identify differentially expressed genes between two groups of cells using the Student's t-test.
- "negbinom" : Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson" : Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR" : Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST" : Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2" : Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014).This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/D

`min.pct` only test genes that are detected in a minimum fraction of min.pct cells in either of the two populations. Meant to speed up the function by not testing genes that are very infrequently expressed. Default is 0.1

`min.diff.pct` only test genes that show a minimum difference in the fraction of detection between the two groups. Set to -Inf by default

`verbose` Print a progress bar once expression testing begins

`only.pos` Only return positive markers (FALSE by default)

`max.cells.per.ident`
Down sample each identity class to a max number. Default is no downsampling. Not activated by default (set to Inf)

`random.seed` Random seed for downsampling

`latent.vars` Variables to test, used only when `test.use` is one of 'LR', 'negbinom', 'poisson', or 'MAST'

`min.cells.feature`
Minimum number of cells expressing the feature in at least one of the two groups, currently only used for poisson and negative binomial tests

`min.cells.group`
Minimum number of cells in one of the groups

`pseudocount.use`
Pseudocount to add to averaged expression values when calculating logFC. 1 by default.

`ident.1` Identity class to define markers for; pass an object of class `phylo` or 'clustertree' to find markers for a node in a cluster tree; passing 'clustertree' requires `BuildClusterTree` to have been run

ident.2	A second identity class for comparison; if NULL, use all other cells for comparison; if an object of class phylo or 'clustertree' is passed to ident.1, must pass a node to find markers for
group.by	Regroup cells into a different identity class prior to performing differential expression (see example)
subset.ident	Subset a particular identity class prior to regrouping. Only relevant if group.by is set (see example)
assay	Assay to use in differential expression testing

Details

p-value adjustment is performed using bonferroni correction based on the total number of genes in the dataset. Other correction methods are not recommended, as Seurat pre-filters genes using the arguments above, reducing the number of tests performed. Lastly, as Aaron Lun has pointed out, p-values should be interpreted cautiously, as the genes used for clustering are the same genes tested for differential expression.

Value

data.frame with a ranked list of putative markers as rows, and associated statistics as columns (p-values, ROC score, etc., depending on the test used (test.use)). The following columns are always present:

- avg_logFC: log fold-chage of the average expression between the two groups. Positive values indicate that the gene is more highly expressed in the first group
- pct.1: The percentage of cells where the gene is detected in the first group
- pct.2: The percentage of cells where the gene is detected in the second group
- p_val_adj: Adjusted p-value, based on bonferroni correction using all genes in the dataset

References

McDavid A, Finak G, Chattopadyay PK, et al. Data exploration, quality control and testing in single-cell qPCR-based gene expression experiments. Bioinformatics. 2013;29(4):461-467. doi:10.1093/bioinformatics/bts7

Trapnell C, et al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nature Biotechnology volume 32, pages 381-386 (2014)

Andrew McDavid, Greg Finak and Masanao Yajima (2017). MAST: Model-based Analysis of Single Cell Transcriptomics. R package version 1.2.1. https://github.com/RGLab/MAST/

Love MI, Huber W and Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." Genome Biology. https://bioconductor.org/packages/release/bioc/html/DESeq2.html

Examples

```
# Find markers for cluster 2
markers <- FindMarkers(object = pbmc_small, ident.1 = 2)
head(x = markers)

# Take all cells in cluster 2, and find markers that separate cells in the 'g1' group (metadata
```

```
# variable 'group')
markers <- FindMarkers(pbmc_small, ident.1 = "g1", group.by = 'groups', subset.ident = "2")
head(x = markers)

# Pass 'clustertree' or an object of class phylo to ident.1 and
# a node to ident.2 as a replacement for FindMarkersNode
pbmc_small <- BuildClusterTree(object = pbmc_small)
markers <- FindMarkers(object = pbmc_small, ident.1 = 'clustertree', ident.2 = 5)
head(x = markers)
```

FindNeighbors *SNN Graph Construction*

Description

Constructs a Shared Nearest Neighbor (SNN) Graph for a given dataset. We first determine the k-nearest neighbors of each cell. We use this knn graph to construct the SNN graph by calculating the neighborhood overlap (Jaccard index) between every cell and its k.param nearest neighbors.

Usage

```
FindNeighbors(object, ...)

## Default S3 method:
FindNeighbors(
  object,
  distance.matrix = FALSE,
  k.param = 20,
  compute.SNN = TRUE,
  prune.SNN = 1/15,
  nn.method = "rann",
  annoy.metric = "euclidean",
  nn.eps = 0,
  verbose = TRUE,
  force.recalc = FALSE,
  ...
)

## S3 method for class 'Assay'
FindNeighbors(
  object,
  features = NULL,
  k.param = 20,
  compute.SNN = TRUE,
  prune.SNN = 1/15,
  nn.method = "rann",
  annoy.metric = "euclidean",
```

```
  nn.eps = 0,
  verbose = TRUE,
  force.recalc = FALSE,
  ...
)

## S3 method for class 'dist'
FindNeighbors(
  object,
  k.param = 20,
  compute.SNN = TRUE,
  prune.SNN = 1/15,
  nn.method = "rann",
  annoy.metric = "euclidean",
  nn.eps = 0,
  verbose = TRUE,
  force.recalc = FALSE,
  ...
)

## S3 method for class 'Seurat'
FindNeighbors(
  object,
  reduction = "pca",
  dims = 1:10,
  assay = NULL,
  features = NULL,
  k.param = 20,
  compute.SNN = TRUE,
  prune.SNN = 1/15,
  nn.method = "rann",
  annoy.metric = "euclidean",
  nn.eps = 0,
  verbose = TRUE,
  force.recalc = FALSE,
  do.plot = FALSE,
  graph.name = NULL,
  ...
)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`distance.matrix`	Boolean value of whether the provided matrix is a distance matrix; note, for objects of class `dist`, this parameter will be set automatically
`k.param`	Defines k for the k-nearest neighbor algorithm

compute.SNN	also compute the shared nearest neighbor graph
prune.SNN	Sets the cutoff for acceptable Jaccard index when computing the neighborhood overlap for the SNN construction. Any edges with values less than or equal to this will be set to 0 and removed from the SNN graph. Essentially sets the strigency of pruning (0 — no pruning, 1 — prune everything).
nn.method	Method for nearest neighbor finding. Options include: rann, annoy
annoy.metric	Distance metric for annoy. Options include: euclidean, cosine, manhattan, and hamming
nn.eps	Error bound when performing nearest neighbor seach using RANN; default of 0.0 implies exact nearest neighbor search
verbose	Whether or not to print output to the console
force.recalc	Force recalculation of SNN.
features	Features to use as input for building the SNN
reduction	Reduction to use as input for building the SNN
dims	Dimensions of reduction to use as input
assay	Assay to use in construction of SNN
do.plot	Plot SNN graph on tSNE coordinates
graph.name	Optional naming parameter for stored SNN graph. Default is assay.name_snn.

Value

Returns the object with object@snn filled

Examples

```
pbmc_small
# Compute an SNN on the gene expression level
pbmc_small <- FindNeighbors(pbmc_small, features = VariableFeatures(object = pbmc_small))

# More commonly, we build the SNN on a dimensionally reduced form of the data
# such as the first 10 principle components.

pbmc_small <- FindNeighbors(pbmc_small, reduction = "pca", dims = 1:10)
```

FindTransferAnchors *Find transfer anchors*

Description

Find a set of anchors between a reference and query object. These anchors can later be used to transfer data from the reference to query object using the TransferData object.

Usage

```
FindTransferAnchors(
  reference,
  query,
  normalization.method = c("LogNormalize", "SCT"),
  reference.assay = NULL,
  query.assay = NULL,
  reduction = "pcaproject",
  project.query = FALSE,
  features = NULL,
  npcs = 30,
  l2.norm = TRUE,
  dims = 1:30,
  k.anchor = 5,
  k.filter = 200,
  k.score = 30,
  max.features = 200,
  nn.method = "rann",
  eps = 0,
  approx.pca = TRUE,
  verbose = TRUE
)
```

Arguments

reference	Seurat object to use as the reference
query	Seurat object to use as the query
normalization.method	Name of normalization method used: LogNormalize or SCT
reference.assay	Name of the Assay to use from reference
query.assay	Name of the Assay to use from query
reduction	Dimensional reduction to perform when finding anchors. Options are: • pcaproject: Project the PCA from the reference onto the query. We recommend using PCA when reference and query datasets are from scRNA-seq • cca: Run a CCA on the reference and query
project.query	Project the PCA from the query dataset onto the reference. Use only in rare cases where the query dataset has a much larger cell number, but the reference dataset has a unique assay for transfer.
features	Features to use for dimensional reduction. If not specified, set as variable features of the reference object which are also present in the query.
npcs	Number of PCs to compute on reference. If null, then use an existing PCA structure in the reference object
l2.norm	Perform L2 normalization on the cell embeddings after dimensional reduction
dims	Which dimensions to use from the reduction to specify the neighbor search space

k.anchor	How many neighbors (k) to use when finding anchors
k.filter	How many neighbors (k) to use when filtering anchors
k.score	How many neighbors (k) to use when scoring anchors
max.features	The maximum number of features to use when specifying the neighborhood search space in the anchor filtering
nn.method	Method for nearest neighbor finding. Options include: rann, annoy
eps	Error bound on the neighbor finding algorithm (from RANN)
approx.pca	Use truncated singular value decomposition to approximate PCA
verbose	Print progress bars and output

Details

The main steps of this procedure are outlined below. For a more detailed description of the methodology, please see Stuart, Butler, et al Cell 2019. https://doi.org/10.1016/j.cell.2019.05.031; https://doi.org/10.1101/460147

- Perform dimensional reduction. Exactly what is done here depends on the values set for the reduction and project.query parameters. If reduction = "pcaproject", a PCA is performed on either the reference (if project.query = FALSE) or the query (if project.query = TRUE), using the features specified. The data from the other dataset is then projected onto this learned PCA structure. If reduction = "cca", then CCA is performed on the reference and query for this dimensional reduction step. If l2.norm is set to TRUE, perform L2 normalization of the embedding vectors.
- Identify anchors between the reference and query - pairs of cells from each dataset that are contained within each other's neighborhoods (also known as mutual nearest neighbors).
- Filter low confidence anchors to ensure anchors in the low dimension space are in broad agreement with the high dimensional measurements. This is done by looking at the neighbors of each query cell in the reference dataset using max.features to define this space. If the reference cell isn't found within the first k.filter neighbors, remove the anchor.
- Assign each remaining anchor a score. For each anchor cell, determine the nearest k.score anchors within its own dataset and within its pair's dataset. Based on these neighborhoods, construct an overall neighbor graph and then compute the shared neighbor overlap between anchor and query cells (analogous to an SNN graph). We use the 0.01 and 0.90 quantiles on these scores to dampen outlier effects and rescale to range between 0-1.

Value

Returns an AnchorSet object that can be used as input to TransferData

References

Stuart T, Butler A, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177:1888-1902 https://doi.org/10.1016/j.cell.2019.05.031;

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("pbmc3k")

# for demonstration, split the object into reference and query
pbmc.reference <- pbmc3k[, 1:1350]
pbmc.query <- pbmc3k[, 1351:2700]

# perform standard preprocessing on each object
pbmc.reference <- NormalizeData(pbmc.reference)
pbmc.reference <- FindVariableFeatures(pbmc.reference)
pbmc.reference <- ScaleData(pbmc.reference)

pbmc.query <- NormalizeData(pbmc.query)
pbmc.query <- FindVariableFeatures(pbmc.query)
pbmc.query <- ScaleData(pbmc.query)

# find anchors
anchors <- FindTransferAnchors(reference = pbmc.reference, query = pbmc.query)

# transfer labels
predictions <- TransferData(
  anchorset = anchors,
  refdata = pbmc.reference$seurat_annotations
)
pbmc.query <- AddMetaData(object = pbmc.query, metadata = predictions)

## End(Not run)
```

FindVariableFeatures *Find variable features*

Description

Identifies features that are outliers on a 'mean variability plot'.

Usage

```
FindVariableFeatures(object, ...)

## Default S3 method:
FindVariableFeatures(
  object,
  selection.method = "vst",
  loess.span = 0.3,
  clip.max = "auto",
```

```
  mean.function = FastExpMean,
  dispersion.function = FastLogVMR,
  num.bin = 20,
  binning.method = "equal_width",
  verbose = TRUE,
  ...
)

## S3 method for class 'Assay'
FindVariableFeatures(
  object,
  selection.method = "vst",
  loess.span = 0.3,
  clip.max = "auto",
  mean.function = FastExpMean,
  dispersion.function = FastLogVMR,
  num.bin = 20,
  binning.method = "equal_width",
  nfeatures = 2000,
  mean.cutoff = c(0.1, 8),
  dispersion.cutoff = c(1, Inf),
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
FindVariableFeatures(
  object,
  assay = NULL,
  selection.method = "vst",
  loess.span = 0.3,
  clip.max = "auto",
  mean.function = FastExpMean,
  dispersion.function = FastLogVMR,
  num.bin = 20,
  binning.method = "equal_width",
  nfeatures = 2000,
  mean.cutoff = c(0.1, 8),
  dispersion.cutoff = c(1, Inf),
  verbose = TRUE,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods

`selection.method`
: How to choose top variable features. Choose one of :
  - vst: First, fits a line to the relationship of log(variance) and log(mean) using local polynomial regression (loess). Then standardizes the feature values using the observed mean and expected variance (given by the fitted line). Feature variance is then calculated on the standardized values after clipping to a maximum (see clip.max parameter).
  - mean.var.plot (mvp): First, uses a function to calculate average expression (mean.function) and dispersion (dispersion.function) for each feature. Next, divides features into num.bin (deafult 20) bins based on their average expression, and calculates z-scores for dispersion within each bin. The purpose of this is to identify variable features while controlling for the strong relationship between variability and average expression.
  - dispersion (disp): selects the genes with the highest dispersion values

`loess.span`
: (vst method) Loess span parameter used when fitting the variance-mean relationship

`clip.max`
: (vst method) After standardization values larger than clip.max will be set to clip.max; default is 'auto' which sets this value to the square root of the number of cells

`mean.function`
: Function to compute x-axis value (average expression). Default is to take the mean of the detected (i.e. non-zero) values

`dispersion.function`
: Function to compute y-axis value (dispersion). Default is to take the standard deviation of all values

`num.bin`
: Total number of bins to use in the scaled analysis (default is 20)

`binning.method`
: Specifies how the bins should be computed. Available methods are:
  - equal_width: each bin is of equal width along the x-axis [default]
  - equal_frequency: each bin contains an equal number of features (can increase statistical power to detect overdispersed features at high expression values, at the cost of reduced resolution along the x-axis)

`verbose`
: show progress bar for calculations

`nfeatures`
: Number of features to select as top variable features; only used when `selection.method` is set to `'dispersion'` or `'vst'`

`mean.cutoff`
: A two-length numeric vector with low- and high-cutoffs for feature means

`dispersion.cutoff`
: A two-length numeric vector with low- and high-cutoffs for feature dispersions

`assay`
: Assay to use

Details

For the mean.var.plot method: Exact parameter settings may vary empirically from dataset to dataset, and based on visual inspection of the plot. Setting the y.cutoff parameter to 2 identifies features that are more than two standard deviations away from the average dispersion within a bin. The default X-axis function is the mean expression level, and for Y-axis it is the log(Variance/mean). All mean/variance calculations are not performed in log-space, but the results are reported in log-space - see relevant functions for exact details.

GetAssay *Get an Assay object from a given Seurat object.*

Description

Get an Assay object from a given Seurat object.

Usage

```
GetAssay(object, ...)

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

Arguments

object	An object
...	Arguments passed to other methods
assay	Assay to get

Value

Returns an Assay object

Examples

```
GetAssay(object = pbmc_small, assay = "RNA")
```

GetAssayData *General accessor function for the Assay class*

Description

This function can be used to pull information from any of the slots in the Assay class. For example, pull one of the data matrices("counts", "data", or "scale.data").

Usage

```
GetAssayData(object, ...)

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

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

Arguments

object	An object
...	Arguments passed to other methods
slot	Specific information to pull (i.e. counts, data, scale.data, ...)
assay	Name of assay to pull data from

Value

Returns info from requested slot

Examples

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

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

GetIntegrationData *Get integation data*

Description

Get integation data

Usage

```
GetIntegrationData(object, integration.name, slot)
```

Arguments

object	Seurat object
integration.name	Name of integration object
slot	Which slot in integration object to get

Value

Returns data from the requested slot within the integrated object

GetResidual *Calculate pearson residuals of features not in the scale.data*

Description

This function calls sctransform::get_residuals.

Usage

```
GetResidual(
  object,
  features,
  assay = "SCT",
  umi.assay = NULL,
  clip.range = NULL,
  replace.value = FALSE,
  verbose = TRUE
)
```

Arguments

object	A seurat object
features	Name of features to add into the scale.data
assay	Name of the assay of the seurat object generated by SCTransform
umi.assay	Name of the assay of the seurat object containing UMI matrix and the default is RNA
clip.range	Numeric of length two specifying the min and max values the Pearson residual will be clipped to
replace.value	Recalculate residuals for all features, even if they are already present. Useful if you want to change the clip.range.
verbose	Whether to print messages and progress bars

Value

Returns a Seurat object containing pearson residuals of added features in its scale.data

See Also

get_residuals

Examples

```
pbmc_small <- SCTransform(object = pbmc_small, variable.features.n = 20)
pbmc_small <- GetResidual(object = pbmc_small, features = c('MS4A1', 'TCL1A'))
```

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`

HoverLocator *Hover Locator*

Description

Get quick information from a scatterplot by hovering over points

Usage

```
HoverLocator(plot, information = NULL, dark.theme = FALSE, ...)
```

Arguments

`plot`	A ggplot2 plot
`information`	An optional dataframe or matrix of extra information to be displayed on hover
`dark.theme`	Plot using a dark theme?
`...`	Extra parameters to be passed to `plotly::layout`

See Also

`layout` `ggplot_build` `DimPlot` `FeaturePlot`

Examples

```
## Not run:
plot <- DimPlot(object = pbmc_small)
HoverLocator(plot = plot, information = FetchData(object = pbmc_small, vars = 'percent.mito'))

## End(Not run)
```

HTODemux *Demultiplex samples based on data from cell 'hashing'*

Description

Assign sample-of-origin for each cell, annotate doublets.

Usage

```
HTODemux(
  object,
  assay = "HTO",
  positive.quantile = 0.99,
  init = NULL,
  nstarts = 100,
  kfunc = "clara",
  nsamples = 100,
  seed = 42,
  verbose = TRUE
)
```

Arguments

object	Seurat object. Assumes that the hash tag oligo (HTO) data has been added and normalized.
assay	Name of the Hashtag assay (HTO by default)
positive.quantile	The quantile of inferred 'negative' distribution for each hashtag - over which the cell is considered 'positive'. Default is 0.99
init	Initial number of clusters for hashtags. Default is the # of hashtag oligo names + 1 (to account for negatives)
nstarts	nstarts value for k-means clustering (for kfunc = "kmeans"). 100 by default
kfunc	Clustering function for initial hashtag grouping. Default is "clara" for fast k-medoids clustering on large applications, also support "kmeans" for kmeans clustering
nsamples	Number of samples to be drawn from the dataset used for clustering, for kfunc = "clara"
seed	Sets the random seed. If NULL, seed is not set
verbose	Prints the output

Value

The Seurat object with the following demultiplexed information stored in the meta data:

hash.maxID Name of hashtag with the highest signal

hash.secondID Name of hashtag with the second highest signal

hash.margin The difference between signals for hash.maxID and hash.secondID

classification Classification result, with doublets/multiplets named by the top two highest hashtags

classification.global Global classification result (singlet, doublet or negative)

hash.ID Classification result where doublet IDs are collapsed

See Also

HTOHeatmap

Examples

```
## Not run:
object <- HTODemux(object)

## End(Not run)
```

HTOHeatmap *Hashtag oligo heatmap*

Description

Draws a heatmap of hashtag oligo signals across singlets/doublets/negative cells. Allows for the visualization of HTO demultiplexing results.

Usage

```
HTOHeatmap(
  object,
  assay = "HTO",
  classification = paste0(assay, "_classification"),
  global.classification = paste0(assay, "_classification.global"),
  ncells = 5000,
  singlet.names = NULL,
  raster = TRUE
)
```

Arguments

object	Seurat object. Assumes that the hash tag oligo (HTO) data has been added and normalized, and demultiplexing has been run with HTODemux().
assay	Hashtag assay name.
classification	The naming for metadata column with classification result from HTODemux().
global.classification	The slot for metadata column specifying a cell as singlet/doublet/negative.

ncells	Number of cells to plot. Default is to choose 5000 cells by random subsampling, to avoid having to draw exceptionally large heatmaps.
singlet.names	Namings for the singlets. Default is to use the same names as HTOs.
raster	If true, plot with geom_raster, else use geom_tile. geom_raster may look blurry on some viewing applications such as Preview due to how the raster is interpolated. Set this to FALSE if you are encountering that issue (note that plots may take longer to produce/render).

Value

Returns a ggplot2 plot object.

See Also

HTODemux

Examples

```
## Not run:
object <- HTODemux(object)
HTOHeatmap(object)

## End(Not run)
```

HVFInfo *Get highly variable feature information*

Description

Get highly variable feature information

Usage

```
HVFInfo(object, ...)

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

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

Arguments

object	An object
...	Arguments passed to other methods
selection.method	Which method to pull; choose one from c('sctransform','sct') or c('mean.var.plot','dispersio
status	Add variable status to the resulting data.frame
assay	Name of assay to pull highly variable feature information for

Value

A dataframe with feature means, dispersion, and scaled dispersion

Examples

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

# Get the HVF info from a specific Assay in a Seurat object
HVFInfo(object = pbmc_small, assay = "RNA")[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'
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 identites 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 identies

`Idents<-`: An object with the cell identites 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(object = pbmc_small)

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

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

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

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

## End(Not run)

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

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

IntegrateData *Integrate data*

Description

Perform dataset integration using a pre-computed AnchorSet.

Usage

```
IntegrateData(
  anchorset,
  new.assay.name = "integrated",
  normalization.method = c("LogNormalize", "SCT"),
  features = NULL,
  features.to.integrate = NULL,
  dims = 1:30,
  k.weight = 100,
  weight.reduction = NULL,
  sd.weight = 1,
  sample.tree = NULL,
  preserve.order = FALSE,
  do.cpp = TRUE,
  eps = 0,
  verbose = TRUE
)
```

Arguments

anchorset	An AnchorSet object generated by FindIntegrationAnchors
new.assay.name	Name for the new assay containing the integrated data
normalization.method	Name of normalization method used: LogNormalize or SCT
features	Vector of features to use when computing the PCA to determine the weights. Only set if you want a different set from those used in the anchor finding process
features.to.integrate	Vector of features to integrate. By default, will use the features used in anchor finding.
dims	Number of dimensions to use in the anchor weighting procedure
k.weight	Number of neighbors to consider when weighting anchors
weight.reduction	Dimension reduction to use when calculating anchor weights. This can be one of: • A string, specifying the name of a dimension reduction present in all objects to be integrated

- A vector of strings, specifying the name of a dimension reduction to use for each object to be integrated
- A vector of DimReduc objects, specifying the object to use for each object in the integration
- NULL, in which case a new PCA will be calculated and used to calculate anchor weights

Note that, if specified, the requested dimension reduction will only be used for calculating anchor weights in the first merge between reference and query, as the merged object will subsequently contain more cells than was in query, and weights will need to be calculated for all cells in the object.

sd.weight	Controls the bandwidth of the Gaussian kernel for weighting
sample.tree	Specify the order of integration. If NULL, will compute automatically.
preserve.order	Do not reorder objects based on size for each pairwise integration.
do.cpp	Run cpp code where applicable
eps	Error bound on the neighbor finding algorithm (from RANN)
verbose	Print progress bars and output

Details

The main steps of this procedure are outlined below. For a more detailed description of the methodology, please see Stuart, Butler, et al Cell 2019. https://doi.org/10.1016/j.cell.2019.05.031; https://doi.org/10.1101/460147

For pairwise integration:

- Construct a weights matrix that defines the association between each query cell and each anchor. These weights are computed as 1 - the distance between the query cell and the anchor divided by the distance of the query cell to the k.weightth anchor multiplied by the anchor score computed in FindIntegrationAnchors. We then apply a Gaussian kernel width a bandwidth defined by sd.weight and normalize across all k.weight anchors.
- Compute the anchor integration matrix as the difference between the two expression matrices for every pair of anchor cells
- Compute the transformation matrix as the product of the integration matrix and the weights matrix.
- Subtract the transformation matrix from the original expression matrix.

For multiple dataset integration, we perform iterative pairwise integration. To determine the order of integration (if not specified via sample.tree), we

- Define a distance between datasets as the total number of cells in the smaller dataset divided by the total number of anchors between the two datasets.
- Compute all pairwise distances between datasets
- Cluster this distance matrix to determine a guide tree

Value

Returns a Seurat object with a new integrated Assay. If normalization.method = "LogNormalize", the integrated data is returned to the data slot and can be treated as log-normalized, corrected data. If normalization.method = "SCT", the integrated data is returned to the scale.data slot and can be treated as centered, corrected Pearson residuals.

References

Stuart T, Butler A, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177:1888-1902 https://doi.org/10.1016/j.cell.2019.05.031

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("panc8")

# panc8 is a merged Seurat object containing 8 separate pancreas datasets
# split the object by dataset
pancreas.list <- SplitObject(panc8, split.by = "tech")

# perform standard preprocessing on each object
for (i in 1:length(pancreas.list)) {
  pancreas.list[[i]] <- NormalizeData(pancreas.list[[i]], verbose = FALSE)
  pancreas.list[[i]] <- FindVariableFeatures(
    pancreas.list[[i]], selection.method = "vst",
    nfeatures = 2000, verbose = FALSE
  )
}

# find anchors
anchors <- FindIntegrationAnchors(object.list = pancreas.list)

# integrate data
integrated <- IntegrateData(anchorset = anchors)

## End(Not run)
```

IntegrationData-class *The IntegrationData Class*

Description

The IntegrationData object is an intermediate storage container used internally throughout the integration procedure to hold bits of data that are useful downstream.

Slots

`neighbors` List of neighborhood information for cells (outputs of `RANN::nn2`)

`weights` Anchor weight matrix

`integration.matrix` Integration matrix

`anchors` Anchor matrix

`offsets` The offsets used to enable cell look up in downstream functions

`objects.ncell` Number of cells in each object in the object.list

`sample.tree` Sample tree used for ordering multi-dataset integration

`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']])
```

JackStraw *Determine statistical significance of PCA scores.*

Description

Randomly permutes a subset of data, and calculates projected PCA scores for these 'random' genes. Then compares the PCA scores for the 'random' genes with the observed PCA scores to determine statistical signifance. End result is a p-value for each gene's association with each principal component.

Usage

```
JackStraw(
  object,
  reduction = "pca",
  assay = NULL,
  dims = 20,
  num.replicate = 100,
  prop.freq = 0.01,
  verbose = TRUE,
  maxit = 1000
)
```

Arguments

object	Seurat object
reduction	DimReduc to use. ONLY PCA CURRENTLY SUPPORTED.
assay	Assay used to calculate reduction.
dims	Number of PCs to compute significance for
num.replicate	Number of replicate samplings to perform
prop.freq	Proportion of the data to randomly permute for each replicate
verbose	Print progress bar showing the number of replicates that have been processed.
maxit	maximum number of iterations to be performed by the irlba function of RunPCA

Value

Returns a Seurat object where JS(object = object[['pca']], slot = 'empirical') represents p-values for each gene in the PCA analysis. If ProjectPCA is subsequently run, JS(object = object[['pca']], slot = 'full') then represents p-values for all genes.

References

Inspired by Chung et al, Bioinformatics (2014)

Examples

```
## Not run:
pbmc_small = suppressWarnings(JackStraw(pbmc_small))
head(JS(object = pbmc_small[['pca']], slot = 'empirical'))

## End(Not run)
```

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

JackStrawPlot *JackStraw Plot*

Description

Plots the results of the JackStraw analysis for PCA significance. For each PC, plots a QQ-plot comparing the distribution of p-values for all genes across each PC, compared with a uniform distribution. Also determines a p-value for the overall significance of each PC (see Details).

Usage

```
JackStrawPlot(object, dims = 1:5, reduction = "pca", xmax = 0.1, ymax = 0.3)
```

Arguments

`object`	Seurat object
`dims`	Dims to plot
`reduction`	reduction to pull jackstraw info from
`xmax`	X-axis maximum on each QQ plot.
`ymax`	Y-axis maximum on each QQ plot.

Details

Significant PCs should show a p-value distribution (black curve) that is strongly skewed to the left compared to the null distribution (dashed line) The p-value for each PC is based on a proportion test comparing the number of genes with a p-value below a particular threshold (score.thresh), compared with the proportion of genes expected under a uniform distribution of p-values.

Value

A ggplot object

Author(s)

Omri Wurtzel

See Also

ScoreJackStraw

Examples

```
JackStrawPlot(object = pbmc_small)
```

JS *Get JackStraw information*

Description

Get JackStraw information

Set JackStraw information

Usage

```
JS(object, ...)

JS(object, ...) <- value

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

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

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

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

Key *Get a key*

Description

Get a key

Set a key

Usage

```
Key(object, ...)

Key(object, ...) <- value

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

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

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

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

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

Arguments

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

Examples

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

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

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

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

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

L2CCA *L2-Normalize CCA*

Description

Perform l2 normalization on CCs

Usage

```
L2CCA(object, ...)
```

Arguments

object	Seurat object
...	Additional parameters to L2Dim.

L2Dim *L2-normalization*

Description

Perform l2 normalization on given dimensional reduction

Usage

```
L2Dim(object, reduction, new.dr = NULL, new.key = NULL)
```

Arguments

object	Seurat object
reduction	Dimensional reduction to normalize
new.dr	name of new dimensional reduction to store (default is olddr.l2)
new.key	name of key for new dimensional reduction

Value

Returns a Seurat object

LabelClusters *Label clusters on a ggplot2-based scatter plot*

Description

Label clusters on a ggplot2-based scatter plot

Usage

```
LabelClusters(
  plot,
  id,
  clusters = NULL,
  labels = NULL,
  split.by = NULL,
  repel = TRUE,
  ...
)
```

Arguments

plot	A ggplot2-based scatter plot
id	Name of variable used for coloring scatter plot
clusters	Vector of cluster ids to label
labels	Custom labels for the clusters
split.by	Split labels by some grouping label, useful when using facet_wrap or facet_grid
repel	Use geom_text_repel to create nicely-repelled labels
...	Extra parameters to geom_text_repel, such as size

Value

A ggplot2-based scatter plot with cluster labels

See Also

geom_text_repel geom_text

Examples

```
plot <- DimPlot(object = pbmc_small)
LabelClusters(plot = plot, id = 'ident')
```

LabelPoints *Add text labels to a ggplot2 plot*

Description

Add text labels to a ggplot2 plot

Usage

```
LabelPoints(
  plot,
  points,
  labels = NULL,
  repel = FALSE,
  xnudge = 0.3,
  ynudge = 0.05,
  ...
)
```

Arguments

plot	A ggplot2 plot with a GeomPoint layer
points	A vector of points to label; if NULL, will use all points in the plot
labels	A vector of labels for the points; if NULL, will use rownames of the data provided to the plot at the points selected
repel	Use geom_text_repel to create a nicely-repelled labels; this is slow when a lot of points are being plotted. If using repel, set xnudge and ynudge to 0
xnudge, ynudge	Amount to nudge X and Y coordinates of labels by
...	Extra parameters passed to geom_text

Value

A ggplot object

See Also

geom_text

Examples

```
ff <- TopFeatures(object = pbmc_small[['pca']])
cc <- TopCells(object = pbmc_small[['pca']])
plot <- FeatureScatter(object = pbmc_small, feature1 = ff[1], feature2 = ff[2])
LabelPoints(plot = plot, points = cc)
```

Loadings *Get feature loadings*

Description

Get feature loadings

Add feature loadings

Usage

```
Loadings(object, ...)

Loadings(object, ...) <- value

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

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

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

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

Examples

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

# Get the feature loadings for a specified DimReduc in a Seurat object
Loadings(object = pbmc_small, reduction = "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
```

`LocalStruct` *Calculate the local structure preservation metric*

Description

Calculates a metric that describes how well the local structure of each group prior to integration is preserved after integration. This procedure works as follows: For each group, compute a PCA, compute the top num.neighbors in pca space, compute the top num.neighbors in corrected pca space, compute the size of the intersection of those two sets of neighbors. Return the average over all groups.

Usage

```
LocalStruct(
  object,
  grouping.var,
  idents = NULL,
  neighbors = 100,
  reduction = "pca",
  reduced.dims = 1:10,
  orig.dims = 1:10,
  verbose = TRUE
)
```

Arguments

`object`	Seurat object
`grouping.var`	Grouping variable
`idents`	Optionally specify a set of idents to compute metric for
`neighbors`	Number of neighbors to compute in pca/corrected pca space
`reduction`	Dimensional reduction to use for corrected space
`reduced.dims`	Number of reduced dimensions to use
`orig.dims`	Number of PCs to use in original space
`verbose`	Display progress bar

Value

Returns the average preservation metric

LogNormalize *Normalize raw data*

Description

Normalize count data per cell and transform to log scale

Usage

```
LogNormalize(data, scale.factor = 10000, verbose = TRUE)
```

Arguments

data	Matrix with the raw count data
scale.factor	Scale the data. Default is 1e4
verbose	Print progress

Value

Returns a matrix with the normalize and log transformed data

Examples

```
mat <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
mat
mat_norm <- LogNormalize(data = mat)
mat_norm
```

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`

LogVMR *Calculate the variance to mean ratio of logged values*

Description

Calculate the variance to mean ratio (VMR) in non-logspace (return answer in log-space)

Usage

```
LogVMR(x, ...)
```

Arguments

`x`	A vector of values
`...`	Other arguments (not used)

Value

Returns the VMR in log-space

Examples

```
LogVMR(x = c(1, 2, 3))
```

---

merge.Assay *Merge Seurat Objects*

Description

Merge two or more objects.

Usage

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

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

Arguments

x	Object
y	Object (or a list of multiple 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.
...	Arguments passed to other methods
project	Sets the project name for the Seurat object.

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 not preserve reductions, 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).

Value

Merged object

Examples

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

MetaFeature *Aggregate expression of multiple features into a single feature*

Description

Calculates relative contribution of each feature to each cell for given set of features.

Usage

```
MetaFeature(
  object,
  features,
  meta.name = "metafeature",
  cells = NULL,
  assay = NULL,
  slot = "data"
)
```

Arguments

object	A Seurat object
features	List of features to aggregate
meta.name	Name of column in metadata to store metafeature
cells	List of cells to use (default all cells)
assay	Which assay to use
slot	Which slot to take data from (default data)

Value

Returns a Seurat object with metafeature stored in objct metadata

Examples

```
pbmc_small <- MetaFeature(
  object = pbmc_small,
  features = c("LTB", "EAF2"),
  meta.name = 'var.aggregate'
)
head(pbmc_small[[]])
```

MinMax *Apply a ceiling and floor to all values in a matrix*

Description

Apply a ceiling and floor to all values in a matrix

Usage

```
MinMax(data, min, max)
```

Arguments

data	Matrix or data frame
min	all values below this min value will be replaced with min
max	all values above this max value will be replaced with max

Value

Returns matrix after performing these floor and ceil operations

Examples

```
mat <- matrix(data = rbinom(n = 25, size = 20, prob = 0.2 ), nrow = 5)
mat
MinMax(data = mat, min = 4, max = 5)
```

Misc *Access miscellaneous data*

Description

Access miscellaneous data

Set miscellaneous data

Usage

```
Misc(object, ...)

Misc(object, ...) <- value

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

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

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

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

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

## 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"
```

MixingMetric *Calculates a mixing metric*

Description

Here we compute a measure of how well mixed a composite dataset is. To compute, we first examine the local neighborhood for each cell (looking at max.k neighbors) and determine for each group (could be the dataset after integration) the k nearest neighbor and what rank that neighbor was in the overall neighborhood. We then take the median across all groups as the mixing metric per cell.

Usage

```
MixingMetric(
  object,
  grouping.var,
  reduction = "pca",
  dims = 1:2,
  k = 5,
  max.k = 300,
  eps = 0,
  verbose = TRUE
)
```

Arguments

object	Seurat object
grouping.var	Grouping variable for dataset
reduction	Which dimensionally reduced space to use
dims	Dimensions to use
k	Neighbor number to examine per group
max.k	Maximum size of local neighborhood to compute
eps	Error bound on the neighbor finding algorithm (from RANN)
verbose	Displays progress bar

Value

Returns a vector of values of the mixing metric for each cell

---

MULTIseqDemux *Demultiplex samples based on classification method from MULTI-seq (McGinnis et al., bioRxiv 2018)*

Description

Identify singlets, doublets and negative cells from multiplexing experiments. Annotate singlets by tags.

Usage

```
MULTIseqDemux(
  object,
  assay = "HTO",
  quantile = 0.7,
  autoThresh = FALSE,
  maxiter = 5,
  qrange = seq(from = 0.1, to = 0.9, by = 0.05),
  verbose = TRUE
)
```

Arguments

object	Seurat object. Assumes that the specified assay data has been added
assay	Name of the multiplexing assay (HTO by default)
quantile	The quantile to use for classification
autoThresh	Whether to perform automated threshold finding to define the best quantile. Default is FALSE
maxiter	Maximum number of iterations if autoThresh = TRUE. Default is 5
qrange	A range of possible quantile values to try if autoThresh = TRUE
verbose	Prints the output

Value

A Seurat object with demultiplexing results stored at `object$MULTI_ID`

References

https://www.biorxiv.org/content/early/2018/08/08/387241

Examples

```
## Not run:
object <- MULTIseqDemux(object)

## End(Not run)
```

NormalizeData *Normalize Data*

Description

Normalize the count data present in a given assay.

Usage

```
NormalizeData(object, ...)

## Default S3 method:
NormalizeData(
  object,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  block.size = NULL,
  verbose = TRUE,
  ...
```

```
)

## S3 method for class 'Assay'
NormalizeData(
  object,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
NormalizeData(
  object,
  assay = NULL,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	An object
`...`	Arguments passed to other methods
`normalization.method`	Method for normalization. • LogNormalize: Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. This is then natural-log transformed using log1p. • CLR: Applies a centered log ratio transformation • RC: Relative counts. Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. No log-transformation is applied. For counts per million (CPM) set `scale.factor = 1e6`
`scale.factor`	Sets the scale factor for cell-level normalization
`margin`	If performing CLR normalization, normalize across features (1) or cells (2)
`block.size`	How many cells should be run in each chunk, will try to split evenly across threads
`verbose`	display progress bar for normalization procedure
`assay`	Name of assay to use

Value

Returns object after normalization

Examples

```
## Not run:
pbmc_small
pmbc_small <- NormalizeData(object = pbmc_small)

## End(Not run)
```

`OldWhichCells` *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, ect..

Usage

```
OldWhichCells(object, ...)

## S3 method for class 'Assay'
OldWhichCells(
  object,
  cells,
  subset.name = NULL,
  low.threshold = -Inf,
  high.threshold = Inf,
  accept.value = NULL,
  ...
)

## S3 method for class 'Seurat'
OldWhichCells(
  object,
  cells = NULL,
  subset.name = NULL,
  low.threshold = -Inf,
  high.threshold = Inf,
  accept.value = NULL,
  ident.keep = NULL,
  ident.remove = NULL,
  max.cells.per.ident = Inf,
  random.seed = 1,
  assay = NULL,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods and FetchData
cells	Subset of cell names
subset.name	Parameter to subset on. Eg, the name of a gene, PC_1, a column name in object@meta.data, etc. Any argument that can be retreived using FetchData
low.threshold	Low cutoff for the parameter (default is -Inf)
high.threshold	High cutoff for the parameter (default is Inf)
accept.value	Returns all cells with the subset name equal to this value
ident.keep	Create a cell subset based on the provided identity classes
ident.remove	Subtract out cells from these identity classes (used for filtration)
max.cells.per.ident	Can be used to downsample the data to a certain max per cell ident. Default is INF.
random.seed	Random seed for downsampling
assay	Which assay to filter on

Value

A vector of cell names

See Also

FetchData

Examples

```
## Not run:
OldWhichCells(object = pbmc_small, ident.keep = 2)

## End(Not run)
```

---

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

---

PCASigGenes *Significant genes from a PCA*

Description

Returns a set of genes, based on the JackStraw analysis, that have statistically significant associations with a set of PCs.

Usage

```
PCASigGenes(
  object,
  pcs.use,
  pval.cut = 0.1,
  use.full = FALSE,
  max.per.pc = NULL
)
```

Arguments

object	Seurat object
pcs.use	PCS to use.
pval.cut	P-value cutoff
use.full	Use the full list of genes (from the projected PCA). Assumes that ProjectDim has been run. Currently, must be set to FALSE.
max.per.pc	Maximum number of genes to return per PC. Used to avoid genes from one PC dominating the entire analysis.

Value

A vector of genes whose p-values are statistically significant for at least one of the given PCs.

See Also

ProjectDim JackStraw

Examples

```
PCASigGenes(pbmc_small, pcs.use = 1:2)
```

PercentageFeatureSet *Calculate the percentage of all counts that belong to a given set of features*

Description

This function enables you to easily calculate the percentage of all the counts belonging to a subset of the possible features for each cell. This is useful when trying to compute the percentage of transcripts that map to mitochondrial genes for example. The calculation here is simply the column sum of the matrix present in the counts slot for features belonging to the set divided by the column sum for all features times 100.

Usage

```
PercentageFeatureSet(
  object,
  pattern = NULL,
  features = NULL,
  col.name = NULL,
  assay = NULL
)
```

Arguments

object	A Seurat object
pattern	A regex pattern to match features against
features	A defined feature set. If features provided, will ignore the pattern matching
col.name	Name in meta.data column to assign. If this is not null, returns a Seurat object with the proportion of the feature set stored in metadata.
assay	Assay to use

Value

Returns a vector with the proportion of the feature set or if md.name is set, returns a Seurat object with the proportion of the feature set stored in metadata.

Examples

```
# Calculate the proportion of transcripts mapping to mitochondrial genes
# NOTE: The pattern provided works for human gene names. You may need to adjust depending on your
# system of interest
pbmc_small[["percent.mt"]] <- PercentageFeatureSet(object = pbmc_small, pattern = "^MT-")
```

PlotClusterTree *Plot clusters as a tree*

Description

Plots previously computed tree (from BuildClusterTree)

Usage

```
PlotClusterTree(object, ...)
```

Arguments

object	Seurat object
...	Additional arguments to ape::plot.phylo

Value

Plots dendogram (must be precomputed using BuildClusterTree), returns no value

Examples

```
pbmc_small <- BuildClusterTree(object = pbmc_small)
PlotClusterTree(object = pbmc_small)
```

PolyDimPlot *Polygon DimPlot*

Description

Plot cells as polygons, rather than single points. Color cells by identity, or a categorical variable in metadata

Usage

```
PolyDimPlot(
  object,
  group.by = NULL,
  cells = NULL,
  poly.data = "spatial",
  flip.coords = FALSE
)
```

Arguments

object	Seurat object
group.by	A grouping variable present in the metadata. Default is to use the groupings present in the current cell identities (Idents(object = object))
cells	Vector of cells to plot (default is all cells)
poly.data	Name of the polygon dataframe in the misc slot
flip.coords	Flip x and y coordinates

Value

Returns a ggplot object

PolyFeaturePlot *Polygon FeaturePlot*

Description

Plot cells as polygons, rather than single points. Color cells by any value accessible by FetchData.

Usage

```
PolyFeaturePlot(
  object,
  features,
  cells = NULL,
  poly.data = "spatial",
  ncol = ceiling(x = length(x = features)/2),
  min.cutoff = 0,
  max.cutoff = NA,
  common.scale = TRUE,
  flip.coords = FALSE
)
```

Arguments

object	Seurat object
features	Vector of features to plot. Features can come from: • An Assay feature (e.g. a gene name - "MS4A1") • A column name from meta.data (e.g. mitochondrial percentage - "percent.mito") • A column name from a DimReduc object corresponding to the cell embedding values (e.g. the PC 1 scores - "PC_1")
cells	Vector of cells to plot (default is all cells)
poly.data	Name of the polygon dataframe in the misc slot
ncol	Number of columns to split the plot into
min.cutoff	Vector of minimum and maximum cutoff values for each feature, may specify quantile in the form of 'q##' where '##' is the quantile (eg, 'q1', 'q10')
max.cutoff	Vector of minimum and maximum cutoff values for each feature, may specify quantile in the form of 'q##' where '##' is the quantile (eg, 'q1', 'q10')
common.scale	...
flip.coords	Flip x and y coordinates

Value

Returns a ggplot object

PrepSCTIntegration *Prepare an object list normalized with sctransform for integration.*

Description

This function takes in a list of objects that have been normalized with the SCTransform method and performs the following steps:

- If anchor.features is a numeric value, calls SelectIntegrationFeatures to determine the features to use in the downstream integration procedure.
- Ensures that the sctransform residuals for the features specified to anchor.features are present in each object in the list. This is necessary because the default behavior of SCTransform is to only store the residuals for the features determined to be variable. Residuals are recomputed for missing features using the stored model parameters via the GetResidual function.
- Subsets the scale.data slot to only contain the residuals for anchor.features for efficiency in downstream processing.

Usage

```
PrepSCTIntegration(
  object.list,
  assay = NULL,
  anchor.features = 2000,
  sct.clip.range = NULL,
  verbose = TRUE
)
```

Arguments

object.list	A list of Seurat objects to prepare for integration
assay	The name of the Assay to use for integration. This can be a single name if all the assays to be integrated have the same name, or a character vector containing the name of each Assay in each object to be integrated. The specified assays must have been normalized using SCTransform. If NULL (default), the current default assay for each object is used.
anchor.features	Can be either: • A numeric value. This will call SelectIntegrationFeatures to select the provided number of features to be used in anchor finding • A vector of features to be used as input to the anchor finding process
sct.clip.range	Numeric of length two specifying the min and max values the Pearson residual will be clipped to
verbose	Display output/messages

Value

A list of Seurat objects with the appropriate scale.data slots containing only the required anchor.features.

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("panc8")

# panc8 is a merged Seurat object containing 8 separate pancreas datasets
# split the object by dataset and take the first 2 to integrate
pancreas.list <- SplitObject(panc8, split.by = "tech")[1:2]

# perform SCTransform normalization
pancreas.list <- lapply(X = pancreas.list, FUN = SCTransform)

# select integration features and prep step
features <- SelectIntegrationFeatures(pancreas.list)
pancreas.list <- PrepSCTIntegration(
  pancreas.list,
  anchor.features = features
)

# downstream integration steps
anchors <- FindIntegrationAnchors(
  pancreas.list,
  normalization.method = "SCT",
  anchor.features = features
)
pancreas.integrated <- IntegrateData(anchors)

## End(Not run)
```

---

print.DimReduc *Print the results of a dimensional reduction analysis*

Description

Prints a set of features that most strongly define a set of components

Usage

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

Arguments

x	An object
dims	Number of dimensions to display
nfeatures	Number of genes to display
projected	Use projected slot
...	Arguments passed to other methods

Value

Set of features defining the components

See Also

cat

`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

`ProjectDim` *Project Dimensional reduction onto full dataset*

Description

Takes a pre-computed dimensional reduction (typically calculated on a subset of genes) and projects this onto the entire dataset (all genes). Note that the cell loadings will remain unchanged, but now there are gene loadings for all genes.

Usage

```
ProjectDim(
  object,
  reduction = "pca",
  assay = NULL,
  dims.print = 1:5,
  nfeatures.print = 20,
  overwrite = FALSE,
  do.center = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	Seurat object
`reduction`	Reduction to use
`assay`	Assay to use
`dims.print`	Number of dims to print features for
`nfeatures.print`	Number of features with highest/lowest loadings to print for each dimension
`overwrite`	Replace the existing data in feature.loadings
`do.center`	Center the dataset prior to projection (should be set to TRUE)
`verbose`	Print top genes associated with the projected dimensions

Value

Returns Seurat object with the projected values

Examples

```
pbmc_small
pbmc_small <- ProjectDim(object = pbmc_small, reduction = "pca")
# Vizualize top projected genes in heatmap
DimHeatmap(object = pbmc_small, reduction = "pca", dims = 1, balanced = TRUE)
```

`Read10X` *Load in data from 10X*

Description

Enables easy loading of sparse data matrices provided by 10X genomics.

Usage

```
Read10X(
  data.dir = NULL,
  gene.column = 2,
  unique.features = TRUE,
  strip.suffix = FALSE
)
```

Arguments

`data.dir`	Directory containing the matrix.mtx, genes.tsv (or features.tsv), and barcodes.tsv files provided by 10X. A vector or named vector can be given in order to load several data directories. If a named vector is given, the cell barcode names will be prefixed with the name.
`gene.column`	Specify which column of genes.tsv or features.tsv to use for gene names; default is 2
`unique.features`	Make feature names unique (default TRUE)
`strip.suffix`	Remove trailing "-1" if present in all cell barcodes.

Value

If features.csv indicates the data has multiple data types, a list containing a sparse matrix of the data from each type will be returned. Otherwise a sparse matrix containing the expression data will be returned.

Examples

```
## Not run:
# For output from CellRanger < 3.0
data_dir <- 'path/to/data/directory'
list.files(data_dir) # Should show barcodes.tsv, genes.tsv, and matrix.mtx
expression_matrix <- Read10X(data.dir = data_dir)
seurat_object = CreateSeuratObject(counts = expression_matrix)

# For output from CellRanger >= 3.0 with multiple data types
data_dir <- 'path/to/data/directory'
list.files(data_dir) # Should show barcodes.tsv.gz, features.tsv.gz, and matrix.mtx.gz
data <- Read10X(data.dir = data_dir)
seurat_object = CreateSeuratObject(counts = data$`Gene Expression`)
```

```
seurat_object[['Protein']] = CreateAssayObject(counts = data$`Antibody Capture`)

## End(Not run)
```

`Read10X_h5` *Read 10X hdf5 file*

Description

Read count matrix from 10X CellRanger hdf5 file. This can be used to read both scATAC-seq and scRNA-seq matrices.

Usage

```
Read10X_h5(filename, use.names = TRUE, unique.features = TRUE)
```

Arguments

`filename`	Path to h5 file
`use.names`	Label row names with feature names rather than ID numbers.
`unique.features`	Make feature names unique (default TRUE)

Value

Returns a sparse matrix with rows and columns labeled. If multiple genomes are present, returns a list of sparse matrices (one per genome).

`ReadAlevin` *Load in data from Alevin pipeline*

Description

Enables easy loading of binary format matrix provided by Alevin

Usage

```
ReadAlevin(base.path)
```

Arguments

`base.path`	Directory containing the alevin/quant_mat* files provided by Alevin.

Value

Returns a matrix with rows and columns labeled

Author(s)

Avi Srivastava

Examples

```
## Not run:
data_dir <- 'path/to/output/directory'
list.files(data_dir) # Should show alevin/quants_mat* files
expression_matrix <- ReadAlevin(base.path = data_dir)
seurat_object = CreateSeuratObject(counts = expression_matrix)

## End(Not run)
```

ReadAlevinCsv *Load in data from Alevin pipeline*

Description

Enables easy loading of csv format matrix provided by Alevin ran with '–dumpCsvCounts' flags.

Usage

```
ReadAlevinCsv(base.path)
```

Arguments

base.path	Directory containing the alevin/quant_mat* files provided by Alevin.

Value

Returns a matrix with rows and columns labeled

Author(s)

Avi Srivastava

Examples

```
## Not run:
data_dir <- 'path/to/output/directory'
list.files(data_dir) # Should show alevin/quants_mat* files
expression_matrix <- ReadAlevinCsv(base.path = data_dir)
seurat_object = CreateSeuratObject(counts = expression_matrix)

## End(Not run)
```

ReadH5AD *Read from and write to h5ad files*

Description

Utilize the Anndata h5ad file format for storing and sharing single-cell expression data. Provided are tools for writing objects to h5ad files, as well as reading h5ad files into a Seurat object

Usage

```
ReadH5AD(file, ...)

WriteH5AD(object, ...)

## S3 method for class 'character'
ReadH5AD(file, assay = "RNA", layers = "data", verbose = TRUE, ...)

## S3 method for class 'H5File'
ReadH5AD(file, assay = "RNA", layers = "data", verbose = TRUE, ...)

## S3 method for class 'Seurat'
WriteH5AD(
  object,
  file,
  assay = NULL,
  graph = NULL,
  verbose = TRUE,
  overwrite = FALSE,
  ...
)
```

Arguments

file	Name of h5ad file, or an H5File object for reading in
...	arguments passed to other methods
object	An object
assay	Name of assay to store
layers	Slot to store layers as; choose from 'counts' or 'data'; pass FALSE to not pull layers; may pass one value of 'counts' or 'data' for each layer in the H5AD file, must be in order
verbose	Show progress updates
graph	Name of graph to write out, defaults to paste0(assay,'_snn')
overwrite	Overwrite existing file

Details

ReadH5AD and WriteH5AD will try to automatically fill slots based on data type and presence. For example, objects will be filled with scaled and normalized data if adata.X is a dense matrix and raw is present (when reading), or if the scale.data slot is filled (when writing). The following is a list of how objects will be filled

adata.X **is dense and** adata.raw **is filled;** ScaleData **is filled** Objects will be filled with scaled and normalized data

adata.X **is sparse and** adata.raw **is filled;** NormalizeData **has been run,** ScaleData **has not been run** Objects will be filled with normalized and raw data

adata.X **is sparse and** adata.raw **is not filled;** NormalizeData **has not been run** Objects will be filled with raw data only

In addition, dimensional reduction information and nearest-neighbor graphs will be searched for and added if and only if scaled data is being added.

When reading: project name is basename(file); identity classes will be set as the project name; all cell-level metadata from adata.obs will be taken; feature level metadata from data.var and adata.raw.var (if present) will be merged and stored in assay meta.features; highly variable features will be set if highly_variable is present in feature-level metadata; dimensional reduction objects will be given the assay name provided to the function call; graphs will be named assay_method if method is present, otherwise assay_adata

When writing: only one assay will be written; all dimensional reductions and graphs associated with that assay will be stored, no other reductions or graphs will be written; active identity classes will be stored in adata.obs as active_ident

Value

ReadH5AD: A Seurat object with data from the h5ad file

WriteH5AD: None, writes to disk

Note

WriteH5AD is not currently functional, please use as.loom instead

See Also

as.loom

Reductions *Pull DimReducs or DimReduc names*

Description

Lists the names of DimReduc objects present in a Seurat object. If slot is provided, pulls specified DimReduc object.

Usage

```
Reductions(object, slot = NULL)
```

Arguments

object	A Seurat object
slot	Name of DimReduc

Value

If slot is NULL, the names of all DimReduc objects in this Seurat object. Otherwise, the DimReduc object requested

Examples

```
Reductions(object = pbmc_small)
```

RegroupIdents *Regroup idents based on meta.data info*

Description

For cells in each ident, set a new identity based on the most common value of a specified metadata column.

Usage

```
RegroupIdents(object, metadata)
```

Arguments

object	Seurat object
metadata	Name of metadata column

Value

A Seurat object with the active idents regrouped

Examples

```
pbmc_small <- RegroupIdents(pbmc_small, metadata = "groups")
```

RelativeCounts *Normalize raw data to fractions*

Description

Normalize count data to relative counts per cell by dividing by the total per cell. Optionally use a scale factor, e.g. for counts per million (CPM) use `scale.factor = 1e6`.

Usage

```
RelativeCounts(data, scale.factor = 1, verbose = TRUE)
```

Arguments

`data`	Matrix with the raw count data
`scale.factor`	Scale the result. Default is 1
`verbose`	Print progress

Value

Returns a matrix with the relative counts

Examples

```
mat <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
mat
mat_norm <- RelativeCounts(data = mat)
mat_norm
```

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 '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
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(
    object = 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))
```

RidgePlot *Single cell ridge plot*

Description

Draws a ridge plot of single cell data (gene expression, metrics, PC scores, etc.)

Usage

```
RidgePlot(
  object,
  features,
  cols = NULL,
  idents = NULL,
  sort = FALSE,
  assay = NULL,
  group.by = NULL,
```

```
  y.max = NULL,
  same.y.lims = FALSE,
  log = FALSE,
  ncol = NULL,
  slot = "data",
  combine = TRUE
)
```

Arguments

object	Seurat object
features	Features to plot (gene expression, metrics, PC scores, anything that can be retreived by FetchData)
cols	Colors to use for plotting
idents	Which classes to include in the plot (default is all)
sort	Sort identity classes (on the x-axis) by the average expression of the attribute being potted, can also pass 'increasing' or 'decreasing' to change sort direction
assay	Name of assay to use, defaults to the active assay
group.by	Group (color) cells in different ways (for example, orig.ident)
y.max	Maximum y axis value
same.y.lims	Set all the y-axis limits to the same values
log	plot the feature axis on log scale
ncol	Number of columns if multiple plots are displayed
slot	Use non-normalized counts data for plotting
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

Examples

```
RidgePlot(object = pbmc_small, features = 'PC_1')
```

RowMergeSparseMatrices

Merge two matrices by rowname

Description

This function is for use on sparse matrices and should not be run on a Seurat object.

Usage

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

Arguments

mat1	First matrix
mat2	Second matrix

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

A merged matrix

Returns a sparse matrix

RunALRA *Run Adaptively-thresholded Low Rank Approximation (ALRA)*

Description

Runs ALRA, a method for imputation of dropped out values in scRNA-seq data. Computes the k-rank approximation to A_norm and adjusts it according to the error distribution learned from the negative values. Described in Linderman, G. C., Zhao, J., Kluger, Y. (2018). "Zero-preserving imputation of scRNA-seq data using low rank approximation." (bioRxiv:138677)

Usage

```
RunALRA(object, ...)

## Default S3 method:
RunALRA(
  object,
  k = NULL,
  q = 10,
  quantile.prob = 0.001,
  use.mkl = FALSE,
  mkl.seed = -1,
  ...
)

## S3 method for class 'Seurat'
RunALRA(
  object,
```

```
  k = NULL,
  q = 10,
  quantile.prob = 0.001,
  use.mkl = FALSE,
  mkl.seed = -1,
  assay = NULL,
  slot = "data",
  setDefaultAssay = TRUE,
  genes.use = NULL,
  K = NULL,
  thresh = 6,
  noise.start = NULL,
  q.k = 2,
  k.only = FALSE,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
k	The rank of the rank-k approximation. Set to NULL for automated choice of k.
q	The number of additional power iterations in randomized SVD when computing rank k approximation. By default, q=10.
quantile.prob	The quantile probability to use when calculating threshold. By default, quantile.prob = 0.001.
use.mkl	Use the Intel MKL based implementation of SVD. Needs to be installed from https://github.com/KlugerLab/rpca-mkl. **Note**: this requires the SeuratWrappers implementation of RunALRA
mkl.seed	Only relevant if use.mkl = TRUE. Set the seed for the random generator for the Intel MKL implementation of SVD. Any number <0 will use the current timestamp. If use.mkl = FALSE, set the seed using set.seed() function as usual.
assay	Assay to use
slot	slot to use
setDefaultAssay	If TRUE, will set imputed results as default Assay
genes.use	genes to impute
K	Number of singular values to compute when choosing k. Must be less than the smallest dimension of the matrix. Default 100 or smallest dimension.
thresh	The threshold for "significance" when choosing k. Default 1e-10.
noise.start	Index for which all smaller singular values are considered noise. Default K - 20.
q.k	Number of additional power iterations when choosing k. Default 2.
k.only	If TRUE, only computes optimal k WITHOUT performing ALRA

Note

RunALRA and associated functions are being moved to SeuratWrappers; for more information on SeuratWrappers, please see https://github.com/satijalab/seurat-wrappers

Author(s)

Jun Zhao, George Linderman

References

Linderman, G. C., Zhao, J., Kluger, Y. (2018). "Zero-preserving imputation of scRNA-seq data using low rank approximation." (bioRxiv:138677)

See Also

ALRAChooseKPlot

Examples

```
pbmc_small
# Example 1: Simple usage, with automatic choice of k.
pbmc_small_alra <- RunALRA(object = pbmc_small)
## Not run:
# Example 2: Visualize choice of k, then run ALRA
# First, choose K
pbmc_small_alra <- RunALRA(pbmc_small, k.only=TRUE)
# Plot the spectrum, spacings, and p-values which are used to choose k
ggouts <- ALRAChooseKPlot(pbmc_small_alra)
do.call(gridExtra::grid.arrange, c(ggouts, nrow=1))
# Run ALRA with the chosen k
pbmc_small_alra <- RunALRA(pbmc_small_alra)

## End(Not run)
```

RunCCA *Perform Canonical Correlation Analysis*

Description

Runs a canonical correlation analysis using a diagonal implementation of CCA. For details about stored CCA calculation parameters, see PrintCCAParams.

Usage

```
RunCCA(object1, object2, ...)

## Default S3 method:
RunCCA(
  object1,
  object2,
  standardize = TRUE,
  num.cc = 20,
  seed.use = 42,
  verbose = FALSE,
  ...
)

## S3 method for class 'Seurat'
RunCCA(
  object1,
  object2,
  assay1 = NULL,
  assay2 = NULL,
  num.cc = 20,
  features = NULL,
  renormalize = FALSE,
  rescale = FALSE,
  compute.gene.loadings = TRUE,
  add.cell.id1 = NULL,
  add.cell.id2 = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object1	First Seurat object
object2	Second Seurat object.
...	Extra parameters (passed onto MergeSeurat in case with two objects passed, passed onto ScaleData in case with single object and rescale.groups set to TRUE)
standardize	Standardize matrices - scales columns to have unit variance and mean 0
num.cc	Number of canonical vectors to calculate
seed.use	Random seed to set. If NULL, does not set a seed
verbose	Show progress messages
assay1, assay2	Assays to pull from in the first and second objects, respectively
features	Set of genes to use in CCA. Default is the union of both the variable features sets present in both objects.
renormalize	Renormalize raw data after merging the objects. If FALSE, merge the data matrices also.

rescale Rescale the datasets prior to CCA. If FALSE, uses existing data in the scale data slots.

compute.gene.loadings
Also compute the gene loadings. NOTE - this will scale every gene in the dataset which may impose a high memory cost.

add.cell.id1, add.cell.id2
Add ...

Value

Returns a combined Seurat object with the CCA results stored.

See Also

merge.Seurat

Examples

```
pbmc_small
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- subset(pbmc_small, cells = colnames(pbmc_small)[1:40])
pbmc2 <- subset(pbmc_small, cells = colnames(x = pbmc_small)[41:80])
pbmc1[["group"]] <- "group1"
pbmc2[["group"]] <- "group2"
pbmc_cca <- RunCCA(object1 = pbmc1, object2 = pbmc2)
# Print results
print(x = pbmc_cca[["cca"]])
```

RunICA *Run Independent Component Analysis on gene expression*

Description

Run fastica algorithm from the ica package for ICA dimensionality reduction. For details about stored ICA calculation parameters, see PrintICAParams.

Usage

```
RunICA(object, ...)

## Default S3 method:
RunICA(
  object,
  assay = NULL,
  nics = 50,
  rev.ica = FALSE,
  ica.function = "icafast",
```

```
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.name = "ica",
  reduction.key = "ica_",
  seed.use = 42,
  ...
)

## S3 method for class 'Assay'
RunICA(
  object,
  assay = NULL,
  features = NULL,
  nics = 50,
  rev.ica = FALSE,
  ica.function = "icafast",
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.name = "ica",
  reduction.key = "ica_",
  seed.use = 42,
  ...
)

## S3 method for class 'Seurat'
RunICA(
  object,
  assay = NULL,
  features = NULL,
  nics = 50,
  rev.ica = FALSE,
  ica.function = "icafast",
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.name = "ica",
  reduction.key = "IC_",
  seed.use = 42,
  ...
)
```

Arguments

object	Seurat object
...	Additional arguments to be passed to fastica
assay	Name of Assay ICA is being run on

nics	Number of ICs to compute
rev.ica	By default, computes the dimensional reduction on the cell x feature matrix. Setting to true will compute it on the transpose (feature x cell matrix).
ica.function	ICA function from ica package to run (options: icafast, icaimax, icajade)
verbose	Print the top genes associated with high/low loadings for the ICs
ndims.print	ICs to print genes for
nfeatures.print	Number of genes to print for each IC
reduction.name	dimensional reduction name
reduction.key	dimensional reduction key, specifies the string before the number for the dimension names.
seed.use	Set a random seed. Setting NULL will not set a seed.
features	Features to compute ICA on

RunLSI *Run Latent Semantic Indexing on binary count matrix*

Description

For details about stored LSI calculation parameters, see PrintLSIParams.

Usage

```
RunLSI(object, ...)

## Default S3 method:
RunLSI(
  object,
  assay = NULL,
  n = 50,
  reduction.key = "LSI_",
  scale.max = NULL,
  seed.use = 42,
  verbose = TRUE,
  ...
)

## S3 method for class 'Assay'
RunLSI(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "LSI_",
```

```
  scale.max = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
RunLSI(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "LSI_",
  reduction.name = "lsi",
  scale.max = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	Seurat object
...	Arguments passed to other methods
assay	Which assay to use. If NULL, use the default assay
n	Number of singular values to compute
reduction.key	Key for dimension reduction object
scale.max	Clipping value for cell embeddings. Default (NULL) is no clipping.
seed.use	Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed.
verbose	Print messages
features	Which features to use. If NULL, use variable features
reduction.name	Name for stored dimension reduction object. Default 'lsi'

Note

RunLSI is being moved to Signac. Equivalent functionality can be achieved via the Signac::RunTFIDF and Signac::RunSVD functions; for more information on Signac, please see https://github.com/timoast/Signac

Examples

```
lsi <- RunLSI(object = pbmc_small, n = 5)
```

RunPCA *Run Principal Component Analysis*

Description

Run a PCA dimensionality reduction. For details about stored PCA calculation parameters, see PrintPCAParams.

Usage

```
RunPCA(object, ...)

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

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

## S3 method for class 'Seurat'
RunPCA(
  object,
```

```
  assay = NULL,
  features = NULL,
  npcs = 50,
  rev.pca = FALSE,
  weight.by.var = TRUE,
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.name = "pca",
  reduction.key = "PC_",
  seed.use = 42,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods and IRLBA
assay	Name of Assay PCA is being run on
npcs	Total Number of PCs to compute and store (50 by default)
rev.pca	By default computes the PCA on the cell x gene matrix. Setting to true will compute it on gene x cell matrix.
weight.by.var	Weight the cell embeddings by the variance of each PC (weights the gene loadings if rev.pca is TRUE)
verbose	Print the top genes associated with high/low loadings for the PCs
ndims.print	PCs to print genes for
nfeatures.print	Number of genes to print for each PC
reduction.key	dimensional reduction key, specifies the string before the number for the dimension names. PC by default
seed.use	Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed.
approx	Use truncated singular value decomposition to approximate PCA
features	Features to compute PCA on. If features=NULL, PCA will be run using the variable features for the Assay. Note that the features must be present in the scaled data. Any requested features that are not scaled or have 0 variance will be dropped, and the PCA will be run using the remaining features.
reduction.name	dimensional reduction name, pca by default

Value

Returns Seurat object with the PCA calculation stored in the reductions slot

RunTSNE *Run t-distributed Stochastic Neighbor Embedding*

Description

Run t-SNE dimensionality reduction on selected features. Has the option of running in a reduced dimensional space (i.e. spectral tSNE, recommended), or running based on a set of genes. For details about stored TSNE calculation parameters, see `PrintTSNEParams`.

Usage

```
RunTSNE(object, ...)

## S3 method for class 'matrix'
RunTSNE(
  object,
  assay = NULL,
  seed.use = 1,
  tsne.method = "Rtsne",
  add.iter = 0,
  dim.embed = 2,
  reduction.key = "tSNE_",
  ...
)

## S3 method for class 'DimReduc'
RunTSNE(
  object,
  cells = NULL,
  dims = 1:5,
  seed.use = 1,
  tsne.method = "Rtsne",
  add.iter = 0,
  dim.embed = 2,
  reduction.key = "tSNE_",
  ...
)

## S3 method for class 'dist'
RunTSNE(
  object,
  assay = NULL,
  seed.use = 1,
  tsne.method = "Rtsne",
  add.iter = 0,
  dim.embed = 2,
  reduction.key = "tSNE_",
```

```
  ...
)

## S3 method for class 'Seurat'
RunTSNE(
  object,
  reduction = "pca",
  cells = NULL,
  dims = 1:5,
  features = NULL,
  seed.use = 1,
  tsne.method = "Rtsne",
  add.iter = 0,
  dim.embed = 2,
  distance.matrix = NULL,
  reduction.name = "tsne",
  reduction.key = "tSNE_",
  ...
)
```

Arguments

object	Seurat object
...	Arguments passed to other methods and to t-SNE call (most commonly used is perplexity)
assay	Name of assay that that t-SNE is being run on
seed.use	Random seed for the t-SNE. If NULL, does not set the seed
tsne.method	Select the method to use to compute the tSNE. Available methods are: • Rtsne: Use the Rtsne package Barnes-Hut implementation of tSNE (default) • FIt-SNE: Use the FFT-accelerated Interpolation-based t-SNE. Based on Kluger Lab code found here: https://github.com/KlugerLab/FIt-SNE
add.iter	If an existing tSNE has already been computed, uses the current tSNE to seed the algorithm and then adds additional iterations on top of this
dim.embed	The dimensional space of the resulting tSNE embedding (default is 2). For example, set to 3 for a 3d tSNE
reduction.key	dimensional reduction key, specifies the string before the number for the dimension names. tSNE_ by default
cells	Which cells to analyze (default, all cells)
dims	Which dimensions to use as input features
reduction	Which dimensional reduction (e.g. PCA, ICA) to use for the tSNE. Default is PCA
features	If set, run the tSNE on this subset of features (instead of running on a set of reduced dimensions). Not set (NULL) by default; dims must be NULL to run on features

distance.matrix
: If set, runs tSNE on the given distance matrix instead of data matrix (experimental)

reduction.name
: dimensional reduction name, specifies the position in the object$dr list. tsne by default

---

RunUMAP *Run UMAP*

Description

Runs the Uniform Manifold Approximation and Projection (UMAP) dimensional reduction technique. To run, you must first install the umap-learn python package (e.g. via `pip install umap-learn`). Details on this package can be found here: https://github.com/lmcinnes/umap. For a more in depth discussion of the mathematics underlying UMAP, see the ArXiv paper here: https://arxiv.org/abs/1802.03426.

Usage

```
RunUMAP(object, ...)

## Default S3 method:
RunUMAP(
  object,
  reduction.model = NULL,
  assay = NULL,
  umap.method = "uwot",
  n.neighbors = 30L,
  n.components = 2L,
  metric = "cosine",
  n.epochs = NULL,
  learning.rate = 1,
  min.dist = 0.3,
  spread = 1,
  set.op.mix.ratio = 1,
  local.connectivity = 1L,
  repulsion.strength = 1,
  negative.sample.rate = 5,
  a = NULL,
  b = NULL,
  uwot.sgd = FALSE,
  seed.use = 42,
  metric.kwds = NULL,
  angular.rp.forest = FALSE,
  reduction.key = "UMAP_",
  verbose = TRUE,
  ...
```

```
)

## S3 method for class 'Graph'
RunUMAP(
  object,
  assay = NULL,
  umap.method = "umap-learn",
  n.components = 2L,
  metric = "correlation",
  n.epochs = 0L,
  learning.rate = 1,
  min.dist = 0.3,
  spread = 1,
  repulsion.strength = 1,
  negative.sample.rate = 5L,
  a = NULL,
  b = NULL,
  uwot.sgd = FALSE,
  seed.use = 42L,
  metric.kwds = NULL,
  verbose = TRUE,
  reduction.key = "UMAP_",
  ...
)

## S3 method for class 'Seurat'
RunUMAP(
  object,
  reduction.model = NULL,
  dims = NULL,
  reduction = "pca",
  features = NULL,
  graph = NULL,
  assay = "RNA",
  umap.method = "uwot",
  n.neighbors = 30L,
  n.components = 2L,
  metric = "cosine",
  n.epochs = NULL,
  learning.rate = 1,
  min.dist = 0.3,
  spread = 1,
  set.op.mix.ratio = 1,
  local.connectivity = 1L,
  repulsion.strength = 1,
  negative.sample.rate = 5L,
  a = NULL,
  b = NULL,
```

```
  uwot.sgd = FALSE,
  seed.use = 42L,
  metric.kwds = NULL,
  angular.rp.forest = FALSE,
  verbose = TRUE,
  reduction.name = "umap",
  reduction.key = "UMAP_",
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods and UMAP
reduction.model	DimReduc object that contains the umap model
assay	Assay to pull data for when using features, or assay used to construct Graph if running UMAP on a Graph
umap.method	UMAP implementation to run. Can be uwot: Runs umap via the uwot R package uwot-learn: Runs umap via the uwot R package and return the learned umap model umap-learn: Run the Seurat wrapper of the python umap-learn package
n.neighbors	This determines the number of neighboring points used in local approximations of manifold structure. Larger values will result in more global structure being preserved at the loss of detailed local structure. In general this parameter should often be in the range 5 to 50.
n.components	The dimension of the space to embed into.
metric	metric: This determines the choice of metric used to measure distance in the input space. A wide variety of metrics are already coded, and a user defined function can be passed as long as it has been JITd by numba.
n.epochs	he number of training epochs to be used in optimizing the low dimensional embedding. Larger values result in more accurate embeddings. If NULL is specified, a value will be selected based on the size of the input dataset (200 for large datasets, 500 for small).
learning.rate	The initial learning rate for the embedding optimization.
min.dist	This controls how tightly the embedding is allowed compress points together. Larger values ensure embedded points are moreevenly distributed, while smaller values allow the algorithm to optimise more accurately with regard to local structure. Sensible values are in the range 0.001 to 0.5.
spread	The effective scale of embedded points. In combination with min.dist this determines how clustered/clumped the embedded points are.
set.op.mix.ratio	Interpolate between (fuzzy) union and intersection as the set operation used to combine local fuzzy simplicial sets to obtain a global fuzzy simplicial sets. Both

fuzzy set operations use the product t-norm. The value of this parameter should be between 0.0 and 1.0; a value of 1.0 will use a pure fuzzy union, while 0.0 will use a pure fuzzy intersection.

`local.connectivity`
: The local connectivity required - i.e. the number of nearest neighbors that should be assumed to be connected at a local level. The higher this value the more connected the manifold becomes locally. In practice this should be not more than the local intrinsic dimension of the manifold.

`repulsion.strength`
: Weighting applied to negative samples in low dimensional embedding optimization. Values higher than one will result in greater weight being given to negative samples.

`negative.sample.rate`
: The number of negative samples to select per positive sample in the optimization process. Increasing this value will result in greater repulsive force being applied, greater optimization cost, but slightly more accuracy.

`a`
: More specific parameters controlling the embedding. If NULL, these values are set automatically as determined by min. dist and spread. Parameter of differentiable approximation of right adjoint functor.

`b`
: More specific parameters controlling the embedding. If NULL, these values are set automatically as determined by min. dist and spread. Parameter of differentiable approximation of right adjoint functor.

`uwot.sgd`
: Set `uwot::umap(fast_sgd = TRUE)`; see umap for more details

`seed.use`
: Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed

`metric.kwds`
: A dictionary of arguments to pass on to the metric, such as the p value for Minkowski distance. If NULL then no arguments are passed on.

`angular.rp.forest`
: Whether to use an angular random projection forest to initialise the approximate nearest neighbor search. This can be faster, but is mostly on useful for metric that use an angular style distance such as cosine, correlation etc. In the case of those metrics angular forests will be chosen automatically.

`reduction.key`
: dimensional reduction key, specifies the string before the number for the dimension names. UMAP by default

`verbose`
: Controls verbosity

`dims`
: Which dimensions to use as input features, used only if `features` is NULL

`reduction`
: Which dimensional reduction (PCA or ICA) to use for the UMAP input. Default is PCA

`features`
: If set, run UMAP on this subset of features (instead of running on a set of reduced dimensions). Not set (NULL) by default; `dims` must be NULL to run on features

`graph`
: Name of graph on which to run UMAP

`reduction.name`
: Name to store dimensional reduction under in the Seurat object

Value

Returns a Seurat object containing a UMAP representation

References

McInnes, L, Healy, J, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, ArXiv e-prints 1802.03426, 2018

Examples

```
## Not run:
pbmc_small
# Run UMAP map on first 5 PCs
pbmc_small <- RunUMAP(object = pbmc_small, dims = 1:5)
# Plot results
DimPlot(object = pbmc_small, reduction = 'umap')

## End(Not run)
```

SampleUMI *Sample UMI*

Description

Downsample each cell to a specified number of UMIs. Includes an option to upsample cells below specified UMI as well.

Usage

```
SampleUMI(data, max.umi = 1000, upsample = FALSE, verbose = FALSE)
```

Arguments

data	Matrix with the raw count data
max.umi	Number of UMIs to sample to
upsample	Upsamples all cells with fewer than max.umi
verbose	Display the progress bar

Value

Matrix with downsampled data

Examples

```
counts = as.matrix(x = GetAssayData(object = pbmc_small, assay = "RNA", slot = "counts"))
downsampled = SampleUMI(data = counts)
head(x = downsampled)
```

ScaleData *Scale and center the data.*

Description

Scales and centers features in the dataset. If variables are provided in vars.to.regress, they are individually regressed against each feautre, and the resulting residuals are then scaled and centered.

Usage

```
ScaleData(object, ...)

## Default S3 method:
ScaleData(
  object,
  features = NULL,
  vars.to.regress = NULL,
  latent.data = NULL,
  split.by = NULL,
  model.use = "linear",
  use.umi = FALSE,
  do.scale = TRUE,
  do.center = TRUE,
  scale.max = 10,
  block.size = 1000,
  min.cells.to.block = 3000,
  verbose = TRUE,
  ...
)

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

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

Arguments

object	An object
...	Arguments passed to other methods
features	Vector of features names to scale/center. Default is variable features.
vars.to.regress	Variables to regress out (previously latent.vars in RegressOut). For example, nUMI, or percent.mito.
latent.data	Extra data to regress out, should be cells x latent data
split.by	Name of variable in object metadata or a vector or factor defining grouping of cells. See argument f in split for more details
model.use	Use a linear model or generalized linear model (poisson, negative binomial) for the regression. Options are 'linear' (default), 'poisson', and 'negbinom'
use.umi	Regress on UMI count data. Default is FALSE for linear modeling, but automatically set to TRUE if model.use is 'negbinom' or 'poisson'
do.scale	Whether to scale the data.
do.center	Whether to center the data.
scale.max	Max value to return for scaled data. The default is 10. Setting this can help reduce the effects of feautres that are only expressed in a very small number of cells. If regressing out latent variables and using a non-linear model, the default is 50.
block.size	Default size for number of feautres to scale at in a single computation. Increasing block.size may speed up calculations but at an additional memory cost.
min.cells.to.block	If object contains fewer than this number of cells, don't block for scaling calculations.
verbose	Displays a progress bar for scaling procedure
assay	Name of Assay to scale

Details

ScaleData now incorporates the functionality of the function formerly known as RegressOut (which regressed out given the effects of provided variables and then scaled the residuals). To make use of the regression functionality, simply pass the variables you want to remove to the vars.to.regress parameter.

Setting center to TRUE will center the expression for each feautre by subtracting the average expression for that feautre. Setting scale to TRUE will scale the expression level for each feautre by dividing the centered feautre expression levels by their standard deviations if center is TRUE and by their root mean square otherwise.

ScoreJackStraw *Compute Jackstraw scores significance.*

Description

Significant PCs should show a p-value distribution that is strongly skewed to the left compared to the null distribution. The p-value for each PC is based on a proportion test comparing the number of features with a p-value below a particular threshold (score.thresh), compared with the proportion of features expected under a uniform distribution of p-values.

Usage

```
ScoreJackStraw(object, ...)

## S3 method for class 'JackStrawData'
ScoreJackStraw(object, dims = 1:5, score.thresh = 1e-05, ...)

## S3 method for class 'DimReduc'
ScoreJackStraw(object, dims = 1:5, score.thresh = 1e-05, ...)

## S3 method for class 'Seurat'
ScoreJackStraw(
  object,
  reduction = "pca",
  dims = 1:5,
  score.thresh = 1e-05,
  do.plot = FALSE,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
dims	Which dimensions to examine

score.thresh	Threshold to use for the proportion test of PC significance (see Details)
reduction	Reduction associated with JackStraw to score
do.plot	Show plot. To return ggplot object, use JackStrawPlot after running ScoreJackStraw.

Value

Returns a Seurat object

Author(s)

Omri Wurtzel

See Also

JackStrawPlot

JackStrawPlot

SCTransform *Use regularized negative binomial regression to normalize UMI count data*

Description

This function calls sctransform::vst. The sctransform package is available at https://github.com/ChristophH/sctransform. Use this function as an alternative to the NormalizeData, FindVariableFeatures, ScaleData workflow. Results are saved in a new assay (named SCT by default) with counts being (corrected) counts, data being log1p(counts), scale.data being pearson residuals; sctransform::vst intermediate results are saved in misc slot of new assay.

Usage

```
SCTransform(
  object,
  assay = "RNA",
  new.assay.name = "SCT",
  do.correct.umi = TRUE,
  ncells = NULL,
  variable.features.n = 3000,
  variable.features.rv.th = 1.3,
  vars.to.regress = NULL,
  do.scale = FALSE,
  do.center = TRUE,
  clip.range = c(-sqrt(x = ncol(x = object[[assay]])/30), sqrt(x = ncol(x =
    object[[assay]])/30)),
  conserve.memory = FALSE,
  return.only.var.genes = TRUE,
```

```
  seed.use = 1448145,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	A seurat object
`assay`	Name of assay to pull the count data from; default is 'RNA'
`new.assay.name`	Name for the new assay containing the normalized data
`do.correct.umi`	Place corrected UMI matrix in assay counts slot; default is TRUE
`ncells`	Number of subsampling cells used to build NB regression; default is NULL
`variable.features.n`	Use this many features as variable features after ranking by residual variance; default is 3000
`variable.features.rv.th`	Instead of setting a fixed number of variable features, use this residual variance cutoff; this is only used when `variable.features.n` is set to NULL; default is 1.3
`vars.to.regress`	Variables to regress out in a second non-regularized linear regression. For example, percent.mito. Default is NULL
`do.scale`	Whether to scale residuals to have unit variance; default is FALSE
`do.center`	Whether to center residuals to have mean zero; default is TRUE
`clip.range`	Range to clip the residuals to; default is `c(-sqrt(n/30),sqrt(n/30))`, where n is the number of cells
`conserve.memory`	If set to TRUE the residual matrix for all genes is never created in full; useful for large data sets, but will take longer to run; this will also set return.only.var.genes to TRUE; default is FALSE
`return.only.var.genes`	If set to TRUE the scale.data matrices in output assay are subset to contain only the variable genes; default is TRUE
`seed.use`	Set a random seed. By default, sets the seed to 1448145. Setting NULL will not set a seed.
`verbose`	Whether to print messages and progress bars
`...`	Additional parameters passed to `sctransform::vst`

Value

Returns a Seurat object with a new assay (named SCT by default) with counts being (corrected) counts, data being log1p(counts), scale.data being pearson residuals; sctransform::vst intermediate results are saved in misc slot of the new assay.

See Also

`correct_counts` `get_residuals`

Examples

```
SCTransform(object = pbmc_small)
```

`SelectIntegrationFeatures` *Select integration features*

Description

Choose the features to use when integrating multiple datasets. This function ranks features by the number of datasets they are deemed variable in, breaking ties by the median variable feature rank across datasets. It returns the top scoring features by this ranking.

Usage

```
SelectIntegrationFeatures(
  object.list,
  nfeatures = 2000,
  assay = NULL,
  verbose = TRUE,
  fvf.nfeatures = 2000,
  ...
)
```

Arguments

`object.list`	List of seurat objects
`nfeatures`	Number of features to return
`assay`	Name or vector of assay names (one for each object) from which to pull the variable features.
`verbose`	Print messages
`fvf.nfeatures`	nfeatures for `FindVariableFeatures`. Used if `VariableFeatures` have not been set for any object in `object.list`.
`...`	Additional parameters to `FindVariableFeatures`

Details

If for any assay in the list, `FindVariableFeatures` hasn't been run, this method will try to run it using the `fvf.nfeatures` parameter and any additional ones specified through the ....

Value

A vector of selected features

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("panc8")

# panc8 is a merged Seurat object containing 8 separate pancreas datasets
# split the object by dataset and take the first 2
pancreas.list <- SplitObject(panc8, split.by = "tech")[1:2]

# perform SCTransform normalization
pancreas.list <- lapply(X = pancreas.list, FUN = SCTransform)

# select integration features
features <- SelectIntegrationFeatures(pancreas.list)

## End(Not run)
```

SetAssayData *Setter for multimodal data*

Description

Setter for multimodal data

Usage

```
SetAssayData(object, ...)

## S3 method for class 'Assay'
SetAssayData(object, slot, new.data, ...)

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

Arguments

object	An object
...	Arguments passed to other methods
slot	Where to store the new data
new.data	New data to insert
assay	Name of assay whose data should be set

Value

object with the assay data set

Examples

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

# 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"
)
```

SetIntegrationData *Set integation data*

Description

Set integation data

Usage

```
SetIntegrationData(object, integration.name, slot, new.data)
```

Arguments

`object`	Seurat object
`integration.name`	Name of integration object
`slot`	Which slot in integration object to set
`new.data`	New data to insert

Value

Returns a `Seurat` object

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-class 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-class objects. Seurat objects also store additional meta data, both at the cell and feature level (contained within individual assays). The object was designed to be as self-contained as possible, and easily extendible to new methods.

Slots

`assays` A list of assays for this project

`meta.data` Contains meta-information about each cell, starting with number of genes detected (nGene) 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-class` objects

`neighbors` ...

`reductions` A list of dimmensional reduction objects for this object

`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-class *The Seurat Class*

Description

The Seurat object is the center of each single cell analysis. It stores all information associated with the dataset, including data, annotations, analyes, etc. All that is needed to construct a Seurat object is an expression matrix (rows are genes, columns are cells), which should be log-scale

Details

Each Seurat object has a number of slots which store information. Key slots to access are listed below.

Slots

`raw.data` The raw project data

`data` The normalized expression matrix (log-scale)

`scale.data` scaled (default is z-scoring each gene) expression matrix; used for dimmensional reduction and heatmap visualization

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

`is.expr` Expression threshold to determine if a gene is expressed (0 by default)

`ident` THe 'identity class' for each cell

`meta.data` Contains meta-information about each cell, starting with number of genes detected (nGene) and the original identity class (orig.ident); more information is added using `AddMetaData`

`project.name` Name of the project (for record keeping)

`dr` List of stored dimmensional reductions; named by technique

`assay` List of additional assays for multimodal analysis; named by technique

`hvg.info` The output of the mean/variability analysis for all genes

`imputed` Matrix of imputed gene scores

`cell.names` Names of all single cells (column names of the expression matrix)

`cluster.tree` List where the first element is a phylo object containing the phylogenetic tree relating different identity classes

`snn` Spare matrix object representation of the SNN graph

`calc.params` Named list to store all calculation-related parameter choices

`kmeans` Stores output of gene-based clustering from `DoKMeans`

`spatial` Stores internal data and calculations for spatial mapping of single cells

`misc` Miscellaneous spot to store any data alongisde the object (for example, gene lists)

`version` Version of package used in object creation

`SeuratCommand-class` *The SeuratCommand Class*

Description

The SeuratCommand is used for logging commands that are run on a SeuratObject. 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

SeuratTheme *Seurat Themes*

Description

Various themes to be applied to ggplot2-based plots

`SeuratTheme` The curated Seurat theme, consists of ...

`DarkTheme` A dark theme, axes and text turn to white, the background becomes black

`NoAxes` Removes axis lines, text, and ticks

`NoLegend` Removes the legend

`FontSize` Sets axis and title font sizes

`NoGrid` Removes grid lines

`SeuratAxes` Set Seurat-style axes

`SpatialTheme` A theme designed for spatial visualizations (eg `PolyFeaturePlot`, `PolyDimPlot`)

`RestoreLegend` Restore a legend after removal

`RotatedAxis` Rotate X axis text 45 degrees

`BoldTitle` Enlarges and emphasizes the title

Usage

```
SeuratTheme()

DarkTheme(...)

FontSize(
  x.text = NULL,
  y.text = NULL,
  x.title = NULL,
  y.title = NULL,
  main = NULL,
  ...
)

NoAxes(..., keep.text = FALSE, keep.ticks = FALSE)

NoLegend(...)

NoGrid(...)

SeuratAxes(...)

SpatialTheme(...)
```

```
RestoreLegend(..., position = "right")

RotatedAxis(...)

BoldTitle(...)

WhiteBackground(...)
```

Arguments

...	Extra parameters to be passed to theme
x.text, y.text	X and Y axis text sizes
x.title, y.title	X and Y axis title sizes
main	Plot title size
keep.text	Keep axis text
keep.ticks	Keep axis ticks
position	A position to restore the legend to

Value

A ggplot2 theme object

See Also

theme

Examples

```
# Generate a plot with a dark theme
library(ggplot2)
df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
p <- ggplot(data = df, mapping = aes(x = x, y = y)) + geom_point(mapping = aes(color = 'red'))
p + DarkTheme(legend.position = 'none')

# Generate a plot with no axes
library(ggplot2)
df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
p <- ggplot(data = df, mapping = aes(x = x, y = y)) + geom_point(mapping = aes(color = 'red'))
p + NoAxes()

# Generate a plot with no legend
library(ggplot2)
df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
p <- ggplot(data = df, mapping = aes(x = x, y = y)) + geom_point(mapping = aes(color = 'red'))
p + NoLegend()

# Generate a plot with no grid lines
library(ggplot2)
```

```
df <- data.frame(x = rnorm(n = 100, mean = 20, sd = 2), y = rbinom(n = 100, size = 100, prob = 0.2))
p <- ggplot(data = df, mapping = aes(x = x, y = y)) + geom_point(mapping = aes(color = 'red'))
p + NoGrid()
```

SplitObject *Splits object into a list of subsetted objects.*

Description

Splits object based on a single attribute into a list of subsetted objects, one for each level of the attribute. For example, useful for taking an object that contains cells from many patients, and subdividing it into patient-specific objects.

Usage

```
SplitObject(object, split.by = "ident")
```

Arguments

object	Seurat object
split.by	Attribute for splitting. Default is "ident". Currently only supported for class-level (i.e. non-quantitative) attributes.

Value

A named list of Seurat objects, each containing a subset of cells from the original object.

Examples

```
# Assign the test object a three level attribute
groups <- sample(c("group1", "group2", "group3"), size = 80, replace = TRUE)
names(groups) <- colnames(pbmc_small)
pbmc_small <- AddMetaData(object = pbmc_small, metadata = groups, col.name = "group")
obj.list <- SplitObject(pbmc_small, split.by = "group")
```

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

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")
```

StopCellbrowser *Stop Cellbrowser web server*

Description

Stop Cellbrowser web server

Usage

```
StopCellbrowser()
```

Examples

```
## Not run:
StopCellbrowser()

## End(Not run)
```

SubsetByBarcodeInflections *Subset a Seurat Object based on the Barcode Distribution Inflection Points*

Description

This convenience function subsets a Seurat object based on calculated inflection points.

Usage

```
SubsetByBarcodeInflections(object)
```

Arguments

object	Seurat object

Details

See [CalculateBarcodeInflections()] to calculate inflection points and [BarcodeInflectionsPlot()] to visualize and test inflection point calculations.

Value

Returns a subsetted Seurat object.

Author(s)

Robert A. Amezquita, <robert.amezquita@fredhutch.org>

See Also

CalculateBarcodeInflections BarcodeInflectionsPlot

Examples

```
pbmc_small <- CalculateBarcodeInflections(
  object = pbmc_small,
  group.column = 'groups',
  threshold.low = 20,
  threshold.high = 30
)
SubsetByBarcodeInflections(object = pbmc_small)
```

SubsetData *Return a subset of the Seurat object*

Description

Creates a Seurat object containing only a subset of the cells in the original object. Takes either a list of cells to use as a subset, or a parameter (for example, a gene), to subset on.

Usage

```
SubsetData(object, ...)

## S3 method for class 'Assay'
SubsetData(
  object,
  cells = NULL,
  subset.name = NULL,
  low.threshold = -Inf,
  high.threshold = Inf,
  accept.value = NULL,
  ...
)

## S3 method for class 'Seurat'
SubsetData(
  object,
  assay = NULL,
  cells = NULL,
  subset.name = NULL,
  ident.use = NULL,
  ident.remove = NULL,
  low.threshold = -Inf,
  high.threshold = Inf,
  accept.value = NULL,
  max.cells.per.ident = Inf,
  random.seed = 1,
  ...
)
```

Arguments

object	An object
...	Arguments passed to other methods
cells	A vector of cell names to use as a subset. If NULL (default), then this list will be computed based on the next three arguments. Otherwise, will return an object consisting only of these cells
subset.name	Parameter to subset on. Eg, the name of a gene, PC_1, a column name in object@meta.data, etc. Any argument that can be retreived using FetchData
low.threshold	Low cutoff for the parameter (default is -Inf)
high.threshold	High cutoff for the parameter (default is Inf)
accept.value	Returns cells with the subset name equal to this value
assay	Assay to subset on
ident.use	Create a cell subset based on the provided identity classes
ident.remove	Subtract out cells from these identity classes (used for filtration)
max.cells.per.ident	Can be used to downsample the data to a certain max per cell ident. Default is INF.
random.seed	Random seed for downsampling

Value

Returns a Seurat object containing only the relevant subset of cells

Examples

```
## Not run:
pbmc1 <- SubsetData(object = pbmc_small, cells = colnames(x = pbmc_small)[1:40])
pbmc1

## End(Not run)
```

TF.IDF *Term frequency-inverse document frequency*

Description

Normalize binary data per cell using the term frequency-inverse document frequency normalization method (TF-IDF). This is suitable for the normalization of binary ATAC peak datasets.

Usage

```
TF.IDF(data, verbose = TRUE)
```

Arguments

data	Matrix with the raw count data
verbose	Print progress

Value

Returns a matrix with the normalized data

Examples

```
mat <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
mat_norm <- TF.IDF(data = mat)
```

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)
```

TopCells *Find cells with highest scores for a given dimensional reduction technique*

Description

Return a list of genes with the strongest contribution to a set of components

Usage

```
TopCells(object, dim = 1, ncells = 20, balanced = FALSE, ...)
```

Arguments

`object`	DimReduc object
`dim`	Dimension to use
`ncells`	Number of cells to return
`balanced`	Return an equal number of cells with both + and - scores.
`...`	Extra parameters passed to `Embeddings`

Value

Returns a vector of cells

Examples

```
pbmc_small
head(TopCells(object = pbmc_small[["pca"]]))
# Can specify which dimension and how many cells to return
TopCells(object = pbmc_small[["pca"]], dim = 2, ncells = 5)
```

TopFeatures *Find features with highest scores for a given dimensional reduction technique*

Description

Return a list of features with the strongest contribution to a set of components

Usage

```
TopFeatures(
  object,
  dim = 1,
  nfeatures = 20,
  projected = FALSE,
  balanced = FALSE,
  ...
)
```

Arguments

object	DimReduc object
dim	Dimension to use
nfeatures	Number of features to return
projected	Use the projected feature loadings
balanced	Return an equal number of features with both + and - scores.
...	Extra parameters passed to Loadings

Value

Returns a vector of features

Examples

```
pbmc_small
TopFeatures(object = pbmc_small[["pca"]], dim = 1)
# After projection:
TopFeatures(object = pbmc_small[["pca"]], dim = 1,  projected = TRUE)
```

`TransferData` *Transfer data*

Description

Transfer categorical or continuous data across single-cell datasets. For transferring categorical information, pass a vector from the reference dataset (e.g. `refdata = reference$celltype`). For transferring continuous information, pass a matrix from the reference dataset (e.g. `refdata = GetAssayData(reference[['RNA']])`).

Usage

```
TransferData(
  anchorset,
  refdata,
  weight.reduction = "pcaproject",
  l2.norm = FALSE,
  dims = 1:30,
  k.weight = 50,
  sd.weight = 1,
  eps = 0,
  do.cpp = TRUE,
  verbose = TRUE,
  slot = "data"
)
```

Arguments

`anchorset`	An `AnchorSet` object generated by `FindTransferAnchors`
`refdata`	Data to transfer. Should be either a vector where the names correspond to reference cells, or a matrix, where the column names correspond to the reference cells.
`weight.reduction`	Dimensional reduction to use for the weighting anchors. Options are: • pcaproject: Use the projected PCA used for anchor building • pca: Use an internal PCA on the query only • cca: Use the CCA used for anchor building • custom DimReduc: User provided `DimReduc` object computed on the query cells
`l2.norm`	Perform L2 normalization on the cell embeddings after dimensional reduction
`dims`	Number of dimensions to use in the anchor weighting procedure
`k.weight`	Number of neighbors to consider when weighting anchors
`sd.weight`	Controls the bandwidth of the Gaussian kernel for weighting
`eps`	Error bound on the neighbor finding algorithm (from `RANN`)

do.cpp	Run cpp code where applicable
verbose	Print progress bars and output
slot	Slot to store the imputed data. Must be either "data" (default) or "counts"

Details

The main steps of this procedure are outlined below. For a more detailed description of the methodology, please see Stuart, Butler, et al Cell 2019. https://doi.org/10.1016/j.cell.2019.05.031; https://doi.org/10.1101/460147

For both transferring discrete labels and also feature imputation, we first compute the weights matrix.

- Construct a weights matrix that defines the association between each query cell and each anchor. These weights are computed as 1 - the distance between the query cell and the anchor divided by the distance of the query cell to the k.weightth anchor multiplied by the anchor score computed in FindIntegrationAnchors. We then apply a Gaussian kernel width a bandwidth defined by sd.weight and normalize across all k.weight anchors.

The main difference between label transfer (classification) and feature imputation is what gets multiplied by the weights matrix. For label transfer, we perform the following steps:

- Create a binary classification matrix, the rows corresponding to each possible class and the columns corresponding to the anchors. If the reference cell in the anchor pair is a member of a certain class, that matrix entry is filled with a 1, otherwise 0.
- Multiply this classification matrix by the transpose of weights matrix to compute a prediction score for each class for each cell in the query dataset.

For feature imputation, we perform the following step:

- Multiply the expression matrix for the reference anchor cells by the weights matrix. This returns a predicted expression matrix for the specified features for each cell in the query dataset.

Value

If refdata is a vector, returns a data.frame with label predictions. If refdata is a matrix, returns an Assay object where the imputed data has been stored in the provided slot.

References

Stuart T, Butler A, et al. Comprehensive Integration of Single-Cell Data. Cell. 2019;177:1888-1902 https://doi.org/10.1016/j.cell.2019.05.031

Examples

```
## Not run:
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data("pbmc3k")

# for demonstration, split the object into reference and query
```

```
pbmc.reference <- pbmc3k[, 1:1350]
pbmc.query <- pbmc3k[, 1351:2700]

# perform standard preprocessing on each object
pbmc.reference <- NormalizeData(pbmc.reference)
pbmc.reference <- FindVariableFeatures(pbmc.reference)
pbmc.reference <- ScaleData(pbmc.reference)

pbmc.query <- NormalizeData(pbmc.query)
pbmc.query <- FindVariableFeatures(pbmc.query)
pbmc.query <- ScaleData(pbmc.query)

# find anchors
anchors <- FindTransferAnchors(reference = pbmc.reference, query = pbmc.query)

# transfer labels
predictions <- TransferData(anchorset = anchors, refdata = pbmc.reference$seurat_annotations)
pbmc.query <- AddMetaData(object = pbmc.query, metadata = predictions)

## End(Not run)
```

`UpdateSeuratObject` *Update old Seurat object to accomodate 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)
```

`UpdateSymbolList` *Get updated synonyms for gene symbols*

Description

Find current gene symbols based on old or alias symbols using the gene names database from the HUGO Gene Nomenclature Committee (HGNC)

Usage

```
GeneSymbolThesarus(
  symbols,
  timeout = 10,
  several.ok = FALSE,
  verbose = TRUE,
  ...
)

UpdateSymbolList(
  symbols,
  timeout = 10,
  several.ok = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`symbols`	A vector of gene symbols
`timeout`	Time to wait before cancelling query in seconds
`several.ok`	Allow several current gene sybmols for each provided symbol
`verbose`	Show a progress bar depicting search progress
`...`	Extra parameters passed to `GET`

Details

For each symbol passed, we query the HGNC gene names database for current symbols that have the provided symbol as either an alias (`alias_symbol`) or old (`prev_symbol`) symbol. All other queries are **not** supported.

Value

For `GeneSymbolThesarus`, if `several.ok`, a named list where each entry is the current symbol found for each symbol provided and the names are the provided symbols. Otherwise, a named vector with the same information.

For `UpdateSymbolList`, `symbols` with updated symbols from HGNC's gene names database

Note

This function requires internet access

Source

https://www.genenames.org/ http://rest.genenames.org/

See Also

GET

Examples

```
## Not run:
GeneSybmolThesarus(symbols = c("FAM64A"))

## End(Not run)

## Not run:
UpdateSymbolList(symbols = cc.genes$s.genes)

## End(Not run)
```

VariableFeaturePlot *View variable features*

Description

View variable features

Usage

```
VariableFeaturePlot(
  object,
  cols = c("black", "red"),
  pt.size = 1,
  log = NULL,
  selection.method = NULL,
  assay = NULL
)
```

Arguments

object	Seurat object
cols	Colors to specify non-variable/variable status
pt.size	Size of the points on the plot

log	Plot the x-axis in log scale
selection.method	Which method to pull; choose one from c('sctransform','sct') or c('mean.var.plot','dispersio
assay	Assay to pull variable features from

Value

A ggplot object

See Also

FindVariableFeatures

Examples

```
VariableFeaturePlot(object = pbmc_small)
```

VariableFeatures *Get and set variable feature information*

Description

Get and set variable feature information

Usage

```
VariableFeatures(object, ...)

VariableFeatures(object, ...) <- value

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

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

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

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

Arguments

object	An object
...	Arguments passed to other methods
value	A character vector of variable features
selection.method	Method used to set variable features
assay	Name of assay to pull variable features for

VizDimLoadings *Visualize Dimensional Reduction genes*

Description

Visualize top genes associated with reduction components

Usage

```
VizDimLoadings(
  object,
  dims = 1:5,
  nfeatures = 30,
  col = "blue",
  reduction = "pca",
  projected = FALSE,
  balanced = FALSE,
  ncol = NULL,
  combine = TRUE
)
```

Arguments

object	Seurat object
dims	Number of dimensions to display
nfeatures	Number of genes to display
col	Color of points to use
reduction	Reduction technique to visualize results for
projected	Use reduction values for full dataset (i.e. projected dimensional reduction values)
balanced	Return an equal number of genes with + and - scores. If FALSE (default), returns the top genes ranked by the scores absolute values
ncol	Number of columns to display
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

Examples

```
VizDimLoadings(object = pbmc_small)
```

VlnPlot *Single cell violin plot*

Description

Draws a violin plot of single cell data (gene expression, metrics, PC scores, etc.)

Usage

```
VlnPlot(
  object,
  features,
  cols = NULL,
  pt.size = 1,
  idents = NULL,
  sort = FALSE,
  assay = NULL,
  group.by = NULL,
  split.by = NULL,
  adjust = 1,
  y.max = NULL,
  same.y.lims = FALSE,
  log = FALSE,
  ncol = NULL,
  slot = "data",
  split.plot = FALSE,
  combine = TRUE
)
```

Arguments

object	Seurat object
features	Features to plot (gene expression, metrics, PC scores, anything that can be retreived by FetchData)
cols	Colors to use for plotting
pt.size	Point size for geom_violin
idents	Which classes to include in the plot (default is all)

sort	Sort identity classes (on the x-axis) by the average expression of the attribute being potted, can also pass 'increasing' or 'decreasing' to change sort direction
assay	Name of assay to use, defaults to the active assay
group.by	Group (color) cells in different ways (for example, orig.ident)
split.by	A variable to split the violin plots by,
adjust	Adjust parameter for geom_violin
y.max	Maximum y axis value
same.y.lims	Set all the y-axis limits to the same values
log	plot the feature axis on log scale
ncol	Number of columns if multiple plots are displayed
slot	Use non-normalized counts data for plotting
split.plot	plot each group of the split violin plots by multiple or single violin shapes.
combine	Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects

Value

A patchworked ggplot object if combine = TRUE; otherwise, a list of ggplot objects

See Also

FetchData

Examples

```
VlnPlot(object = pbmc_small, features = 'PC_1')
VlnPlot(object = pbmc_small, features = 'LYZ', split.by = 'groups')
```

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, ect..

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 to other methods
cells	Subset of cell names
expression	A predicate expression for feature/variable expression, can evalue 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(object = pbmc_small, idents = 2)
WhichCells(object = pbmc_small, expression = MS4A1 > 3)
levels(x = pbmc_small)
```

```
WhichCells(object = pbmc_small, idents = c(1, 2), invert = TRUE)
```

`[.Seurat` *Subset a Seurat object*

Description

Subset a Seurat object

Usage

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

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

Arguments

`x`	Seurat object to be subsetted
`i, features`	A vector of features to keep
`j, cells`	A vector of cells to keep
`...`	Extra parameters passed to `WhichCells`, such as `slot`, `invert`, or `downsample`
`subset`	Logical expression indicating features/variables to keep
`idents`	A vector of identity classes to keep

Value

A subsetted Seurat object

See Also

`subset` `WhichCells`

Examples

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

subset(x = pbmc_small, subset = MS4A1 > 4)
subset(x = pbmc_small, subset = `DLGAP1-AS1` > 2)
subset(x = pbmc_small, idents = '0', invert = TRUE)
subset(x = pbmc_small, subset = MS4A1 > 3, slot = 'counts')
subset(x = pbmc_small, features = VariableFeatures(object = pbmc_small))
```

Index