Package ‘Seurat’

April 15, 2019

Version  3.0.0
Date     2019-04-16
Title    Tools for Single Cell Genomics

URL httpZOOwwwNsatijalabNorgOseurat,
                               httpsZOOgithubNcomOsatijalabOseurat
BugReports https://github.com/satijalab/seurat
Additional_repositories https://mojaveazure.github.io/loomR
Depends  R (>= 3.4.0), methods,
Imports  ape, cluster, cowplot, fitdistrplus, future, futureapply,
          ggplot2 (>= 3.0.0), ggrepel, ggridges, graphics, grDevices,
          grid, igraph, irlba, KernSmooth, lmtest, MASS, Matrix (>= 1.2.14),
          metap, pbapply, plotly, png, RANN, RColorBrewer, Rcpp,
          reticulate, rlang, ROCR, Rtsne, scales, sctransform (>= 0.2.0),
          SDMTools, stats, tools, tsne, utils
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'
RoxygenNote 6.1.1
Encoding UTF-8
Suggests  loomR, testthat, hdf5r, S4Vectors, SummarizedExperiment,
SingleCellExperiment, MAST, DESeq2, GenomicRanges,
GenomeInfoDb, IRanges, rtracklayer

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]

Maintainer  Paul Hoffman <nygcSatijalab@nygenome.org>

Repository  CRAN

Date/Publication  2019-04-15 21:54:16 UTC

R topics documented:

AddMetaData ..................................................... 5
AddModuleScore ................................................... 6
ALRAChooseKPlot .................................................. 7
AnchorSet-class ................................................... 8
as.Graph .......................................................... 9
as.loom ............................................................ 9
as.Seurat .......................................................... 11
as.SingleCellExperiment ........................................ 13
as.sparse ........................................................ 13
Assay-class ......................................................... 14
AugmentPlot ......................................................... 15
AverageExpression ................................................. 15
BarcodeInflectionsPlot .......................................... 16
BlackAndWhite ..................................................... 17
BuildClusterTree .................................................. 18
CalculateBarcodeInflections ..................................... 19
CaseMatch .......................................................... 20
cc.genes .......................................................... 21
CellCycleScoring .................................................. 21
Cells ............................................................... 22
CellScatter .......................................................... 23
CellSelector ........................................................ 24
CollapseSpeciesExpressionMatrix ................................ 25
CombinePlots ......................................................... 26
Command ............................................................ 27
CreateAssayObject ................................................ 27
CreateDimReducObject ............................................. 28
CreateGeneActivityMatrix ........................................ 29
topics documented:

- CreateSeuratObject .............................................. 30
- CustomDistance .................................................. 31
- DefaultAssay ..................................................... 32
- DietSeurat .......................................................... 33
- DimHeatmap ......................................................... 33
- DimPlot ............................................................. 35
- DimReduc-class .................................................... 36
- DoHeatmap .......................................................... 37
- DotPlot .............................................................. 38
- ElbowPlot ........................................................... 39
- Embeddings ......................................................... 40
- ExpMean ............................................................. 41
- ExportToCellbrowser ............................................. 41
- ExpSD ............................................................... 43
- ExpVar ............................................................... 43
- FeaturePlot ........................................................ 44
- FeatureScatter .................................................... 45
- FetchData .......................................................... 46
- FindAllMarkers .................................................... 47
- FindClusters ....................................................... 49
- FindConservedMarkers ......................................... 51
- FindIntegrationAnchors ........................................ 52
- FindMarkers ....................................................... 53
- FindNeighbors ..................................................... 56
- FindTransferAnchors .......................................... 58
- FindVariableFeatures .......................................... 59
- GetAssay ........................................................... 61
- GetAssayData ..................................................... 62
- GetIntegrationData ............................................... 63
- Graph-class ......................................................... 63
- HoverLocator ...................................................... 64
- HTODemux .......................................................... 64
- HTOHeatmap ....................................................... 66
- HVFInfo ............................................................. 67
- Idents ............................................................... 67
- IntegrateData ....................................................... 70
- IntegrationData .................................................... 71
- JackStraw .......................................................... 72
- JackStrawData-class ............................................. 73
- JackStrawPlot ...................................................... 73
- JS ................................................................. 74
- Key ................................................................. 75
- L2CCA .............................................................. 76
- L2Dim ............................................................... 76
- LabelClusters ...................................................... 77
- LabelPoints ......................................................... 78
- Loadings ............................................................ 79
- LocalStruct ......................................................... 80
R topics documented:

LogNormalize ......................................................... 80
LogVMR ................................................................. 81
merge.Assay ............................................................ 82
MetaFeature ............................................................ 83
MinMax ................................................................. 83
Misc ................................................................. 84
MixingMetric ......................................................... 85
MULTIseqDemux ...................................................... 86
NormalizeData ......................................................... 87
OldWhichCells ......................................................... 88
pbmc_small ............................................................. 89
PCASigGenes ............................................................ 90
PercentageFeatureSet ................................................ 91
PlotClusterTree ......................................................... 92
PolyDimPlot .............................................................. 92
PolyFeaturePlot ......................................................... 93
print.DimReduc ......................................................... 94
Project ................................................................. 94
ProjectDim ............................................................... 95
Read10X ................................................................. 95
Read10X_h5 ............................................................... 96
ReadAlevin ............................................................... 97
ReadAlevinCsv ............................................................ 97
ReadH5AD ................................................................. 98
RelativeCounts ......................................................... 99
RenameCells .............................................................. 100
RidgePlot ............................................................... 101
RunALRA ................................................................. 103
RunCCA ................................................................. 105
RunICA ................................................................. 106
RunLSI ................................................................. 108
RunPCA ................................................................. 109
RunTSNE ................................................................. 110
RunUMAP ................................................................. 111
SampleUMI .............................................................. 114
ScaleData ............................................................... 115
ScoreJackStraw ........................................................ 117
SCTransform ........................................................... 118
SelectIntegrationFeatures ........................................... 119
SetAssayData .......................................................... 120
SetIntegrationData .................................................... 121
Seurat-class ............................................................ 121
seurat-class ........................................................... 122
SeuratCommand-class ................................................ 123
SeuratTheme ............................................................ 124
SplitObject ............................................................. 126
Stdev ................................................................. 126
StopCellbrowser ....................................................... 127
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

```r
cluster_letters <- LETTERS[Idsents(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[[1]])
```

---

**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**

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

**Arguments**

- `object`: Seurat object
- `features`: Feature expression programs in list
- `pool`: List of features to check expression levels against, 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
ALRAChooseKPlot

**Value**

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

**References**

Tirosh et al, Science (2016)

**Examples**

```r
## Not run:

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)`
AnchorSet-class

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 gg object; note that if TRUE, themeing will not work when plotting multiple features

Value

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

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.
- **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.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.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 de-
    fault

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_attr/CellID; feature names will be stored in row_attr/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_attr; all periods ‘.’ in metadata will be replaced
  with underscores ‘_’
- Clustering information from Idents(object = x) will be stored in col_attr/ClusterID
  and col_attr/ClusterName for the numeric and string representation of the factor, respectively
- Feature-level metadata will be stored in Feature_attr; all periods ‘.’ in metadata will be
  replaced with underscores ‘_’
- Variable features, if set, will be stored in row_attr/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 colAttrs for cell embeddings and rowAttrs 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 '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", ...)

Arguments
x An object to convert to class Seurat
... Arguments passed to other methods
cells The name of the dataset within col_attrs containing cell names
features The name of the dataset within row_attrs containing feature names
The name of the dataset within layers containing the normalized expression matrix
The name of the dataset within layers containing the scaled expression matrix
Name of the assay to create
Display progress updates
name of the SingleCellExperiment assay to store as counts
name of the SingleCellExperiment assay to slot as data

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 (character 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 scaleNdata; all rows entirely consisting of NAs 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
```r
## Not run:
1file <- as.loom(x = pbmc_small)
```
as.SingleCellExperiment

pbmc <- as.Seurat(x = 1f1e)
## 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, ...)
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
- **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
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
width, height
dpi
A ggplot object
Width and height of PNG version of plot
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, use.scale = FALSE, use.counts = FALSE, verbose = TRUE, ...)
BarcodeInflectionsPlot

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).
use.scale Use scaled values for gene expression
use.counts Use count values for gene 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
BlackAndWhite

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
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, features = NULL, dims = NULL, graph = NULL,
                   reorder = FALSE, reorder.numeric = FALSE, verbose = TRUE)
```

**Arguments**
- **object**  
  Seurat object
- **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
- **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

**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())
```
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

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

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

```r
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 this 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(pbm_small, group.column = 'groups')

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

---

**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
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**

```r
Cells(x = 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**

```r
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)
- `...`: Ignored for now

**Value**

A ggplot object

**Examples**

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

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
```r
## 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)
```
**CollapseSpeciesExpressionMatrix**

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

---

**Description**

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

**Usage**

```r
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**

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

## End(Not run)
```
## Description

Combine ggplot2-based plots into a single plot

## Usage

```r
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

```r
pbmc_small["group"] <- sample(
  x = c("g1", "g2"),
  size = ncol(x = pbmc_small),
  replace = TRUE
)
plots <- FeaturePlot(
  object = pbmc_small,
  features = c("MS4A1", "FCN1"),
  split.by = "group",
  combine = FALSE
)
CombinePlots(
  plots = plots,
  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 parameter value

CreateAssayObject
Create an Assay object

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

Usage
CreateAssayObject(counts, data, min.cells = 0, min.features = 0)

Arguments
- counts: Un normalized data such as raw counts or TPMs
- data: Pre normalized 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.
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, 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
jackstraw Results from the JackStraw function
misc list for the user to store any additional information associated with the dimensional reduction
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, 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
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.
assey  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 'DimReduc'
DefaultAssay(object, ...)

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

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

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>An object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
<tr>
<td>value</td>
<td>Name of assay to set as default</td>
</tr>
</tbody>
</table>

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

```r
DietSeurat(object, counts = TRUE, data = TRUE, scale.data = FALSE, 
features = NULL, assays = 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

---

**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

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

```r
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 dimensional 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
- **combine**: Combine plots into a single gg object; note that if TRUE; themeing will not work when plotting multiple dimensions
- **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
- **...**: Extra parameters passed to `DimHeatmap`

Value

No return value by default. If using fast = FALSE, will return a ggplot object.

See Also

- `image geom_raster`

Examples

```r
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**

```r
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 = "red", sizes.highlight = 1, na.value = "grey50",
        combine = TRUE, ncol = NULL, ...)

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. By default, ggplot2 assigns colors
- **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)
**DimReduc-class**

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.

### Description

- **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
- **combine** Combine plots into a single gg object; note that if TRUE; themeing will not work when plotting multiple features
- **ncol** Number of columns for display when combining plots
- **...** Extra parameters passed on to `CombinePlots`

### Value

A ggplot object

### Note

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

### See Also

- FeaturePlot HoverLocator CellSelector link{FetchData}

### Examples

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

---

**DoHeatmap**  
**Feature expression heatmap**

**Description**

Draws a heatmap of single cell feature expression.

**Usage**

```r
DoHeatmap(object, features = NULL, cells = NULL, group.by = "ident", 
            group.bar = TRUE, disp.min = -2.5, disp.max = NULL, 
            slot = "scale.data", assay = NULL, label = TRUE, size = 5.5, 
            hjust = 0, angle = 45, raster = TRUE, 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
- `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 identities above the color bar
- `size`: Size of text above color bar
- `hjust`: Horizontal justification of text above color bar
DotPlot

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).

combine

Combine plots into a single gg object; note that if TRUE; themeing will not work when plotting multiple dimensions

Value

A ggplot object

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 of cells within a class (blue is high).

Usage

DotPlot(object, 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.by = "radius",
  scale.min = NA, scale.max = NA, ...)

Arguments

object Seurat object
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.
ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPlot

ElbowPl...
Embeddings

Value

A ggplot object

Examples

ElbowPlot(object = pbmc_small)

Description

Get cell embeddings

Usage

Embeddings(object, ...)

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

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

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>An object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
<tr>
<td>reduction</td>
<td>Name of reduction to pull cell embeddings for</td>
</tr>
</tbody>
</table>

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

**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, ...)
```
ExportToCellbrowser

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

```r
## 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 = c("lightgrey", "blue"), pt.size = NULL, order = FALSE, 
min.cutoff = NA, max.cutoff = NA, reduction = NULL, 
split.by = NULL, shape.by = NULL, blend = FALSE, 
blend.threshold = 0.5, label = FALSE, label.size = 4, 
ncol = NULL, combine = TRUE, coord.fixed = FALSE, by.col = 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.
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
FeatureScatter

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

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

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

combine
Combine plots into a single gg object; note that if TRUE, themeing will not work when plotting multiple features

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.

Value
A ggplot object

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

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, slot = "data", ...)
Arguments

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

Value

A ggplot object

Examples

```r
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**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>vars</td>
<td>List of all variables to fetch, use keyword 'ident' to pull identity classes</td>
</tr>
<tr>
<td>cells</td>
<td>Cells to collect data for (default is all cells)</td>
</tr>
<tr>
<td>slot</td>
<td>Slot to pull feature data for</td>
</tr>
</tbody>
</table>
FindAllMarkers

**Value**

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

**Examples**

```r
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**

```r
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). 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)) 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/DESeq2.html

| 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
FindClusters

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, algorithm = 1, n.start = 10, n.iter = 10,
  random.seed = 0, 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, algorithm = 1, n.start = 10,
              n.iter = 10, random.seed = 0, 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.
- **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.
- **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 two groups

Description

Finds markers that are conserved between the two groups

Usage

FindConservedMarkers(object, ident.1, ident.2 = NULL, grouping.var, assay.type = "RNA", slot = "data", meta.method = 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.type Type of assay to fetch data for (default is RNA)
slot Slot to pull data from; note that if test.use is "negbinom", "poisson", or "DE-Seq2", 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

Matrix 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 MetaDE package), percentage of cells expressing the marker, average differences)

Examples

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

## End(Not run)
FindIntegrationAnchors

*Find integration anchors*

**Description**

Finds the integration anchors

**Usage**

```r
FindIntegrationAnchors(object.list = NULL, assay = NULL,
anchor.features = 2000, scale = TRUE, l2.norm = TRUE,
dims = 1:30, k.anchor = 5, k.filter = 200, k.score = 30,
max.features = 200, eps = 0, verbose = TRUE)
```

**Arguments**

- `object.list`: A list of 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.
- `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
- `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
- `eps`: Error bound on the neighbor finding algorithm (from RANN)
- `verbose`: Print progress bars and output

**Value**

Returns an AnchorSet object
FindMarkers

Gene expression markers of identity classes

Description

Finds markers (differentially expressed genes) for identity classes

Usage

FindMarkers(object, ...)

## Default S3 method:
FindMarkers(object, cells.1 = NULL, cells.2 = 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, ...)

## 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
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
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)
FindMarkers

- "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)) 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/DESeq2.html

  - 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
FindMarkers

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

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

reduction
Reduction to use in differential expression testing - will test for DE on cell embeddings

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-change 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


## Examples

```r
# 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(pbm_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)
```

## 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

```r
FindNeighbors(object, ...)  
```

## Default S3 method:

```r
FindNeighbors(object, distance.matrix = FALSE,
              k.param = 20, compute.SNN = TRUE, prune.SNN = 1/15, nn.eps = 0,
              verbose = TRUE, force.recalc = FALSE, ...)
```

## S3 method for class 'Assay'

```r
FindNeighbors(object, features = NULL, k.param = 20,
              compute.SNN = TRUE, prune.SNN = 1/15, nn.eps = 0, verbose = TRUE,
              force.recalc = FALSE, ...)
```

## S3 method for class 'dist'

```r
FindNeighbors(object, k.param = 20, compute.SNN = TRUE,
              ...)
```
FindNeighbors

```r
prune.SNN = 1/15, 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.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 stringency of pruning (0 — no pruning, 1 — prune everything).
- **nn.eps**: Error bound when performing nearest neighbor search 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

```r
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.
```
FindTransferAnchors

Find transfer anchors

Description

Finds the transfer anchors

Usage

```r
FindTransferAnchors(reference, query, 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, eps = 0, approx.pca = TRUE, verbose = TRUE)
```

Arguments

- **reference**: Seurat object to use as the reference
- **query**: Seurat object to use as the query
- **reference.assay**: Assay to use from reference
- **query.assay**: 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
- **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 picking anchors
- **k.filter**: How many neighbors (k) to use when filtering anchors
- **k.score**: How many neighbors (k) to use when scoring anchors

pbmc_small <- FindNeighbors(pbmc_small, reduction = "pca", dims = 1:10)
max.features | The maximum number of features to use when specifying the neighborhood search space in the anchor filtering
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

Value

Returns an AnchorSet object

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: First, uses a function to calculate average expression (mean.function) and dispersion (dispersion.function) for each feature. Next, divides features into num.bin (default 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: 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**

```r
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. raw.data, 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 integration data

Description
Get integration 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

Graph-class  The Graph Class

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

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

```r
## 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
- **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

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

## End(Not run)```
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

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

## End(Not run)```
HVFInfo  

*Get highly variable feature information*

---

**Description**

Get highly variable feature information

**Usage**

```r
HVFInfo(object, ...)  
```

```r
## S3 method for class 'Assay'
HVFInfo(object, ...)
```

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

**Arguments**

- `object`: An object
- `...`: Arguments passed to other methods
- `assay`: Name of assay to pull highly variable feature information for

**Value**

A dataframe with feature means, dispersion, and scaled dispersion

**Examples**

```r
# Get the HVF info directly from an Assay object
HVFInfo(object = pbmc_small[["RNA"]][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 them-
  selves

var Feature or variable to order on

save.name Store current identity information under this name
cells  Set cell identities for specific cells
drop  Drop unused levels
reverse  Reverse ordering
afxn  Function to evaluate each identity class based on; default is mean
reorder.numeric  Rename all identity classes to be increasing numbers starting from 1 (default is FALSE)
x, object  An object

Value

Idents: The cell identities
Idents<=: An object with the cell identities changed
RenameIdents: An object with selected identity classes renamed
ReorderIdent: An object with
SetIdent: An object with new identity classes set
StashIdent: An object with the identities stashed

Examples

# Get cell identity classes
Idents(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 ids from a value in object metadata
colnames(x = pbmc_small[[1]])
Idents(object = pbmc_small) <- 'RNA_snn_res.1'
levels(x = pbmc_small)

# Rename cell identity classes
# Can provide an arbitrary amount of ids 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[[1]])

pbmc_small <- StashIdent(object = pbmc_small, save.name = 'idents')

head(x = pbmc_small[[1]])

# 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

Integrates the data

### Usage

IntegrateData(anchorset, new.assay.name = "integrated",
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: Results from FindIntegrationAnchors
- new.assay.name: Name for the new assay containing the integrated data
- 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 PCs to use in the weighting procedure
- k.weight: Number of neighbors to consider when weighting
- weight.reduction: Dimension reduction to use when calculating anchor weights. This can be either:
  - 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
• 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

Value

Returns a Seurat object with a new integrated Assay

---

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
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 significance. End result is a p-value for each gene's association with each principal component.

Usage
```r
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
```r
## Not run:
pbmcsmall = suppressWarnings(JackStraw(pbmcsmall))
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)

Description

Get 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
Key

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

```r
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)
```
### 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**

```r
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**

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

Arguments

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

Value

Returns a matrix with the normalize and log transformed data

Examples

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

---

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

Value

Returns the VMR in log-space

Examples

```r
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 renormal-
            ization). 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 poten-

tially 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.

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 object 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

```r
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

```r
Misc(object, ...)
```

```r
Misc(object, ...) <- value
```

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

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

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

```r
## 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
MixingMetric

Value

Miscellaneous data
An object with miscellaneous data added

Examples

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

# Add misc info
Mix(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 representing the entropy metric from each bootstrapped iteration.
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
- 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**

```r
NormalizeData(object, ...)
```

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

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

```r
## 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
  - `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

```r
## 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, etc.

**Usage**

```r
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 retrieved 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
pbmc_small

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
OldWhichCells(object = pbmc_small, ident.keep = 2)

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
raw.data Raw expression data
data Normalized expression data
scale.data Scaled expression data
var.genes Variable genes
dr Dimmensional reductions: currently PCA and tSNE
hvg.info Information about highly variable genes
cluster.tree Cluster tree
calc.params Parameters for calculations done thus far

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

```r
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

```r
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**

```R
pbmcsmall <- 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 (Idsents(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
**Description**

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

**Usage**

```r
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
**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**

```r
## 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**

```r
Project(object, ...)

Project(object, ...) <- value
```

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

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

**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**

```r
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")
# Visualize top projected genes in heatmap
DimHeatmap(object = pbmc_small, reduction = "pca", dims = 1, balanced = TRUE)

Description

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

Usage

Read10X(data.dir = NULL, gene.column = 2, unique.features = TRUE)

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)

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)

### 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).

---

### Description
Load in data from Alevin pipeline

### 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**

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

---

**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
ReadH5AD

## Examples

```r
## 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, ...)`

#### 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
- `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 adatayXy 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; NormalizedData 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; NormalizedData 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 adatayobs will be taken; feature level metadata from data.yvar 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 adatayobs 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

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
- **...**: Ignored

**Value**

Returns a matrix with the relative counts

**Examples**

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

---

**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, ...)`
RidgePlot

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))
pbmce_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, combine = TRUE, slot = "data", ...)

Arguments

object | Seurat object
features | Features to plot (gene expression, metrics, PC scores, anything that can be retrieved 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 plotted, 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
combine | Combine plots into a single gg object; note that if TRUE; themeing will not work when plotting multiple features
slot | Use non-normalized counts data for plotting
... | Extra parameters passed on to CombinePlots

Value

A ggplot object

Examples

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

---

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, ...)

## S3 method for class 'Seurat'
RunALRA(object, k = NULL, q = 10, assay = NULL,
  slot = "data",setDefaultAssay = TRUE, genes.use = NULL,
  K = NULL, p.val.th = 1e-10, 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.

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.

p.val.th  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

Author(s)

Jun Zhao, George Linderman

References

  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, verbose = FALSE, use.cpp = TRUE, ...)

## 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, use.cpp = 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
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`.

Value

Returns a combined Seurat object with the CCA results stored.

See Also

`mergeSeurat`

Examples

```r
pbmc_small
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small, cells = colnames(x = pbmc_small)[1:40])
pbmc2 <- SubsetData(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']])
```
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

```r
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'

Examples

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

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

**Usage**

```r
runpca(object, ...)
```

```r
## 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, ...)
```

```r
## 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, ...)
```

```r
## 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
runTSNE

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

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, dms = 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,
  dms = 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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>assay</td>
<td>Name of assay that t-SNE is being run on</td>
</tr>
<tr>
<td>seed.use</td>
<td>Random seed for the t-SNE</td>
</tr>
<tr>
<td>tsne.method</td>
<td>Select the method to use to compute the t-SNE. Available methods are:</td>
</tr>
<tr>
<td>add.iter</td>
<td>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</td>
</tr>
<tr>
<td>dim.embed</td>
<td>The dimensional space of the resulting tSNE embedding (default is 2). For example, set to 3 for a 3d tSNE</td>
</tr>
<tr>
<td>reduction.key</td>
<td>Dimensional reduction key, specifies the string before the number for the dimension names. tSNE_ by default</td>
</tr>
<tr>
<td>cells</td>
<td>Which cells to analyze (default, all cells)</td>
</tr>
<tr>
<td>dims</td>
<td>Which dimensions to use as input features</td>
</tr>
<tr>
<td>reduction</td>
<td>Which dimensional reduction (e.g. PCA, ICA) to use for the tSNE. Default is PCA</td>
</tr>
<tr>
<td>features</td>
<td>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</td>
</tr>
<tr>
<td>distance.matrix</td>
<td>If set, runs tSNE on the given distance matrix instead of data matrix (experimental)</td>
</tr>
<tr>
<td>reduction.name</td>
<td>Dimensional reduction name, specifies the position in the object$dr list. tsne by default</td>
</tr>
</tbody>
</table>

**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](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](https://arxiv.org/abs/1802.03426).
Usage

RunUMAP(object, ...)

### Default S3 method:
RunUMAP(object, assay = NULL, n.neighbors = 30L,
  n.components = 2L, metric = "correlation", 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, seed.use = 42, metric.kwds = NULL,
  angular.rp.forest = FALSE, reduction.key = "UMAP_", verbose = TRUE,
  ...)  

### S3 method for class 'Graph'
RunUMAP(object, assay = NULL, 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, seed.use = 42L,
  metric.kwds = NULL, verbose = TRUE, reduction.key = "UMAP_", ...)  

### S3 method for class 'Seurat'
RunUMAP(object, dims = NULL, reduction = "pca",
  features = NULL, graph = NULL, assay = "RNA", n.neighbors = 30L,
  n.components = 2L, metric = "correlation", 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, 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

assay Assay to pull data for when using features, or assay used to construct Graph if running UMAP on a Graph

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
The 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 to compress points together. Larger values ensure embedded points are more evenly 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.

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

```r
## 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)
**ScaleData**

**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**

```r
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 feature, and the resulting residuals are then scaled and centered.

**Usage**

```r
ScaleData(object, ...)
```

## Default S3 method:

```r
ScaleData(object, features = NULL,
  vars.to.regress = NULL, latent.data = 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'

```r
ScaleData(object, features = NULL,
  vars.to.regress = NULL, latent.data = 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'

```r
ScaleData(object, features = NULL, assay = NULL,
  vars.to.regress = 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 all 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
- **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 features 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 features 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 feature by subtracting the average expression for that feature. Setting scale to TRUE will scale the expression level for each feature by dividing the centered feature 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**

```r
ScoreJackStraw(object, ...)  
```

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

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

```r
## 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
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, 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
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

See Also

JackStrawPlot
JackStrawPlot
Description

Choose the features to use when integrating multiple datasets. This function ranks features by the number of datasets they appear in, breaking ties by the median rank across datasets. It returns the highest 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 of assay from which to pull the variable features.
- verbose: Print messages
**SetAssayData**

**Setter for multimodal data**

**Description**

Setter for multimodal data

**Usage**

```r
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 integration data*

**Description**  
Set integration 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.iden**: The active cluster identity for this Seurat object; settable using `Idents`.
- **graphs**: A list of `Graph-class` objects.
- **neighbors**: 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, analyses, 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`.
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 run
call.string String of the command call
params List of parameters used in the command call
SeuratTheme

**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.
... Ignored

Value

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

Examples

```r
# 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 arguments
... 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")
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

... arguments to be passed to [CalculateBarcodeInflections()]; if provided, will re-calculate the inflection points, else will use those already in `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**

```r
SubsetData(object, ...)  
```

```r  
## 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 retrieved 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**

```r
pbmc1 <- SubsetData(object = pbmc_small, cells = colnames(x = pbmc_small)[1:40])
pbmc1
data <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
mat_norm <- TF.IDF(data = mat)
```

---

**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**

```r
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**

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

```r
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 Labels

Description

Transfers the labels

Usage

TransferData(anchorse, refdata, weight.reduction = "pcaproject",
12.norm = FALSE, dims = 1:30, k.weight = 50, sd.weight = 1,
eps = 0, do.ccpp = TRUE, verbose = TRUE, slot = "data")
Arguments

anchorset
Results from 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. 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 PCs to use in the weighting procedure

k.weight
Number of neighbors to consider when weighting

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

Value

If refdata is a vector, returns a dataframe with label predictions. If refdata is a matrix, returns an Assay object where the imputed data has been stored in the provided slot.

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

```r
## Not run:
updated_seurat_object = UpdateSeuratObject(object = old_seurat_object)

## End(Not run)
```

---

**VariableFeaturePlot**  
*View variable features*

**Description**

View variable features

**Usage**

```r
VariableFeaturePlot(object, cols = c("black", "red"), pt.size = 1, 
log = 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
- `assay` Assay to pull variable features from

**Value**

A ggplot object

**See Also**

`FindVariableFeatures`

**Examples**

```r
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, ...)

## S3 method for class 'Seurat'
VariableFeatures(object, assay = 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
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 gg object; note that if TRUE; themeing will not work when plotting multiple features
- **...**: Ignored

Value

A ggplot object

Examples

```r
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

```r
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, combine = TRUE, slot = "data", ...)
```
**Arguments**

- **object**: Seurat object
- **features**: Features to plot (gene expression, metrics, PC scores, anything that can be retrieved 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, see FetchData for more details
- **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
- **combine**: Combine plots into a single gg object; note that if TRUE; themeing will not work when plotting multiple features
- **slot**: Use non-normalized counts data for plotting
- **...**: Extra parameters passed on to CombinePlots

**Value**

A ggplot object

**See Also**

FetchData

**Examples**

```r
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, etc.

Usage

WhichCells(object, ...)

## S3 method for class 'Assay'
WhichCells(object, cells = NULL, expression, invert = FALSE, ...)

## S3 method for class 'Seurat'
WhichCells(object, cells = NULL, idents = NULL, expression, slot = "data", invert = FALSE, downsample = Inf, seed = 1, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
<tr>
<td>cells</td>
<td>Subset of cell names</td>
</tr>
<tr>
<td>expression</td>
<td>A predicate expression for feature/variable expression, can evaluate anything that can be pulled by <code>FetchData</code>; please note, you may need to wrap feature names in backticks (``) if dashes between numbers are present in the feature name</td>
</tr>
<tr>
<td>invert</td>
<td>Invert the selection of cells</td>
</tr>
<tr>
<td>idents</td>
<td>A vector of identity classes to keep</td>
</tr>
<tr>
<td>slot</td>
<td>Slot to pull feature data for</td>
</tr>
<tr>
<td>downsample</td>
<td>Maximum number of cells per identity class, default is Inf; downsampling will happen after all other operations, including inverting the cell selection</td>
</tr>
<tr>
<td>seed</td>
<td>Random seed for downsampling</td>
</tr>
</tbody>
</table>

Value

A vector of cell names

See Also

FetchData
Examples

```r
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)
```
Examples

```r
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

*Topic datasets
  cc_genes, 21
  pbmc_small, 89
  [.Seurat, 140
  [[<], Assay-method (AddMetaData), 5
  [[<], Seurat-method (AddMetaData), 5

  AddMetaData, 5
  AddModuleScore, 6, 21
  AddSamples (merge.Assay), 82
  ALRAChooseKPlot, 7, 104
  AnchorSet (AnchorSet-class), 8
  Annotation_raster, 15
  as.data.frame.Matrix (as.sparse), 13
  as.Graph, 9
  as.loom, 9, 100
  as.Seurat, 11
  as.SingleCellExperiment, 13
  as.sparse, 13
  Assay (Assay-class), 14
  Assay-class, 14
  AugmentPlot, 15
  AverageExpression, 15

  BarcodeInflectionsPlot, 16, 20, 128
  BlackAndWhite, 17
  BlueAndRed (BlackAndWhite), 17
  BoldTitle (SeuratTheme), 124
  BuildClusterTree, 18, 48, 55

  CalculateBarcodeInflections, 17, 19, 128
  CaseMatch, 20
  cat, 94
  cc_genes, 21
  CellCycleScoring, 21
  CellPlot (CellScatter), 23
  Cells, 22
  CellScatter, 23
  CellSelector, 24, 36, 45

  CollapseSpeciesExpressionMatrix, 25
  CombinePlots, 26, 36, 103, 138
  Command, 27
  create, 11
  CreateAssayObject, 27
  CreateDimReducObject, 28
  CreateGeneActivityMatrix, 29
  CreateSeuratObject, 16, 30
  CustomDistance, 31
  CustomPalette (BlackAndWhite), 17

  DarkTheme (SeuratTheme), 124
  data.frame, 14
  DefaultAssay, 32, 122
  DefaultAssay<- (DefaultAssay), 32
  DietSeurat, 33
  DimHeatmap, 33
  DimPlot, 24, 35, 45, 64
  DimReduc (DimReduc-class), 36
  DimReduc-class, 36
  DoHeatmap, 37
  DotPlot, 38

  ElbowPlot, 39
  Embeddings, 40, 132
  ExpMean, 41
  ExportToCellbrowser, 41
  ExpSD, 43
  ExpVar, 43

  facet_grid, 77
  facet_wrap, 77
  FeatureHeatmap (FeaturePlot), 44
  FeatureLocator (CellSelector), 24
  FeaturePlot, 24, 36, 44, 64
  FeatureScatter, 45
  FetchData, 35, 39, 46, 68, 89, 93, 138, 139
  FindAllMarkers, 47
  FindAllMarkersNode (FindAllMarkers), 47
  FindClusters, 49
FindConservedMarkers, 51
FindIntegrationAnchors, 52
FindMarkers, 53
FindMarkersNode (FindMarkers), 53
FindNeighbors, 56
FindTransferAnchors, 58
FindVariableFeatures, 59, 120, 135
FindVariableGenes
  (FindVariableFeatures), 59
FontSize (SeuratTheme), 124

GenePlot (FeatureScatter), 45
geom_raster, 34
geom_text, 77, 78
geom_text_repel, 77
GetAssay, 61
GetAssayData, 62
GetIntegrationData, 63
ggplot_build, 24, 64
Graph (Graph-class), 63
Graph-class, 63

HoverLocator, 36, 45, 64
HTODemux, 64, 66
HTOHeatmap, 65, 66
HVFInfo, 67

ICAPlot (DimPlot), 35
Idents, 67, 122
Idents<- (Idents), 67
image, 34
IntegrateData, 8, 70
IntegrationData
  (IntegrationData-class), 71
IntegrationData-class, 71

JackStraw, 72, 90
JackStrawData (JackStrawData-class), 73
JackStrawData-class, 73
JackStrawPlot, 73, 118
JS, 74
JS<- (JS), 74

Key, 75
Key<- (Key), 75

L2CCA, 76
L2Dim, 76
LabelClusters, 77
Labeler (LabelPoints), 78

LabelPoints, 78
layout, 64
levels.Seurat (Idents), 67
levels<- .Seurat (Idents), 67
Loadings, 79, 133
Loadings<- (Loadings), 79
LocalStruct, 80
locator, 24
LogNormalize, 80
LogVMR, 81

make.names, 14
mean, 69
MeanVarPlot (VariableFeaturePlot), 135
merge (merge.Assay), 82
merge.Assay, 82
merge.Seurat, 106
MergeSeurat (merge.Assay), 82
MetaFeature, 83
MinMax, 83
Misc, 84
Misc<- (Misc), 84
MixingMetric, 85
MULTIseqDemux, 86

NoAxes (SeuratTheme), 124
NoLegend (SeuratTheme), 124
NormalizeData, 87

OldWhichCells, 88

pbmc_small, 89
PCAPlot (DimPlot), 35
PCASigGenes, 90
PCHeatmap (DimHeatmap), 33
PercentageFeatureSet, 91
PlotClusterTree, 92
pnt.in.poly, 24
PolyDimPlot, 92, 124
PolyFeaturePlot, 93, 124
print (print.DimReduc), 94
print.DimReduc, 94
Project, 94
Project<- (Project), 94
ProjectDim, 90, 95
PurpleAndYellow (BlackAndWhite), 17
RANN, 71