Package ‘Seurat’

March 18, 2024

Version 5.0.3
Date 2024-03-18
Title Tools for Single Cell Genomics


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

Additional_repositories

Depends R (>= 4.0.0),
        methods,
        SeuratObject (>= 5.0.0)

Imports cluster,
        cowplot,
        fastDummies,
        fitdistrplus,
        future,
        future.apply,
        generics (>= 0.1.3),
        ggplot2 (>= 3.3.0),
        ggrepel,
        ggridges,
        graphics,
        grDevices,
        grid,
        htr,
        ica,
        igraph,
irlba,
jsonlite,
KernSmooth,
leiden (>= 0.3.1),
lifecycle,
lmtest,
MASS,
Matrix (>= 1.5-0),
matrixStats,
miniUI,
patchwork,
pbapply,
plotly (>= 4.9.0),
png,
progressr,
purrr,
RANN,
RColorBrewer,
Rcpp (>= 1.0.7),
RcppAnnoy (>= 0.0.18),
RcppHNSW,
reticulate,
rlang,
ROCR,
RSpectra,
Rtsne,
scales,
scattermore (>= 1.2),
scattertransform (>= 0.4.1),
shiny,
spatstat.explore,
spatstat.geom,
stats,
tibble,
tools,
tools,
utils,
uwot (>= 0.1.10)

**LinkingTo** Rcpp (>= 0.11.0), RcppEigen, RcppProgress

**License** MIT + file LICENSE

**LazyData** true

**Collate** 'RcppExports.R'
  'reexports.R'
  'generics.R'
  'clustering.R'
  'visualization.R'
  'convenience.R'
  'data.R'
'differential_expression.R'
'dimensional_reduction.R'
'integration.R'
'zzz.R'
'integration5.R'
'mixscape.R'
'objects.R'
'preprocessing.R'
'preprocessing5.R'
'roxygen.R'
'sketching.R'
'tree.R'
'utilities.R'

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**Suggests** ape,
BPCells,
rsvd,
testthat,
hdf5r,
S4Vectors,
SummarizedExperiment,
SingleCellExperiment,
MAST,
DESeq2,
BiocGenerics,
GenomicRanges,
GenomeInfoDb,
IRanges,
rtracklayer,
Rfast2,
monocle,
Biobase,
VGAM,
limma,
metap,
enrichR,
mixtools,
ggrastr,
data.table,
R.utils,
presto,
DelayedArray,
harmony
R topics documented:

Seurat-package .................................................. 8
AddAzimuthResults ........................................... 10
AddAzimuthScores ............................................. 11
AddModuleScore .................................................. 11
AggregateExpression ........................................... 13
AnchorSet-class .................................................. 15
AnnotateAnchors .................................................. 15
as.CellDataSet ................................................... 16
as.Seurat.CellDataSet .......................................... 17
as.SingleCellExperiment ....................................... 18
Assay-class .......................................................... 18
AugmentPlot .......................................................... 19
AutoPointSize ...................................................... 20
AverageExpression .............................................. 21
BarcodeInflectionsPlot ........................................... 22
BGTextColor ......................................................... 23
BlackAndWhite ..................................................... 24
BridgeCellsRepresentation ..................................... 25
BridgeReferenceSet-class ....................................... 26
BuildClusterTree .................................................. 26
BuildNicheAssay ................................................... 28
CalcPerturbSig .................................................... 28
CalculateBarcodeInflections .................................... 30
CaseMatch .......................................................... 31
cc.genes .............................................................. 32
cc.genes.updated.2019 ........................................... 32
CCAIntegration .................................................... 33
CellCycleScoring .................................................. 35
Cells.SCTModel ................................................... 37
CellsByImage ....................................................... 37
CellScatter .......................................................... 38
CellSelector .......................................................... 39
CollapseEmbeddingOutliers .................................... 40
CollapseSpeciesExpressionMatrix .............................. 41
ColorDimSplit ....................................................... 42
CombinePlots ....................................................... 44
contrast-theory ................................................... 45
CreateCategoryMatrix ............................................ 46
CreateSCTAssayObject .......................................... 46
CustomDistance ................................................... 47
DEenrichRPlot ..................................................... 48
DietSeurat .......................................................... 50
DimHeatmap .......................................................... 51
DimPlot .............................................................. 52
DimReduc-class ................................................... 55
R topics documented:

DiscretePalette .............................................. 55
DoHeatmap ....................................................... 56
DotPlot .......................................................... 58
ElbowPlot ......................................................... 59
ExpMean ........................................................... 60
ExpSD .............................................................. 60
ExpVar ............................................................. 61
FastRowScale ..................................................... 62
FastRPCAIntegration ........................................... 62
FeaturePlot ......................................................... 63
FeatureScatter ................................................... 66
FetchResiduals .................................................... 68
FetchResidualSCTModel ........................................ 69
FetchResiduals_reference ....................................... 70
FilterSlideSeq ................................................... 71
FindAllMarkers ................................................ 72
FindBridgeAnchor .............................................. 75
FindBridgeIntegrationAnchors ................................. 77
FindBridgeTransferAnchors .................................... 78
FindClusters ....................................................... 79
FindConservedMarkers ......................................... 81
FindIntegrationAnchors ......................................... 82
FindMarkers ....................................................... 85
FindMultiModalNeighbors ...................................... 91
FindNeighbors .................................................... 93
FindSpatiallyVariableFeatures ................................ 96
FindSubCluster .................................................. 98
FindTransferAnchors ........................................... 99
FindVariableFeatures .......................................... 102
FoldChange ....................................................... 105
GetAssay ......................................................... 107
GetImage.SlideSeq ............................................... 108
GetIntegrationData ............................................. 109
GetResidual ....................................................... 109
GetTissueCoordinates.SlideSeq ................................. 110
GetTransferPredictions ........................................ 111
Graph-class ....................................................... 112
GroupCorrelation ............................................... 112
GroupCorrelationPlot .......................................... 113
HarmonyIntegration ............................................. 114
HoverLocator ..................................................... 116
HTODemux ......................................................... 117
HTOHeatmap ....................................................... 118
HVFInfo.SCTAssay ............................................... 120
IFeaturePlot ....................................................... 120
ImageDimPlot ..................................................... 121
ImageFeaturePlot ............................................... 123
IntegrateData .................................................... 125
IntegrateEmbeddings .................................................. 128
IntegrateLayers ...................................................... 130
IntegrationAnchorSet-class ......................................... 131
IntegrationData-class ................................................. 131
ISpatialDimPlot ....................................................... 132
ISpatialFeaturePlot .................................................. 133
JackStraw ............................................................. 133
JackStrawData-class .................................................. 135
JackStrawPlot ......................................................... 135
JointPCAIntegration ................................................... 136
L2CCA ................................................................. 138
L2Dim ................................................................. 138
LabelClusters .......................................................... 139
LabelPoints ............................................................ 140
LeverageScore .......................................................... 141
LinkedPlots ............................................................ 143
Load10X_Spatial ....................................................... 144
LoadAnnoyIndex ....................................................... 145
LoadCurioSeeker ....................................................... 146
LoadSTARmap ........................................................... 146
LoadXenium ............................................................ 147
LocalStruct ............................................................ 148
LogNormalize .......................................................... 149
LogVMR ................................................................. 150
MappingScore .......................................................... 151
MapQuery ............................................................... 153
merge.SCTAssay ....................................................... 154
MetaFeature ............................................................ 155
MinMax ................................................................. 156
MixingMetric ............................................................ 157
MixscapeHeatmap ...................................................... 158
MixscapeLDA ............................................................. 160
ModalityWeights-class ............................................... 161
MULTIseqDemux .......................................................... 161
Neighbor-class .......................................................... 163
NNPlot ................................................................. 163
NNtoGraph .............................................................. 164
NormalizeData ........................................................... 165
PCASigGenes ............................................................ 166
PercentAbove ............................................................ 167
PercentageFeatureSet ................................................... 168
PlotClusterTree ......................................................... 169
PlotPerturbScore ....................................................... 169
PolyDimPlot ............................................................. 170
PolyFeaturePlot ......................................................... 171
PredictAssay ............................................................. 172
PrepareBridgeReference ............................................... 173
PrepLDA ................................................................. 175
R topics documented:

PrepSCTFindMarkers .................................................. 176
PrepSCTIntegration ..................................................... 177
ProjectData ................................................................. 179
ProjectDim ................................................................. 180
ProjectDimReduce ......................................................... 181
ProjectIntegration ......................................................... 182
ProjectUMAP ................................................................. 183
PseudobulkExpression ..................................................... 185
Radius.SlideSeq .............................................................. 186
Read10X ...................................................................... 186
Read10X_h5 ................................................................. 188
Read10X_Image .............................................................. 188
Read10X_probe_metadata .................................................... 189
ReadAkoya ................................................................. 189
ReadMtx ................................................................. 191
ReadNanostring ............................................................ 192
ReadParseBio .............................................................. 194
ReadSlideSeq .............................................................. 195
ReadSTARsolo .............................................................. 195
ReadVitessce .............................................................. 196
ReadVizgen ................................................................. 197
RegroupIdsents ............................................................. 199
RelativeCounts ............................................................. 200
RenameCells.SCTAssay ..................................................... 201
RidgePlot ................................................................. 201
RPCAIntegration ........................................................... 203
RunCCA ................................................................. 205
RunGraphLaplacian ........................................................ 207
RunICA ................................................................. 208
RunLDA ................................................................. 210
RunMarkVario .............................................................. 211
RunMixscape .............................................................. 212
RunMoransI ............................................................... 213
RunPCA ................................................................. 214
RunSLSI ................................................................. 216
RunSPCA ................................................................. 217
RunTSNE ................................................................. 219
RunUMAP ................................................................. 221
SampleUMI ............................................................... 226
SaveAnnoyIndex .......................................................... 227
ScaleData ................................................................. 228
ScaleFactors .............................................................. 230
ScoreJackStraw ............................................................ 231
SCTAssay-class ............................................................ 232
SCTTransform ............................................................. 234
SCTResults ............................................................... 237
SelectIntegrationFeatures ................................................ 238
SelectIntegrationFeatures5 ............................................... 240
Seurat-package

Description


Package options

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

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

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

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

Seurat.Rfast2.msg  Show message about more efficient Moran’s I function available via the Rfast2 package

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

Author(s)

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

Other contributors:

• Andrew Butler <abutler@nygenome.org> (ORCID) [contributor]
• Saket Choudhary <schoudhary@nygenome.org> (ORCID) [contributor]
• Charlotte Darby <cdarby@nygenome.org> (ORCID) [contributor]
• Jeff Farrell <jfarrell@g.harvard.edu> [contributor]
• Isabella Grabski <igrabski@nygenome.org> (ORCID) [contributor]
• Christoph Hafemeister <chafemeister@nygenome.org> (ORCID) [contributor]
• Yuhan Hao <yhao@nygenome.org> (ORCID) [contributor]
• Austin Hartman <ahartman@nygenome.org> (ORCID) [contributor]
• Paul Hoffman <hoff0792@umn.edu> (ORCID) [contributor]
• Jaison Jain <jjain@nygenome.org> (ORCID) [contributor]
• Longda Jiang <ljiang@nygenome.org> (ORCID) [contributor]
• Madeline Kowalski <mkowalski@nygenome.org> (ORCID) [contributor]
• Skylar Li <sli@nygenome.org> [contributor]
• Gesmira Molla <gmolla@nygenome.org> (ORCID) [contributor]
• Efthymia Papalexi <epapalexi@nygenome.org> (ORCID) [contributor]
• Patrick Roelli <proelli@nygenome.org> [contributor]
• Karthik Shekhar <kshekhar@berkeley.edu> [contributor]
• Avi Srivastava <asrivastava@nygenome.org> (ORCID) [contributor]
• Tim Stuart <tstuart@nygenome.org> (ORCID) [contributor]
• Kristof Torkenczy (ORCID) [contributor]
• Shiwei Zheng <szheng@nygenome.org> (ORCID) [contributor]
• Satija Lab and Collaborators [funder]
## AddAzimuthResults

### Description

Add mapping and prediction scores, UMAP embeddings, and imputed assay (if available) from Azimuth to an existing or new `Seurat` object.

### Usage

```r
AddAzimuthResults(object = NULL, filename)
```

### Arguments

- **object**: A `Seurat` object.
- **filename**: Path to Azimuth mapping scores file.

### Value

- `object` with Azimuth results added.

### Examples

```r
## Not run:
object <- AddAzimuthResults(object, filename = "azimuth_results.Rds")
## End(Not run)
```

---

**See Also**

Useful links:

- [https://satijalab.org/seurat](https://satijalab.org/seurat)
- [https://github.com/satijalab/seurat](https://github.com/satijalab/seurat)
- Report bugs at [https://github.com/satijalab/seurat/issues](https://github.com/satijalab/seurat/issues)
**AddAzimuthScores**

**Description**
Add mapping and prediction scores from Azimuth to a Seurat object

**Usage**
AddAzimuthScores(object, filename)

**Arguments**
- object: A Seurat object
- filename: Path to Azimuth mapping scores file

**Value**
object with the mapping scores added

**Examples**
```r
## Not run:
object <- AddAzimuthScores(object, filename = "azimuth_pred.tsv")
## End(Not run)
```

---

**AddModuleScore**

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

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

**Usage**
AddModuleScore(
  object,
  features,
  pool = NULL,
  nbin = 24,
  ctrl = 100,
)
AddModuleScore

k = FALSE,
assay = NULL,
name = "Cluster",
seed = 1,
search = FALSE,
slot = "data",
...
)

Arguments

object Seurat object
features A list of vectors of features for expression programs; each entry should be a vector of feature names
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; will append a number to the end for each entry in features (eg. if features has three programs, the results will be stored as name1, name2, name3, respectively)
seed Set a random seed. If NULL, seed is not set.
search Search for symbol synonyms for features in features that don’t match features in object? Searches the HGNC’s gene names database; see UpdateSymbolList for more details
slot Slot to calculate score values off of. Defaults to data slot (i.e log-normalized counts)
... Extra parameters passed to UpdateSymbolList

Value

Returns a Seurat object with module scores added to object meta data; each module is stored as name# for each module program present in features

References

Tirosh et al, Science (2016)

Examples

## Not run:
data("pbmc_small")
cd_features <- list(c("Var CD79B",
...
pbmc_small <- AddModuleScore(
  object = pbmc_small,
  features = cd_features,
  ctrl = 5,
  name = 'CD_Features'
)
head(x = pbmc_small[])

## End(Not run)

---

**AggregateExpression**

*Aggregated feature expression by identity class*

**Description**

Returns summed counts ("pseudobulk") for each identity class.

**Usage**

```r
AggregateExpression(
  object,
  assays = NULL,
  features = NULL,
  return.seurat = FALSE,
  group.by = "ident",
  add.ident = NULL,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  verbose = TRUE,
  ...
)
```
AggregateExpression

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
group.by  Category (or vector of categories) for grouping (e.g, ident, replicate, celltype); 'ident' by default To use multiple categories, specify a vector, such as c('ident', 'replicate', 'celltype')
add.ident  (Deprecated). Place an additional label on each cell prior to pseudobulking
normalization.method  Method for normalization, see NormalizeData
scale.factor  Scale factor for normalization, see NormalizeData
margin  Margin to perform CLR normalization, see NormalizeData
verbose  Print messages and show progress bar
...  Arguments to be passed to methods such as CreateSeuratObject

Details

If return.seurat = TRUE, aggregated values are placed in the 'counts' layer of the returned object. The data is then normalized by running NormalizeData on the aggregated counts. ScaleData is then run on the default assay before returning the object.

Value

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

Examples

```r
## Not run:
data("pbmc_small")
head(AggregateExpression(object = pbmc_small)$RNA)
head(AggregateExpression(object = pbmc_small, group.by = c('ident', 'groups'))$RNA)
## End(Not run)
```
The AnchorSet Class

Description

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

Slots

- **object.list**: List of objects used to create anchors
- **reference.cells**: List of cell names in the reference dataset - needed when performing data transfer.
- **reference.objects**: Position of reference object/s in object.list
- **query.cells**: List of cell names in the query dataset - needed when performing data transfer
- **anchors**: The anchor matrix. This contains the cell indices of both anchor pair cells, the anchor score, and the index of the original dataset in the object.list for cell1 and cell2 of the anchor.
- **offsets**: The offsets used to enable cell look up in downstream functions
- **weight.reduction**: The weight dimensional reduction used to calculate weight matrix
- **anchor.features**: The features used when performing anchor finding.
- **neighbors**: List containing Neighbor objects for reuse later (e.g. mapping)
- **command**: Store log of parameters that were used

AnnotateAnchors

Add info to anchor matrix

Description

Add info to anchor matrix

Usage

AnnotateAnchors(anchors, vars, slot, ...)

## Default S3 method:
AnnotateAnchors(
  anchors,
  vars = NULL,
  slot = NULL,
  object.list,
  assay = NULL,
Arguments

- **anchors**: An `AnchorSet` object
- **vars**: Variables to pull for each object via `FetchData`
- **slot**: Slot to pull feature data for
- **object.list**: List of Seurat objects
- **assay**: Specify the Assay per object if annotating with expression data
- **reference**: Reference object used in `FindTransferAnchors`
- **query**: Query object used in `FindTransferAnchors`

Value

Returns the anchor dataframe with additional columns for annotation metadata

---

**as.CellDataSet**  
*Convert objects to CellDataSet objects*

**Description**

Convert objects to CellDataSet objects
Usage

as.CellDataSet(x, ...)

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

Arguments

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

as.Seurat.CellDataSet Convert objects to Seurat objects

Description

Convert objects to Seurat objects

Usage

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

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

Arguments

x An object to convert to class Seurat
slot Slot to store expression data as
assay Name of assays to convert; set to NULL for all assays to be converted
verbose Show progress updates
... Arguments passed to other methods
counts name of the SingleCellExperiment assay to store as counts; set to NULL if only normalized data are present
data name of the SingleCellExperiment assay to slot as data. Set to NULL if only counts are present
project Project name for new Seurat object
Value

A Seurat object generated from x

See Also

SeuratObject::as.Seurat

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 Assays to convert

as.sparse.H5Group Cast to Sparse

Description

Cast to Sparse

Usage

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

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

- **x**: An object
- **...**: Arguments passed to other methods
- **row.names**: NULL or a character vector giving the row names for the data; missing values are not allowed
- **optional**: logical. If TRUE, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R’s base package `as.data.frame()` methods use optional only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`. See also the `make.names` argument of the `matrix` method.
- **stringsAsFactors**: logical: should the character vector be converted to a factor?

**Value**

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

**See Also**

`SeuratObject::as.sparse`

---

**Assay-class**

The Assay Class

**Description**

The Assay object is the basic unit of Seurat; for more details, please see the documentation in `SeuratObject`

**See Also**

`SeuratObject::Assay-class`

---

**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)`
AutoPointSize

Arguments

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

Value

A ggplot object

Examples

## Not run:
data("pbmc_small")
plot <- DimPlot(object = pbmc_small)
AugmentPlot(plot = plot)
## End(Not run)

AutoPointSize(data, raster = NULL)

Arguments

data A data frame being passed to ggplot2
raster If TRUE, point size is set to 1

Value

The "optimal" point size for visualizing these data

Examples

df <- data.frame(x = rnorm(n = 10000), y = runif(n = 10000))
AutoPointSize(data = df)
AverageExpression

Averaged feature expression by identity class

Description

Returns averaged expression values for each identity class.

Usage

AverageExpression(
  object,
  assays = NULL,
  features = NULL,
  return.seurat = FALSE,
  group.by = "ident",
  add.ident = NULL,
  layer = "data",
  slot = deprecated(),
  verbose = TRUE,
  ...
)

Arguments

object Seurat object
assays Which assays to use. Default is all assays
features Features to analyze. Default is all features in the assay
return.seurat Whether to return the data as a Seurat object. Default is FALSE
group.by Category (or vector of categories) for grouping (e.g, ident, replicate, celltype);
  'ident' by default To use multiple categories, specify a vector, such as c('ident',
  'replicate', 'celltype')
add.ident (Deprecated). Place an additional label on each cell prior to pseudobulking
layer Layer(s) to use; if multiple layers are given, assumed to follow the order of
  'assays' (if specified) or object’s assays
slot (Deprecated). Slots(s) to use
verbose Print messages and show progress bar
...
Arguments to be passed to methods such as CreateSeuratObject

Details

If layer is set to 'data', this function assumes that the data has been log normalized and there-fore feature values are exponentiated prior to averaging so that averaging is done in non-log space. Otherwise, if layer is set to either 'counts' or 'scale.data', no exponentiation is performed prior to averaging. If return.seurat = TRUE and layer is not 'scale.data', averaged values are placed
in the 'counts' layer of the returned object and 'log1p' is run on the averaged counts and placed in the 'data' layer. `ScaleData` is then run on the default assay before returning the object. If `return.seurat = TRUE` and layer is 'scale.data', the 'counts' layer contains average counts and 'scale.data' is set to the averaged values of 'scale.data'.

Value

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

Examples

```r
data("pbmc_small")
head(AverageExpression(object = pbmc_small)$RNA)
head(AverageExpression(object = pbmc_small, group.by = c('ident', 'groups'))$RNA)
```

BarcodeInflectionsPlot

Plot the Barcode Distribution and Calculated Inflection Points

Description

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

Usage

```r
BarcodeInflectionsPlot(object)
```

Arguments

- **object** Seurat object

Details

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

Value

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

Author(s)

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

Determine text color based on background color

### Usage

```r
BGTextColor(
  background,
  threshold = 186,
  w3c = FALSE,
  dark = "black",
  light = "white"
)
```

### Arguments

- `background`: A vector of background colors; supports R color names and hexadecimal codes
- `threshold`: Intensity threshold for light/dark cutoff; intensities greater than `threshold` yield dark, others yield light
- `w3c`: Use W3C formula for calculating background text color; ignores `threshold`
- `dark`: Color for dark text
- `light`: Color for light text

### Value

A named vector of either dark or light, depending on background; names of vector are `background`

### Source


### Examples

```r
BGTextColor(background = c('black', 'white', '#E76BF3'))
```
Create a custom color palette

Description

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

Usage

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

Arguments

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

Value

A color palette for plotting

Examples

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

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

myPalette <- CustomPalette()
myPalette

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

Construct a dictionary representation for each unimodal dataset

Description

Construct a dictionary representation for each unimodal dataset

Usage

BridgeCellsRepresentation(
  object.list,
  bridge.object,
  object.reduction,
  bridge.reduction,
  laplacian.reduction = "lap",
  laplacian.dims = 1:50,
  bridge.assay.name = "Bridge",
  return.all.assays = FALSE,
  l2.norm = TRUE,
  verbose = TRUE
)

 Arguments

object.list A list of Seurat objects
bridge.object A multi-omic bridge Seurat which is used as the basis to represent unimodal datasets
object.reduction A list of dimensional reductions from object.list used to be reconstructed by bridge.object
bridge.reduction A list of dimensional reductions from bridge.object used to reconstruct object.reduction
laplacian.reduction Name of bridge graph laplacian dimensional reduction
laplacian.dims Dimensions used for bridge graph laplacian dimensional reduction
bridge.assay.name Assay name used for bridge object reconstruction value (default is 'Bridge')
return.all.assays Whether to return all assays in the object.list. Only bridge assay is returned by default.
l2.norm Whether to l2 normalize the dictionary representation
verbose Print messages and progress
BuildClusterTree

Value
Returns a object list in which each object has a bridge cell derived assay

BridgeReferenceSet-class
The BridgeReferenceSet Class The BridgeReferenceSet is an output from PrepareBridgeReference

Description
The BridgeReferenceSet Class The BridgeReferenceSet is an output from PrepareBridgeReference

Slots
bridge The multi-omic object
reference The Reference object only containing bridge representation assay
params A list of parameters used in the PrepareBridgeReference command Store log of parameters that were used

BuildClusterTree Phylogenetic Analysis of Identity Classes

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

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

Arguments

- object: Seurat object
- assay: Assay to use for the analysis.
- features: Genes to use for the analysis. Default is the set of variable genes (VariableFeatures(object = object))
- dims: If set, tree is calculated in dimension reduction space; overrides features
- reduction: Name of dimension reduction to use. Only used if dims is not NULL.
- graph: If graph is passed, build tree based on graph connectivity between clusters; overrides dims and features
- slot: (Deprecated). Slots(s) to use
- reorder: Re-order identity classes (factor ordering), according to position on the tree. This groups similar classes together which can be helpful, for example, when drawing violin plots.
- reorder.numeric: Re-order identity classes according to position on the tree, assigning a numeric value ('1' is the leftmost node)
- verbose: Show progress updates

Details

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

Value

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

Examples

```r
## Not run:
if (requireNamespace("ape", quietly = TRUE)) {
  data("pbmc_small")
  pbmc_small <- BuildClusterTree(object = pbmc_small)
  Tool(object = pbmc_small, slot = 'BuildClusterTree')
}
## End(Not run)
```
BuildNicheAssay  
*Construct an assay for spatial niche analysis*

**Description**

This function will construct a new assay where each feature is a cell label. The values represent the sum of a particular cell label neighboring a given cell.

**Usage**

```r
BuildNicheAssay(
  object,  
  fov,  
  group.by,  
  assay = "niche",  
  neighbors.k = 20,  
  niches.k = 4
)
```

**Arguments**

- **object**  
  A Seurat object

- **fov**  
  FOV object to gather cell positions from

- **group.by**  
  Cell classifications to count in spatial neighborhood

- **assay**  
  Name for spatial neighborhoods assay

- **neighbors.k**  
  Number of neighbors to consider for each cell

- **niches.k**  
  Number of clusters to return based on the niche assay

**Value**

Seurat object containing a new assay

---

CalcPerturbSig  
*Calculate a perturbation Signature*

**Description**

Function to calculate perturbation signature for pooled CRISPR screen datasets. For each target cell (expressing one target gRNA), we identified 20 cells from the control pool (non-targeting cells) with the most similar mRNA expression profiles. The perturbation signature is calculated by subtracting the averaged mRNA expression profile of the non-targeting neighbors from the mRNA expression profile of the target cell.
CalcPerturbSig

Usage

CalcPerturbSig(
  object,
  assay = NULL,
  features = NULL,
  slot = "data",
  gd.class = "guide_ID",
  nt.cell.class = "NT",
  split.by = NULL,
  num.neighbors = NULL,
  reduction = "pca",
  ndims = 15,
  new.assay.name = "PRTB",
  verbose = TRUE
)

Arguments

object    An object of class Seurat.
assay     Name of Assay PRTB signature is being calculated on.
features  Features to compute PRTB signature for. Defaults to the variable features set in
           the assay specified.
slot      Data slot to use for PRTB signature calculation.
gd.class  Metadata column containing target gene classification.
nt.cell.class  Non-targeting gRNA cell classification identity.
split.by  Provide metadata column if multiple biological replicates exist to calculate PRTB
           signature for every replicate separately.
um.neighbors Number of nearest neighbors to consider.
reduction Reduction method used to calculate nearest neighbors.
ndims     Number of dimensions to use from dimensionality reduction method.
new.assay.name Name for the new assay.
verbose   Display progress + messages

Value

Returns a Seurat object with a new assay added containing the perturbation signature for all cells in
the data slot.
CalculateBarcodeInflections

*Calculate the Barcode Distribution Inflection*

**Description**

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

**Usage**

```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
CaseMatch

Author(s)
Robert A. Amezquita, <robert.amezquita@fredhutch.org>

See Also
BarcodeInflectionsPlot SubsetByBarcodeInflections

Examples

data("pbmc_small")
CalculateBarcodeInflections(pbmc_small, group.column = 'groups')

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

CaseMatch

Match the case of character vectors

Description

Match the case of character vectors

Usage

CaseMatch(search, match)

Arguments

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

Value

Values from search present in match with the case of match

Examples

data("pbmc_small")
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**


---

**cc.genes.updated.2019**

*Cell cycle genes: 2019 update*

---

**Description**

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

**Usage**

cc.genes.updated.2019

**Format**

A list of two vectors

- **s.genes** Genes associated with S-phase
- **g2m.genes** Genes associated with G2M-phase
Updated symbols

The following symbols were updated from `cc.genes`

`s.genes`  
• `MCM2`: `MCM7`  
• `MLF1IP`: `CENPU`  
• `RPA2`: `POLR1B`  
• `BRIP1`: `MRPL36`

`g2m.genes`  
• `FAM64A`: `PIMREG`  
• `HN1`: `JPT1`

Source

https://www.science.org/doi/abs/10.1126/science.aad0501

See Also

`cc.genes`

Examples

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

## End(Not run)
```

CCAIntegration | Seurat-CCA Integration

Description

Seurat-CCA Integration

Usage

```r
CCAIntegration(
  object = NULL,
  assay = NULL,
  layers = NULL,
  orig = NULL,
  new.reduction = "integrated.dr",
  reference = NULL,
  features = NULL,
  normalization.method = c("LogNormalize", "SCT"),
  dims = 1:30,
)```
k.filter = NA,
scale.layer = "scale.data",
dims.to.integrate = NULL,
k.weight = 100,
weight.reduction = NULL,
sd.weight = 1,
sample.tree = NULL,
preserve.order = FALSE,
verbose = TRUE,
...

Arguments

object        A Seurat object
assay         Name of Assay in the Seurat object
layers        Names of layers in assay
orig          A dimensional reduction to correct
new.reduction Name of new integrated dimensional reduction
reference     A reference Seurat object
features      A vector of features to use for integration
normalization.method
               Name of normalization method used: LogNormalize or SCT
dims          Dimensions of dimensional reduction to use for integration
k.filter      Number of anchors to filter
scale.layer   Name of scaled layer in Assay
dims.to.integrate
               Number of dimensions to return integrated values for
k.weight      Number of neighbors to consider when weighting anchors
weight.reduction
               Dimension reduction to use when calculating anchor weights. This can be one of:
               • A string, specifying the name of a dimension reduction present in all objects
to be integrated
               • A vector of strings, specifying the name of a dimension reduction to use for
each object to be integrated
               • A vector of DimReduc objects, specifying the object to use for each object
in the integration
               • NULL, in which case the full corrected space is used for computing anchor
weights.
sd.weight     Controls the bandwidth of the Gaussian kernel for weighting
sample.tree   Specify the order of integration. Order of integration should be encoded in a ma-
               trix, where each row represents one of the pairwise integration steps. Negative
numbers specify a dataset, positive numbers specify the integration results from
a given row (the format of the merge matrix included in the `hclust` function output). For example: matrix(c(-2, 1, -3, -1), ncol = 2) gives:

```
[,1] [,2]
[1,] -2 -3
[2,] 1  -1
```

Which would cause dataset 2 and 3 to be integrated first, then the resulting object integrated with dataset 1.

If NULL, the sample tree will be computed automatically.

- `preserve.order` Do not reorder objects based on size for each pairwise integration.
- `verbose` Print progress
- `...` Arguments passed on to `FindIntegrationAnchors`

### Examples

```r
## Not run:
# Preprocessing
obj <- SeuratData::LoadData("pbmcscsa")
obj[["RNA"]]<- split(obj[["RNA"]], f = obj$Method)
obj <- NormalizeData(obj)
obj <- FindVariableFeatures(obj)
obj <- ScaleData(obj)
obj <- RunPCA(obj)

# After preprocessing, we integrate layers.
obj <- IntegrateLayers(object = obj, method = CCAIntegration,
orig.reduction = "pca", new.reduction = "integrated.cca",
verbose = FALSE)

# Modifying parameters
# We can also specify parameters such as `k.anchor` to increase the strength of integration
obj <- IntegrateLayers(object = obj, method = CCAIntegration,
orig.reduction = "pca", new.reduction = "integrated.cca",
k.anchor = 20, verbose = FALSE)

# Integrating SCTransformed data
obj <- SCTransform(object = obj)
obj <- IntegrateLayers(object = obj, method = CCAIntegration,
orig.reduction = "pca", new.reduction = "integrated.cca",
assay = "SCT", verbose = FALSE)

## End(Not run)
```
Description

Score cell cycle phases

Usage

CellCycleScoring(
  object,
  s.features,
  g2m.features,
  ctrl = NULL,
  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
ctrl Number of control features selected from the same bin per analyzed feature supplied to AddModuleScore. Defaults to value equivalent to minimum number of features present in 's.features' and 'g2m.features'.
set.ident If true, sets identity to phase assignments Stashes old identities in 'old.ident'
... Arguments to be passed to AddModuleScore

Value

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

See Also

AddModuleScore

Examples

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

## End(Not run)
Cells.SCTModel  Get Cell Names

Description

Get Cell Names

Usage

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

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

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

## S3 method for class 'VisiumV1'
Cells(x, ...)

Arguments

x             An object
...

Arguments passed to other methods

See Also

SeuratObject::Cells

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

Description

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

Usage

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

Arguments

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

Value

A vector of cell names

Examples

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

CellScatter  

Cell-cell scatter plot

Description

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

Usage

CellScatter(
  object,
  cell1,
  cell2,
  features = NULL,
  highlight = NULL,
  cols = NULL,
  pt.size = 1,
  smooth = FALSE,
  raster = NULL,
  raster.dpi = c(512, 512)
)

Arguments

object Seurat object
cell1 Cell 1 name
cell2 Cell 2 name
CellSelector

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)
raster: Convert points to raster format, default is NULL which will automatically use raster if the number of points plotted is greater than 100,000
raster.dpi: Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).

Value
A ggplot object

Examples

data("pbmc_small")
CellScatter(object = pbmc_small, cell1 = "ATAGGAGAACAGA", cell2 = "CATCAGGATGCACA")

CellSelector

Cell Selector

Description
Select points on a scatterplot and get information about them

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

FeatureLocator(plot, ...)

Arguments

plot: A ggplot2 plot
object: An optional Seurat object; if passes, will return an object with the identities of selected cells set to ident
ident: An optional new identity class to assign the selected cells
...: Ignored

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

Move outliers towards center on dimension reduction plot

Description

Move outliers towards center on dimension reduction plot

Usage

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

Arguments

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

See Also

  DimPlot FeaturePlot

Examples

## Not run:
data("pbmc_small")
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)
Value

Returns a DimReduc object with the modified embeddings

Examples

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

---

### ColorDimSplit

**Color dimensional reduction plot by tree split**

Description

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

Usage

```
ColorDimSplit(
  object,  # Seurat object
  node,    # Node in cluster tree on which to base the split
  left.color = "red",  # Color for the left side of the split
  right.color = "blue",  # Color for the right side of the split
  other.color = "grey50",  # Color for all other cells
  ...  # Arguments passed on to DimPlot
)
```

Arguments

- **object**: Seurat object
- **node**: Node in cluster tree on which to base the split
- **left.color**: Color for the left side of the split
- **right.color**: Color for the right side of the split
- **other.color**: Color for all other cells
- **...**: Arguments passed on to DimPlot
- **dims**: Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions
- **cells**: Vector of cells to plot (default is all cells)
cols  Vector of colors, each color corresponds to an identity class. This may also be a single character or numeric value corresponding to a palette as specified by `brewer.pal.info`. By default, ggplot2 assigns colors. We also include a number of palettes from the pals package. See `DiscretePalette` for details.

pt.size  Adjust point size for plotting

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

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

split.by  A factor in object metadata to split the plot by, pass 'ident' to split by cell identity

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. Only applicable if raster = FALSE.

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)

shuffle  Whether to randomly shuffle the order of points. This can be useful for crowded plots if points of interest are being buried. (default is FALSE)

seed  Sets the seed if randomly shuffling the order of points.

label  Whether to label the clusters

label.size  Sets size of labels

label.color  Sets the color of the label text

label.box  Whether to put a box around the label text (geom_text vs geom_label)

alpha  Alpha value for plotting (default is 1)

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. If sizes.highlight = TRUE size of all points will be this value.

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

ncol  Number of columns for display when combining plots

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

raster  Convert points to raster format, default is NULL which automatically rasterizes if plotting more than 100,000 cells

raster.dpi  Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).
CombinePlots

Combine ggplot2-based plots into a single plot

Description

Combine ggplot2-based plots into a single plot

Usage

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

Arguments

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

Value

A combined plot

See Also

DimPlot

Examples

## Not run:
if (requireNamespace("ape", quietly = TRUE)) {
  data("pbmc_small")
  pbmc_small <- BuildClusterTree(object = pbmc_small, verbose = FALSE)
  PlotClusterTree(pbmc_small)
  ColorDimSplit(pbmc_small, node = 5)
}
## End(Not run)
Examples

data("pbmc_small")
pbmc_small[['group']] <- sample(
  x = c('g1', 'g2'),
  size = ncol(x = pbmc_small),
  replace = TRUE
)
plot1 <- FeaturePlot(
  object = pbmc_small,
  features = 'MS4A1',
  split.by = 'group'
)
plot2 <- FeaturePlot(
  object = pbmc_small,
  features = 'FCN1',
  split.by = 'group'
)
CombinePlots(
  plots = list(plot1, plot2),
  legend = 'none',
  nrow = length(x = unique(x = pbmc_small[['group']]), drop = TRUE))

---

contrast-theory

Get the intensity and/or luminance of a color

Description

Get the intensity and/or luminance of a color

Usage

Intensity(color)

Luminance(color)

Arguments

  color       A vector of colors

Value

  A vector of intensities/luminances for each color

Source

https://stackoverflow.com/questions/3942878/how-to-decide-font-color-in-white-or-black-depending-on-background-color
CreateSCTAssayObject

Create a SCT Assay object

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

Usage
CreateSCTAssayObject(
  counts,
  data,
  scale.data = NULL,
  umi.assay = "RNA",
  min.cells = 0,
  min.features = 0,
  SCTModel.list = NULL
)

CreateCategoryMatrix
Create one hot matrix for a given label

Description
Create one hot matrix for a given label

Usage
CreateCategoryMatrix(
  labels,
  method = c("aggregate", "average"),
  cells.name = NULL
)

Arguments
labels A vector of labels
method Method to aggregate cells with the same label. Either 'aggregate' or 'average'
cells.name A vector of cell names

Examples
Intensity(color = c('black', 'white', '#E76BF3'))
Luminance(color = c('black', 'white', '#E76BF3'))
Arguments

- counts: Unnormalized data such as raw counts or TPMs
- data: Prenormalized data; if provided, do not pass counts
- scale.data: a residual matrix
- umi.assay: The UMI assay name. Default is RNA
- 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
- SCTModel.list: list of SCTModels

Details

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

---

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**

```r
data("pbmc_small")
# 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)
```

---

**DEenrichRPlot**

**DE and EnrichR pathway visualization barplot**

**Description**

DE and EnrichR pathway visualization barplot

**Usage**

```r
DEenrichRPlot(
  object,  # Name of object class Seurat.
  ident.1 = NULL,  # Cell class identity 1.
  ident.2 = NULL,  # Cell class identity 2.
  balanced = TRUE,  # Option to display pathway enrichments for both negative and positive DE genes. If false, only positive DE gene will be displayed.
  logfc.threshold = 0.25,  # 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.
  assay = NULL,  # Assay
  max.genes,  # Maximum number of genes to display
  test.use = "wilcox",  # Test to use for differential expression
  p.val.cutoff = 0.05,  # P-value cutoff for differential expression
  cols = NULL,  # colors
  enrich.database = NULL,  # Enrichment database
  num.pathway = 10,  # Number of pathways to display
  return.gene.list = FALSE,  # Return gene list
  ...)
```

**Arguments**

- **object**
  - Name of object class Seurat.
- **ident.1**
  - Cell class identity 1.
- **ident.2**
  - Cell class identity 2.
- **balanced**
  - Option to display pathway enrichments for both negative and positive DE genes. If false, only positive DE gene will be displayed.
- **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.
assay: Assay to use in differential expression testing
max.genes: Maximum number of genes to use as input to enrichR.
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); will use a fast implementation by Presto if installed
- "wilcox_limma": Identifies differentially expressed genes between two groups of cells using the limma implementation of the Wilcoxon Rank Sum test; set this option to reproduce results from Seurat v4
- "bimod": Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc": Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
- "t": Identify differentially expressed genes between two groups of cells using the Student’s t-test.
- "negbinom": Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson": Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR": Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST": Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2": Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014). This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/DESeq2.html

p.val.cutoff: Cutoff to select DE genes.
cols: A list of colors to use for barplots.
enrich.database: Database to use from enrichR.
DietSeurat

Number of pathways to display in barplot.

Return list of DE genes

Arguments passed to other methods and to specific DE methods

Value

Returns one (only enriched) or two (both enriched and depleted) barplots with the top enriched/depleted GO terms from EnrichR.

DietSeurat

Slim down a Seurat object

Description

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

Usage

DietSeurat(object, layers = NULL, features = NULL, assays = NULL, dimreducs = NULL, graphs = NULL, misc = TRUE, counts = deprecated(), data = deprecated(), scale.data = deprecated(), ...)

Arguments

object A Seurat object
layers A vector or named list of layers to keep
features Only keep a subset of features, defaults to all features
assays Only keep a subset of assays specified here
dimreducs Only keep a subset of DimReducs specified here (if NULL, remove all DimReduces)
graphs Only keep a subset of Graphs specified here (if NULL, remove all Graphs)
misc Preserve the misc slot; default is TRUE
counts Preserve the count matrices for the assays specified
**DimHeatmap**

data Preserve the data matrices for the assays specified
scale.data Preserve the scale data matrices for the assays specified
...

**Value**

object with only the sub-object specified retained

---

**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,
  fast = TRUE,
  raster = TRUE,
  slot = "scale.data",
  assays = NULL,
  combine = TRUE
)
```

PCHeatmap(object, ...)

**Arguments**

- `object` Seurat object
- `dims` Dimensions to plot
- `nfeatures` Number of genes to plot
- `cells` A list of cells to plot. If numeric, just plots the top cells.
- `reduction` Which dimensional reduction to use
DimPlot

**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).

**disp.min** Minimum display value (all values below are clipped)

**disp.max** Maximum display value (all values above are clipped); defaults to 2.5 if slot is 'scale.data', 6 otherwise

**balanced** Plot an equal number of genes with both + and - scores.

**projected** Use the full projected dimensional reduction

**ncol** Number of columns to plot

**fast** If true, use image to generate plots; faster than using ggplot2, but not customizable

**raster** If true, plot with geom_raster, else use geom_tile. geom_raster may look blurry on some viewing applications such as Preview due to how the raster is interpolated. Set this to FALSE if you are encountering that issue (note that plots may take longer to produce/render).

**slot** Data slot to use, choose from 'raw.data', 'data', or 'scale.data'

**assays** A vector of assays to pull data from

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

**...** Extra parameters passed to DimHeatmap

**Value**

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

**See Also**

image geom_raster

**Examples**

```r
data("pbmc_small")
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).
**DimPlot**

**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,
  shuffle = FALSE,
  seed = 1,
  label = FALSE,
  label.size = 4,
  label.color = "black",
  label.box = FALSE,
  repel = FALSE,
  alpha = 1,
  cells.highlight = NULL,
  colshighlight = "#DE2D26",
  sizes.highlight = 1,
  na.value = "grey50",
  ncol = NULL,
  combine = TRUE,
  raster = NULL,
  raster.dpi = c(512, 512)
)
```

**PCAPlot**

**TSNEPlot**

**UMAPPlot**

**Arguments**

- **object**: Seurat object
- **dims**: Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions
- **cells**: Vector of cells to plot (default is all cells)
- **cols**: Vector of colors, each color corresponds to an identity class. This may also be a single character or numeric value corresponding to a palette as specified by `brewer.pal.info`. By default, ggplot2 assigns colors. We also include a number of palettes from the pals package. See `DiscretePalette` for details.
- **pt.size**: Adjust point size for plotting
reduction

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

group.by

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

split.by

A factor in object metadata to split the plot by, pass 'ident' to split by cell identity.

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. Only applicable if raster = FALSE.

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).

shuffle

Whether to randomly shuffle the order of points. This can be useful for crowded plots if points of interest are being buried. (default is FALSE)

seed

Sets the seed if randomly shuffling the order of points.

label

Whether to label the clusters.

label.size

Sets size of labels.

label.color

Sets the color of the label text.

label.box

Whether to put a box around the label text (geom_text vs geom_label).

repel

Repel labels.

alpha

Alpha value for plotting (default is 1).

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. If sizes.highlight = TRUE size of all points will be this value.

na.value

Color value for NA points when using custom scale.

ncol

Number of columns for display when combining plots.

combine

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

raster

Convert points to raster format, default is NULL which automatically rasterizes if plotting more than 100,000 cells.

raster.dpi

Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).

...

Extra parameters passed to DimPlot

Value

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

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

**See Also**

`FeaturePlot` `HoverLocator` `CellSelector` `FetchData`

**Examples**

```r
data("pbmc_small")
DimPlot(object = pbmc_small)
DimPlot(object = pbmc_small, split.by = 'letter.idents')
```

---

**DimReduc-class**

**The DimReduc Class**

**Description**

The `DimReduc` object stores a dimensionality reduction taken out in Seurat; for more details, please see the documentation in `SeuratObject`.

**See Also**

`SeuratObject::DimReduc-class`

---

**DiscretePalette**

**Discrete colour palettes from pals**

**Description**

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

**Usage**

```r
DiscretePalette(n, palette = NULL, shuffle = FALSE)
```

**Arguments**

- `n` Number of colours to be generated.
- `palette` Options are "alphabet", "alphabet2", "glasbey", "polychrome", "stepped", and "parade". Can be omitted and the function will use the one based on the requested `n`.
- `shuffle` Shuffle the colors in the selected palette.
Details

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

Many thanks to Kevin Wright for writing the pals package.

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

Value

A vector of colors

---

DoHeatmap  
Feature expression heatmap

Description

Draws a heatmap of single cell feature expression.

Usage

DoHeatmap(
  object,
  features = NULL,
  cells = NULL,
  group.by = "ident",
  group.bar = TRUE,
  group.colors = NULL,
  disp.min = -2.5,
  disp.max = NULL,
  slot = "scale.data",
  assay = NULL,
  label = TRUE,
  size = 5.5,
  hjust = 0,
  vjust = 0,
  angle = 45,
  raster = TRUE,
  draw.lines = TRUE,
  lines.width = NULL,
  group.bar.height = 0.02,
  combine = TRUE
)

DoHeatmap

Arguments

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

Value

A `patchwork`ed ggplot object if `combine = TRUE`; otherwise, a list of ggplot objects

Examples

```r
data("pbmc_small")
DoHeatmap(object = pbmc_small)
```
DotPlot

*Dot plot visualization*

**Description**

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

**Usage**

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

**Arguments**

- `object`: Seurat object
- `features`: Input vector of features, or named list of feature vectors if feature-grouped panels are desired (replicates the functionality of the old SplitDotPlotGG)
- `assay`: Name of assay to use, defaults to the active assay
- `cols`: Colors to plot: the name of a palette from RColorBrewer::brewer_pal.info, a pair of colors defining a gradient, or 3+ colors defining multiple gradients (if `split.by` is set)
- `col.min`: Minimum scaled average expression threshold (everything smaller will be set to this)
- `col.max`: Maximum scaled average expression threshold (everything larger will be set to this)
- `dot.min`: The fraction of cells at which to draw the smallest dot (default is 0). All cell groups with less than this expressing the given gene will have no dot drawn.
dot.scale Scale the size of the points, similar to cex
idents Identity classes to include in plot (default is all)
group.by Factor to group the cells by
split.by A factor in object metadata to split the plot by, pass 'ident' to split by cell identity' see FetchData for more details
cluster.idents Whether to order identities by hierarchical clusters based on given features, default is FALSE
scale Determine whether the data is scaled, TRUE for default
scale.by Scale the size of the points by 'size' or by 'radius'
scale.min Set lower limit for scaling, use NA for default
scale.max Set upper limit for scaling, use NA for default

Value
A ggplot object

See Also
RCOLORBrewer::brewer.pal.info

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

---

ElbowPlot

Quickly Pick Relevant Dimensions

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

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

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

Value
A ggplot object

Examples

data("pbmc_small")
ElbowPlot(object = pbmc_small)

ExpMean

Calculate the mean of logged values

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

Usage
ExpMean(x, ...)

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

Value
Returns the mean in log-space

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

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)
**ExpVar**

**Arguments**

\[ x \]  
A vector of values

**Value**

Returns the standard deviation in log-space

**Examples**

\[
\text{ExpSD}(x = c(1, 2, 3))
\]

---

**ExpVar**

*Calculate the variance of logged values*

**Description**

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

**Usage**

\[
\text{ExpVar}(x)
\]

**Arguments**

\[ x \]  
A vector of values

**Value**

Returns the variance in log-space

**Examples**

\[
\text{ExpVar}(x = c(1, 2, 3))
\]
FastRowScale  
Scale and/or center matrix rowwise

Description
Performs row scaling and/or centering. Equivalent to using \( t(\text{scale}(t(\text{mat}))) \) in R except in the case of NA values.

Usage
FastRowScale(mat, center = TRUE, scale = TRUE, scale_max = 10)

Arguments
- mat: A matrix
- center: a logical value indicating whether to center the rows
- scale: a logical value indicating whether to scale the rows
- scale_max: clip all values greater than scale_max to scale_max. Don’t clip if Inf.

Value
Returns the center/scaled matrix

FastRPCAIntegration  
Perform integration on the joint PCA cell embeddings.

Description
This is a convenience wrapper function around the following three functions that are often run together when perform integration. #'FindIntegrationAnchors, RunPCA, IntegrateEmbeddings.

Usage
FastRPCAIntegration(
  object.list,  
  reference = NULL,  
  anchor.features = 2000,  
  k.anchor = 20,  
  dims = 1:30,  
  scale = TRUE,  
  normalization.method = c("LogNormalize", "SCT"),  
  new.reduction.name = "integrated_dr",  
 npcs = 50,  
  findintegrationanchors.args = list(),  
  verbose = TRUE
)
Arguments

object.list A list of Seurat objects between which to find anchors for downstream integration.

reference A vector specifying the object/s to be used as a reference during integration. If NULL (default), all pairwise anchors are found (no reference/s). If not NULL, the corresponding objects in object.list will be used as references. When using a set of specified references, anchors are first found between each query and each reference. The references are then integrated through pairwise integration. Each query is then mapped to the integrated reference.

anchor.features Can be either:

• A numeric value. This will call SelectIntegrationFeatures to select the provided number of features to be used in anchor finding
• A vector of features to be used as input to the anchor finding process

k.anchor How many neighbors (k) to use when picking anchors

dims Which dimensions to use from the CCA to specify the neighbor search space

scale Whether or not to scale the features provided. Only set to FALSE if you have previously scaled the features you want to use for each object in the object.list

normalization.method Name of normalization method used: LogNormalize or SCT

ew.new.reduction.name Name of integrated dimensional reduction

npcs Total Number of PCs to compute and store (50 by default)

findintegrationanchors.args A named list of additional arguments to FindIntegrationAnchors

verbose Print messages and progress

Value

Returns a Seurat object with integrated dimensional reduction

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,
  dms = c(1, 2),
  cells = NULL,
  cols = if (blend) {
    c("lightgrey", "#ff0000", 
    "#00ff00")
  } else {
    c("lightgrey", "blue")
  },
  pt.size = NULL,
  alpha = 1,
  order = FALSE,
  min.cutoff = NA,
  max.cutoff = NA,
  reduction = NULL,
  split.by = NULL,
  keep.scale = "feature",
  shape.by = NULL,
  slot = "data",
  blend = FALSE,
  blend.threshold = 0.5,
  label = FALSE,
  label.size = 4,
  label.color = "black",
  repel = FALSE,
  ncol = NULL,
  coord.fixed = FALSE,
  by.col = TRUE,
  sort.cell = deprecated(),
  interactive = FALSE,
  combine = TRUE,
  raster = NULL,
  raster.dpi = c(512, 512)
)

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")
**FeaturePlot**

- **dms**: Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions.
- **cells**: Vector of cells to plot (default is all cells).
- **cols**: The two colors to form the gradient over. Provide as string vector with the first color corresponding to low values, the second to high. Also accepts a Brewer color scale or vector of colors. Note: this will bin the data into number of colors provided. When blend is TRUE, takes anywhere from 1-3 colors:
  - **1 color**: Treated as color for double-negatives, will use default colors 2 and 3 for per-feature expression.
  - **2 colors**: Treated as colors for per-feature expression, will use default color 1 for double-negatives.
  - **3+ colors**: First color used for double-negatives, colors 2 and 3 used for per-feature expression, all others ignored.
- **pt.size**: Adjust point size for plotting.
- **alpha**: Alpha value for plotting (default is 1).
- **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 plot by, pass 'ident' to split by cell identity.
- **keep.scale**: How to handle the color scale across multiple plots. Options are:
  - “feature” (default; by row/feature scaling): The plots for each individual feature are scaled to the maximum expression of the feature across the conditions provided to split.by.
  - “all” (universal scaling): The plots for all features and conditions are scaled to the maximum expression value for the feature with the highest overall expression.
  - all (no scaling): Each individual plot is scaled to the maximum expression value of the feature in the condition provided to split.by. Be aware setting NULL will result in color scales that are not comparable between plots.
- **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. Only applicable if raster = FALSE.
- **slot**: Which slot to pull expression data from?
- **blend**: Scale and blend expression values to visualize coexpression of two features.
- **blend.threshold**: The color cutoff from weak signal to strong signal; ranges from 0 to 1.
- **label**: Whether to label the clusters.
- **label.size**: Sets size of labels.
label.color  Sets the color of the label text
repel        Repel labels
ncol         Number of columns to combine multiple feature plots to, ignored if split.by is not NULL
coord.fixed  Plot cartesian coordinates with fixed aspect ratio
by.col       If splitting by a factor, plot the splits per column with the features as rows; ignored if blend = TRUE
sort.cell    Redundant with order. This argument is being deprecated. Please use order instead.
interactive  Launch an interactive FeaturePlot
combine      Combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot objects
raster       Convert points to raster format, default is NULL which automatically rasterizes if plotting more than 100,000 cells
raster.dpi   Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).

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

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

See Also
DimPlot HoverLocator CellSelector

Examples
```r
data("pbmc_small")
FeaturePlot(object = pbmc_small, features = \"PC_1\")
```

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.
FeatureScatter

Usage

FeatureScatter(
  object,
  feature1,    # First feature to plot. Typically feature expression but can also be metrics, PC
  feature2,    # scores, etc. - anything that can be retrieved with FetchData
  cells = NULL,  # Cells to include on the scatter plot.
  shuffle = FALSE,  # Whether to randomly shuffle the order of points. This can be useful for crowded
  seed = 1,      # plots if points of interest are being buried. (default is FALSE)
  group.by = NULL,  
  split.by = NULL,  
  cols = NULL,  # Colors to use for identity class plotting.
  pt.size = 1,    # Size of the points on the plot
  shape.by = NULL,  
  span = NULL,    # Spline span in loess function call, if NULL, no spline added
  smooth = FALSE,  # Smooth the graph (similar to smoothScatter)
  combine = TRUE,  
  slot = "data",  
  plot.cor = TRUE,  
  ncol = NULL,    # Sets the seed if randomly shuffling the order of points.
  raster = NULL,  
  raster.dpi = c(512, 512),
  jitter = FALSE,
  log = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>Name</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>shuffle</td>
<td>Whether to randomly shuffle the order of points. This can be useful for crowded plots if points of interest are being buried. (default is FALSE)</td>
</tr>
<tr>
<td>seed</td>
<td>Sets the seed if randomly shuffling the order of points.</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>split.by</td>
<td>A factor in object metadata to split the feature plot by, pass 'ident' to split by cell identity'</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>
</tbody>
</table>
FetchResiduals

Combine plots into a single `patchworked` slot.

Slot to pull data from, should be one of 'counts', 'data', or 'scale.data'.

Display correlation in plot title.

Number of columns if plotting multiple plots.

Convert points to raster format, default is NULL which will automatically use raster if the number of points plotted is greater than 100,000.

Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).

Jitter for easier visualization of crowded points (default is FALSE).

Plot features on the log scale (default is FALSE).

Value

A ggplot object

Examples

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

FetchResiduals

Calculate pearson residuals of features not in the scale.data

Description

This function calls sctransform::get_residuals.

Usage

FetchResiduals(
  object, 
  features, 
  assay = NULL, 
  umi.assay = "RNA", 
  layer = "counts", 
  clip.range = NULL, 
  reference.SCT.model = NULL, 
  replace.value = FALSE, 
  na.rm = TRUE, 
  verbose = TRUE
)
Arguments

object A seurat object
features Name of features to add into the scale.data
assay Name of the assay of the seurat object generated by SCTTransform
umi.assay Name of the assay of the seurat object containing UMI matrix and the default is RNA
layer Name (prefix) of the layer to pull counts from
clip.range Numeric of length two specifying the min and max values the Pearson residual will be clipped to
reference.SCT.model reference.SCT.model If a reference SCT model should be used for calculating the residuals. When set to not NULL, ignores the ‘SCTModel’ parameter.
replace.value Recalculate residuals for all features, even if they are already present. Useful if you want to change the clip.range.
na.rm For features where there is no feature model stored, return NA for residual value in scale.data when na.rm = FALSE. When na.rm is TRUE, only return residuals for features with a model stored for all cells.
verbose Whether to print messages and progress bars

Value

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

See Also

get_residuals

FetchResidualSCTModel  Calculate pearson residuals of features not in the scale.data This function is the secondary function under FetchResiduals

Description

Calculate pearson residuals of features not in the scale.data This function is the secondary function under FetchResiduals

Usage

FetchResidualSCTModel(object,
  assay = "SCT",
  umi.assay = "RNA",
  layer = "counts",
  chunk_size = 2000,
layer.cells = NULL,
SCTModel = NULL,
reference.SCT.model = NULL,
new_features = NULL,
clip.range = NULL,
replace.value = FALSE,
verbose = FALSE
)

Arguments

object A seurat object
assay Name of the assay of the seurat object generated by SCTTransform. Default is "SCT"
umi.assay Name of the assay of the seurat object to fetch UMIs from. Default is "RNA"
layer Name of the layer under 'umi.assay' to fetch UMIs from. Default is "counts"
chunk.size Number of cells to load in memory for calculating residuals
layer.cells Vector of cells to calculate the residual for. Default is NULL which uses all cells in the layer
SCTModel Which SCTmodel to use from the object for calculating the residual. Will be ignored if reference.SCT.model is set
reference.SCT.model If a reference SCT model should be used for calculating the residuals. When set to not NULL, ignores the 'SCTModel' parameter.
new_features A vector of features to calculate the residuals for
clip.range Numeric of length two specifying the min and max values the Pearson residual will be clipped to. Useful if you want to change the clip.range.
replace.value Whether to replace the value of residuals if it already exists
verbose Whether to print messages and progress bars

Value

Returns a matrix containing centered pearson residuals of added features

FetchResiduals_reference

temporal function to get residuals from reference

Description

temporal function to get residuals from reference
Usage

FetchResiduals_reference(
  object,
  reference.SCT.model = NULL,
  features = NULL,
  nCount_Umi = NULL,
  verbose = FALSE
)

Arguments

- **object**: A seurat object
- **reference.SCT.model**: a reference SCT model that should be used for calculating the residuals
- **features**: Names of features to compute
- **nCount_Umi**: UMI counts. If not specified, defaults to column sums of object
- **verbose**: Whether to print messages and progress bars

Description

This function is useful for removing stray beads that fall outside the main Slide-seq puck area. Essentially, it's a circular filter where you set a center and radius defining a circle of beads to keep. If the center is not set, it will be estimated from the bead coordinates (removing the 1st and 99th quantile to avoid skewing the center by the stray beads). By default, this function will display a `SpatialDimPlot` showing which cells were removed for easy adjustment of the center and/or radius.

Usage

FilterSlideSeq(
  object,
  image = "image",
  center = NULL,
  radius = NULL,
  do.plot = TRUE
)

Arguments

- **object**: Seurat object with slide-seq data
- **image**: Name of the image where the coordinates are stored
- **center**: Vector specifying the x and y coordinates for the center of the inclusion circle
- **radius**: Radius of the circle of inclusion
- **do.plot**: Display a `SpatialDimPlot` with the cells being removed labeled.
Value

Returns a Seurat object with only the subset of cells that pass the circular filter

Examples

```r
## Not run:
# This example uses the ssHippo dataset which you can download
# using the SeuratData package.
library(SeuratData)
data('ssHippo')
# perform filtering of beads
ssHippo.filtered <- FilterSlideSeq(ssHippo, radius = 2300)
# This radius looks to small so increase and repeat until satisfied

## End(Not run)
```

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 = 0.1,
  test.use = "wilcox",
  slot = "data",
  min.pct = 0.01,
  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,
  mean.fxn = NULL,
  fc.name = NULL,
  base = 2,
  return.thresh = 0.01,
  densify = FALSE,
  ...
)
```
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 0.1 Increasing logfc.threshold speeds up the function, but can miss weaker signals. If the slot parameter is "scale.data" no filtering is performed.

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); will use a fast implementation by Presto if installed
• "wilcox_limma" : Identifies differentially expressed genes between two groups of cells using the limma implementation of the Wilcoxon Rank Sum test; set this option to reproduce results from Seurat v4
• "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
• "roc" : Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
• "t" : Identify differentially expressed genes between two groups of cells using the Student’s t-test.
• "negbinom" : Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
• "poisson" : Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
• "LR" : Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
• "MAST" : Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
• "DESeq2" : Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014). This test does not support pre-
filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/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 0.01

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

`node`  
A node to find markers for and all its children; requires BuildClusterTree to have been run previously; replaces FindAllMarkersNode

`verbose`  
Print a progress bar once expression testing begins

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

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

`random.seed`  
Random seed for downsampling

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

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

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

`mean.fxn`  
Function to use for fold change or average difference calculation. The default depends on the the value of `fc.slot`:

- "counts" : difference in the log of the mean counts, with pseudocount.
- "data" : difference in the log of the average exponentiated data, with pseudocount. This adjusts for differences in sequencing depth between cells, and assumes that "data" has been log-normalized.
- "scale.data" : difference in the means of scale.data.

`fc.name`  
Name of the fold change, average difference, or custom function column in the output data.frame. If NULL, the fold change column will be named according to the logarithm base (eg, "avg_log2FC"), or if using the scale.data slot "avg_diff".

`base`  
The base with respect to which logarithms are computed.

`return.thresh`  
Only return markers that have a p-value < return.thresh, or a power > return.thresh (if the test is ROC)

`densify`  
Convert the sparse matrix to a dense form before running the DE test. This can provide speedups but might require higher memory; default is FALSE

...  
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

data("pbmc_small")
# 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)

FindBridgeAnchor

Find bridge anchors between two unimodal datasets

Description

First, bridge object is used to reconstruct two single-modality profiles and then project those cells into bridage graph laplacian space. Next, find a set of anchors between two single-modality objects. These anchors can later be used to integrate embeddings or transfer data from the reference to query object using the MapQuery object.

Usage

FindBridgeAnchor(
  object.list, 
  bridge.object, 
  object.reduction, 
  bridge.reduction, 
  anchor.type = c("Transfer", "Integration"), 
  reference = NULL, 
  laplacian.reduction = "lap", 
  laplacian.dims = 1:50, 
  reduction = c("direct", "cca"), 
  bridge.assay.name = "Bridge", 
  reference.bridge.stored = FALSE, 
  k.anchor = 20, 
  k.score = 50, 
  verbose = TRUE, 
  ... 
)
FindBridgeAnchor

**Arguments**

- **object.list**
  A list of Seurat objects

- **bridge.object**
  A multi-omic bridge Seurat which is used as the basis to represent unimodal datasets

- **object.reduction**
  A list of dimensional reductions from object.list used to be reconstructed by bridge.object

- **bridge.reduction**
  A list of dimensional reductions from bridge.object used to reconstruct object.reduction

- **anchor.type**
  The type of anchors. Can be one of:
  - Integration: Generate IntegrationAnchors for integration
  - Transfer: Generate TransferAnchors for transferring data

- **reference**
  A vector specifying the object/s to be used as a reference during integration or transfer data.

- **laplacian.reduction**
  Name of bridge graph laplacian dimensional reduction

- **laplacian.dims**
  Dimensions used for bridge graph laplacian dimensional reduction

- **reduction**
  Dimensional reduction to perform when finding anchors. Can be one of:
  - cca: Canonical correlation analysis
  - direct: Use assay data as a dimensional reduction

- **bridge.assay.name**
  Assay name used for bridge object reconstruction value (default is 'Bridge')

- **refernece.bridge.stored**
  If refernece has stored the bridge dictionary representation

- **k.anchor**
  How many neighbors (k) to use when picking anchors

- **k.score**
  How many neighbors (k) to use when scoring anchors

- **verbose**
  Print messages and progress

- **Additional parameters** passed to FindIntegrationAnchors or FindTransferAnchors

**Details**

- Bridge cells reconstruction
- Find anchors between objects. It can be either IntegrationAnchors or TransferAnchor.

**Value**

Returns an AnchorSet object that can be used as input to IntegrateEmbeddings or MapQuery
**FindBridgeIntegrationAnchors**

*Find integration bridge anchors between query and extended bridge-reference*

---

**Description**

Find a set of anchors between unimodal query and the other unimodal reference using a pre-computed BridgeReferenceSet. These integration anchors can later be used to integrate query and reference using the IntegrateEmbeddings object.

**Usage**

```r
FindBridgeIntegrationAnchors(
  extended.reference,
  query,
  query.assay = NULL,
  dims = 1:30,
  scale = FALSE,
  reduction = c("lsiproject", "pcaproject"),
  integration.reduction = c("direct", "cca"),
  verbose = TRUE
)
```

**Arguments**

- **extended.reference**: BridgeReferenceSet object generated from PrepareBridgeReference
- **query**: A query Seurat object
- **query.assay**: Assay name for query-bridge integration
- **dims**: Number of dimensions for query-bridge integration
- **scale**: Determine if scale the query data for projection
- **reduction**: Dimensional reduction to perform when finding anchors. Options are:
  - pcaproject: Project the PCA from the bridge onto the query. We recommend using PCA when bridge and query datasets are from scRNA-seq
  - lsiproject: Project the LSI from the bridge onto the query. We recommend using LSI when bridge and query datasets are from scATAC-seq or scCUT&TAG data. This requires that LSI or supervised LSI has been computed for the bridge dataset, and the same features (eg, peaks or genome bins) are present in both the bridge and query.
- **integration.reduction**: Dimensional reduction to perform when finding anchors between query and reference. Options are:
  - direct: find anchors directly on the bridge representation space
FindBridgeTransferAnchors

- cca: perform cca on the on the bridge representation space and then find anchors

verbose  Print messages and progress

Value

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

---

FindBridgeTransferAnchors

*Find bridge anchors between query and extended bridge-reference*

**Description**

Find a set of anchors between unimodal query and the other unimodal reference using a pre-computed BridgeReferenceSet. This function performs three steps: 1. Harmonize the bridge and query cells in the bridge query reduction space 2. Construct the bridge dictionary representations for query cells 3. Find a set of anchors between query and reference in the bridge graph laplacian eigenspace. These anchors can later be used to integrate embeddings or transfer data from the reference to query object using the MapQuery object.

**Usage**

```r
FindBridgeTransferAnchors(
  extended.reference,
  query,
  query.assay = NULL,
  dims = 1:30,
  scale = FALSE,
  reduction = c("lsiproject", "pcaproject"),
  bridge.reduction = c("direct", "cca"),
  verbose = TRUE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extended.reference</td>
<td>BridgeReferenceSet object generated from PrepareBridgeReference</td>
</tr>
<tr>
<td>query</td>
<td>A query Seurat object</td>
</tr>
<tr>
<td>query.assay</td>
<td>Assay name for query-bridge integration</td>
</tr>
<tr>
<td>dims</td>
<td>Number of dimensions for query-bridge integration</td>
</tr>
<tr>
<td>scale</td>
<td>Determine if scale the query data for projection</td>
</tr>
<tr>
<td>reduction</td>
<td>Dimensional reduction to perform when finding anchors. Options are:</td>
</tr>
<tr>
<td></td>
<td>- pcaproject: Project the PCA from the bridge onto the query. We recommend</td>
</tr>
<tr>
<td></td>
<td>using PCA when bridge and query datasets are from scRNA-seq</td>
</tr>
</tbody>
</table>
FindClusters

• lsiproject: Project the LSI from the bridge onto the query. We recommend using LSI when bridge and query datasets are from scATAC-seq or scCUT&TAG data. This requires that LSI or supervised LSI has been computed for the bridge dataset, and the same features (e.g., peaks or genome bins) are present in both the bridge and query.

bridge.reduction

Dimensional reduction to perform when finding anchors. Can be one of:

• cca: Canonical correlation analysis
• direct: Use assay data as a dimensional reduction

verbose

Print messages and progress

Value

Returns an AnchorSet object that can be used as input to TransferData, IntegrateEmbeddings and MapQuery.

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,
  node.sizes = NULL,
  resolution = 0.8,
  method = "matrix",
  algorithm = 1,
  n.start = 10,
  n.iter = 10,
  random.seed = 0,
  group.singletons = TRUE,
  temp.file.location = NULL,
  edge.file.name = NULL,
## S3 method for class 'Seurat'

```r
FindClusters(
  object,
  graph.name = NULL,
  cluster.name = NULL,
  modularity.fxn = 1,
  initialmembership = NULL,
  nodesizes = NULL,
  resolution = 0.8,
  method = "matrix",
  algorithm = 1,
  n.start = 10,
  n.iter = 10,
  random.seed = 0,
  group.singletons = TRUE,
  temp.file.location = NULL,
  edge.file.name = NULL,
  verbose = TRUE,
  ...
)
```

### Arguments

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

Details

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

Value

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

FindConservedMarkers  Finds markers that are conserved between the groups

Description

Finds markers that are conserved between the groups

Usage

FindConservedMarkers(
  object,
  ident.1,
  ident.2 = NULL,
  grouping.var,
  assay = "RNA",
  slot = "data",
  min.cells.group = 3,
  meta.method = metap::minimump,
  verbose = TRUE,
  ...
)
FindIntegrationAnchors

Arguments

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

Value

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

Examples

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

FindIntegrationAnchors

*Find integration anchors*

Description

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

FindIntegrationAnchors(
  object.list = NULL,
  assay = NULL,
  reference = NULL,
  anchor.features = 2000,
  scale = TRUE,
  normalization.method = c("LogNormalize", "SCT"),
  sct.clip.range = NULL,
  reduction = c("cca", "rpca", "jPCA", "rlsi"),
  l2.norm = TRUE,
  dims = 1:30,
  k.anchor = 5,
  k.filter = 200,
  k.score = 30,
  max.features = 200,
  nn.method = "annoy",
  n.trees = 50,
  eps = 0,
  verbose = TRUE
)

Arguments

object.list  A list of Seurat objects between which to find anchors for downstream integration.
assay        A vector of assay names specifying which assay to use when constructing anchors. If NULL, the current default assay for each object is used.
reference    A vector specifying the object/s to be used as a reference during integration. If NULL (default), all pairwise anchors are found (no reference/s). If not NULL, the corresponding objects in object.list will be used as references. When using a set of specified references, anchors are first found between each query and each reference. The references are then integrated through pairwise integration. Each query is then mapped to the integrated reference.
anchor.features  Can be either:
  • A numeric value. This will call SelectIntegrationFeatures to select the provided number of features to be used in anchor finding
  • A vector of features to be used as input to the anchor finding process
scale        Whether or not to scale the features provided. Only set to FALSE if you have previously scaled the features you want to use for each object in the object.list
normalization.method  Name of normalization method used: LogNormalize or SCT
sct.clip.range Numeric of length two specifying the min and max values the Pearson residual will be clipped to
reduction    Dimensional reduction to perform when finding anchors. Can be one of:
FindIntegrationAnchors

- cca: Canonical correlation analysis
- rpca: Reciprocal PCA
- j pca: Joint PCA
- rlsi: Reciprocal LSI

**12.norm** Perform L2 normalization on the CCA cell embeddings after dimensional reduction

**dims** Which dimensions to use from the CCA to specify the neighbor search space

**k.anchor** How many neighbors (k) to use when picking anchors

**k.filter** How many neighbors (k) to use when filtering anchors

**k.score** How many neighbors (k) to use when scoring anchors

**max.features** The maximum number of features to use when specifying the neighborhood search space in the anchor filtering

**nn.method** Method for nearest neighbor finding. Options include: rann, annoy

**n.trees** More trees gives higher precision when using annoy approximate nearest neighbor search

**eps** Error bound on the neighbor finding algorithm (from RANN/Annoy)

**verbose** Print progress bars and output

**Details**

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

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

- Perform dimensional reduction on the dataset pair as specified via the `reduction` parameter. If `12.norm` is set to `TRUE`, perform L2 normalization of the embedding vectors.
- Identify anchors - pairs of cells from each dataset that are contained within each other’s neighborhoods (also known as mutual nearest neighbors).
- Filter low confidence anchors to ensure anchors in the low dimension space are in broad agreement with the high dimensional measurements. This is done by looking at the neighbors of each query cell in the reference dataset using `max.features` to define this space. If the reference cell isn’t found within the first `k.filter` neighbors, remove the anchor.
- Assign each remaining anchor a score. For each anchor cell, determine the nearest `k.score` anchors within its own dataset and within its pair’s dataset. Based on these neighborhoods, construct an overall neighbor graph and then compute the shared neighbor overlap between anchor and query cells (analogous to an SNN graph). We use the 0.01 and 0.90 quantiles on these scores to dampen outlier effects and rescale to range between 0-1.

**Value**

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

Gene expression markers of identity classes

Description

Finds markers (differentially expressed genes) for identity classes

Usage

FindMarkers(object, ...)

## Default S3 method:
FindMarkers(
  object,
  slot = "data",
  cells.1 = NULL,
  ...)
cells.2 = NULL,
features = NULL,
logfc.threshold = 0.1,
test.use = "wilcox",
min.pct = 0.01,
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,
fctest.results = NULL,
densify = FALSE,
...
)

## S3 method for class 'Assay'
FindMarkers(
  object,
  slot = "data",
cells.1 = NULL,
cells.2 = NULL,
features = NULL,
test.use = "wilcox",
fc.slot = "data",
pseudocount.use = 1,
norm.method = NULL,
mean.fxn = NULL,
fctest.name = NULL,
base = 2,
...
)

## S3 method for class 'SCTAssay'
FindMarkers(
  object,
cells.1 = NULL,
cells.2 = NULL,
features = NULL,
test.use = "wilcox",
pseudocount.use = 1,
slot = "data",
fc.slot = "data",
mean.fxn = NULL,
fctest.name = NULL,
base = 2,
FindMarkers

recorrect_umi = TRUE,
...
)

## S3 method for class 'DimReduc'
FindMarkers(
  object,
  cells.1 = NULL,
  cells.2 = NULL,
  features = NULL,
  logfc.threshold = 0.1,
  test.use = "wilcox",
  min.pct = 0.01,
  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,
  densify = FALSE,
  mean.fxn = rowMeans,
  fc.name = NULL,
  ...
)

## S3 method for class 'Seurat'
FindMarkers(
  object,
  latent.vars = NULL,
  ident.1 = NULL,
  ident.2 = NULL,
  group.by = NULL,
  subset.ident = NULL,
  assay = NULL,
  reduction = NULL,
  ...
)

Arguments

object An object
...
Arguments passed to other methods and to specific DE methods
slot Slot to pull data from; note that if test.use is "negbinom", "poisson", or "DESeq2", slot will be set to "counts"
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 0.1 Increasing logfc.threshold speeds up the function, but can miss weaker signals. If the slot parameter is "scale.data" no filtering is performed.

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); will use a fast implementation by Presto if installed
- "wilcox_limma": Identifies differentially expressed genes between two groups of cells using the limma implementation of the Wilcoxon Rank Sum test; set this option to reproduce results from Seurat v4
- "bimod": Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc": Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
- "t": Identify differentially expressed genes between two groups of cells using the Student’s t-test.
- "negbinom": Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson": Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR": Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST": Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2": Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014). This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/DESeq2.html
FindMarkers

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 0.01

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

verbose  Print a progress bar once expression testing begins

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

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

random.seed  Random seed for downsampling

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

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

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

fc.results  data.frame from FoldChange

densify  Convert the sparse matrix to a dense form before running the DE test. This can provide speedups but might require higher memory; default is FALSE

fc.slot  Slot used to calculate fold-change - will also affect the default for mean.fxn, see below for more details.

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

norm.method  Normalization method for fold change calculation when slot is “data”

mean.fxn  Function to use for fold change or average difference calculation. The default depends on the the value of fc.slot:

- "counts" : difference in the log of the mean counts, with pseudocount.
- "data" : difference in the log of the average exponentiated data, with pseudocount. This adjusts for differences in sequencing depth between cells, and assumes that "data" has been log-normalized.
- "scale.data" : difference in the means of scale.data.

fc.name  Name of the fold change, average difference, or custom function column in the output data.frame. If NULL, the fold change column will be named according to the logarithm base (eg. "avg_log2FC"), or if using the scale.data slot "avg_diff".

base  The base with respect to which logarithms are computed.

recorrect_umi  Recalculate corrected UMI counts using minimum of the median UMIs when performing DE using multiple SCT objects; default is TRUE

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
FindMarkers

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

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


See Also

FoldChange
FindMultiModalNeighbors

## Examples

```r
## Not run:
data("pbmc_small")
# Find markers for cluster 2
markers <- FindMarkers(object = pbmc_small, ident.1 = 2)
head(x = markers)

# Take all cells in cluster 2, and find markers that separate cells in the 'g1' group (metadata
# variable 'group')
markers <- FindMarkers(pbmc_small, ident.1 = "g1", group.by = 'groups', subset.ident = "2")
head(x = markers)

# Pass 'clustertree' or an object of class phylo to ident.1 and
# a node to ident.2 as a replacement for FindMarkersNode
if (requireNamespace("ape", quietly = TRUE)) {
pbmc_small <- BuildClusterTree(object = pbmc_small)
markers <- FindMarkers(object = pbmc_small, ident.1 = 'clustertree', ident.2 = 5)
head(x = markers)
}
## End(Not run)
```

---

**FindMultiModalNeighbors**

*Construct weighted nearest neighbor graph*

### Description

This function will construct a weighted nearest neighbor (WNN) graph. For each cell, we identify
the nearest neighbors based on a weighted combination of two modalities. Takes as input two
dimensional reductions, one computed for each modality. Other parameters are listed for debugging,
but can be left as default values.

### Usage

```r
FindMultiModalNeighbors(
  object,
  reduction.list,
  dims.list,
  k.nn = 20,
  l2.norm = TRUE,
  knn.graph.name = "wknn",
  snn.graph.name = "wsnn",
  weighted.nn.name = "weighted.nn",
  modality.weight.name = NULL,
  knn.range = 200,
  prune.SNN = 1/15,
```
FindMultiModalNeighbors

sd.scale = 1,
cross.contant.list = NULL,
smooth = FALSE,
return.intermediate = FALSE,
modality.weight = NULL,
verbose = TRUE
)

Arguments

object
reduction.list
dims.list
k.nn
l2.norm
knn.graph.name
snn.graph.name
weighted.nn.name
modality.weight.name
knn.range
prune.SNN
sd.scale
cross.contant.list
smooth
return.intermediate
modality.weight
verbose

Value

Seurat object containing a nearest-neighbor object, KNN graph, and SNN graph - each based on a weighted combination of modalities.
FindNeighbors

(Shared) Nearest-neighbor graph construction

Description

Computes the \( k \) nearest neighbors for a given dataset. Can also optionally (via `compute.SNN`), construct a shared nearest neighbor graph by calculating the neighborhood overlap (Jaccard index) between every cell and its \( k \) nearest neighbors.

Usage

FindNeighbors(object, ...)

## Default S3 method:
FindNeighbors(
  object,
  query = NULL,
  distance.matrix = FALSE,
  k.param = 20,
  return.neighbor = FALSE,
  compute.SNN = !return.neighbor,
  prune.SNN = 1/15,
  nn.method = "annoy",
  n.trees = 50,
  annoy.metric = "euclidean",
  nn.eps = 0,
  verbose = TRUE,
  l2.norm = FALSE,
  cache.index = FALSE,
  index = NULL,
  ...
)

## S3 method for class 'Assay'
FindNeighbors(
  object,
  features = NULL,
  k.param = 20,
  return.neighbor = FALSE,
  compute.SNN = !return.neighbor,
  prune.SNN = 1/15,
  nn.method = "annoy",
  n.trees = 50,
  annoy.metric = "euclidean",
  nn.eps = 0,
  verbose = TRUE,
  l2.norm = FALSE,
cache.index = FALSE,
...
)

## S3 method for class 'dist'
FindNeighbors(
    object,
    k.param = 20,
    return.neighbor = FALSE,
    compute.SNN = !return.neighbor,
    prune.SNN = 1/15,
    nn.method = "annoy",
    n.trees = 50,
    annoy.metric = "euclidean",
    nn.eps = 0,
    verbose = TRUE,
    l2.norm = FALSE,
    cache.index = FALSE,
...
)

## S3 method for class 'Seurat'
FindNeighbors(
    object,
    reduction = "pca",
    dims = 1:10,
    assay = NULL,
    features = NULL,
    k.param = 20,
    return.neighbor = FALSE,
    compute.SNN = !return.neighbor,
    prune.SNN = 1/15,
    nn.method = "annoy",
    n.trees = 50,
    annoy.metric = "euclidean",
    nn.eps = 0,
    verbose = TRUE,
    do.plot = FALSE,
    graph.name = NULL,
    l2.norm = FALSE,
    cache.index = FALSE,
...
)

Arguments

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

- **query**: Matrix of data to query against object. If missing, defaults to object.
- **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.
- **return.neighbor**: Return result as Neighbor object. Not used with distance matrix input.
- **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.method**: Method for nearest neighbor finding. Options include: rann, annoy
- **n.trees**: More trees gives higher precision when using annoy approximate nearest neighbor search
- **annoy.metric**: Distance metric for annoy. Options include: euclidean, cosine, manhattan, and hamming
- **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
- **l2.norm**: Take L2Norm of the data
- **cache.index**: Include cached index in returned Neighbor object (only relevant if return.neighbor = TRUE)
- **index**: Precomputed index. Useful if querying new data against existing index to avoid recomputing.
- **features**: Features to use as input for building the (S)NN; used only when dims is NULL
- **reduction**: Reduction to use as input for building the (S)NN
- **dims**: Dimensions of reduction to use as input
- **assay**: Assay to use in construction of (S)NN; used only when dims is NULL
- **do.plot**: Plot SNN graph on tSNE coordinates
- **graph.name**: Optional naming parameter for stored (S)NN graph (or Neighbor object, if return.neighbor = TRUE). Default is assay.name_(s)nn. To store both the neighbor graph and the shared nearest neighbor (SNN) graph, you must supply a vector containing two names to the graph.name parameter. The first element in the vector will be used to store the nearest neighbor (NN) graph, and the second element used to store the SNN graph. If only one name is supplied, only the NN graph is stored.

**Value**

This function can either return a Neighbor object with the KNN information or a list of Graph objects with the KNN and SNN depending on the settings of return.neighbor and compute.SNN. When running on a Seurat object, this returns the Seurat object with the Graphs or Neighbor objects stored in their respective slots. Names of the Graph or Neighbor object can be found with Graphs or Neighbors.
Examples

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

# More commonly, we build the SNN on a dimensionally reduced form of the data
# such as the first 10 principle components.
pbmc_small <- FindNeighbors(pbmc_small, reduction = "pca", dims = 1:10)

FindSpatiallyVariableFeatures

Find spatially variable features

Description
Identify features whose variability in expression can be explained to some degree by spatial location.

Usage

FindSpatiallyVariableFeatures(object, ...)

## Default S3 method:
FindSpatiallyVariableFeatures(
  object,
  spatial.location,
  selection.method = c("markvariogram", "moransi"),
  r.metric = 5,
  x.cuts = NULL,
  y.cuts = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'Assay'
FindSpatiallyVariableFeatures(
  object,
  slot = "scale.data",
  spatial.location,
  selection.method = c("markvariogram", "moransi"),
  features = NULL,
  r.metric = 5,
  x.cuts = NULL,
  y.cuts = NULL,
  nfeatures = nfeatures,
  ...
## FindSpatiallyVariableFeatures

```r
FindSpatiallyVariableFeatures(
  object,
  assay = NULL,
  slot = "scale.data",
  features = NULL,
  image = NULL,
  selection.method = c("markvariogram", "moransi"),
  r.metric = 5,
  x.cuts = NULL,
  y.cuts = NULL,
  nfeatures = 2000,
  verbose = TRUE,
  ...
)
```

### S3 method for class 'Seurat'

```r
FindSpatiallyVariableFeatures(  
  object,
  assay = NULL,
  slot = "scale.data",
  features = NULL,
  image = NULL,
  selection.method = c("markvariogram", "moransi"),
  r.metric = 5,
  x.cuts = NULL,
  y.cuts = NULL,
  nfeatures = 2000,
  verbose = TRUE,
  ...
)
```

### S3 method for class 'StdAssay'

```r
FindSpatiallyVariableFeatures(  
  object,
  layer = "scale.data",
  spatial.location,
  selection.method = c("markvariogram", "moransi"),
  features = NULL,
  r.metric = 5,
  x.cuts = NULL,
  y.cuts = NULL,
  nfeatures = nfeatures,
  verbose = TRUE,
  ...
)
```

## Arguments

- **object**: A Seurat object, assay, or expression matrix
- **...**: Arguments passed to other methods
- **spatial.location**: Coordinates for each cell/spot/bead
- **selection.method**: Method for selecting spatially variable features.
  - markvariogram: See `RunMarkVario` for details
  - moransi: See `RunMoransI` for details.
- **r.metric**: r value at which to report the "trans" value of the mark variogram
- **x.cuts**: Number of divisions to make in the x direction, helps define the grid over which binning is performed

---

**Arguments**

- **object**: A Seurat object, assay, or expression matrix
- **...**: Arguments passed to other methods
- **spatial.location**: Coordinates for each cell/spot/bead
- **selection.method**: Method for selecting spatially variable features.
  - markvariogram: See `RunMarkVario` for details
  - moransi: See `RunMoransI` for details.
- **r.metric**: r value at which to report the "trans" value of the mark variogram
- **x.cuts**: Number of divisions to make in the x direction, helps define the grid over which binning is performed
FindSubCluster

Find subclusters under one cluster

**Usage**

```r
FindSubCluster(
  object,
  cluster,
  graph.name,
  subcluster.name = "sub.cluster",
  resolution = 0.5,
  algorithm = 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>cluster</td>
<td>the cluster to be sub-clustered</td>
</tr>
<tr>
<td>graph.name</td>
<td>Name of graph to use for the clustering algorithm</td>
</tr>
<tr>
<td>subcluster.name</td>
<td>the name of sub cluster added in the meta.data</td>
</tr>
<tr>
<td>resolution</td>
<td>Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.</td>
</tr>
<tr>
<td>algorithm</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Value**

return a object with sub cluster labels in the sub-cluster.name variable
FindTransferAnchors

Description

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

Usage

FindTransferAnchors(
  reference,
  query,
  normalization.method = "LogNormalize",
  recompute.residuals = TRUE,
  reference.assay = NULL,
  reference.neighbors = NULL,
  query.assay = NULL,
  reduction = "pcaproject",
  reference.reduction = NULL,
  project.query = FALSE,
  features = NULL,
  scale = TRUE,
 npcs = 30,
  l2.norm = TRUE,
  dims = 1:30,
  k.anchor = 5,
  k.filter = NA,
  k.score = 30,
  max.features = 200,
  nn.method = "annoy",
  n.trees = 50,
  eps = 0,
  approx.pca = TRUE,
  mapping.score.k = NULL,
  verbose = TRUE
)

Arguments

reference Seurat object to use as the reference
query Seurat object to use as the query
normalization.method Name of normalization method used: LogNormalize or SCT.
recompute.residuals If using SCT as a normalization method, compute query Pearson residuals using the reference SCT model parameters.
reference.assay
Name of the Assay to use from reference

reference.neighbors
Name of the Neighbor to use from the reference. Optionally enables reuse of precomputed neighbors.

query.assay
Name of the Assay to use from query

reduction
Dimensional reduction to perform when finding anchors. Options are:

- pcaproject: Project the PCA from the reference onto the query. We recommend using PCA when reference and query datasets are from scRNA-seq
- lsiproject: Project the LSI from the reference onto the query. We recommend using LSI when reference and query datasets are from scATAC-seq. This requires that LSI has been computed for the reference dataset, and the same features (e.g., peaks or genome bins) are present in both the reference and query. See RunTFIDF and RunSVD
- rpca: Project the PCA from the reference onto the query, and the PCA from the query onto the reference (reciprocal PCA projection).
- cca: Run a CCA on the reference and query

reference.reduction
Name of dimensional reduction to use from the reference if running the pcaproject workflow. Optionally enables reuse of precomputed reference dimensional reduction. If NULL (default), use a PCA computed on the reference object.

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. In this case, the default features will be set to the variable features of the query object that are also present in the reference.

features
Features to use for dimensional reduction. If not specified, set as variable features of the reference object which are also present in the query.

scale
Scale query data.

npcs
Number of PCs to compute on reference if reference.reduction is not provided.

l2.norm
Perform L2 normalization on the cell embeddings after dimensional reduction

dims
Which dimensions to use from the reduction to specify the neighbor search space

k.anchor
How many neighbors (k) to use when finding anchors

k.filter
How many neighbors (k) to use when filtering anchors. Set to NA to turn off filtering.

k.score
How many neighbors (k) to use when scoring anchors

max.features
The maximum number of features to use when specifying the neighborhood search space in the anchor filtering

nn.method
Method for nearest neighbor finding. Options include: rann, annoy

n.trees
More trees gives higher precision when using annoy approximate nearest neighbor search

eps
Error bound on the neighbor finding algorithm (from RANN or RcppAnnoy)

approx.pca
Use truncated singular value decomposition to approximate PCA
FindTransferAnchors

mapping.score.k
Compute and store nearest k query neighbors in the AnchorSet object that is returned. You can optionally set this if you plan on computing the mapping score and want to enable reuse of some downstream neighbor calculations to make the mapping score function more efficient.

verbose
Print progress bars and output

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

• Perform dimensional reduction. Exactly what is done here depends on the values set for the reduction and project.query parameters. If reduction = "pcaproject", a PCA is performed on either the reference (if project.query = FALSE) or the query (if project.query = TRUE), using the features specified. The data from the other dataset is then projected onto this learned PCA structure. If reduction = "cca", then CCA is performed on the reference and query for this dimensional reduction step. If reduction = "lsiproject", the stored LSI dimension reduction in the reference object is used to project the query dataset onto the reference. If l2.norm is set to TRUE, perform L2 normalization of the embedding vectors.

• Identify anchors between the reference and query - pairs of cells from each dataset that are contained within each other’s neighborhoods (also known as mutual nearest neighbors).

• Filter low confidence anchors to ensure anchors in the low dimension space are in broad agreement with the high dimensional measurements. This is done by looking at the neighbors of each query cell in the reference dataset using max.features to define this space. If the reference cell isn’t found within the first k.filter neighbors, remove the anchor.

• Assign each remaining anchor a score. For each anchor cell, determine the nearest k.score anchors within its own dataset and within its pair’s dataset. Based on these neighborhoods, construct an overall neighbor graph and then compute the shared neighbor overlap between anchor and query cells (analogous to an SNN graph). We use the 0.01 and 0.90 quantiles on these scores to dampen outlier effects and rescale to range between 0-1.

Value
Returns an AnchorSet object that can be used as input to TransferData, IntegrateEmbeddings and MapQuery. The dimension reduction used for finding anchors is stored in the AnchorSet object and can be used for computing anchor weights in downstream functions. Note that only the requested dimensions are stored in the dimension reduction object in the AnchorSet. This means that if dims=2:20 is used, for example, the dimension of the stored reduction is 1:19.

References
### Not run:

```r
# to install the SeuratData package see https://github.com/satijalab/seurat-data
library(SeuratData)
data(“pbmc3k”)

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

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

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

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

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

## End(Not run)
```

---

### FindVariableFeatures  
*Find variable features*

#### Description

Identifies features that are outliers on a ‘mean variability plot’.

#### Usage

```r
FindVariableFeatures(object, ...)
```

---

### Examples

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

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

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

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

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

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

## End(Not run)
```
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 'SCTAssay'
FindVariableFeatures(object, nfeatures = 2000, ...)

## 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 \( \log(\text{variance}) \) and \( \log(\text{mean}) \) using local polynomial regression (loess). Then standardizes the feature values using the observed mean and expected variance (given by the fitted line). Feature variance is then calculated on the standardized values after clipping to a maximum (see clip.max parameter).

• “mean.var.plot” (mvp): First, uses a function to calculate average expression (mean.function) and dispersion (dispersion.function) for each feature. Next, divides features into num.bin (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” (disp): selects the genes with the highest dispersion values

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

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

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

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

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

binning.method Specifies how the bins should be computed. Available methods are:

• “equal_width”: each bin is of equal width along the x-axis (default)

• “equal_frequency”: each bin contains an equal number of features (can increase statistical power to detect overdispersed features at high expression values, at the cost of reduced resolution along the x-axis)

verbose show progress bar for calculations

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

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

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

assay Assay to use

Details

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

### Description

Calculate log fold change and percentage of cells expressing each feature for different identity classes.

### Usage

FoldChange(object, ...)

## Default S3 method:
FoldChange(object, cells.1, cells.2, mean.fxn, fc.name, features = NULL, ...)

## S3 method for class 'Assay'
FoldChange(
  object,
  cells.1,
  cells.2,
  features = NULL,
  slot = "data",
  pseudocount.use = 1,
  fc.name = NULL,
  mean.fxn = NULL,
  base = 2,
  norm.method = NULL,
  ...
)

## S3 method for class 'SCTAssay'
FoldChange(
  object,
  cells.1,
  cells.2,
  features = NULL,
  slot = "data",
  pseudocount.use = 1,
  fc.name = NULL,
  mean.fxn = NULL,
  base = 2,
  ...
## S3 method for class 'DimReduc'

FoldChange(
  object,
  cells.1,
  cells.2,
  features = NULL,
  slot = NULL,
  pseudocount.use = 1,
  fc.name = NULL,
  mean.fxn = NULL,
  ...
)

## S3 method for class 'Seurat'

FoldChange(
  object,
  ident.1 = NULL,
  ident.2 = NULL,
  group.by = NULL,
  subset.ident = NULL,
  assay = NULL,
  slot = "data",
  reduction = NULL,
  features = NULL,
  pseudocount.use = 1,
  mean.fxn = NULL,
  base = 2,
  fc.name = NULL,
  ...
)

### Arguments

- **object**: A Seurat object
- **...**: Arguments passed to other methods
- **cells.1**: Vector of cell names belonging to group 1
- **cells.2**: Vector of cell names belonging to group 2
- **mean.fxn**: Function to use for fold change or average difference calculation
- **fc.name**: Name of the fold change, average difference, or custom function column in the output data.frame
- **features**: Features to calculate fold change for. If NULL, use all features
- **slot**: Slot to pull data from
- **pseudocount.use**: Pseudocount to add to averaged expression values when calculating logFC.
- **base**: The base with respect to which logarithms are computed.
### GetAssay

Get an Assay object from a given Seurat object.

#### Description

Get an Assay object from a given Seurat object.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm.method</td>
<td>Normalization method for mean function selection when slot is “data”</td>
</tr>
<tr>
<td>ident.1</td>
<td>Identity class to calculate fold change for; pass an object of class phylo or 'clustertree' to calculate fold change for a node in a cluster tree; passing 'clustertree' requires BuildClusterTree to have been run</td>
</tr>
<tr>
<td>ident.2</td>
<td>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 calculate fold change for</td>
</tr>
<tr>
<td>group.by</td>
<td>Regroup cells into a different identity class prior to calculating fold change (see example in FindMarkers)</td>
</tr>
<tr>
<td>subset.ident</td>
<td>Subset a particular identity class prior to regrouping. Only relevant if group.by is set (see example in FindMarkers)</td>
</tr>
<tr>
<td>assay</td>
<td>Assay to use in fold change calculation</td>
</tr>
<tr>
<td>reduction</td>
<td>Reduction to use - will calculate average difference on cell embeddings</td>
</tr>
</tbody>
</table>

#### Details

If the slot is `scale.data` or a reduction is specified, average difference is returned instead of log fold change and the column is named "avg_diff". Otherwise, log2 fold change is returned with column named "avg_log2_FC".

#### Value

Returns a data.frame

#### See Also

FindMarkers

#### Examples

```r
## Not run:
data("pbmc_small")
FoldChange(pbmc_small, ident.1 = 1)

## End(Not run)
```
Usage

GetAssay(object, ...)

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

Arguments

object An object
...
assay Assay to get

Value

Returns an Assay object

Examples

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

Description

Get Image Data

Usage

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

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

## S3 method for class 'VisiumV1'
GetImage(object, mode = c("grob", "raster", "plotly", "raw"), ...)

Arguments

object An object
mode How to return the image; should accept one of "grob", "raster", "plotly", or "raw"
...
Arguments passed to other methods
**GetIntegrationData**

*Get integration data*

### Description

Get integration data

### Usage

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

#### Arguments

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

### Value

Returns data from the requested slot within the integrated object

---

**GetResidual**

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

### Description

This function calls sctransform::get_residuals.

### Usage

```r
GetResidual(
    object,
    features,
    assay = NULL,
    umi.assay = "RNA",
    clip.range = NULL,
    replace.value = FALSE,
    na.rm = TRUE,
    verbose = TRUE
)
```
GetTissueCoordinates.SlideSeq

Arguments

- **object**: A seurat object
- **features**: Name of features to add into the scale.data
- **assay**: Name of the assay of the seurat object generated by SCTransform
- **umi.assay**: Name of the assay of the seurat object containing UMI matrix and the default is RNA
- **clip.range**: Numeric of length two specifying the min and max values the Pearson residual will be clipped to
- **replace.value**: Recalculate residuals for all features, even if they are already present. Useful if you want to change the clip.range.
- **na.rm**: For features where there is no feature model stored, return NA for residual value in scale.data when na.rm = FALSE. When na.rm is TRUE, only return residuals for features with a model stored for all cells.
- **verbose**: Whether to print messages and progress bars

Value

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

See Also

- `get_residuals`

Examples

```r
## Not run:
data("pbmc_small")
pbmc_small <- SCTransform(object = pbmc_small, variable.features.n = 20)
pbmc_small <- GetResidual(object = pbmc_small, features = c("MS4A1", "TCL1A"))
## End(Not run)
```

GetTissueCoordinates.SlideSeq

*Get Tissue Coordinates*

Description

Get Tissue Coordinates
Usage

## S3 method for class 'SlideSeq'
GetTissueCoordinates(object, ...)

## S3 method for class 'STARmap'
GetTissueCoordinates(object, qhulls = FALSE, ...)

## S3 method for class 'VisiumV1'
GetTissueCoordinates(
  object,
  scale = "lowres",
  cols = c("imagerow", "imagecol"),
  ...
)

Arguments

object An object
...
Arguments passed to other methods
qhulls return qhulls instead of centroids
scale A factor to scale the coordinates by; choose from: 'tissue', 'fiducial', 'hires', 'lowres', or NULL for no scaling
cols Columns of tissue coordinates data.frame to pull

See Also

SeuratObject::GetTissueCoordinates

GetTransferPredictions

Get the predicted identity

Description

Utility function to easily pull out the name of the class with the maximum prediction. This is useful if you’ve set prediction.assay = TRUE in TransferData and want to have a vector with the predicted class.

Usage

GetTransferPredictions(
  object,
  assay = "predictions",
  slot = "data",
  score.filter = 0.75
)
**Arguments**

- **object**: Seurat object
- **assay**: Name of the assay holding the predictions
- **slot**: Slot of the assay in which the prediction scores are stored
- **score.filter**: Return "Unassigned" for any cell with a score less than this value

**Value**

Returns a vector of predicted class names

**Examples**

```r
## Not run:
prediction.assay <- TransferData(anchorset = anchors, refdata = reference$class)
query[["predictions"]]] <- prediction.assay
query$predicted.id <- GetTransferPredictions(query)
## End(Not run)
```

---

**Graph-class**

*The Graph Class*

**Description**

For more details, please see the documentation in `SeuratObject`

**See Also**

`SeuratObject::Graph-class`

---

**GroupCorrelation**

*Compute the correlation of features broken down by groups with another covariate*

**Description**

Compute the correlation of features broken down by groups with another covariate
Usage

GroupCorrelation(
  object,
  assay = NULL,
  slot = "scale.data",
  var = NULL,
  group.assay = NULL,
  min.cells = 5,
  ngroups = 6,
  do.plot = TRUE
)

Arguments

  object          Seurat object
  assay           Assay to pull the data from
  slot            Slot in the assay to pull feature expression data from (counts, data, or scale.data)
  var             Variable with which to correlate the features
  group.assay     Compute the gene groups based off the data in this assay.
  min.cells       Only compute for genes in at least this many cells
  ngroups         Number of groups to split into
  do.plot         Display the group correlation boxplot (via GroupCorrelationPlot)

Value

  A Seurat object with the correlation stored in metafeatures

GroupCorrelationPlot  Boxplot of correlation of a variable (e.g. number of UMIs) with expression data

Description

  Boxplot of correlation of a variable (e.g. number of UMIs) with expression data

Usage

GroupCorrelationPlot(
  object,
  assay = NULL,
  feature.group = "feature.grp",
  cor = "nCount_RNA_cor"
)
**Arguments**

- `object` - Seurat object
- `assay` - Assay where the feature grouping info and correlations are stored
- `feature.group` - Name of the column in meta.features where the feature grouping info is stored
- `cor` - Name of the column in meta.features where correlation info is stored

**Value**

Returns a ggplot boxplot of correlations split by group

---

**Harmony Integration**

**Description**

Harmony Integration

**Usage**

```r
HarmonyIntegration(
  object,
  orig,
  features = NULL,
  scale.layer = "scale.data",
  new.reduction = "harmony",
  layers = NULL,
 npcs = 50L,
  key = "harmony_",
  theta = NULL,
  lambda = NULL,
  sigma = 0.1,
  nclust = NULL,
  tau = 0,
  block.size = 0.05,
  max.iter.harmony = 10L,
  max.iter.cluster = 20L,
  epsilon.cluster = 1e-05,
  epsilon.harmony = 1e-04,
  verbose = TRUE,
  ...
)
```

...
Arguments

- **object**: An Assay5 object
- **orig**: A dimensional reduction to correct
- **features**: Ignored
- **scale.layer**: Ignored
- **new.reduction**: Name of new integrated dimensional reduction
- **layers**: Ignored
- **npcs**: If doing PCA on input matrix, number of PCs to compute
- **key**: Key for Harmony dimensional reduction
- **theta**: Diversity clustering penalty parameter
- **lambda**: Ridge regression penalty parameter
- **sigma**: Width of soft kmeans clusters
- **nclust**: Number of clusters in model
- **tau**: Protection against overclustering small datasets with large ones
- **block.size**: What proportion of cells to update during clustering
- **max.iter.harmony**: Maximum number of rounds to run Harmony
- **max.iter.cluster**: Maximum number of rounds to run clustering at each round of Harmony
- **epsilon.cluster**: Convergence tolerance for clustering round of Harmony
- **epsilon.harmony**: Convergence tolerance for Harmony
- **verbose**: Whether to print progress messages. TRUE to print, FALSE to suppress
- **...**: Ignored

Value

... 

Note

This function requires the harmony package to be installed

See Also

harmony::HarmonyMatrix()
Examples

```r
## Not run:
# Preprocessing
obj <- SeuratData::LoadData("pbmcscsa")
obj[['RNA']] <- split(obj[['RNA']], f = obj$Method)
obj <- NormalizeData(obj)
obj <- FindVariableFeatures(obj)
obj <- ScaleData(obj)
obj <- RunPCA(obj)

# After preprocessing, we integrate layers with added parameters specific to Harmony:
obj <- IntegrateLayers(object = obj, method = HarmonyIntegration, orig.reduction = "pca",
                      new.reduction = 'harmony', verbose = FALSE)

# Modifying Parameters
# We can also add arguments specific to Harmony such as theta, to give more diverse clusters
obj <- IntegrateLayers(object = obj, method = HarmonyIntegration, orig.reduction = "pca",
                      new.reduction = 'harmony', verbose = FALSE, theta = 3)

# Integrating SCTransformed data
obj <- SCTransform(object = obj)
obj <- IntegrateLayers(object = obj, method = HarmonyIntegration,
                      orig.reduction = "pca", new.reduction = 'harmony',
                      assay = "SCT", verbose = FALSE)
```

## End(Not run)

---

**HoverLocator**

**Hover Locator**

**Description**

Get quick information from a scatterplot by hovering over points

**Usage**

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

**Arguments**

- `plot`: A `ggplot2` plot
- `information`: An optional dataframe or matrix of extra information to be displayed on hover
- `axes`: Display or hide x- and y-axes
- `dark.theme`: Plot using a dark theme?
- `...`: Extra parameters to be passed to `layout`
See Also

    layout ggplot_build DimPlot FeaturePlot

Examples

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

## End(Not run)
```

### Description

Demultiplex samples based on data from cell ‘hashing’

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

### Usage

```r
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
HTOHeatmap

kfunc  Clustering function for initial hashtag grouping. Default is "clara" for fast k-medoids clustering on large applications, also support "kmeans" for kmeans clustering
	nsamples  Number of samples to be drawn from the dataset used for clustering, for kfunc = "clara"

seed  Sets the random seed. If NULL, seed is not set

verbose  Prints the output

Value

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

hash.maxID  Name of hashtag with the highest signal

hash.secondID  Name of hashtag with the second highest signal

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

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

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

hash.ID  Classification result where doublet IDs are collapsed

See Also

HTOHeatmap

Examples

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

## End(Not run)
**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.SCTAssay  

Get Variable Feature Information

Description
Get variable feature information from SCTAssay objects

Usage
## S3 method for class 'SCTAssay'
HVFInfo(object, method, status = FALSE, ...)

Arguments
object  
An object
method  
method to determine variable features
status  
Add variable status to the resulting data frame
...  
Arguments passed to other methods

See Also
HVFInfo

Examples
## Not run:
# Get the HVF info directly from an SCTAssay object
pbmc_small <- SCTransform(pbmc_small)
HVFInfo(pbmc_small[["SCT"]], method = 'sct')[1:5, ]

## End(Not run)

IFeaturePlot  

Visualize features in dimensional reduction space interactively

Description
Visualize features in dimensional reduction space interactively

Usage
IFeaturePlot(object, feature, dims = c(1, 2), reduction = NULL, slot = "data")
**ImageDimPlot**

**Arguments**
- `object` Seurat object
- `feature` Feature to plot
- `dims` Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions
- `reduction` Which dimensionality reduction to use. If not specified, first searches for umap, then tsne, then pca
- `slot` Which slot to pull expression data from?

**Value**
Returns the final plot as a ggplot object

---

**Spatial Cluster Plots**

**Description**
Visualize clusters or other categorical groupings in a spatial context

**Usage**

```r
ImageDimPlot(
  object,
  fov = NULL,
  boundaries = NULL,
  group.by = NULL,
  split.by = NULL,
  cols = NULL,
  shuffle.cols = FALSE,
  size = 0.5,
  molecules = NULL,
  mols.size = 0.1,
  mols.cols = NULL,
  mols.alpha = 1,
  nmols = 1000,
  alpha = 1,
  border.color = "white",
  border.size = NULL,
  na.value = "grey50",
  dark.background = TRUE,
  crop = FALSE,
  cells = NULL,
  overlap = FALSE,
  axes = FALSE,
)```
combine = TRUE,
coord.fixed = TRUE,
flip.xy = TRUE
)

Arguments

object A Seurat object
fov Name of FOV to plot
boundaries A vector of segmentation boundaries per image to plot; can be a character vector,
a named character vector, or a named list. Names should be the names of FOVs
and values should be the names of segmentation boundaries
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 A factor in object metadata to split the plot by, pass ‘ident’ to split by cell identity’
cols Vector of colors, each color corresponds to an identity class. This may also
be a single character or numeric value corresponding to a palette as specified
by brewer_pal.info. By default, ggplot2 assigns colors. We also include a
number of palettes from the pals package. See DiscretePalette for details.
shuffle.cols Randomly shuffle colors when a palette or vector of colors is provided to cols
size Point size for cells when plotting centroids
molecules A vector of molecules to plot
mols.size Point size for molecules
mols.cols A vector of color for molecules. The "Set1" palette from RColorBrewer is used
by default.
mols.alpha Alpha value for molecules, should be between 0 and 1
nmols Max number of each molecule specified in ‘molecules’ to plot
alpha Alpha value for plotting (default is 1)
border.color Color of cell segmentation border; pass NA to suppress borders for segmentation-based plots
border.size Thickness of cell segmentation borders; pass NA to suppress borders for centroid-based plots
na.value Color value for NA points when using custom scale
dark.background Set plot background to black
crop Crop the plots to area with cells only
cells Vector of cells to plot (default is all cells)
overlap Overlay boundaries from a single image to create a single plot; if TRUE, then
boundaries are stacked in the order they’re given (first is lowest)
axes Keep axes and panel background
combine Combine plots into a single patchwork ggplot object. If FALSE, return a list of
ggplot objects
coord.fixed Plot cartesian coordinates with fixed aspect ratio
flip.xy Flag to flip X and Y axes. Default is FALSE.
**Value**

If `combine = TRUE`, a patchwork `ggplot` object; otherwise, a list of `ggplot` objects.

---

**ImageFeaturePlot**

*Spatial Feature Plots*

**Description**

Visualize expression in a spatial context.

**Usage**

```r
ImageFeaturePlot(
  object,  # object
  features,  # features
  fov = NULL,  # field of view
  boundaries = NULL,  # boundaries
  cols = if (isTRUE(x = blend)) {
    c("lightgrey", "#ff0000", "#00ff00")
  } else {
    c("lightgrey", "firebrick1")
  },
  size = 0.5,  # size
  min.cutoff = NA,  # minimum cutoff
  max.cutoff = NA,  # maximum cutoff
  split.by = NULL,  # split by
  molecules = NULL,  # molecules
  mols.size = 0.1,  # molecule size
  mols.cols = NULL,  # molecule colors
  nmols = 1000,  # number of molecules
  alpha = 1,  # transparency
  border.color = "white",  # border color
  border.size = NULL,  # border size
  dark.background = TRUE,  # dark background
  blend = FALSE,  # blend
  blend.threshold = 0.5,  # blend threshold
  crop = FALSE,  # crop
  cells = NULL,  # cells
  scale = c("feature", "all", "none"),  # scale
  overlap = FALSE,  # overlap
  axes = FALSE,  # axes
  combine = TRUE,  # combine
  coord.fixed = TRUE  # coordinate fixed
)
```
ImageFeaturePlot

**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")
- **fov**: Name of FOV to plot
- **boundaries**: A vector of segmentation boundaries per image to plot; can be a character vector, a named character vector, or a named list. Names should be the names of FOVs and values should be the names of segmentation boundaries
- **cols**: The two colors to form the gradient over. Provide as string vector with the first color corresponding to low values, the second to high. Also accepts a Brewer color scale or vector of colors. Note: this will bin the data into number of colors provided. When blend is TRUE, takes anywhere from 1-3 colors:
  - **1 color**: Treated as color for double-negatives, will use default colors 2 and 3 for per-feature expression
  - **2 colors**: Treated as colors for per-feature expression, will use default color 1 for double-negatives
  - **3+ colors**: First color used for double-negatives, colors 2 and 3 used for per-feature expression, all others ignored
- **size**: Point size for cells when plotting centroids
- **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’)
- **split.by**: A factor in object metadata to split the plot by, pass ‘ident’ to split by cell identity
- **molecules**: A vector of molecules to plot
- **mols.size**: Point size for molecules
- **mols.cols**: A vector of color for molecules. The "Set1" palette from RColorBrewer is used by default.
- **nmols**: Max number of each molecule specified in ‘molecules’ to plot
- **alpha**: Alpha value for plotting (default is 1)
- **border.color**: Color of cell segmentation border; pass NA to suppress borders for segmentation-based plots
- **border.size**: Thickness of cell segmentation borders; pass NA to suppress borders for centroid-based plots
- **dark.background**: Set plot background to black
- **blend**: Scale and blend expression values to visualize coexpression of two features
IntegrateData

blend.threshold
  The color cutoff from weak signal to strong signal; ranges from 0 to 1.
crop
  Crop the plots to area with cells only
cells
  Vector of cells to plot (default is all cells)
scale
  Set color scaling across multiple plots; choose from:
  - "feature": Plots per-feature are scaled across splits
  - "all": Plots per-feature are scaled across all features
  - "none": Plots are not scaled; **note**: setting scale to "none" will result in color scales that are not comparable between plots
  Ignored if blend = TRUE
overlap
  Overlay boundaries from a single image to create a single plot; if TRUE, then boundaries are stacked in the order they’re given (first is lowest)
axes
  Keep axes and panel background
combine
  Combine plots into a single patchwork ggplot object. If FALSE, return a list of ggplot objects
coord.fixed
  Plot cartesian coordinates with fixed aspect ratio

Value

If combine = TRUE, a patchwork ggplot object; otherwise, a list of ggplot objects

---

IntegrateData

Integrate data

Description

Perform dataset integration using a pre-computed AnchorSet.

Usage

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

anchorset An AnchorSet object generated by FindIntegrationAnchors
new.assay.name Name for the new assay containing the integrated data
normalization.method Name of normalization method used: LogNormalize or SCT
features Vector of features to use when computing the PCA to determine the weights. Only set if you want a different set from those used in the anchor finding process
features.to.integrate Vector of features to integrate. By default, will use the features used in anchor finding.
dims Number of dimensions to use in the anchor weighting procedure
k.weight Number of neighbors to consider when weighting anchors
weight.reduction Dimension reduction to use when calculating anchor weights. This can be one of:
  • A string, specifying the name of a dimension reduction present in all objects to be integrated
  • A vector of strings, specifying the name of a dimension reduction to use for each object to be integrated
  • A vector of DimReduc objects, specifying the object to use for each object in the integration
  • NULL, in which case a new PCA will be calculated and used to calculate anchor weights
Note that, if specified, the requested dimension reduction will only be used for calculating anchor weights in the first merge between reference and query, as the merged object will subsequently contain more cells than was in query, and weights will need to be calculated for all cells in the object.

sd.weight Controls the bandwidth of the Gaussian kernel for weighting
sample.tree Specify the order of integration. Order of integration should be encoded in a matrix, where each row represents one of the pairwise integration steps. Negative numbers specify a dataset, positive numbers specify the integration results from a given row (the format of the merge matrix included in the hclust function output). For example: matrix(c(-2, 1, -3, -1), ncol = 2) gives:

\[
\begin{bmatrix}
[1,] & [1,] & [1,] & [1,] \\
[2,] & -2 & -2 & -2 \\
[3,] & 1 & -1 & -1 \\
\end{bmatrix}
\]

Which would cause dataset 2 and 3 to be integrated first, then the resulting object integrated with dataset 1.
If NULL, the sample tree will be computed automatically.
preserve.order Do not reorder objects based on size for each pairwise integration.
eps Error bound on the neighbor finding algorithm (from RANN)
verbose Print progress bars and output
IntegrateData

Details

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

For pairwise integration:

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

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

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

Value

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

References


Examples

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

# panc8 is a merged Seurat object containing 8 separate pancreas datasets
# split the object by dataset
pancreas.list <- SplitObject(panc8, split.by = "tech")
```
# perform standard preprocessing on each object
for (i in 1:length(pancreas.list)) {
  pancreas.list[[i]] <- NormalizeData(pancreas.list[[i]], verbose = FALSE)
  pancreas.list[[i]] <- FindVariableFeatures(
    pancreas.list[[i]], selection.method = "vst",
    nfeatures = 2000, verbose = FALSE
  )
}

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

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

## End(Not run)

---

**IntegrateEmbeddings**

*Integrate low dimensional embeddings*

**Description**

Perform dataset integration using a pre-computed Anchorset of specified low dimensional representations.

**Usage**

IntegrateEmbeddings(anchorset, ...)

## S3 method for class 'IntegrationAnchorSet'
IntegrateEmbeddings(
  anchorset,
  new.reduction.name = "integrated_dr",
  reductions = NULL,
  dims.to.integrate = NULL,
  k.weight = 100,
  weight.reduction = NULL,
  sd.weight = 1,
  sample.tree = NULL,
  preserve.order = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'TransferAnchorSet'
IntegrateEmbeddings(
  anchorset,
reference,
query,
query.assay = NULL,
new.reduction.name = "integrated_dr",
reductions = "pcaproject",
dims.to.integrate = NULL,
k.weight = 100,
weight.reduction = NULL,
reuse.weights.matrix = TRUE,
.sd.weight = 1,
preserve.order = FALSE,
verbose = TRUE,
...
)

Arguments

anchorset

An AnchorSet object

... Reserved for internal use

new.reduction.name

Name for new integrated dimensional reduction.

terms

Name of reductions to be integrated. For a TransferAnchorSet, this should be the
name of a reduction present in the anchorset object (for example, "pcaproject").
For an IntegrationAnchorSet, this should be a DimReduc object containing all
cells present in the anchorset object.

dims.to.integrate

Number of dimensions to return integrated values for

k.weight

Number of neighbors to consider when weighting anchors

weight.reduction

Dimension reduction to use when calculating anchor weights. This can be one
of:

• A string, specifying the name of a dimension reduction present in all objects
to be integrated
• A vector of strings, specifying the name of a dimension reduction to use for
each object to be integrated
• A vector of DimReduc objects, specifying the object to use for each object
in the integration
• NULL, in which case the full corrected space is used for computing anchor
weights.

sd.weight

Controls the bandwidth of the Gaussian kernel for weighting

sample.tree

Specify the order of integration. Order of integration should be encoded in a ma-
trix, where each row represents one of the pairwise integration steps. Negative
numbers specify a dataset, positive numbers specify the integration results from
given row (the format of the merge matrix included in the hclust function
output). For example: matrix(c(-2, 1, -3, -1), ncol = 2) gives:
Which would cause dataset 2 and 3 to be integrated first, then the resulting object integrated with dataset 1. If NULL, the sample tree will be computed automatically.

**Details**

The main steps of this procedure are identical to `IntegrateData` with one key distinction. When computing the weights matrix, the distance calculations are performed in the full space of integrated embeddings when integrating more than two datasets, as opposed to a reduced PCA space which is the default behavior in `IntegrateData`.

**Value**

When called on a TransferAnchorSet (from FindTransferAnchors), this will return the query object with the integrated embeddings stored in a new reduction. When called on an IntegrationAnchorSet (from IntegrateData), this will return a merged object with the integrated reduction stored.

```r
IntegrateLayers(object, method, orig.reduction = "pca", assay = NULL, features = NULL, layers = NULL, scale.layer = "scale.data", ...
)
```
Arguments

- **object**: A Seurat object
- **method**: Integration method function
- **orig.reduction**: Name of dimensional reduction for correction
- **assay**: Name of assay for integration
- **features**: A vector of features to use for integration
- **layers**: Names of normalized layers in assay
- **scale.layer**: Name(s) of scaled layer(s) in assay
- **...**: Arguments passed on to method

Value

- **object** with integration data added to it

Integration Method Functions

The following integration method functions are available:

See Also

- Writing integration method functions

---

IntegrationAnchorSet-class

*The IntegrationAnchorSet Class*

Description

Inherits from the Anchorset class. Implemented mainly for method dispatch purposes. See AnchorSet for slot details.

---

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.n-cell**: Number of cells in each object in the object.list
- **sample.tree**: Sample tree used for ordering multi-dataset integration

---

**ISpatialDimPlot**  
*Visualize clusters spatially and interactively*

Description

Visualize clusters spatially and interactively

Usage

`ISpatialDimPlot(object, image = NULL, group.by = NULL, alpha = c(0.3, 1))`

Arguments

- **object**: A Seurat object
- **image**: Name of the image to use in the plot
- **group.by**: Name of meta.data column to group the data by
- **alpha**: Controls opacity of spots. Provide as a vector specifying the min and max for SpatialFeaturePlot. For SpatialDimPlot, provide a single alpha value for each plot.

Value

Returns final plot as a ggplot object
**ISpatialFeaturePlot**  
*Visualize features spatially and interactively*

**Description**

Visualize features spatially and interactively

**Usage**

```r
ISpatialFeaturePlot(
  object,
  feature,
  image = NULL,
  slot = "data",
  alpha = c(0.1, 1)
)
```

**Arguments**

- **object**: A Seurat object
- **feature**: Feature to visualize
- **image**: Name of the image to use in the plot
- **slot**: If plotting a feature, which data slot to pull from (counts, data, or scale.data)
- **alpha**: Controls opacity of spots. Provide as a vector specifying the min and max for SpatialFeaturePlot. For SpatialDimPlot, provide a single alpha value for each plot.

**Value**

Returns final plot as a ggplot object

---

**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

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:
data("pbmc_small")
pbmc_small = suppressWarnings(JackStraw(pbmc_small))
head(JS(object = pbmc_small[['pca']], slot = 'empirical'))

## End(Not run)
```
JackStrawData-class  The JackStrawData Class

Description
For more details, please see the documentation in SeuratObject.

See Also
SeuratObject::JackStrawData-class

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,
  cols = NULL,
  reduction = "pca",
  xmax = 0.1,
  ymax = 0.3
)

Arguments
object  Seurat object
dims  Dims to plot
cols  Vector of colors, each color corresponds to an individual PC. This may also be a single character or numeric value corresponding to a palette as specified by brewer.pal.info. By default, ggplot2 assigns colors. We also include a number of palettes from the pals package. See DiscretePalette for details.
reduction  reduction to pull jackstraw info from
xmax  X-axis maximum on each QQ plot.
ymax  Y-axis maximum on each QQ plot.
JointPCAIntegration

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

data("pbmc_small")
JackStrawPlot(object = pbmc_small)

JointPCAIntegration Seurat-Joint PCA Integration

Description

Seurat-Joint PCA Integration

Usage

JointPCAIntegration(
  object = NULL,
  assay = NULL,
  layers = NULL,
  orig = NULL,
  new.reduction = "integrated.dr",
  reference = NULL,
  features = NULL,
  normalization.method = c("LogNormalize", "SCT"),
  dims = 1:30,
  k.anchor = 20,
  scale.layer = "scale.data",
  dims.to.integrate = NULL,
  k.weight = 100,
JointPCAIntegration

weight.reduction = NULL,
sd.weight = 1,
sample.tree = NULL,
preserve.order = FALSE,
verbose = TRUE,
...

Arguments

object A Seurat object
assay Name of Assay in the Seurat object
layers Names of layers in assay
orig A dimensional reduction to correct
new.reduction Name of new integrated dimensional reduction
reference A reference Seurat object
features A vector of features to use for integration
normalization.method Name of normalization method used: LogNormalize or SCT
dims Dimensions of dimensional reduction to use for integration
k.anchor How many neighbors (k) to use when picking anchors
scale.layer Name of scaled layer in Assay
dims.to.integrate Number of dimensions to return integrated values for
k.weight Number of neighbors to consider when weighting anchors
weight.reduction Dimension reduction to use when calculating anchor weights. This can be one of:
  • A string, specifying the name of a dimension reduction present in all objects to be integrated
  • A vector of strings, specifying the name of a dimension reduction to use for each object to be integrated
  • A vector of DimReduc objects, specifying the object to use for each object in the integration
  • NULL, in which case the full corrected space is used for computing anchor weights.
sd.weight Controls the bandwidth of the Gaussian kernel for weighting
sample.tree Specify the order of integration. Order of integration should be encoded in a matrix, where each row represents one of the pairwise integration steps. Negative numbers specify a dataset, positive numbers specify the integration results from a given row (the format of the merge matrix included in the hclust function output). For example: matrix(c(-2, 1, -3, -1), ncol = 2) gives:
```
[,1] [,2]
[1,] -2 -3
[2,]  1 -1
```

Which would cause dataset 2 and 3 to be integrated first, then the resulting object integrated with dataset 1.

If NULL, the sample tree will be computed automatically.

- `preserve.order` Do not reorder objects based on size for each pairwise integration.
- `verbose` Print progress
- `...` Arguments passed on to `FindIntegrationAnchors`

---

### L2CCA

**L2-Normalize CCA**

**Description**

Perform l2 normalization on CCs

**Usage**

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

**Arguments**

- `object` Seurat object
- `...` Additional parameters to `L2Dim`.

---

### L2Dim

**L2-normalization**

**Description**

Perform l2 normalization on given dimensional reduction

**Usage**

```r
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,
  box = FALSE,
  geom = "GeomPoint",
  position = "median",
  ...
)

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
box Use geom_label/geom_label_repel (includes a box around the text labels)
geom Name of geom to get X/Y aesthetic names for
position How to place the label if repel = FALSE. If "median", place the label at the median position. If "nearest" place the label at the position of the nearest data point to the median.
...
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

```r
data("pbmc_small")
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

```r
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`
LeverageScore

Examples

data("pbmc_small")
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

This function computes the leverage scores for a given object. It uses the concept of sketching and random projections. The function provides an approximation to the leverage scores using a scalable method suitable for large matrices.

Usage

LeverageScore(object, ...)

## Default S3 method:
LeverageScore(
  object,
  nsketch = 5000L,
  ndims = NULL,
  method = CountSketch,
  eps = 0.5,
  seed = 123L,
  verbose = TRUE,
  ...
)

## S3 method for class 'StdAssay'
LeverageScore(
  object,
  nsketch = 5000L,
  ndims = NULL,
  method = CountSketch,
  vf.method = NULL,
  layer = "data",
  eps = 0.5,
  seed = 123L,
  verbose = TRUE,
  ...
)
## S3 method for class 'Assay'
LeverageScore(
  object,
  nsketch = 5000L,
  ndims = NULL,
  method = CountSketch,
  vf.method = NULL,
  layer = "data",
  eps = 0.5,
  seed = 123L,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
LeverageScore(
  object,
  assay = NULL,
  nsketch = 5000L,
  ndims = NULL,
  var.name = "leverage.score",
  over.write = FALSE,
  method = CountSketch,
  vf.method = NULL,
  layer = "data",
  eps = 0.5,
  seed = 123L,
  verbose = TRUE,
  ...
)

### Arguments

- **object**: A matrix-like object
- **...**: Arguments passed to other methods
- **nsketch**: A positive integer. The number of sketches to be used in the approximation. Default is 5000.
- **ndims**: A positive integer or NULL. The number of dimensions to use. If NULL, the number of dimensions will default to the number of columns in the object.
- **method**: The sketching method to use, defaults to CountSketch.
- **eps**: A numeric. The error tolerance for the approximation in Johnson–Lindenstrauss embeddings, defaults to 0.5.
- **seed**: A positive integer. The seed for the random number generator, defaults to 123.
- **verbose**: Print progress and diagnostic messages
- **vf.method**: VariableFeatures method
- **layer**: layer to use
assay  assay to use
var.name  name of slot to store leverage scores
over.write  whether to overwrite slot that currently stores leverage scores. Defaults to FALSE, in which case the ‘var.name’ is modified if it already exists in the object

References

LinkedPlots  Visualize spatial and clustering (dimensional reduction) data in a linked, interactive framework

Description
Visualize spatial and clustering (dimensional reduction) data in a linked, interactive framework

Usage
LinkedDimPlot(
  object,
  dims = 1:2,
  reduction = NULL,
  image = NULL,
  group.by = NULL,
  alpha = c(0.1, 1),
  combine = TRUE
)

LinkedFeaturePlot(
  object,
  feature,
  dims = 1:2,
  reduction = NULL,
  image = NULL,
  slot = “data”,
  alpha = c(0.1, 1),
  combine = TRUE
)

Arguments
  object  A Seurat object
dims  Dimensions to plot, must be a two-length numeric vector specifying x- and y-dimensions
Load10X_Spatial

Load a 10x Genomics Visium Spatial Experiment into a Seurat object

Description

Load a 10x Genomics Visium Spatial Experiment into a Seurat object

Usage

Load10X_Spatial(
  data.dir,
  filename = "filtered_feature_bc_matrix.h5",
  assay = "Spatial",
  slice = "slice1",
  filter.matrix = TRUE,
  to.upper = FALSE,
  image = NULL,
  ...
)
LoadAnnoyIndex

Load the Annoy index file

Description
Load the Annoy index file

Usage
LoadAnnoyIndex(object, file)

Arguments

object Neighbor object
file Path to file with annoy index

Value
Returns the Neighbor object with the index stored
LoadCurioSeeker  Load Curio Seeker data

Description
Load Curio Seeker data

Usage
LoadCurioSeeker(data.dir, assay = "Spatial")

Arguments
- data.dir: location of data directory that contains the counts matrix, gene names, barcodes/beads, and barcodes/bead location files.
- assay: Name of assay to associate spatial data to

Value
A Seurat object

LoadSTARmap  Load STARmap data

Description
Load STARmap data

Usage
LoadSTARmap(
  data.dir,
  counts.file = "cell_barcode_count.csv",
  gene.file = "genes.csv",
  qhull.file = "qhulls.tsv",
  centroid.file = "centroids.tsv",
  assay = "Spatial",
  image = "image"
)

## Description

Read and Load 10x Genomics Xenium in-situ data

### Usage

```r
LoadXenium(data.dir, fov = "fov", assay = "Xenium")

ReadXenium(
  data.dir,
  outs = c("matrix", "microns"),
  type = "centroids",
  mols.qv.threshold = 20
)
```

### Arguments

- `data.dir`: Directory containing all Xenium output files with default filenames
- `fov`: FOV name
- `assay`: Assay name
- `outs`: Types of molecular outputs to read; choose one or more of:
  - "matrix": the counts matrix
  - "microns": molecule coordinates
LocalStruct

**Type**

Type of cell spatial coordinate matrices to read; choose one or more of:

- “centroids”: cell centroids in pixel coordinate space
- “segmentations”: cell segmentations in pixel coordinate space

**mols.qv.threshold**

Remove transcript molecules with a QV less than this threshold. QV >= 20 is the standard threshold used to construct the cell x gene count matrix.

**Value**

LoadXenium: A Seurat object

ReadXenium: A list with some combination of the following values:

- “matrix”: a sparse matrix with expression data; cells are columns and features are rows
- “centroids”: a data frame with cell centroid coordinates in three columns: “x”, “y”, and “cell”
- “pixels”: a data frame with molecule pixel coordinates in three columns: “x”, “y”, and “gene”

---

**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  
)
```
LogNormalize

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>grouping.var</td>
<td>Grouping variable</td>
</tr>
<tr>
<td>idents</td>
<td>Optionally specify a set of idents to compute metric for</td>
</tr>
<tr>
<td>neighbors</td>
<td>Number of neighbors to compute in pca/corrected pca space</td>
</tr>
<tr>
<td>reduction</td>
<td>Dimensional reduction to use for corrected space</td>
</tr>
<tr>
<td>reduced.dims</td>
<td>Number of reduced dimensions to use</td>
</tr>
<tr>
<td>orig.dims</td>
<td>Number of PCs to use in original space</td>
</tr>
<tr>
<td>verbose</td>
<td>Display progress bar</td>
</tr>
</tbody>
</table>

Value

Returns the average preservation metric

Description

Normalize Raw Data

Usage

LogNormalize(data, scale.factor = 10000, margin = 2L, verbose = TRUE, ...)

## S3 method for class 'data.frame'
LogNormalize(data, scale.factor = 10000, margin = 2L, verbose = TRUE, ...)

## S3 method for class 'V3Matrix'
LogNormalize(data, scale.factor = 10000, margin = 2L, verbose = TRUE, ...)

## Default S3 method:
LogNormalize(data, scale.factor = 10000, margin = 2L, verbose = TRUE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Matrix with the raw count data</td>
</tr>
<tr>
<td>scale.factor</td>
<td>Scale the data; default is 1e4</td>
</tr>
<tr>
<td>margin</td>
<td>Margin to normalize over</td>
</tr>
<tr>
<td>verbose</td>
<td>Print progress</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
</tbody>
</table>
LogVMR

Value

A matrix with the normalized 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
- `...`: Other arguments (not used)

Value

Returns the VMR in log-space

Examples

```r
LogVMR(x = c(1, 2, 3))
```
**MappingScore**

**Metric for evaluating mapping success**

**Description**

This metric was designed to help identify query cells that aren’t well represented in the reference dataset. The intuition for the score is that we are going to project the query cells into a reference-defined space and then project them back onto the query. By comparing the neighborhoods before and after projection, we identify cells who’s local neighborhoods are the most affected by this transformation. This could be because there is a population of query cells that aren’t present in the reference or the state of the cells in the query is significantly different from the equivalent cell type in the reference.

**Usage**

```r
MappingScore(anchors, ...)
```

## Default S3 method:

```r
MappingScore(
  anchors,
  combined.object,
  query.neighbors,
  ref.embeddings,
  query.embeddings,
  kanchors = 50,
  ndim = 50,
  ksmooth = 100,
  ksnn = 20,
  snn.prune = 0,
  subtract.first.nn = TRUE,
  nn.method = "annoy",
  n.trees = 50,
  query.weights = NULL,
  verbose = TRUE,
  ...
)
```

## S3 method for class 'AnchorSet'

```r
MappingScore(  
  anchors,
  kanchors = 50,
  ndim = 50,
  ksmooth = 100,
  ksnn = 20,
  snn.prune = 0,
  subtract.first.nn = TRUE,
  nn.method = "annoy",
  n.trees = 50,
  query.weights = NULL,
  verbose = TRUE,
  ...
  )
```
n.trees = 50,  
query.weights = NULL,  
verbose = TRUE,
...
)

**Arguments**

- **anchors**  
  AnchorSet object or just anchor matrix from the Anchorset object returned from FindTransferAnchors
  
- **combined.object**  
  Combined object (ref + query) from the Anchorset object returned

- **query.neighbors**  
  Neighbors object computed on query cells

- **ref.embeddings**  
  Reference embeddings matrix

- **query.embeddings**  
  Query embeddings matrix

- **kanchors**  
  Number of anchors to use in projection steps when computing weights

- **ndim**  
  Number of dimensions to use when working with low dimensional projections of the data

- **ksmooth**  
  Number of cells to average over when computing transition probabilities

- **ksnn**  
  Number of cells to average over when determining the kernel bandwidth from the SNN graph

- **snn.prune**  
  Amount of pruning to apply to edges in SNN graph

- **subtract.first.nn**  
  Option to the scoring function when computing distances to subtract the distance to the first nearest neighbor

- **nn.method**  
  Nearest neighbor method to use (annoy or RANN)

- **n.trees**  
  More trees gives higher precision when using annoy approximate nearest neighbor search

- **query.weights**  
  Query weights matrix for reuse

- **verbose**  
  Display messages/progress

**Value**

Returns a vector of cell scores
MapQuery

### Map query cells to a reference

**Description**

This is a convenience wrapper function around the following three functions that are often run together when mapping query data to a reference: `TransferData`, `IntegrateEmbeddings`, `ProjectUMAP`. Note that by default, the `weight.reduction` parameter for all functions will be set to the dimension reduction method used in the `FindTransferAnchors` function call used to construct the anchor object, and the `dims` parameter will be the same dimensions used to find anchors.

**Usage**

```r
MapQuery(
  anchorset,  # An AnchorSet object
  query,     # Query object used in anchorset construction
  reference, # Reference object used in anchorset construction
  refdata = NULL,  # Data to transfer. This can be specified in one of two ways:
                   # The reference data itself as either a vector where the names correspond to
                   # the reference cells, or a matrix, where the column names correspond to
                   # the reference cells.
                   # The name of the metadata field or assay from the reference object provided.
                   # This requires the reference parameter to be specified. If pulling assay data
                   # in this manner, it will pull the data from the data slot. To transfer data from
                   # other slots, please pull the data explicitly with `GetAssayData` and provide
                   # that matrix here.
  new.reduction.name = NULL,  # Name for new integrated dimensional reduction.
  reference.reduction = NULL,
  reference.dims = NULL,
  query.dims = NULL,
  store.weights = FALSE,
  reduction.model = NULL,
  transferdata.args = list(),
  integrateembeddings.args = list(),
  projectumap.args = list(),
  verbose = TRUE
)
```

**Arguments**

- `anchorset`: An AnchorSet object
- `query`: Query object used in anchorset construction
- `reference`: Reference object used in anchorset construction
- `refdata`: Data to transfer. This can be specified in one of two ways:
  - The reference data itself as either a vector where the names correspond to
    the reference cells, or a matrix, where the column names correspond to
    the reference cells.
  - The name of the metadata field or assay from the reference object provided.
    This requires the reference parameter to be specified. If pulling assay data
    in this manner, it will pull the data from the data slot. To transfer data from
    other slots, please pull the data explicitly with `GetAssayData` and provide
    that matrix here.
- `new.reduction.name`: Name for new integrated dimensional reduction.
merge.SCTAssay

merge.SCTAssay

S3 method for class 'SCTAssay'

merge(
  x = NULL,
  y = NULL,
  add.cell.ids = NULL,
  merge.data = TRUE,
  na.rm = TRUE,
  ...
)

merge.SCTAssay

Merge SCTAssay objects

Description

Merge SCTAssay objects

Usage

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

Value

Returns a modified query Seurat object containing:

- New Assays corresponding to the features transferred and/or their corresponding prediction scores from TransferData
- An integrated reduction from IntegrateEmbeddings
- A projected UMAP reduction of the query cells projected into the reference UMAP using ProjectUMAP
**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**

```r
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

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

---

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

Description

Apply a ceiling and floor to all values in a matrix

Usage

MinMax(data, min, max)

Arguments

data  
Matrix or data frame

min  
all values below this min value will be replaced with min

max  
all values above this max value will be replaced with max

Value

Returns matrix after performing these floor and ceil operations

Examples

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

Calculates a mixing metric

Description

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

Usage

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

Arguments

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

Value

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

Differential expression heatmap for mixscape

Description

Draws a heatmap of single cell feature expression with cells ordered by their mixscape ko probabilities.

Usage

MixscapeHeatmap(
  object,
  ident.1 = NULL,
  ident.2 = NULL,
  balanced = TRUE,
  logfc.threshold = 0.25,
  assay = "RNA",
  max.genes = 100,
  test.use = "wilcox",
  max.cells.group = NULL,
  order.by.prob = TRUE,
  group.by = NULL,
  mixscape.class = "mixscape_class",
  prtb.type = "KO",
  fc.name = "avg_log2FC",
  pval.cutoff = 0.05,
  ...
)

Arguments

object An object
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
balanced Plot an equal number of genes with both groups of cells.
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.
assay Assay to use in differential expression testing
max.genes Total number of DE genes to plot.
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); will use a fast implementation by Presto if installed
- "wilcox_limma": Identifies differentially expressed genes between two groups of cells using the limma implementation of the Wilcoxon Rank Sum test; set this option to reproduce results from Seurat v4
- "bimod": Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc": Identifies 'markers' of gene expression using ROC analysis. For each gene, evaluates (using AUC) a classifier built on that gene alone, to classify between two groups of cells. An AUC value of 1 means that expression values for this gene alone can perfectly classify the two groupings (i.e. Each of the cells in cells.1 exhibit a higher level than each of the cells in cells.2). An AUC value of 0 also means there is perfect classification, but in the other direction. A value of 0.5 implies that the gene has no predictive power to classify the two groups. Returns a 'predictive power' (abs(AUC-0.5) * 2) ranked matrix of putative differentially expressed genes.
- "t": Identify differentially expressed genes between two groups of cells using the Student’s t-test.
- "negbinom": Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model. Use only for UMI-based datasets
- "poisson": Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model. Use only for UMI-based datasets
- "LR": Uses a logistic regression framework to determine differentially expressed genes. Constructs a logistic regression model predicting group membership based on each feature individually and compares this to a null model with a likelihood ratio test.
- "MAST": Identifies differentially expressed genes between two groups of cells using a hurdle model tailored to scRNA-seq data. Utilizes the MAST package to run the DE testing.
- "DESeq2": Identifies differentially expressed genes between two groups of cells based on a model using DESeq2 which uses a negative binomial distribution (Love et al, Genome Biology, 2014). This test does not support pre-filtering of genes based on average difference (or percent detection rate) between cell groups. However, genes may be pre-filtered based on their minimum detection rate (min.pct) across both cell groups. To use this method, please install DESeq2, using the instructions at https://bioconductor.org/packages/release/bioc/html/DESeq2.html

max.cells.group  Number of cells per identity to plot.

order.by.prob  Order cells on heatmap based on their mixscape knockout probability from highest to lowest score.

(group.by  (Deprecated) Option to split densities based on mixscape classification. Please use mixscape.class instead)
MixscapeLDA function performs unsupervised PCA on each mixscape class separately and projects each subspace onto all cells in the data. Finally, it uses the first 10 principle components from each projection as input to lda in MASS package together with mixscape class labels.

**Arguments**

- `object`: An object of class Seurat.
- `assay`: Assay to use for performing Linear Discriminant Analysis (LDA).
- `ndims.print`: Number of LDA dimensions to print.
- `nfeatures.print`: Number of features to print for each LDA component.
ModalityWeights-class

reduction.key  Reduction key name.
seed  Value for random seed
pc.assay  Assay to use for running Principle components analysis.
labels  Meta data column with target gene class labels.
nt.label  Name of non-targeting cell class.
npcs  Number of principle components to use.
verbose  Print progress bar.
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.

Value

Returns a Seurat object with LDA added in the reduction slot.

ModalityWeights-class  The ModalityWeights Class

Description

The ModalityWeights class is an intermediate data storage class that stores the modality weight and other related information needed for performing downstream analyses - namely data integration (FindModalityWeights) and data transfer (FindMultiModalNeighbors).

Slots

modal.weight.list  A list of modality weights value from all modalities
modal.assay  Names of assays for the list of dimensional reductions
params  A list of parameters used in the FindModalityWeights
score.matrix  a list of score matrices representing cross and within-modality prediction score, and kernel value
command  Store log of parameters that were used
MULTIseqDemux

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

Description

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

Usage

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

Arguments

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

Value

A Seurat object with demultiplexing results stored at object$MULTI_ID

References

https://www.biorxiv.org/content/10.1101/387241v1

Examples

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

## End(Not run)
The Neighbor Class

Description

For more details, please see the documentation in `SeuratObject`.

See Also

`SeuratObject::Neighbor-class`

```
NNPlot

Highlight Neighbors in DimPlot

Description

It will color the query cells and the neighbors of the query cells in the DimPlot

Usage

NNPlot(
  object,  
  reduction,  
  nn.idx,  
  query.cells,  
  dims = 1:2,  
  label = FALSE,  
  label.size = 4,  
  repel = FALSE,  
  sizes.highlight = 2,  
  pt.size = 1,  
  cols.highlight = c("#377eb8", "#e41a1c"),  
  na.value = "#bdbdbd",  
  order = c("self", "neighbors", "other"),  
  show.all.cells = TRUE,  
  ...
)
```

Arguments

- `object`: `Seurat` object
- `reduction`: Which dimensionality reduction to use. If not specified, first searches for umap, then tsne, then pca
- `nn.idx`: the neighbor index of all cells
query.cells: cells used to find their neighbors

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

label: Whether to label the clusters

label.size: Sets size of labels

repel: Repel labels

sizes.highlight: Size of highlighted cells; will repeat to the length groups in cells.highlight. If
sizes.highlight = TRUE size of all points will be this value.

pt.size: Adjust point size for plotting

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

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

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)

show.all.cells: Show all cells or only query and neighbor cells

...: Extra parameters passed to DimPlot

Value

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

---

**NNtoGraph**

*Convert Neighbor class to an asymmetrical Graph class*

**Description**

Convert Neighbor class to an asymmetrical Graph class

**Usage**

```r
NNtoGraph(nn.object, col.cells = NULL, weighted = FALSE)
```

**Arguments**

- **nn.object**: A neighbor class object
- **col.cells**: Cells names of the neighbors, cell names in `nn.object` is used by default
- **weighted**: Determine if use distance in the Graph

**Value**

Returns a Graph object
Description

Normalize the count data present in a given assay.

Usage

NormalizeData(object, ...)

## S3 method for class 'V3Matrix'
NormalizeData(
  object,
  normalization.method = "LogNormalize",
  scale.factor = 10000,
  margin = 1,
  block.size = NULL,
  verbose = TRUE,
  ...
)

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

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

Arguments

object An object
... Arguments passed to other methods
normalization.method
Method for normalization.

- "LogNormalize": Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. This is then natural-log transformed using \( \log(1+p) \)
- "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:
data("pbmc_small")
pbmc_small
pbmc_small <- NormalizeData(object = pbmc_small)

## End(Not run)
```

---

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

---

### PercentAbove

**Calculate the percentage of a vector above some threshold**

**Description**

Calculate the percentage of a vector above some threshold

**Usage**

```r
PercentAbove(x, threshold)
```

**Arguments**

- **x**: Vector of values
- **threshold**: Threshold to use when calculating percentage

**Value**

Returns the percentage of x values above the given threshold

**Examples**

```r
set.seed(42)
PercentAbove(sample(1:100, 10), 75)
```
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

data("pbmc_small")
# 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, direction = "downwards", ...)

Arguments

object    Seurat object
direction  A character string specifying the direction of the tree (default is downwards)
           Possible options: "rightwards", "leftwards", "upwards", and "downwards".
...       Additional arguments to \texttt{ape::plot.phylo}

Value

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

Examples

```r
## Not run:
if (requireNamespace("ape", quietly = TRUE)) {
  data("pbmc_small")
  pbmc_small <- BuildClusterTree(object = pbmc_small)
  PlotClusterTree(object = pbmc_small)
}
## End(Not run)
```

PlotPerturbScore

Function to plot perturbation score distributions.

Description

Density plots to visualize perturbation scores calculated from RunMixscape function.
Usage

PlotPerturbScore(
  object,
  target.gene.class = "gene",
  target.gene.ident = NULL,
  mixscape.class = "mixscape_class",
  col = "orange2",
  split.by = NULL,
  before.mixscape = FALSE,
  prtb.type = "KO"
)

Arguments

  object An object of class Seurat.
  target.gene.class meta data column specifying all target gene names in the experiment.
  target.gene.ident Target gene name to visualize perturbation scores for.
  mixscape.class meta data column specifying mixscape classifications.
  col Specify color of target gene class or knockout cell class. For control non-targeting and non-perturbed cells, colors are set to different shades of grey.
  split.by For datasets with more than one cell type. Set equal TRUE to visualize perturbation scores for each cell type separately.
  before.mixscape Option to split densities based on mixscape classification (default) or original target gene classification. Default is set to NULL and plots cells by original class ID.
  prtb.type specify type of CRISPR perturbation expected for labeling mixscape classifications. Default is KO.

Value

A ggplot object.

PolyDimPlot  Polygon DimPlot

Description

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

**Usage**

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

**Arguments**

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

**Value**

Returns a ggplot object

---

**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 DimReduce 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, 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

---

**PredictAssay**

**Predict value from nearest neighbors**

Description

This function will predict expression or cell embeddings from its k nearest neighbors index. For each cell, it will average its k neighbors value to get its new imputed value. It can average expression value in assays and cell embeddings from dimensional reductions.

Usage

```r
PredictAssay(
  object,  # Seurat object
  nn.idx,  # k nearest neighbors index
  assay,   # Assay object
  reduction = NULL,  # Reduction object
  dims = NULL,  # Dimensional reduction values
  return.assay = TRUE,  # Return Assay object
  slot = "scale.data",  # Slot for scaling data
  features = NULL,  # Features to plot
  mean.function = rowMeans,  # Function to average neighbors
  seed = 4273,  # Random seed
  verbose = TRUE  # Verbose output
)
```
Arguments

- **object**: The object used to calculate knn
- **nn.idx**: k near neighbour indices. A cells x k matrix.
- **assay**: Assay used for prediction
- **reduction**: Cell embedding of the reduction used for prediction
- **dims**: Number of dimensions of cell embedding
- **return.assay**: Return an assay or a predicted matrix
- **slot**: slot used for prediction
- **features**: features used for prediction
- **mean.function**: the function used to calculate row mean
- **seed**: Sets the random seed to check if the nearest neighbor is query cell
- **verbose**: Print progress

Value

return an assay containing predicted expression value in the data slot

---

PrepareBridgeReference

Prepare the bridge and reference datasets

Description

Preprocess the multi-omic bridge and unimodal reference datasets into an extended reference. This function performs the following three steps: 1. Performs within-modality harmonization between bridge and reference 2. Performs dimensional reduction on the SNN graph of bridge datasets via Laplacian Eigendecomposition 3. Constructs a bridge dictionary representation for unimodal reference cells

Usage

```r
PrepareBridgeReference(
  reference,
  bridge,
  reference.reduction = "pca",
  reference.dims = 1:50,
  normalization.method = c("SCT", "LogNormalize"),
  reference.assay = NULL,
  bridge.ref.assay = "RNA",
  bridge.query.assay = "ATAC",
  supervised.reduction = c("slsi", "spca", NULL),
  bridge.query.reduction = NULL,
  bridge.query.features = NULL,
)```

PrepareBridgeReference

```r
laplacian.reduction.name = "lap",
laplacian.reduction.key = "lap_",
laplacian.reduction.dims = 1:50,
verbose = TRUE
)
```

Arguments

- `reference` A reference Seurat object
- `bridge` A multi-omic bridge Seurat object
- `reference.reduction` Name of dimensional reduction of the reference object (default is `pca`)
- `reference.dims` Number of dimensions used for the reference.reduction (default is 50)
- `normalization.method` Name of normalization method used: LogNormalize or SCT
- `reference.assay` Assay name for reference (default is `DefaultAssay`)
- `bridge.ref.assay` Assay name for bridge used for reference mapping. RNA by default
- `bridge.query.assay` Assay name for bridge used for query mapping. ATAC by default
- `supervised.reduction` Type of supervised dimensional reduction to be performed for integrating the bridge and query. # Options are:
  - `slsi`: Perform supervised LSI as the dimensional reduction for the bridge-query integration
  - `spca`: Perform supervised PCA as the dimensional reduction for the bridge-query integration
  - `NULL`: no supervised dimensional reduction will be calculated. `bridge.query.reduction` is used for the bridge-query integration
- `bridge.query.reduction` Name of dimensions used for the bridge-query harmonization. `bridge.query.reduction` and `supervised.reduction` cannot be NULL together.
- `bridge.query.features` Features used for bridge query dimensional reduction (default is NULL which uses VariableFeatures from the bridge object)
- `laplacian.reduction.name` Name of dimensional reduction name of graph laplacian eigenspace (default is `lap`)
- `laplacian.reduction.key` Dimensional reduction key (default is `lap_`)
- `laplacian.reduction.dims` Number of dimensions used for graph laplacian eigenspace (default is 50)
- `verbose` Print progress and message (default is TRUE)
PrepLDA

Value

Returns a BridgeReferenceSet that can be used as input to FindBridgeTransferAnchors. The parameters used are stored in the BridgeReferenceSet as well.

Description

This function performs unsupervised PCA on each mixscape class separately and projects each subspace onto all cells in the data.

Usage

PrepLDA(
  object,
  de.assay = "RNA",
  pc.assay = "PRTB",
  labels = "gene",
  nt.label = "NT",
 npcs = 10,
  verbose = TRUE,
  logfc.threshold = 0.25
)

Arguments

object An object of class Seurat.
de.assay Assay to use for selection of DE genes.
pc.assay Assay to use for running Principle components analysis.
labels Meta data column with target gene class labels.
nt.label Name of non-targeting cell class.
npcs Number of principle components to use.
verbose Print progress bar.
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.

Value

Returns a list of the first 10 PCs from each projection.
PrepSCTFindMarkers

Prepare object to run differential expression on SCT assay with multiple models

Description

Given a merged object with multiple SCT models, this function uses minimum of the median UMI (calculated using the raw UMI counts) of individual objects to reverse the individual SCT regression model using minimum of median UMI as the sequencing depth covariate. The counts slot of the SCT assay is replaced with recorrected counts and the data slot is replaced with log1p of recorrected counts.

Usage

PrepSCTFindMarkers(object, assay = "SCT", verbose = TRUE)

Arguments

- object: Seurat object with SCT assays
- assay: Assay name where for SCT objects are stored; Default is 'SCT'
- verbose: Print messages and progress

Value

Returns a Seurat object with recorrected counts and data in the SCT assay.

Progress Updates with progressr

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

Parallelization with future

This function uses future to enable parallelization. Parallelization strategies can be set using plan. Common plans include “sequential” for non-parallelized processing or “multisession” for parallel evaluation using multiple R sessions; for other plans, see the “Implemented evaluation strategies” section of ?future::plan. For a more thorough introduction to future, see vignette("future-1-overview")

Examples

data("pbmc_small")
pbmc_small1 <- SCTransform(object = pbmc_small, variable.features.n = 20, vst.flavor="v1")
pbmc_small2 <- SCTransform(object = pbmc_small, variable.features.n = 20, vst.flavor="v1")
pbmc_merged <- merge(x = pbmc_small1, y = pbmc_small2)
pbmc_merged <- PrepSCTFindMarkers(object = pbmc_merged)
markers <- FindMarkers(
object = pbmc_merged,
ident.1 = "0",
ident.2 = "1",
assay = "SCT"
)
pbm_subset <- subset(pbmc_merged, idents = c("0", "1"))
markers_subset <- FindMarkers(
  object = pbm_subset,
  ident.1 = "0",
  ident.2 = "1",
  assay = "SCT",
  recorrect_umi = FALSE
)

---

**PrepSCTIntegration**

Prepare an object list normalized with sctransform for integration.

**Description**

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

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

**Usage**

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

**Arguments**

- `object.list`: A list of Seurat objects to prepare for integration
**PrepSCTIntegration**

- **assay**
  The name of the **Assay** to use for integration. This can be a single name if all the assays to be integrated have the same name, or a character vector containing the name of each **Assay** in each object to be integrated. The specified assays must have been normalized using **SCTransform**. If NULL (default), the current default assay for each object is used.

- **anchor.features**
  Can be either:
  - A numeric value. This will call **SelectIntegrationFeatures** to select the provided number of features to be used in anchor finding
  - A vector of features to be used as input to the anchor finding process

- **sct.clip.range**
  Numeric of length two specifying the min and max values the Pearson residual will be clipped to

- **verbose**
  Display output/messages

**Value**

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

**Examples**

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

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

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

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

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

## End(Not run)
```
ProjectData

Project full data to the sketch assay

Description

This function allows projection of high-dimensional single-cell RNA expression data from a full dataset onto the lower-dimensional embedding of the sketch of the dataset.

Usage

```r
ProjectData(
  object,
  assay = "RNA",
  sketched.assay = "sketch",
  sketched.reduction,
  full.reduction,
  dims,
  normalization.method = c("LogNormalize", "SCT"),
  refdata = NULL,
  k.weight = 50,
  umap.model = NULL,
  recompute.neighbors = FALSE,
  recompute.weights = FALSE,
  verbose = TRUE
)
```

Arguments

- **object**: A Seurat object.
- **assay**: Assay name for the full data. Default is 'RNA'.
- **sketched.assay**: Sketched assay name to project onto. Default is 'sketch'.
- **sketched.reduction**: Dimensional reduction results of the sketched assay to project onto.
- **full.reduction**: Dimensional reduction name for the projected full dataset.
- **dims**: Dimensions to include in the projection.
- **normalization.method**: Normalization method to use. Can be 'LogNormalize' or 'SCT'. Default is 'LogNormalize'.
- **refdata**: An optional list for label transfer from sketch to full data. Default is NULL. Similar to refdata in ‘MapQuery’
- **k.weight**: Number of neighbors to consider when weighting labels for transfer. Default is 50.
- **umap.model**: An optional pre-computed UMAP model. Default is NULL.
- **recompute.neighbors**: Whether to recompute the neighbors for label transfer. Default is FALSE.
ProjectDim

-recompute.weights-
Whether to recompute the weights for label transfer. Default is FALSE.

-verbose-
Print progress and diagnostic messages.

Value
A Seurat object with the full data projected onto the sketched dimensional reduction results. The projected data are stored in the specified full reduction.

| ProjectDim | Project Dimensional reduction onto full dataset |

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

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

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

Value
Returns Seurat object with the projected values
Examples

data("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)

ProjectDimReduc  Project query data to reference dimensional reduction

Description

Project query data to reference dimensional reduction

Usage

ProjectDimReduc(
  query,
  reference,
  mode = c("pcaprox", "lsprox"),
  reference.reduction,
  combine = FALSE,
  query.assay = NULL,
  reference.assay = NULL,
  features = NULL,
  do.scale = TRUE,
  reduction.name = NULL,
  reduction.key = NULL,
  verbose = TRUE
)

Arguments

query  Query object
reference  Reference object
mode  Projection mode name for projection
  • pcaprox: PCA projection
  • lsprox: LSI projection
reference.reduction  Name of dimensional reduction in the reference object
combine  Determine if query and reference objects are combined
query.assay  Assay used for query object
reference.assay  Assay used for reference object
ProjectIntegration

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>features</td>
<td>Features used for projection</td>
</tr>
<tr>
<td>do.scale</td>
<td>Determine if scale expression matrix in the pcaproject mode</td>
</tr>
<tr>
<td>reduction.name</td>
<td>Dimensional reduction name, reference.reduction is used by default</td>
</tr>
<tr>
<td>reduction.key</td>
<td>Dimensional reduction key, the key in reference.reduction is used by default</td>
</tr>
<tr>
<td>verbose</td>
<td>Print progress and message</td>
</tr>
</tbody>
</table>

Value

Returns a query-only or query-reference combined seurat object

Usage

ProjectIntegration(
  object,
  sketched.assay = "sketch",
  assay = "RNA",
  reduction = "integrated_dr",
  features = NULL,
  layers = "data",
  reduction.name = NULL,
  reduction.key = NULL,
  method = c("sketch", "data"),
  ratio = 0.8,
  sketched.layers = NULL,
  seed = 123,
  verbose = TRUE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object with all cells for one dataset</td>
</tr>
<tr>
<td>sketched.assay</td>
<td>Assay name for sketched-cell expression (default is 'sketch')</td>
</tr>
<tr>
<td>assay</td>
<td>Assay name for original expression (default is 'RNA')</td>
</tr>
<tr>
<td>reduction</td>
<td>Dimensional reduction name for batch-corrected embeddings in the sketched</td>
</tr>
<tr>
<td></td>
<td>object (default is 'integrated_dr')</td>
</tr>
<tr>
<td>features</td>
<td>Features used for atomic sketch integration</td>
</tr>
<tr>
<td>layers</td>
<td>Names of layers for correction.</td>
</tr>
</tbody>
</table>
ProjectUMAP

reduction.name Name to save new reduction as; defaults to paste0(reduction, '.orig')

reduction.key Key for new dimensional reduction; defaults to creating one from reduction.name

method Methods to construct sketch-cell representation for all cells (default is 'sketch').
Can be one of:
   • "sketch": Use random sketched data slot
   • "data": Use data slot

ratio Sketch ratio of data slot when dictionary.method is set to "sketch"; defaults to 0.8

sketched.layers Names of sketched layers, defaults to all layers of "object[[assay]]"

seed A positive integer. The seed for the random number generator, defaults to 123.

verbose Print progress and message

Details

First learn a atom dictionary representation to reconstruct each cell. Then, using this dictionary representation, reconstruct the embeddings of each cell from the integrated atoms.

Value

Returns a Seurat object with an integrated dimensional reduction

Usage

ProjectUMAP(query, ...)

## Default S3 method:
ProjectUMAP(
    query,
    query.dims = NULL,
    reference,
    reference.dims = NULL,
)

Project query into UMAP coordinates of a reference

Description

This function will take a query dataset and project it into the coordinates of a provided reference UMAP. This is essentially a wrapper around two steps:

   • FindNeighbors - Find the nearest reference cell neighbors and their distances for each query cell.
   • RunUMAP - Perform umap projection by providing the neighbor set calculated above and the umap model previously computed in the reference.
k.param = 30,
nn.method = "annoy",
n.trees = 50,
annoy.metric = "cosine",
l2.norm = FALSE,
cache.index = TRUE,
index = NULL,
neighbor.name = "query_ref.nn",
reduction.model,
...
)

## S3 method for class 'DimReduc'
ProjectUMAP(
  query,
  query.dims = NULL,
  reference,
  reference.dims = NULL,
  k.param = 30,
  nn.method = "annoy",
  n.trees = 50,
  annoy.metric = "cosine",
  l2.norm = FALSE,
  cache.index = TRUE,
  index = NULL,
  neighbor.name = "query_ref.nn",
  reduction.model,
  ...
)

## S3 method for class 'Seurat'
ProjectUMAP(
  query,
  query.reduction,
  query.dims = NULL,
  reference,
  reference.reduction,
  reference.dims = NULL,
  k.param = 30,
  nn.method = "annoy",
  n.trees = 50,
  annoy.metric = "cosine",
  l2.norm = FALSE,
  cache.index = TRUE,
  index = NULL,
  neighbor.name = "query_ref.nn",
  reduction.model,
  reduction.name = "ref.umap",
  ...)
Arguments

query Query dataset

... Additional parameters to RunUMAP

query.dims Dimensions (columns) to use from query

reference Reference dataset

reference.dims Dimensions (columns) to use from reference

k.param Defines k for the k-nearest neighbor algorithm

nn.method Method for nearest neighbor finding. Options include: rann, annoy

n.trees More trees gives higher precision when using annoy approximate nearest neighbor search

annoy.metric Distance metric for annoy. Options include: euclidean, cosine, manhattan, and hamming

l2.norm Take L2Norm of the data

cache.index Include cached index in returned Neighbor object (only relevant if return.neighbor = TRUE)

index Precomputed index. Useful if querying new data against existing index to avoid recomputing.

neighbor.name Name to store neighbor information in the query

reduction.model DimReduc object that contains the umap model

query.reduction Name of reduction to use from the query for neighbor finding

reference.reduction Name of reduction to use from the reference for neighbor finding

reduction.name Name of projected UMAP to store in the query

reduction.key Value for the projected UMAP key

Description

Normalize the count data present in a given assay.

Usage

PseudobulkExpression(object, ...)
Arguments

object      An assay
...        Arguments passed to other methods

Value

Returns object after normalization

---

### Radius.SlideSeq

Get Spot Radius

---

Description

Get Spot Radius

Usage

```r
## S3 method for class 'SlideSeq'
Radius(object)
```

```r
## S3 method for class 'STARmap'
Radius(object)
```

```r
## S3 method for class 'VisiumV1'
Radius(object)
```

Arguments

object      An image object

See Also

SeuratObject::Radius

---

### Read10X

Load in data from 10X

---

Description

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

Read10X(
  data.dir,
  gene.column = 2,
  cell.column = 1,
  unique.features = TRUE,
  strip.suffix = FALSE
)

Arguments

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

Value

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

Examples

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

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

## End(Not run)
Read10X_h5  Read 10X hdf5 file

Description

Read count matrix from 10X CellRanger h5 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).

Read10X_Image  Load a 10X Genomics Visium Image

Description

Load a 10X Genomics Visium Image

Usage

Read10X_Image(image.dir, filter.matrix = TRUE, ...)

Arguments

image.dir  Path to directory with 10X Genomics visium image data; should include files tissue_lowres_image.png, scalefactors_json.json and tissue_positions_list.csv
filter.matrix  Filter spot/feature matrix to only include spots that have been determined to be over tissue.
...  Ignored for now

Value

A VisiumV1 object
Read10X_probe_metadata

Read10x Probe Metadata

Description

This function reads the probe metadata from a 10x Genomics probe barcode matrix file in HDF5 format.

Usage

Read10X_probe_metadata(data.dir, filename = "raw_probe_bc_matrix.h5")

Arguments

data.dir The directory where the file is located.
filename The name of the file containing the raw probe barcode matrix in HDF5 format. The default filename is 'raw_probe_bc_matrix.h5'.

Value

Returns a data.frame containing the probe metadata.

ReadAkoya

Read and Load Akoya CODEX data

Description

Read and Load Akoya CODEX data

Usage

ReadAkoya(
    filename, 
    type = c("inform", "processor", "qupath"),
    filter = "DAPI|Blank|Empty",
    inform.quant = c("mean", "total", "min", "max", "std")
)

LoadAkoya(
    filename, 
    type = c("inform", "processor", "qupath"),
)
```r
fov, assay = "Akoya",
... )
```

### Arguments

- **filename**: Path to matrix generated by upstream processing.
- **type**: Specify which type matrix is being provided.
  - "processor": matrix generated by CODEX Processor
  - "inform": matrix generated by inForm
  - "qupath": matrix generated by QuPath
- **filter**: A pattern to filter features by; pass NA to skip feature filtering
- **inform.quant**: When type is "inform", the quantification level to read in
- **fov**: Name to store FOV as
- **assay**: Name to store expression matrix as
- **...**: Ignored

### Value

**ReadAkoya**: A list with some combination of the following values

- "matrix": a sparse matrix with expression data; cells are columns and features are rows
- "centroids": a data frame with cell centroid coordinates in three columns: "x", "y", and "cell"
- "metadata": a data frame with cell-level meta data; includes all columns in filename that aren't in "matrix" or "centroids"

When type is "inform", additional expression matrices are returned and named using their segmentation type (eg. "nucleus", "membrane"). The "Entire Cell" segmentation type is returned in the "matrix" entry of the list.

**LoadAkoya**: A Seurat object

### Progress Updates with progressr

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

### Note

This function requires the `data.table` package to be installed
ReadMtx

Load in data from remote or local mtx files

Description

Enables easy loading of sparse data matrices

Usage

ReadMtx(
  mtx,  
cells,  
features,  
cell.column = 1,  
feature.column = 2,  
cell.sep = "\t",  
feature.sep = "\t",  
skip.cell = 0,  
skip.feature = 0,  
mtx.transpose = FALSE,  
unique.features = TRUE,  
strip.suffix = FALSE
)

Arguments

mtx          Name or remote URL of the mtx file  
cells        Name or remote URL of the cells/barcodes file  
features      Name or remote URL of the features/genes file  
cell.column  Specify which column of cells file to use for cell names; default is 1  
feature.column  Specify which column of features files to use for feature/gene names; default is 2  
feature.column  Specify the delimiter in the cell name file  
feature.sep    Specify the delimiter in the feature name file  
skip.cell      Number of lines to skip in the cells file before beginning to read cell names  
skip.feature   Number of lines to skip in the features file before beginning to gene names  
 mtx.transpose  Transpose the matrix after reading in  
 unique.features   Make feature names unique (default TRUE)  
 strip.suffix    Remove trailing "-1" if present in all cell barcodes.

Value

A sparse matrix containing the expression data.
Examples

## Not run:
# For local files:

expression_matrix <- ReadMtx(
  mtx = "count_matrix.mtx.gz", features = "features.tsv.gz",
  cells = "barcodes.tsv.gz"
)
seurat_object <- CreateSeuratObject(counts = expression_matrix)

# For remote files:

expression_matrix <- ReadMtx(mtx = "http://localhost/matrix.mtx",
  cells = "http://localhost/barcodes.tsv",
  features = "http://localhost/genes.tsv")
seurat_object <- CreateSeuratObject(counts = data)

## End(Not run)

ReadNanostring  Read and Load Nanostring SMI data

Description

Read and Load Nanostring SMI data

Usage

ReadNanostring(
  data.dir,
  mtx.file = NULL,
  metadata.file = NULL,
  molecules.file = NULL,
  segmentations.file = NULL,
  type = "centroids",
  mol.type = "pixels",
  metadata = NULL,
  mols.filter = NA_character_,
  genes.filter = NA_character_,
  fov.filter = NULL,
  subset.counts.matrix = NULL,
  cell.mols.only = TRUE
)

LoadNanostring(data.dir, fov, assay = "Nanostring")
ReadNanostring

Arguments

data.dir  Path to folder containing Nanostring SMI outputs
mtx.file  Path to Nanostring cell x gene matrix CSV
metadata.file  Contains metadata including cell center, area, and stain intensities
molecules.file  Path to molecules file
segmentations.file  Path to segmentations CSV
type  Type of cell spatial coordinate matrices to read; choose one or more of:
  • “centroids”: cell centroids in pixel coordinate space
  • “segmentations”: cell segmentations in pixel coordinate space
mol.type  Type of molecule spatial coordinate matrices to read; choose one or more of:
  • “pixels”: molecule coordinates in pixel space
metadata  Type of available metadata to read; choose zero or more of:
  • “Area”: number of pixels in cell segmentation
  • “fov”: cell’s fov
  • “Mean.MembraneStain”: mean membrane stain intensity
  • “Mean.DAPI”: mean DAPI stain intensity
  • “Mean.G”: mean green channel stain intensity
  • “Mean.Y”: mean yellow channel stain intensity
  • “Mean.R”: mean red channel stain intensity
  • “Max.MembraneStain”: max membrane stain intensity
  • “Max.DAPI”: max DAPI stain intensity
  • “Max.G”: max green channel stain intensity
  • “Max.Y”: max yellow stain intensity
  • “Max.R”: max red stain intensity
mols.filter  Filter molecules that match provided string
genes.filter  Filter genes from cell x gene matrix that match provided string
fov.filter  Only load in select FOVs. Nanostring SMI data contains 30 total FOVs.
subset.counts.matrix  If the counts matrix should be built from molecule coordinates for a specific segmentation; One of:
  • “Nuclear”: nuclear segmentations
  • “Cytoplasm”: cell cytoplasm segmentations
  • “Membrane”: cell membrane segmentations
cell.mols.only  If TRUE, only load molecules within a cell
fov  Name to store FOV as
assay  Name to store expression matrix as
ReadParseBio

Value

ReadNanostring: A list with some combination of the following values:

- “matrix”: a sparse matrix with expression data; cells are columns and features are rows
- “centroids”: a data frame with cell centroid coordinates in three columns: “x”, “y”, and “cell”
- “pixels”: a data frame with molecule pixel coordinates in three columns: “x”, “y”, and “gene”

LoadNanostring: A Seurat object

Progress Updates with progressr

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

Parallelization with future

This function uses future to enable parallelization. Parallelization strategies can be set using plan. Common plans include “sequential” for non-parallelized processing or “multisession” for parallel evaluation using multiple R sessions; for other plans, see the “Implemented evaluation strategies” section of ?future::plan. For a more thorough introduction to future, see vignette("future-1-overview")

Note

This function requires the data.table package to be installed

Description

Read output from Parse Biosciences

Usage

ReadParseBio(data.dir, ...)

Arguments

data.dir Directory containing the data files
...
Extra parameters passed to ReadMtx
ReadSlideSeq

Load Slide-seq spatial data

Description
Load Slide-seq spatial data

Usage
ReadSlideSeq(coord.file, assay = "Spatial")

Arguments
coord.file Path to csv file containing bead coordinate positions
assay Name of assay to associate image to

Value
A SlideSeq object

See Also

SlideSeq

ReadSTARsolo
Read output from STARsolo

Description
Read output from STARsolo

Usage
ReadSTARsolo(data.dir, ...)

Arguments
data.dir Directory containing the data files
... Extra parameters passed to ReadMtx
ReadVitessce (Read Data From Vitessce)

Description
Read in data from Vitessce-formatted JSON files

Usage
ReadVitessce(
  counts = NULL,
  coords = NULL,
  molecules = NULL,
  type = c("segmentations", "centroids"),
  filter = NA_character_
)

LoadHuBMAPCODEX(data.dir, fov, assay = "CODEX")

Arguments
- **counts**: Path or URL to a Vitessce-formatted JSON file with expression data; should end in "genes.json" or "clusters.json"; pass NULL to skip
- **coords**: Path or URL to a Vitessce-formatted JSON file with cell/spot spatial coordinates; should end in "cells.json"; pass NULL to skip
- **molecules**: Path or URL to a Vitessce-formatted JSON file with molecule spatial coordinates; should end in "molecules.json"; pass NULL to skip
- **type**: Type of cell/spot spatial coordinates to return, choose one or more from:
  - “segmentations” cell/spot segmentations
  - “centroids” cell/spot centroids
- **filter**: A character to filter molecules by, pass NA to skip molecule filtering
- **data.dir**: Path to a directory containing Vitessce cells and clusters JSONs
- **fov**: Name to store FOV as
- **assay**: Name to store expression matrix as

Value
ReadVitessce: A list with some combination of the following values:
- “counts”: if counts is not NULL, an expression matrix with cells as columns and features as rows
- “centroids”: if coords is not NULL and type contains “centroids”, a data frame with cell centroids in three columns: “x”, “y”, and “cell”
- “segmentations”: if coords is not NULL and type contains “centroids”, a data frame with cell segmentations in three columns: “x”, “y” and “cell”
ReadVizgen

Description

Read and load MERFISH input from Vizgen-formatted files

Usage

ReadVizgen(
  data.dir,
  transcripts = NULL,
  spatial = NULL,
  molecules = NULL,
)
type = "segmentations",
mol.type = "microns",
metadata = NULL,
filter = NA_character_,
z = 3L
)

LoadVizgen(data.dir, fov, assay = "Vizgen", z = 3L)

Arguments

data.dir Path to the directory with Vizgen MERFISH files; requires at least one of the following files present:
• "cell_by_gene.csv": used for reading count matrix
• "cell_metadata.csv": used for reading cell spatial coordinate matrices
• "detected_transcripts.csv": used for reading molecule spatial coordinate matrices

transcripts Optional file path for counts matrix; pass NA to suppress reading counts matrix

spatial Optional file path for spatial metadata; pass NA to suppress reading spatial coordinates. If spatial is provided and type is "segmentations", uses dirname(spatial) instead of data.dir to find HDF5 files

molecules Optional file path for molecule coordinates file; pass NA to suppress reading spatial molecule information

type Type of cell spatial coordinate matrices to read; choose one or more of:
• "segmentations": cell segmentation vertices; requires hdf5r to be installed and requires a directory "cell_boundaries" within data.dir. Within "cell_boundaries", there must be one or more HDF5 file named "feature_data_##.hdf5"
• "centroids": cell centroids in micron coordinate space
• "boxes": cell box outlines in micron coordinate space

mol.type Type of molecule spatial coordinate matrices to read; choose one or more of:
• "pixels": molecule coordinates in pixel space
• "microns": molecule coordinates in micron space

metadata Type of available metadata to read; choose zero or more of:
• "volume": estimated cell volume
• "fov": cell's fov

filter A character to filter molecules by, pass NA to skip molecule filtering

z Z-index to load; must be between 0 and 6, inclusive

fov Name to store FOV as

assay Name to store expression matrix as
RegroupIdents

Value

ReadVizgen: A list with some combination of the following values:

- “transcripts”: a sparse matrix with expression data; cells are columns and features are rows
- “segmentations”: a data frame with cell polygon outlines in three columns: “x”, “y”, and “cell”
- “centroids”: a data frame with cell centroid coordinates in three columns: “x”, “y”, and “cell”
- “boxes”: a data frame with cell box outlines in three columns: “x”, “y”, and “cell”
- “microns”: a data frame with molecule micron coordinates in three columns: “x”, “y”, and “gene”
- “pixels”: a data frame with molecule pixel coordinates in three columns: “x”, “y”, and “gene”
- “metadata”: a data frame with the cell-level metadata requested by metadata

LoadVizgen: A Seurat object

Progress Updates with progressr

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

Parallelization with future

This function uses future to enable parallelization. Parallelization strategies can be set using plan. Common plans include “sequential” for non-parallelized processing or “multisession” for parallel evaluation using multiple R sessions; for other plans, see the “Implemented evaluation strategies” section of ?future::plan. For a more thorough introduction to future, see vignette("future-1-overview")

Note

This function requires the data.table package to be installed

RegroupIdents

Regroup Idents based on meta.data info

Description

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

Usage

RegroupIdents(object, metadata)
RelativeCounts

Arguments

object Seurat object
metadata Name of metadata column

Value

A Seurat object with the active idents regrouped

Examples

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

RelativeCounts Normalize raw data to fractions

Description

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

Usage

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

Arguments

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

Value

Returns a matrix with the relative counts

Examples

mat <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
mat
mat_norm <- RelativeCounts(data = mat)
mat_norm
renameCells.SCTAssay  Rename Cells in an Object

Description

Rename Cells in an Object

Usage

## S3 method for class 'SCTAssay'
renameCells(object, new.names = NULL, ...)

## S3 method for class 'SlideSeq'
renameCells(object, new.names = NULL, ...)

## S3 method for class 'STARmap'
renameCells(object, new.names = NULL, ...)

## S3 method for class 'VisiumV1'
renameCells(object, new.names = NULL, ...)

Arguments

object  An object
new.names  vector of new cell names
...  Arguments passed to other methods

See Also

SeuratObject::renameCells

RidgePlot  Single cell ridge plot

Description

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

RidgePlot(
  object,
  features,
  cols = NULL,
  idents = NULL,
  sort = FALSE,
  assay = NULL,
  group.by = NULL,
  y.max = NULL,
  same.y.lims = FALSE,
  log = FALSE,
  ncol = NULL,
  slot = deprecated(),
  layer = "data",
  stack = FALSE,
  combine = TRUE,
  fill.by = "feature"
)

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
  slot Slot to pull expression data from (e.g. "counts" or "data")
  layer Layer to pull expression data from (e.g. "counts" or "data")
  stack Horizontally stack plots for each feature
  combine Combine plots into a single patchwork ggplot object. If FALSE, return a list of ggplot
  fill.by Color violins/ridges based on either ‘feature’ or ‘ident’

Value

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

```r
data("pbmc_small")
RidgePlot(object = pbmc_small, features = 'PC_1')
```

RPCAIntegration

Seurat-RPCA Integration

Description

Seurat-RPCA Integration

Usage

```r
RPCAIntegration(
  object = NULL,
  assay = NULL,
  layers = NULL,
  orig = NULL,
  new.reduction = "integrated.dr",
  reference = NULL,
  features = NULL,
  normalization.method = c("LogNormalize", "SCT"),
  dims = 1:30,
  k.filter = NA,
  scale.layer = "scale.data",
  dims.to.integrate = NULL,
  k.weight = 100,
  weight.reduction = NULL,
  sd.weight = 1,
  sample.tree = NULL,
  preserve.order = FALSE,
  verbose = TRUE,
...
)
```

Arguments

- **object**: A Seurat object
- **assay**: Name of Assay in the Seurat object
- **layers**: Names of layers in assay
- **orig**: A **dimensional reduction** to correct
- **new.reduction**: Name of new integrated dimensional reduction
- **reference**: A reference Seurat object
- **features**: A vector of features to use for integration
normalization.method
  Name of normalization method used: LogNormalize or SCT
dims
  Dimensions of dimensional reduction to use for integration
k.filter
  Number of anchors to filter
scale.layer
  Name of scaled layer in Assay
dims.to.integrate
  Number of dimensions to return integrated values for
k.weight
  Number of neighbors to consider when weighting anchors
weight.reduction
  Dimension reduction to use when calculating anchor weights. This can be one of:
  • A string, specifying the name of a dimension reduction present in all objects
to be integrated
  • A vector of strings, specifying the name of a dimension reduction to use for
each object to be integrated
  • A vector of DimReduc objects, specifying the object to use for each object
    in the integration
  • NULL, in which case the full corrected space is used for computing anchor
    weights.
sd.weight
  Controls the bandwidth of the Gaussian kernel for weighting
sample.tree
  Specify the order of integration. Order of integration should be encoded in a ma-
trix, where each row represents one of the pairwise integration steps. Negative
numbers specify a dataset, positive numbers specify the integration results from
a given row (the format of the merge matrix included in the hclust function
output). For example: matrix(c(-2, 1, -3, -1), ncol = 2) gives:

```
  [,1] [,2]
[1,]  -2  -3
[2,]   1  -1
```

Which would cause dataset 2 and 3 to be integrated first, then the resulting object
integrated with dataset 1.
If NULL, the sample tree will be computed automatically.
preserve.order
  Do not reorder objects based on size for each pairwise integration.
verbose
  Print progress
...
  Arguments passed on to FindIntegrationAnchors

Examples
## Not run:
# Preprocessing
obj <- SeuratData::LoadData("pbmcscsa")
obj[["RNA"]]
  split(obj[["RNA"]], f = obj$Method)
obj <- NormalizeData(obj)
obj <- FindVariableFeatures(obj)
obj <- ScaleData(obj)
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,

obj <- RunPCA(obj)

# After preprocessing, we run integration
obj <- IntegrateLayers(object = obj, method = RPCAIntegration,
  orig.reduction = "pca", new.reduction = 'integrated.rpca',
  verbose = FALSE)

# Reference-based Integration
# Here, we use the first layer as a reference for integration
# Thus, we only identify anchors between the reference and the rest of the datasets,
# saving computational resources
obj <- IntegrateLayers(object = obj, method = RPCAIntegration,
  orig.reduction = "pca", new.reduction = 'integrated.rpca',
  reference = 1, verbose = FALSE)

# Modifying parameters
# We can also specify parameters such as `k.anchor` to increase the strength of
# integration
obj <- IntegrateLayers(object = obj, method = RPCAIntegration,
  orig.reduction = "pca", new.reduction = 'integrated.rpca',
  k.anchor = 20, verbose = FALSE)

# Integrating SCTransformed data
obj <- SCTransform(object = obj)
obj <- IntegrateLayers(object = obj, method = RPCAIntegration,
  orig.reduction = "pca", new.reduction = 'integrated.rpca',
  assay = "SCT", verbose = FALSE)

## End(Not run)
seed.use = 42,
verbose = FALSE,
...
)

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

Arguments

- **object1**: First Seurat object
- **object2**: Second Seurat object.
- **...**: Extra parameters (passed onto MergeSeurat in case with two objects passed, passed onto ScaleData in case with single object and rescale.groups set to TRUE)
- **standardize**: Standardize matrices - scales columns to have unit variance and mean 0
- **num.cc**: Number of canonical vectors to calculate
- **seed.use**: Random seed to set. If NULL, does not set a seed
- **verbose**: Show progress messages
- **assay1, assay2**: Assays to pull from in the first and second objects, respectively
- **features**: Set of genes to use in CCA. Default is the union of both the variable features sets present in both objects.
- **renormalize**: Renormalize raw data after merging the objects. If FALSE, merge the data matrices also.
- **rescale**: Rescale the datasets prior to CCA. If FALSE, uses existing data in the scale data slots.
- **compute.gene.loadings**: Also compute the gene loadings. NOTE - this will scale every gene in the dataset which may impose a high memory cost.
- **add.cell.id1, add.cell.id2**: Add ...
RunGraphLaplacian

Run Graph Laplacian Eigendecomposition

Description

Run a graph laplacian dimensionality reduction. It is used as a low dimensional representation for a cell-cell graph. The input graph should be symmetric.

Usage

RunGraphLaplacian(object, ...)

## S3 method for class 'Seurat'
RunGraphLaplacian(
  object,
  graph,
  reduction.name = "lap",
  reduction.key = "LAP_",
  n = 50,
  verbose = TRUE,
  ...
)

## Default S3 method:
RunGraphLaplacian(object, n = 50, reduction.key = "LAP_", verbose = TRUE, ...)

Value

Returns a combined Seurat object with the CCA results stored.

See Also

merge.Seurat

Examples

## Not run:
data("pbmc_small")
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- subset(pbmc_small, cells = colnames(pbmc_small)[1:40])
pbmc2 <- subset(pbmc_small, cells = colnames(pbmc_small)[41:80])
pbmc1["group"] <- "group1"
pbmc2["group"] <- "group2"
pbmc_cca <- RunCCA(object1 = pbmc1, object2 = pbmc2)
# Print results
print(x = pbmc_cca["cca"])
## End(Not run)
RunICA

Run Independent Component Analysis on gene expression

Description

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

Usage

RunICA(object, ...)

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

## S3 method for class 'Assay'
RunICA(
  object,
RunICA

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to be passed to fastica</td>
</tr>
<tr>
<td>assay</td>
<td>Name of Assay ICA is being run on</td>
</tr>
<tr>
<td>nics</td>
<td>Number of ICs to compute</td>
</tr>
<tr>
<td>rev.ica</td>
<td>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).</td>
</tr>
<tr>
<td>ica.function</td>
<td>ICA function from ica package to run (options: icafast, icaimax, icajade)</td>
</tr>
<tr>
<td>verbose</td>
<td>Print the top genes associated with high/low loadings for the ICs</td>
</tr>
<tr>
<td>ndims.print</td>
<td>ICs to print genes for</td>
</tr>
<tr>
<td>nfeatures.print</td>
<td>Number of genes to print for each IC</td>
</tr>
<tr>
<td>reduction.name</td>
<td>dimensional reduction name</td>
</tr>
</tbody>
</table>
RunLDA

**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

---

**RunLDA**  *Run Linear Discriminant Analysis*

**Description**

Run Linear Discriminant Analysis

Function to perform Linear Discriminant Analysis.

**Usage**

```r
RunLDA(object, ...)
```

## Default S3 method:

```r
RunLDA(
  object,
  labels,
  assay = NULL,
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.key = "LDA_",
  seed = 42,
  ...
)
```

## S3 method for class 'Assay'

```r
RunLDA(
  object,
  assay = NULL,
  labels,
  features = NULL,
  verbose = TRUE,
  ndims.print = 1:5,
  nfeatures.print = 30,
  reduction.key = "LDA_",
  seed = 42,
  ...
)
```

## S3 method for class 'Seurat'

```r
RunLDA(
```

```r
```
RunMarkVario

object, assay = NULL, labels, features = NULL, reduction.name = "lda", reduction.key = "LDA_", seed = 42, verbose = TRUE, ndims.print = 1:5, nfeatures.print = 30, ...

Arguments

object An object of class Seurat.

Arguments passed to other methods

... labels Meta data column with target gene class labels.

assay Assay to use for performing Linear Discriminant Analysis (LDA).

verbose Print the top genes associated with high/low loadings for the PCs

ndims.print Number of LDA dimensions to print.

nfeatures.print Number of features to print for each LDA component.

reduction.key Reduction key name.

seed Value for random seed

features Features to compute LDA on

reduction.name Dimensional reduction name, lda by default

RunMarkVario Run the mark variogram computation on a given position matrix and expression matrix.

Description

Wraps the functionality of markvario from the spatstat package.

Usage

RunMarkVario(spatial.location, data, ...)
Arguments

spatial.location
A 2 column matrix giving the spatial locations of each of the data points also in data
data
Matrix containing the data used as "marks" (e.g. gene expression)
...Arguments passed to markvario

Description

Function to identify perturbed and non-perturbed gRNA expressing cells that accounts for multiple treatments/conditions/chemical perturbations.

Usage

RunMixscape(
object,
assay = "PRTB",
slot = "scale.data",
lables = "gene",
nt.class.name = "NT",
new.class.name = "mixscape_class",
min.de.genes = 5,
min.cells = 5,
de.assay = "RNA",
logfc.threshold = 0.25,
iter.num = 10,
verbose = FALSE,
split.by = NULL,
fine.mode = FALSE,
fine.mode.labels = "guide.ID",
prtb.type = "KO"
)

Arguments

object
An object of class Seurat.
assay
Assay to use for mixscape classification.
slot
Assay data slot to use.
lables
metadata column with target gene labels.
nt.class.name
Classification name of non-targeting gRNA cells.
new.class.name
Name of mixscape classification to be stored in metadata.
RunMoransI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min.de.genes</td>
<td>Required number of genes that are differentially expressed for method to separate perturbed and non-perturbed cells.</td>
</tr>
<tr>
<td>min.cells</td>
<td>Minimum number of cells in target gene class. If fewer than this many cells are assigned to a target gene class during classification, all are assigned NP.</td>
</tr>
<tr>
<td>de.assay</td>
<td>Assay to use when performing differential expression analysis. Usually RNA.</td>
</tr>
<tr>
<td>logfc.threshold</td>
<td>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.</td>
</tr>
<tr>
<td>iter.num</td>
<td>Number of normalmixEM iterations to run if convergence does not occur.</td>
</tr>
<tr>
<td>verbose</td>
<td>Display messages</td>
</tr>
<tr>
<td>split.by</td>
<td>metadata column with experimental condition/cell type classification information. This is meant to be used to account for cases a perturbation is condition/cell type specific.</td>
</tr>
<tr>
<td>fine.mode</td>
<td>When this is equal to TRUE, DE genes for each target gene class will be calculated for each gRNA separately and pooled into one DE list for calculating the perturbation score of every cell and their subsequent classification.</td>
</tr>
<tr>
<td>fine.mode.labels</td>
<td>metadata column with gRNA ID labels.</td>
</tr>
<tr>
<td>prtb.type</td>
<td>specify type of CRISPR perturbation expected for labeling mixscape classifications. Default is KO.</td>
</tr>
</tbody>
</table>

Value

Returns Seurat object with with the following information in the meta data and tools slots:

- **mixscape_class** Classification result with cells being either classified as perturbed (KO, by default) or non-perturbed (NP) based on their target gene class.
- **mixscape_class.global** Global classification result (perturbed, NP or NT)
- **p_ko** Posterior probabilities used to determine if a cell is KO (default). Name of this item will change to match prtb.type parameter setting. (>0.5) or NP
- **perturbation score** Perturbation scores for every cell calculated in the first iteration of the function.

Description

Wraps the functionality of the Moran.I function from the ape package. Weights are computed as 1/distance.

Usage

RunMoransI(data, pos, verbose = TRUE)
RunPCA

Run Principal Component Analysis

Arguments

data Expression matrix
pos Position matrix
verbose Display messages/progress

Description

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

Usage

RunPCA(object, ...)

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

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

...)

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

Arguments

object  An object

...  Arguments passed to other methods and IRLBA

assay  Name of Assay PCA is being run on

npcs  Total Number of PCs to compute and store (50 by default)

rev.pca  By default computes the PCA on the cell x gene matrix. Setting to true will compute it on gene x cell matrix.

weight.by.var  Weight the cell embeddings by the variance of each PC (weights the gene loadings if rev.pca is TRUE)

verbose  Print the top genes associated with high/low loadings for the PCs

ndims.print  PCs to print genes for

nfeatures.print  Number of genes to print for each PC

reduction.key  dimensional reduction key, specifies the string before the number for the dimension names. PC by default

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

approx  Use truncated singular value decomposition to approximate PCA

features  Features to compute PCA on. If features=NULL, PCA will be run using the variable features for the Assay. Note that the features must be present in the scaled data. Any requested features that are not scaled or have 0 variance will be dropped, and the PCA will be run using the remaining features.

reduction.name  dimensional reduction name, pca by default
Value

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

Description

Run a supervised LSI (SLSI) dimensionality reduction supervised by a cell-cell kernel. SLSI is used to capture a linear transformation of peaks that maximizes its dependency to the given cell-cell kernel.

Usage

RunSLSI(object, ...)

## Default S3 method:
RunSLSI(
  object,
  assay = NULL,
  n = 50,
  reduction.key = "SLSI_",
  graph = NULL,
  verbose = TRUE,
  seed.use = 42,
  ...
)

## S3 method for class 'Assay'
RunSLSI(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "SLSI_",
  graph = NULL,
  verbose = TRUE,
  seed.use = 42,
  ...
)

## S3 method for class 'Seurat'
RunSLSI(
  object,
  assay = NULL,
  features = NULL,
n = 50,
reduction.name = "slsi",
reduction.key = "SLSI_",
graph = NULL,
verbose = TRUE,
seed.use = 42,
...
)

Arguments

object An object
... Arguments passed to IRLBA irlba
assay Name of Assay SLSI is being run on
n Total Number of SLSI components to compute and store
reduction.key dimensional reduction key, specifies the string before the number for the dimension names
graph Graph used supervised by SLSI
verbose Display messages
seed.use Set a random seed. Setting NULL will not set a seed.
features Features to compute SLSI on. If NULL, SLSI will be run using the variable features for the Assay.
reduction.name dimensional reduction name

Value

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

RunSPCA

Run Supervised Principal Component Analysis

Description

Run a supervised PCA (SPCA) dimensionality reduction supervised by a cell-cell kernel. SPCA is used to capture a linear transformation which maximizes its dependency to the given cell-cell kernel. We use SNN graph as the kernel to supervise the linear matrix factorization.

Usage

RunSPCA(object, ...)

## Default S3 method:
RunSPCA(
  object,
  assay = NULL,
...
RunSPCA

```r
npcs = 50,
reduction.key = "SPC_",
graph = NULL,
verbose = FALSE,
seed.use = 42,
...
)

## S3 method for class 'Assay'
RunSPCA(
  object,
  assay = NULL,
  features = NULL,
 npcs = 50,
  reduction.key = "SPC_",
  graph = NULL,
  verbose = TRUE,
  seed.use = 42,
  ...
)

## S3 method for class 'Assay5'
RunSPCA(
  object,
  assay = NULL,
  features = NULL,
 npcs = 50,
  reduction.key = "SPC_",
  graph = NULL,
  verbose = TRUE,
  seed.use = 42,
  layer = "scale.data",
  ...
)

## S3 method for class 'Seurat'
RunSPCA(
  object,
  assay = NULL,
  features = NULL,
 npcs = 50,
  reduction.name = "spca",
  reduction.key = "SPC_",
  graph = NULL,
  verbose = TRUE,
  seed.use = 42,
  ...
)
```
RunTSNE

Arguments

object An object
... Arguments passed to other methods and IRLBA
assay Name of Assay SPCA is being run on
npcs Total Number of SPCs to compute and store (50 by default)
reduction.key dimensional reduction key, specifies the string before the number for the dimension names. SPC by default
graph Graph used supervised by SPCA
verbose Print the top genes associated with high/low loadings for the SPCs
seed.use Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed.
features Features to compute SPCA on. If features=NULL, SPCA will be run using the variable features for the Assay.
layer Layer to run SPCA on
reduction.name dimensional reduction name, spca by default

Value

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

References


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",
  dim.embed = 2,
  reduction.key = "tSNE_",
  ...
)

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

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

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

**Arguments**

- object: Seurat object
Arguments passed to other methods and to t-SNE call (most commonly used is perplexity)

assay                Name of assay that t-SNE is being run on
seed.use             Random seed for the t-SNE. If NULL, does not set the seed
tsne.method          Select the method to use to compute the tSNE. Available methods are:
                      • “Rtsne”: Use the Rtsne package Barnes-Hut implementation of tSNE (default)
                      • “FIT-SNE”: Use the FFT-accelerated Interpolation-based t-SNE. Based on Kluger Lab code found here: https://github.com/KlugerLab/FIT-SNE
dim.embed            The dimensional space of the resulting tSNE embedding (default is 2). For example, set to 3 for a 3d tSNE
reduction.key        dimensional reduction key, specifies the string before the number for the dimension names. “tSNE_” by default
cells                Which cells to analyze (default, all cells)
dims                 Which dimensions to use as input features
reduction            Which dimensional reduction (e.g. PCA, ICA) to use for the tSNE. Default is PCA
features             If set, run the tSNE on this subset of features (instead of running on a set of reduced dimensions). Not set (NULL) by default; dims must be NULL to run on features
distance.matrix      If set, runs tSNE on the given distance matrix instead of data matrix (experimental)
reduction.name       dimensional reduction name, specifies the position in the object$dr list. tsne by default

---

**RunUMAP**

**Run UMAP**

---

**Description**

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

**Usage**

RunUMAP(object, ...)

## Default S3 method:
RunUMAP(
RunUMAP

object,
reduction.key = "UMAP_",
assay = NULL,
reduction.model = NULL,
return.model = FALSE,
.umap.method = "uwot",
n.neighbors = 30L,
n.components = 2L,
metric = "cosine",
n.epochs = NULL,
learning.rate = 1,
min.dist = 0.3,
spread = 1,
set.op.mix.ratio = 1,
local.connectivity = 1L,
repulsion.strength = 1,
negative.sample.rate = 5,
a = NULL,
b = NULL,
uwot.sgd = FALSE,
seed.use = 42,
metric.kwds = NULL,
angular.rp.forest = FALSE,
densmap = FALSE,
dens.lambda = 2,
dens.frac = 0.3,
dens.var.shift = 0.1,
verbose = TRUE,
...

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

metric.kwds = NULL,
        densmap = FALSE,
        densmap.kwds = NULL,
        verbose = TRUE,
        reduction.key = "UMAP_",
        ...)

## S3 method for class 'Neighbor'
RunUMAP(object, reduction.model, ...)

## S3 method for class 'Seurat'
RunUMAP(
        object,
        dims = NULL,
        reduction = "pca",
        features = NULL,
        graph = NULL,
        assay = DefaultAssay(object = object),
        nn.name = NULL,
        slot = "data",
        umap.method = "uwot",
        reduction.model = NULL,
        return.model = FALSE,
        n.neighbors = 30L,
        n.components = 2L,
        metric = "cosine",
        n.epochs = NULL,
        learning.rate = 1,
        min.dist = 0.3,
        spread = 1,
        set.op.mix.ratio = 1,
        local.connectivity = 1L,
        repulsion.strength = 1,
        negative.sample.rate = 5L,
        a = NULL,
        b = NULL,
        uwot.sgd = FALSE,
        seed.use = 42L,
        metric.kwds = NULL,
        angular rp.forest = FALSE,
        densmap = FALSE,
        dens.lambda = 2,
        dens.frac = 0.3,
        dens.var.shift = 0.1,
        verbose = TRUE,
        reduction.name = "umap",
        reduction.key = NULL,
Arguments

object: An object

Arguments passed to other methods and UMAP

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

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

reduction.model: DimReduc object that contains the umap model

return.model: Whether UMAP will return the uwot model

umap.method: UMAP implementation to run. Can be

uwot: Runs umap via the uwot R package
uwot-learn: Runs umap via the uwot R package and return the learned umap model
umap-learn: Run the Seurat wrapper of the python umap-learn package

n.neighbors: This determines the number of neighboring points used in local approximations of manifold structure. Larger values will result in more global structure being preserved at the loss of detailed local structure. In general, this parameter should often be in the range 5 to 50.

n.components: The dimension of the space to embed into.

metric: 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 JIT-d 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 optimize 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.
RunUMAP

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

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

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

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

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

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

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

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

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

densmap
Whether to use the density-augmented objective of densMAP. Turning on this option generates an embedding where the local densities are encouraged to be correlated with those in the original space. Parameters below with the prefix ‘dens’ further control the behavior of this extension. Default is FALSE. Only compatible with 'umap-learn' method and version of umap-learn >= 0.5.0

dens.lambda
Specific parameter which controls the regularization weight of the density correlation term in densMAP. Higher values prioritize density preservation over the UMAP objective, and vice versa for values closer to zero. Setting this parameter to zero is equivalent to running the original UMAP algorithm. Default value is 2.

dens.frac
Specific parameter which controls the fraction of epochs (between 0 and 1) where the density-augmented objective is used in densMAP. The first (1 - dens_frac) fraction of epochs optimize the original UMAP objective before introducing the density correlation term. Default is 0.3.

dens.var.shift
Specific parameter which specifies a small constant added to the variance of local radii in the embedding when calculating the density correlation objective to prevent numerical instability from dividing by a small number. Default is 0.1.
SampleUMI

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)
SaveAnnoyIndex

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

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

SaveAnnoyIndex

Save the Annoy index

Description

Save the Annoy index

Usage

SaveAnnoyIndex(object, file)

Arguments

object A Neighbor object with the annoy index stored
file Path to file to write index to
**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,
  split.by = NULL,
  model.use = "linear",
  use.umi = FALSE,
  do.scale = TRUE,
  do.center = TRUE,
  scale.max = 10,
  block.size = 1000,
  min.cells.to.block = 3000,
  verbose = TRUE,
  ...
)
```

### S3 method for class 'IterableMatrix'

```r
ScaleData(
  object,
  features = NULL,
  do.scale = TRUE,
  do.center = TRUE,
  scale.max = 10,
  ...
)
```

### S3 method for class 'Assay'

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

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

Arguments

object       An object
...          Arguments passed to other methods
features     Vector of features names to scale/center. Default is variable features.
vars.to.regress Variables to regress out (previously latent.vars in RegressOut). For example, nUMI, or percent.mito.
latent.data  Extra data to regress out, should be cells x latent data
split.by     Name of variable in object metadata or a vector or factor defining grouping of cells. See argument f in split for more details
model.use    Use a linear model or generalized linear model (poisson, negative binomial) for the regression. Options are 'linear' (default), 'poisson', and 'negbinom'
use.umi      Regress on UMI count data. Default is FALSE for linear modeling, but automatically set to TRUE if model.use is 'negbinom' or 'poisson'
do.scale     Whether to scale the data.
do.center    Whether to center the data.
scale.max  Max value to return for scaled data. The default is 10. Setting this can help reduce the effects of 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.

ScaleFactors  Get image scale factors

Description

Get image scale factors

Usage

ScaleFactors(object, ...)
scalefactors(spot, fiducial, hires, lowres)

## S3 method for class 'VisiumV1'
ScaleFactors(object, ...)

## S3 method for class 'VisiumV1'
ScaleFactors(object, ...)

ScoreJackStraw

Arguments

object An object to get scale factors from
...
Arguments passed to other methods
spot Spot full resolution scale factor
fiducial Fiducial full resolution scale factor
hires High resolution scale factor
lowres Low resolution scale factor

Value

An object of class scalefactors

Note

scalefactors objects can be created with scalefactors()

ScoreJackStraw  Compute Jackstraw scores significance.

Description

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

Usage

ScoreJackStraw(object, ...)

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

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

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

object  An object
... Arguments passed to other methods
dims Which dimensions to examine
score.thresh Threshold to use for the proportion test of PC significance (see Details)
reduction Reduction associated with JackStraw to score
do.plot Show plot. To return ggplot object, use JackStrawPlot after running Score-JackStraw.

Value

Returns a Seurat object

Author(s)

Omri Wurtzel

See Also

JackStrawPlot
JackStrawPlot

SCTAssay-class

The SCTModel Class

Description

The SCTModel object is a model and parameters storage from SCTransform. It can be used to calculate Pearson residuals for new genes.

The SCTAssay object contains all the information found in an Assay object, with extra information from the results of SCTransform

Usage

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

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

Arguments

x An SCTAssay object
value New levels, must be in the same order as the levels present
SCTAssay-class

Value

levels: SCT model names
levels<-: x with updated SCT model names

Slots

feature.attributes A data.frame with feature attributes in SCTransform

clips A list of two numeric of length two specifying the min and max values the Pearson residual will be clipped to. One for vst and one for SCTransform

umi.assay Name of the assay of the seurat object containing UMI matrix and the default is RNA

median_umi Median UMI (or scale factor) used to calculate corrected counts

SCTModel.list A list containing SCT models

Get and set SCT model names

SCT results are named by initial run of `SCTransform` in order to keep SCT parameters straight between runs. When working with merged SCTAssay objects, these model names are important. `levels<-` allows querying the models present. `levels<-` allows the changing of the names of the models present, useful when merging SCTAssay objects. Note: unlike normal `levels<-`, `levels<-.SCTAssay` allows complete changing of model names, not reordering.

Creating an SCTAssay from an Assay

Conversion from an Assay object to an SCTAssay object by is done by adding the additional slots to the object. If `from` has results generated by `SCTransform` from Seurat v3.0.0 to v3.1.1, the conversion will automagically fill the new slots with the data

See Also

Assay

Examples

## Not run:
# SCTAssay objects are generated from SCTransform
pbmc_small <- SCTransform(pbmc_small)

## End(Not run)

## Not run:
# SCTAssay objects are generated from SCTransform
pbmc_small <- SCTransform(pbmc_small)

pbmc_small["SCT"]
### SCTransform

**Perform sctransform-based normalization**

#### Description

This function calls sctransform::vst. The sctransform package is available at https://github.com/satijalab/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

```r
SCTransform(object, ...)
```

**## Default S3 method:**

```r
SCTransform(
  object,
  cell.attr,
  reference.SCT.model = NULL,
  do.correct.umi = TRUE,
  ncells = 5000,
  residual.features = NULL,
  variable.features.n = 3000,
  variable.features.rv.th = 1.3,
  vars.to.regress = NULL,
  latent.data = NULL,
  do.scale = FALSE,
  do.center = TRUE,
  clip.range = c(-sqrt(x = ncol(x = umi)/30), sqrt(x = ncol(x = umi)/30)),
  vst.flavor = "v2",
  conserve.memory = FALSE,
  return.only.var.genes = TRUE,
  seed.use = 1448145,
  verbose = TRUE,
  ...
)```

```
SCTransform

## S3 method for class 'Assay'
SCTransform(
    object,
    cell.attr,
    reference.SCT.model = NULL,
    do.correct.umi = TRUE,
    ncells = 5000,
    residual.features = NULL,
    variable.features.n = 3000,
    variable.features.rv.th = 1.3,
    vars.to.regress = NULL,
    latent.data = NULL,
    do.scale = FALSE,
    do.center = TRUE,
    clip.range = c(-sqrt(x = ncol(x = object)/30), sqrt(x = ncol(x = object)/30)),
    vst.flavor = "v2",
    conserve.memory = FALSE,
    return.only.var.genes = TRUE,
    seed.use = 1448145,
    verbose = TRUE,
    ...
)

## S3 method for class 'Seurat'
SCTransform(
    object,
    assay = "RNA",
    new.assay.name = "SCT",
    reference.SCT.model = NULL,
    do.correct.umi = TRUE,
    ncells = 5000,
    residual.features = NULL,
    variable.features.n = 3000,
    variable.features.rv.th = 1.3,
    vars.to.regress = NULL,
    do.scale = FALSE,
    do.center = TRUE,
    clip.range = c(-sqrt(x = ncol(x = object[[assay]])/30), sqrt(x = ncol(x =
        object[[assay]])/30)),
    vst.flavor = "v2",
    conserve.memory = FALSE,
    return.only.var.genes = TRUE,
    seed.use = 1448145,
    verbose = TRUE,
    ...
)
## S3 method for class 'IterableMatrix'

SCTransform(
  object,
  cell.attr,
  reference.SCT.model = NULL,
  do.correct.umi = TRUE,
  ncells = 5000,
  residual.features = NULL,
  variable.features.n = 3000,
  variable.features.rv.th = 1.3,
  vars.to.regress = NULL,
  latent.data = NULL,
  do.scale = FALSE,
  do.center = TRUE,
  clip.range = \((-\sqrt{x = ncol(x = object)/30}, \sqrt{x = ncol(x = object)/30})\),
  vst.flavor = "v2",
  conserve.memory = FALSE,
  return.only.var.genes = TRUE,
  seed.use = 1448145,
  verbose = TRUE,
  ...
)

### Arguments

- **object**: UMI counts matrix
- **...**: Additional parameters passed to sctransform::vst
- **cell.attr**: A metadata with cell attributes
- **reference.SCT.model**: If not NULL, compute residuals for the object using the provided SCT model; supports only log_umi as the latent variable. If residual.features are not specified, compute for the top variable.features.n specified in the model which are also present in the object. If residual.features are specified, the variable features of the resulting SCT assay are set to the top variable.features.n in the model.
- **do.correct.umi**: Place corrected UMI matrix in assay counts slot; default is TRUE
- **ncells**: Number of subsampling cells used to build NB regression; default is 5000
- **residual.features**: Genes to calculate residual features for; default is NULL (all genes). If specified, will be set to VariableFeatures of the returned object.
- **variable.features.n**: Use this many features as variable features after ranking by residual variance; default is 3000. Only applied if residual.features is not set.
- **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. Only applied if residual.features is not set.
vars.to.regress  Variables to regres out in a second non-regularized linear
latent.data Extra data to regress out, should be cells x latent data regression. For example, percent.mito. Default is NULL
do.scale Whether to scale residuals to have unit variance; default is FALSE
do.center Whether to center residuals to have mean zero; default is TRUE
clip.range Range to clip the residuals to; default is c(-sqrt(n/30), sqrt(n/30)), where n is the number of cells
vst.flavor When set to 'v2' sets method = glmGamPoi_offset, n.cells=2000, and exclude_poisson = TRUE which causes the model to learn theta and intercept only besides excluding poisson genes from learning and regularization
conserve.memory If set to TRUE the residual matrix for all genes is never created in full; useful for large data sets, but will take longer to run; this will also set return.only.var.genes to TRUE; default is FALSE
return.only.var.genes If set to TRUE the scale.data matrices in output assay are subset to contain only the variable genes; default is TRUE
seed.use Set a random seed. By default, sets the seed to 1448145. Setting NULL will not set a seed.
verbose Whether to print messages and progress bars
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; default is 'SCT'

Value

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

See Also

correct_counts get_residuals

---

SCTResults Get SCT results from an Assay

Description

Pull the SCTResults information from an SCTAssay object.
Usage

SCTResults(object, ...)
SCTResults(object, ...) <- value

## S3 method for class 'SCTModel'
SCTResults(object, slot, ...)

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

## S3 method for class 'SCTAssay'
SCTResults(object, slot, model = NULL, ...)

## S3 replacement method for class 'SCTAssay'
SCTResults(object, slot, model = NULL, ...) <- value

## S3 method for class 'Seurat'
SCTResults(object, assay = "SCT", slot, model = NULL, ...)

Arguments

object An object
... Arguments passed to other methods (not used)
value new data to set
slot Which slot to pull the SCT results from
model Name of SCModel to pull result from. Available names can be retrieved with levels.
assay Assay in the Seurat object to pull from

Value

Returns the value present in the requested slot for the requested group. If group is not specified, returns a list of slot results for each group unless there is only one group present (in which case it just returns the slot directly).

SelectIntegrationFeatures

Select integration features

Description

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

Usage

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

Arguments

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

Details

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

Value

A vector of selected features

Examples

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

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

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

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

## End(Not run)
SelectIntegrationFeatures5

Select integration features

Description

Select integration features

Usage

SelectIntegrationFeatures5(
    object,
    nfeatures = 2000,
    assay = NULL,
    method = NULL,
    layers = NULL,
    verbose = TRUE,
    ...
)

Arguments

object       Seurat object
nfeatures     Number of features to return for integration
assay        Name of assay to use for integration feature selection
method       Which method to pull. For HVFInfo and VariableFeatures, choose one from
             one of the following:
             • “vst”
             • “sctransform” or “sct”
             • “mean.var.plot”, “dispersion”, “mvp”, or “disp”
layers       Name of layers to use for integration feature selection
verbose      Print messages
...           Arguments passed on to method
SetIntegrationData

Usage

`SelectSCTIntegrationFeatures(
  object,
  nfeatures = 3000,
  assay = NULL,
  verbose = TRUE,
  ...
)
``

Arguments

- **object**: Seurat object
- **nfeatures**: Number of features to return for integration
- **assay**: Name of assay to use for integration feature selection
- **verbose**: Print messages
- **...**: Arguments passed on to method

---

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
SetQuantile  
*Find the Quantile of Data*

**Description**

Converts a quantile in character form to a number regarding some data. String form for a quantile is represented as a number prefixed with “q”; for example, 10th quantile is “q10” while 2nd quantile is “q2”. Will only take a quantile of non-zero data values.

**Usage**

```
SetQuantile(cutoff, data)
```

**Arguments**

- **cutoff**  
The cutoff to turn into a quantile
- **data**  
The data to turn find the quantile of

**Value**

The numerical representation of the quantile

**Examples**

```
set.seed(42)
SetQuantile('q10', sample(1:100, 10))
```

---

**Seurat-class  
The Seurat Class**

**Description**

The Seurat object is a representation of single-cell expression data for R; for more details, please see the documentation in `SeuratObject`

**See Also**

`SeuratObject::Seurat-class`
The SeuratCommand Class

Description

For more details, please see the documentation in SeuratObject

See Also

SeuratObject::SeuratCommand-class

SeuratTheme

Seurat Themes

Description

Various themes to be applied to ggplot2-based plots

SeuratTheme  The curated Seurat theme, consists of ...
DarkTheme   A dark theme, axes and text turn to white, the background becomes black
NoAxes      Removes axis lines, text, and ticks
NoLegend    Removes the legend
FontSize    Sets axis and title font sizes
NoGrid      Removes grid lines
SeuratAxes  Set Seurat-style axes
SpatialTheme A theme designed for spatial visualizations (eg PolyFeaturePlot, PolyDimPlot)
RestoreLegend Restore a legend after removal
RotatedAxis Rotate X axis text 45 degrees
BoldTitle   Enlarges and emphasizes the title

Usage

SeuratTheme()

CenterTitle(...)

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')
SketchData

# 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()

Description

This function uses sketching methods to downsample high-dimensional single-cell RNA expression data, which can help with scalability for large datasets.

Usage

SketchData(
  object,
  assay = NULL,
  ncells = 5000L,
  sketched.assay = "sketch",
  method = c("LeverageScore", "Uniform"),
  var.name = "leverage.score",
  over.write = FALSE,
  seed = 123L,
  cast = "dgCMatrix",
  verbose = TRUE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object.</td>
</tr>
<tr>
<td>assay</td>
<td>Assay name. Default is NULL, in which case the default assay of the object is used.</td>
</tr>
</tbody>
</table>
ncells  A positive integer indicating the number of cells to sample for the sketching. Default is 5000.
sketched.assay  Sketched assay name. A sketch assay is created or overwrite with the sketch data. Default is 'sketch'.
method  Sketching method to use. Can be 'LeverageScore' or 'Uniform'. Default is 'LeverageScore'.
var.name  A metadata column name to store the leverage scores. Default is 'leverage.score'.
over.write  whether to overwrite existing column in the metadata. Default is FALSE.
seed  A positive integer for the seed of the random number generator. Default is 123.
cast  The type to cast the resulting assay to. Default is 'dgCMatrix'.
verbose  Print progress and diagnostic messages
...  Arguments passed to other methods

Value
A Seurat object with the sketched data added as a new assay.

---

SlideSeq-class  The SlideSeq class

Description
The SlideSeq class represents spatial information from the Slide-seq platform

Slots
coordinates  ...

Slots
assay  Name of assay to associate image data with; will give this image priority for visualization when the assay is set as the active/default assay in a Seurat object
key  A one-length character vector with the object’s key; keys must be one or more alphanumeric characters followed by an underscore “_” (regex pattern “^[a-zA-Z][a-zA-Z0-9]*$”)

---

SpatialImage-class  The SpatialImage Class

Description
For more details, please see the documentation in SeuratObject

See Also
SeuratObject::SpatialImage-class
SpatialPlot

Visualize spatial clustering and expression data.

Description

SpatialPlot plots a feature or discrete grouping (e.g., cluster assignments) as spots over the image that was collected. We also provide SpatialFeaturePlot and SpatialDimPlot as wrapper functions around SpatialPlot for a consistent naming framework.

Usage

SpatialPlot(
  object,
  group.by = NULL,
  features = NULL,
  images = NULL,
  cols = NULL,
  image.alpha = 1,
  crop = TRUE,
  slot = "data",
  keep.scale = "feature",
  min.cutoff = NA,
  max.cutoff = NA,
  cells.highlight = NULL,
  cols.highlight = c("#DE2D26", "grey50"),
  facet.highlight = FALSE,
  label = FALSE,
  label.size = 5,
  label.color = "white",
  label.box = TRUE,
  repel = FALSE,
  ncol = NULL,
  combine = TRUE,
  pt.size.factor = 1.6,
  alpha = c(1, 1),
  stroke = 0.25,
  interactive = FALSE,
  do.identify = FALSE,
  identify.ident = NULL,
  do.hover = FALSE,
  information = NULL
)

SpatialDimPlot(
  object,
  group.by = NULL,
  images = NULL,
SpatialPlot

cols = NULL,
crop = TRUE,
cells.highlight = NULL,
cols.highlight = c("#DE2D26", "grey50"),
facet.highlight = FALSE,
label = FALSE,
label.size = 7,
label.color = "white",
repel = FALSE,
ncol = NULL,
combine = TRUE,
pt.size.factor = 1.6,
alpha = c(1, 1),
image.alpha = 1,
stroke = 0.25,
label.box = TRUE,
interactive = FALSE,
information = NULL
)

SpatialFeaturePlot(
  object,
  features,
  images = NULL,
crop = TRUE,
slot = "data",
keep.scale = "feature",
min.cutoff = NA,
max.cutoff = NA,
ncol = NULL,
combine = TRUE,
pt.size.factor = 1.6,
alpha = c(1, 1),
image.alpha = 1,
stroke = 0.25,
interactive = FALSE,
information = NULL
)

Arguments

object A Seurat object
group.by Name of meta.data column to group the data by
features Name of the feature to visualize. Provide either group.by OR features, not both.
images Name of the images to use in the plot(s)
cols Vector of colors, each color corresponds to an identity class. This may also be a single character or numeric value corresponding to a palette as specified by brewer.pal.info. By default, ggplot2 assigns colors
image.alpha
Crop the plot in to focus on points plotted. Set to FALSE to show entire background image.

slot
If plotting a feature, which data slot to pull from (counts, data, or scale.data)

keep.scale
How to handle the color scale across multiple plots. Options are:

- “feature” (default; by row/feature scaling): The plots for each individual feature are scaled to the maximum expression of the feature across the conditions provided to split.by
- “all” (universal scaling): The plots for all features and conditions are scaled to the maximum expression value for the feature with the highest overall expression
- NULL (no scaling): Each individual plot is scaled to the maximum expression value of the feature in the condition provided to split.by; be aware setting NULL will result in color scales that are not comparable between plots

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’)

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

cols.highlight
A vector of colors to highlight the cells as; ordered the same as the groups in cells.highlight; last color corresponds to unselected cells.

facet.highlight
When highlighting certain groups of cells, split each group into its own plot

label
Whether to label the clusters

label.size
Sets the size of the labels

label.color
Sets the color of the label text

label.box
Whether to put a box around the label text (geom_text vs geom_label)

repel
Repels the labels to prevent overlap

ncol
Number of columns if plotting multiple plots

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

pt.size.factor
Scale the size of the spots.

alpha
Controls opacity of spots. Provide as a vector specifying the min and max for SpatialFeaturePlot. For SpatialDimPlot, provide a single alpha value for each plot.

stroke
Control the width of the border around the spots

interactive
Launch an interactive SpatialDimPlot or SpatialFeaturePlot session, see ISpatialDimPlot or ISpatialFeaturePlot for more details

do.identify, do.hover
DEPRECATED in favor of interactive

identify.ident
DEPRECATED

information
An optional dataframe or matrix of extra information to be displayed on hover
Value

If `do.identify`, either a vector of cells selected or the object with selected cells set to the value of `identify.ident` (if set). Else, if `do.hover`, a plotly object with interactive graphics. Else, a ggplot object.

Examples

```r
## Not run:
# For functionality analagous to FeaturePlot
SpatialPlot(seurat.object, features = "MS4A1")
SpatialFeaturePlot(seurat.object, features = "MS4A1")

# For functionality analagous to DimPlot
SpatialPlot(seurat.object, group.by = "clusters")
SpatialDimPlot(seurat.object, group.by = "clusters")

## End(Not run)
```

---

**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

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

Arguments

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

Value

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

Slots

assay  Name of assay to associate image data with; will give this image priority for visualization when the assay is set as the active/default assay in a Seurat object

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

subset.AnchorSet  Subset an AnchorSet object

Description

Subset an AnchorSet object

Usage

```r
## S3 method for class 'AnchorSet'
subset(
  x,
  score.threshold = NULL,
  disallowed.dataset.pairs = NULL,
  dataset.matrix = NULL,
  group.by = NULL,
  disallowed.ident.pairs = NULL,
  ident.matrix = NULL,
  ...
)
```
SubsetByBarcodeInflections

**Arguments**

- **x**: object to be subsettred.
- **score.threshold**: Only anchor pairs with scores greater than this value are retained.
- **disallowed.dataset.pairs**: Remove any anchors formed between the provided pairs. E.g. `list(c(1, 5), c(1, 2))` filters out any anchors between datasets 1 and 5 and datasets 1 and 2.
- **dataset.matrix**: Provide a binary matrix specifying whether a dataset pair is allowable (1) or not (0). Should be a dataset x dataset matrix.
- **group.by**: Grouping variable to determine allowable ident pairs.
- **disallowed.ident.pairs**: Remove any anchors formed between provided ident pairs. E.g. `list(c("CD4", "CD8"), c("B-cell", "T-cell"))`
- **ident.matrix**: Provide a binary matrix specifying whether an ident pair is allowable (1) or not (0). Should be an ident x ident symmetric matrix.
- **...**: further arguments to be passed to or from other methods.

**Value**

Returns an AnchorSet object with specified anchors filtered out.

---

**SubsetByBarcodeInflections**

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

**Description**

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

**Usage**

`SubsetByBarcodeInflections(object)`

**Arguments**

- **object**: Seurat object

**Details**

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

**Value**

Returns a subsettred Seurat object.
TopCells

Author(s)
Robert A. Amezquita, <robert.amezquita@fredhutch.org>

See Also
CalculateBarcodeInflections BarcodeInflectionsPlot

Examples

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

Description

Find cells with highest scores for a given dimensional reduction technique

Usage

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

Arguments

- `object`: DimReduce 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

```
data("pbmc_small")
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>DimRed object</td>
</tr>
<tr>
<td>dim</td>
<td>Dimension to use</td>
</tr>
<tr>
<td>nfeatures</td>
<td>Number of features to return</td>
</tr>
<tr>
<td>projected</td>
<td>Use the projected feature loadings</td>
</tr>
<tr>
<td>balanced</td>
<td>Return an equal number of features with both + and - scores.</td>
</tr>
<tr>
<td>...</td>
<td>Extra parameters passed to Loadings</td>
</tr>
</tbody>
</table>

Value

Returns a vector of features

Examples

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

*Get nearest neighbors for given cell*

**Description**

Return a vector of cell names of the nearest n cells.

**Usage**

TopNeighbors(object, cell, n = 5)

**Arguments**

- **object**: Neighbor object
- **cell**: Cell of interest
- **n**: Number of neighbors to return

**Value**

Returns a vector of cell names

---

**TransferAnchorSet-class**

*The TransferAnchorSet Class*

**Description**

Inherits from the Anchorset class. Implemented mainly for method dispatch purposes. See AnchorSet for slot details.

---

**TransferData**

*Transfer data*

**Description**

Transfer categorical or continuous data across single-cell datasets. For transferring categorical information, pass a vector from the reference dataset (e.g. refdata = reference$celltype). For transferring continuous information, pass a matrix from the reference dataset (e.g. refdata = GetAssayData(reference[['RNA']])).
Usage

`TransferData(
  anchorset,
  refdata,
  reference = NULL,
  query = NULL,
  query.assay = NULL,
  weight.reduction = "pcaproject",
  l2.norm = FALSE,
  dims = NULL,
  k.weight = 50,
  sd.weight = 1,
  eps = 0,
  n.trees = 50,
  verbose = TRUE,
  slot = "data",
  prediction.assay = FALSE,
  only.weights = FALSE,
  store.weights = TRUE
)

Arguments

anchorset               An AnchorSet object generated by `FindTransferAnchors`
refdata                 Data to transfer. This can be specified in one of two ways:
                         • The reference data itself as either a vector where the names correspond to
                           the reference cells, or a matrix, where the column names correspond to the
                           reference cells.
                         • The name of the metadata field or assay from the reference object provided.
                           This requires the reference parameter to be specified. If pulling assay data
                           in this manner, it will pull the data from the data slot. To transfer data from
                           other slots, please pull the data explicitly with `GetAssayData` and provide
                           that matrix here.
reference               Reference object from which to pull data to transfer
query                   Query object into which the data will be transferred.
query.assay             Name of the Assay to use from query
weight.reduction        Dimensional reduction to use for the weighting anchors. Options are:
                         • pcaproject: Use the projected PCA used for anchor building
                         • lsiproject: Use the projected LSI 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
TransferData

- **dms**: Set of dimensions to use in the anchor weighting procedure. If NULL, the same dimensions that were used to find anchors will be used for weighting.

- **k.weight**: Number of neighbors to consider when weighting anchors

- **sd.weight**: Controls the bandwidth of the Gaussian kernel for weighting

- **eps**: Error bound on the neighbor finding algorithm (from RANN)

- **n.trees**: More trees gives higher precision when using annoy approximate nearest neighbor search

- **verbose**: Print progress bars and output

- **slot**: Slot to store the imputed data. Must be either "data" (default) or "counts"

- **prediction.assay**: Return an Assay object with the prediction scores for each class stored in the data slot.

- **only.weights**: Only return weights matrix

- **store.weights**: Optionally store the weights matrix used for predictions in the returned query object.

**Details**

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

For both transferring discrete labels and also feature imputation, we first compute the weights matrix.

- Construct a weights matrix that defines the association between each query cell and each anchor. These weights are computed as 1 - the distance between the query cell and the anchor divided by the distance of the query cell to the k.weightth anchor multiplied by the anchor score computed in FindIntegrationAnchors. We then apply a Gaussian kernel width a bandwidth defined by sd.weight and normalize across all k.weight anchors.

The main difference between label transfer (classification) and feature imputation is what gets multiplied by the weights matrix. For label transfer, we perform the following steps:

- Create a binary classification matrix, the rows corresponding to each possible class and the columns corresponding to the anchors. If the reference cell in the anchor pair is a member of a certain class, that matrix entry is filled with a 1, otherwise 0.

- Multiply this classification matrix by the transpose of weights matrix to compute a prediction score for each class for each cell in the query dataset.

For feature imputation, we perform the following step:

- Multiply the expression matrix for the reference anchor cells by the weights matrix. This returns a predicted expression matrix for the specified features for each cell in the query dataset.
TransferSketchLabels

**Value**

If query is not provided, for the categorical data in refdata, returns a data.frame with label predictions. If refdata is a matrix, returns an Assay object where the imputed data has been stored in the provided slot.

If query is provided, a modified query object is returned. For the categorical data in refdata, prediction scores are stored as Assays (prediction.score.NAME) and two additional metadata fields: predicted.NAME and predicted.NAME.score which contain the class prediction and the score for that predicted class. For continuous data, an Assay called NAME is returned. NAME here corresponds to the name of the element in the refdata list.

**References**


**Examples**

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

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

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

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

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

## End(Not run)
```

TransferSketchLabels  Transfer data from sketch data to full data
Description

This function transfers cell type labels from a sketched dataset to a full dataset based on the similarities in the lower dimensional space.

Usage

```r
TransferSketchLabels(
  object,
  sketched.assay = "sketch",
  reduction,
  dims,
  refdata = NULL,
  k = 50,
  reduction.model = NULL,
  neighbors = NULL,
  recompute.neighbors = FALSE,
  recompute.weights = FALSE,
  verbose = TRUE
)
```

Arguments

- `object`: A Seurat object.
- `sketched.assay`: Sketched assay name. Default is 'sketch'.
- `reduction`: Dimensional reduction name to use for label transfer.
- `dims`: An integer vector indicating which dimensions to use for label transfer.
- `refdata`: A list of character strings indicating the metadata columns containing labels to transfer. Default is NULL. Similar to refdata in 'MapQuery'
- `k`: Number of neighbors to use for label transfer. Default is 50.
- `reduction.model`: Dimensional reduction model to use for label transfer. Default is NULL.
- `neighbors`: An object storing the neighbors found during the sketching process. Default is NULL.
- `recompute.neighbors`: Whether to recompute the neighbors for label transfer. Default is FALSE.
- `recompute.weights`: Whether to recompute the weights for label transfer. Default is FALSE.
- `verbose`: Print progress and diagnostic messages

Value

A Seurat object with transferred labels stored in the metadata. If a UMAP model is provided, the full data are also projected onto the UMAP space, with the results stored in a new reduction, full.`reduction.model`
UnSketchEmbeddings

*Transfer embeddings from sketched cells to the full data*

**Description**

Transfer embeddings from sketched cells to the full data

**Usage**

```
UnSketchEmbeddings(
  atom.data,
  atom.cells = NULL,
  orig.data,
  embeddings,
  sketch.matrix = NULL
)
```

**Arguments**

- `atom.data` : Atom data
- `atom.cells` : Atom cells
- `orig.data` : Original data
- `embeddings` : Embeddings of atom cells
- `sketch.matrix` : Sketch matrix

UpdateSCTAssays

*Update pre-V4 Assays generated with SCTransform in the Seurat to the new SCTAssay class*

**Description**

Update pre-V4 Assays generated with SCTransform in the Seurat to the new SCTAssay class

**Usage**

```
UpdateSCTAssays(object)
```

**Arguments**

- `object` : A Seurat object

**Value**

A Seurat object with updated SCTAssays
UpdateSymbolList

Get updated synonyms for gene symbols

Description

Find current gene symbols based on old or alias symbols using the gene names database from the HUGO Gene Nomenclature Committee (HGNC)

Usage

GeneSymbolThesarus(
  symbols,
  timeout = 10,
  several.ok = FALSE,
  search.types = c("alias_symbol", "prev_symbol"),
  verbose = TRUE,
  ...
)

UpdateSymbolList(
  symbols,
  timeout = 10,
  several.ok = FALSE,
  verbose = TRUE,
  ...
)

Arguments

- **symbols**: A vector of gene symbols
- **timeout**: Time to wait before canceling query in seconds
- **several.ok**: Allow several current gene symbols for each provided symbol
- **search.types**: Type of query to perform:
  - "alias_symbol": Find alternate symbols for the genes described by symbols
  - "prev_symbol": Find new new symbols for the genes described by symbols
  This parameter accepts multiple options and short-hand options (eg. "prev" for "prev_symbol")
- **verbose**: Show a progress bar depicting search progress
- ... Extra parameters passed to GET

Details

For each symbol passed, we query the HGNC gene names database for current symbols that have the provided symbol as either an alias (alias_symbol) or old (prev_symbol) symbol. All other queries are not supported.
**VariableFeaturePlot**

**Value**

GeneSymbolThesaurus: if several.ok, a named list where each entry is the current symbol found for each symbol provided and the names are the provided symbols. Otherwise, a named vector with the same information.

UpdateSymbolList: symbols with updated symbols from HGNC’s gene names database

**Note**

This function requires internet access

**Source**


**See Also**

GET

**Examples**

```r
## Not run:
GeneSymbolThesaurus(symbols = c("FAM64A"))

## End(Not run)

## Not run:
UpdateSymbolList(symbols = cc.genes$s.genes)

## End(Not run)
```

---

**VariableFeaturePlot**  View variable features

**Description**

View variable features

**Usage**

```r
VariableFeaturePlot(
  object,
  cols = c("black", "red"),
  pt.size = 1,
  log = NULL,
  selection.method = NULL,
  assay = NULL,
  raster = NULL,
)```
raster.dpi = c(512, 512)
)

**Arguments**

- **object**: Seurat object
- **cols**: Colors to specify non-variable/variable status
- **pt.size**: Size of the points on the plot
- **log**: Plot the x-axis in log scale
- **selection.method**
  - [Deprecated]
- **assay**: Assay to pull variable features from
- **raster**: Convert points to raster format, default is NULL which will automatically use raster if the number of points plotted is greater than 100,000
- **raster.dpi**: Pixel resolution for rasterized plots, passed to geom_scattermore(). Default is c(512, 512).

**Value**

A ggplot object

**See Also**

`FindVariableFeatures`

**Examples**

data("pbmc_small")
VariableFeaturePlot(object = pbmc_small)

---

### VisiumV1-class

*The VisiumV1 class*

**Description**

The VisiumV1 class represents spatial information from the 10X Genomics Visium platform

**Slots**

- **image**: A three-dimensional array with PNG image data, see `readPNG` for more details
- **scale.factors**: An object of class `scalefactors`; see `scalefactors` for more information
- **coordinates**: A data frame with tissue coordinate information
- **spot.radius**: Single numeric value giving the radius of the spots
VizDimLoadings

**Visualize Dimensional Reduction genes**

**Description**

Visualize top genes associated with reduction components

**Usage**

```r
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 patchwork ggplot object. If FALSE, return a list of ggplot objects

**Value**

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

**Examples**

```r
data("pbmc_small")
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 = NULL,
  alpha = 1,
  idents = NULL,
  sort = FALSE,
  assay = NULL,
  group.by = NULL,
  split.by = NULL,
  adjust = 1,
  y.max = NULL,
  same.y.lims = FALSE,
  log = FALSE,
  ncol = NULL,
  slot = deprecated(),
  layer = NULL,
  split.plot = FALSE,
  stack = FALSE,
  combine = TRUE,
  fill.by = "feature",
  flip = FALSE,
  add.noise = TRUE,
  raster = NULL
)
```

**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 points
- **alpha**: Alpha value for points
- **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 factor in object metadata to split the plot by, pass 'ident' to split by cell identity
adjust        Adjust parameter for geom_violin
y.max         Maximum y axis value
same.y.lims   Set all the y-axis limits to the same values
log           plot the feature axis on log scale
ncol          Number of columns if multiple plots are displayed
slot          Slot to pull expression data from (e.g. "counts" or "data")
layer         Layer to pull expression data from (e.g. "counts" or "data")
split.plot    plot each group of the split violin plots by multiple or single violin shapes.
stack         Horizontally stack plots for each feature
combine       Combine plots into a single patchwork ggplot object. If FALSE, return a list of ggplot
fill.by       Color violins/ridges based on either 'feature' or 'ident'
flip          flip plot orientation (identities on x-axis)
add.noise     determine if adding a small noise for plotting
raster        Convert points to raster format. Requires 'ggrastr' to be installed.

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

See Also
FetchData

Examples
data("pbmc_small")
VlnPlot(object = pbmc_small, features = 'PC_1')
VlnPlot(object = pbmc_small, features = 'LYZ', split.by = 'groups')
Index

* clustering
  BuildNicheAssay, 28
  FindClusters, 79
  FindMultiModalNeighbors, 91
  FindNeighbors, 93
  FindSubCluster, 98

* convenience
  DimHeatmap, 51
  DimPlot, 52
  ReadParseBio, 194
  ReadSTARsolo, 195
  SpatialPlot, 247

* datasets
  cc.genes, 32
  cc.genes.updated.2019, 32

* data
  cc.genes, 32
  cc.genes.updated.2019, 32

* differential_expression
  FindAllMarkers, 72
  FindConservedMarkers, 81
  FindMarkers, 85
  FoldChange, 105
  PrepSCTFindMarkers, 176

* dimensional_reduction
  JackStraw, 133
  L2CCA, 138
  L2Dim, 138
  PCASigGenes, 166
  ProjectDim, 180
  ProjectUMAP, 183
  RunCCA, 205
  RunGraphLaplacian, 207
  RunICA, 208
  RunPCA, 214
  RunSLSI, 216
  RunSPCA, 217
  RunTSNE, 219
  RunUMAP, 221

  ScoreJackStraw, 231

* future
  PrepSCTFindMarkers, 176
  ReadNanostring, 192
  ReadVizgen, 197

* integration
  AnnotateAnchors, 15
  FindIntegrationAnchors, 82
  FindTransferAnchors, 99
  GetTransferPredictions, 111
  HarmonyIntegration, 114
  IntegrateData, 125
  IntegrateEmbeddings, 128
  IntegrateLayers, 130
  LocalStruct, 148
  MappingScore, 151
  MapQuery, 153
  MixingMetric, 157
  PredictAssay, 172
  PrepSCTIntegration, 177
  SelectIntegrationFeatures, 238
  TransferData, 255

* mixscape
  CalcPerturbSig, 28
  DEenrichRPlot, 48
  MixscapeHeatmap, 158
  MixscapeLDA, 160
  PlotPerturbScore, 169
  PrepLDA, 175
  RunLDA, 210
  RunMixscape, 212

* objects
  AnchorSet-class, 15
  as.CellDataSet, 16
  as.Seurat.CellDataSet, 17
  as.SingleCellExperiment, 18
  as.sparse.H5Group, 18
  BridgeReferenceSet-class, 26
  Cells.SCTModel, 37
CreateSCTAssayObject, 46
DietSeurat, 50
FilterSlideSeq, 71
GetAssay, 107
GetImage.SlideSeq, 108
GetIntegrationData, 109
GetTissueCoordinates.SlideSeq, 110
IntegrationAnchorSet-class, 131
IntegrationData-class, 131
merge.SCTAssay, 154
ModalityWeights-class, 161
Radius.SlideSeq, 186
RenameCells.SCTAssay, 201
ScaleFactors, 230
SCTAssay-class, 232
SCTResults, 237
SetIntegrationData, 241
SplitObject, 250
STARmap-class, 251
subset.AnchorSet, 251
TopCells, 253
TopFeatures, 254
TopNeighbors, 255
TransferAnchorSet-class, 255
UpdateSCTAssays, 260
VisiumV1-class, 263
* preprocessing
  CalculateBarcodeInflections, 30
  FetchResiduals, 68
  FindSpatiallyVariableFeatures, 96
  FindVariableFeatures, 102
  GetResidual, 109
  HTODemux, 117
  Load10X_Spatial, 144
  LoadCurioSeeker, 146
  LoadSTARmap, 146
  LoadXenium, 147
  LogNormalize, 149
  MULTIseqDemux, 162
  NormalizeData, 165
  Read10X, 186
  Read10X_h5, 188
  Read10X_Image, 188
  ReadAkoya, 189
  ReadMtx, 191
  ReadNanostring, 192
  ReadSlideSeq, 195
  ReadVitessce, 196
  ReadVizgen, 197
  RelativeCounts, 200
  RunMarkVario, 211
  RunMoransI, 213
  SampleUMI, 226
  ScaleData, 228
  SCTTransform, 234
  SubsetByBarcodeInflections, 252
* spatial
  Cells.SCTModel, 37
  FilterSlideSeq, 71
  FindSpatiallyVariableFeatures, 96
  GetImage.SlideSeq, 108
  GetTissueCoordinates.SlideSeq, 110
  ISpatialDimPlot, 132
  ISpatialFeaturePlot, 133
  LinkedPlots, 143
  PolyFeaturePlot, 171
  Radius.SlideSeq, 186
  ScaleFactors, 230
  Slide_seq-class, 246
  SpatialPlots, 247
  STARmap-class, 251
  VisiumV1-class, 263
* utilities
  AddModuleScore, 11
  AggregateExpression, 13
  as.sparse.H5Group, 18
  AverageExpression, 21
  CaseMatch, 31
  CellCycleScoring, 35
  CollapseSpeciesExpressionMatrix, 41
  CustomDistance, 47
  ExpMean, 60
  ExpSD, 60
  ExpVar, 61
  FastRowScale, 62
  GroupCorrelation, 112
  LoadAnnoyIndex, 145
  LogVMR, 150
  MetaFeature, 155
 MinMax, 156
  PercentAbove, 167
  PercentageFeatureSet, 168
  RegroupIdents, 199
INDEX

SaveAnnoyIndex, 227
SetQuantile, 242
UpdateSymbolList, 261

* visualization
   AugmentPlot, 19
   AutoPointSize, 20
   BarcodeInflectionsPlot, 22
   BGTextColor, 23
   BlackAndWhite, 24
   CellScatter, 38
   CellSelector, 39
   CollapseEmbeddingOutliers, 40
   ColorDimSplit, 42
   CombinePlots, 44
   contrast-theory, 45
   DimHeatmap, 51
   DimPlot, 52
   DiscretePalette, 55
   DoHeatmap, 56
   DotPlot, 58
   ElbowPlot, 59
   FeaturePlot, 63
   FeatureScatter, 66
   GroupCorrelationPlot, 113
   HoverLocator, 116
   HTOHeatmap, 118
   IFeaturePlot, 120
   ISpatialDimPlot, 132
   ISpatialFeaturePlot, 133
   JackStrawPlot, 135
   LabelClusters, 139
   LabelPoints, 140
   LinkedPlots, 143
   NNPPlot, 163
   PlotClusterTree, 169
   PolyDimPlot, 170
   PolyFeaturePlot, 171
   RidgePlot, 201
   SeuratTheme, 243
   SpatialPlot, 247
   VariableFeaturePlot, 262
   VizDimLoadings, 264
   VlnPlot, 265

?future::plan, 176, 194, 199

AddAzimuthResults, 10
AddAzimuthScores, 11
AddModuleScore, 11, 36
AggregateExpression, 13

AnchorSet, 16, 76, 84, 125, 126, 131, 252, 255, 256
AnchorSet (AnchorSet-class), 15
AnchorSet-class, 15
AnnotateAnchors, 15
annotation_raster, 19
ape::plot.phylo, 169
as.CellDataSet, 16
as.data.frame.Matrix
   (as.sparse.H5Group), 18
as.Seurat.CellDataSet, 17
as.Seurat.SingleCellExperiment
   (as.Seurat.CellDataSet), 17
as.SingleCellExperiment, 18
as.sparse.H5Group, 18
Assay, 127, 178, 232, 233
Assay-class, 19
Assay5, 115
AugmentPlot, 19
AutoPointSize, 20
AverageExpression, 21
BarcodeInflectionsPlot, 22, 31, 253
BGTextColor, 23
BlackAndWhite, 24
BlueAndRed (BlackAndWhite), 24
BoldTitle (SeuratTheme), 243
brewer.pal.info, 43, 53, 122, 135, 248
BridgeCellsRepresentation, 25
BridgeReferenceSet, 77, 78
BridgeReferenceSet
   (BridgeReferenceSet-class), 26
BridgeReferenceSet-class, 26
BuildClusterTree, 26, 74, 89, 107, 158
BuildNicheAssay, 28
CalcPerturbSig, 28
CalculateBarcodeInflections, 23, 30, 253
CaseMatch, 31
cc.genes, 32, 33
cc.genes.updated.2019, 32
CCAIIntegration, 33
CellCycleScoring, 35
CellPlot (CellScatter), 38
Cells.SCTModel, 37
Cells.SlideSeq (Cells.SCTModel), 37
Cells.STARmap (Cells.SCTModel), 37
Cells.VisiumV1 (Cells.SCTModel), 37
CellsByImage, 37
INDEX

CellScatter, 38
CellSelector, 39, 55, 66
CenterTitle (SeuratTheme), 243
CollapseEmbeddingOutliers, 40
CollapseSpeciesExpressionMatrix, 41
ColorDimSplit, 42
CombinePlots, 44, 144
contrast-theory, 45
correct_counts, 237
CreateCategoryMatrix, 46
CreateSCTAssayObject, 46
CreateSeuratObject, 14, 21
CustomDistance, 47
CustomPalette (BlackAndWhite), 24
DarkTheme (SeuratTheme), 243
data.frame, 19
DEenrichRPlot, 48
DefaultAssay, 174
DietSeurat, 50
dimensional reduction, 34, 115, 137, 203
DimHeatmap, 51
DimPlot, 40, 42, 44, 52, 66, 117
DimReduc, 34, 126, 129, 137, 204, 256
DimReduc-class, 55
DiscretePalette, 43, 53, 55, 122, 135
DoHeatmap, 56
DotPlot, 58
ElbowPlot, 59
Embeddings, 253
ExpMean, 60
ExpSD, 60
ExpVar, 61
facet_grid, 139
facet_wrap, 139
FastRowScale, 62
FastRPCAIntegration, 62
FeatureHeatmap (FeaturePlot), 63
FeatureLocator (CellSelector), 39
FeaturePlot, 40, 55, 63, 66, 117
FeatureScatter, 66
FetchData, 55, 59, 171, 266
FetchResiduals, 68
FetchResiduals_reference, 70
FilterSlideSeq, 71
FindAllMarkers, 72
FindAllMarkersNode (FindAllMarkers), 72
FindBridgeAnchor, 75
FindBridgeIntegrationAnchors, 77
FindBridgeTransferAnchors, 78, 175
FindClusters, 79
FindConservedMarkers, 81
FindIntegrationAnchors, 62, 63, 82, 126, 127, 257
FindMarkers, 85, 107
FindMarkersNode (FindMarkers), 85
FindMultiModalNeighbors, 91, 161
FindNeighbors, 93
FindSpatiallyVariableFeatures, 96
FindSubCluster, 98
FindTransferAnchors, 16, 99, 153, 256
FindVariableFeatures, 102, 239, 263
FindVariableGenes (FindVariableFeatures), 102
FoldChange, 105
FontSize (SeuratTheme), 243
GenePlot (FeatureScatter), 66
GeneSymbolThesaurus (UpdateSymbolList), 261
gem_raster, 52
gem_text, 139, 140
gem_text_repel, 139
GET, 261, 262
get_residuals, 69, 110, 237
GetAssay, 107
GetAssayData, 153, 256
GetImage.SlideSeq, 108
GetImage.STARmap (GetImage.SlideSeq), 108
GetImage.VisiumV1 (GetImage.SlideSeq), 108
GetIntegrationData, 109
GetResidual, 109, 177
GetTissueCoordinates.SlideSeq, 110
GetTissueCoordinates.STARmap (GetTissueCoordinates.SlideSeq), 110
GetTissueCoordinates.VisiumV1 (GetTissueCoordinates.SlideSeq), 110
GetTransferPredictions, 111
ggplot_build, 117
Graph, 95
Graph-class, 112
Graphs, 95
GroupCorrelation, 112
GroupCorrelationPlot, 113

harmony::HarmonyMatrix, 115
HarmonyIntegration, 114
hclust, 35, 126, 129, 137, 204
HoverLocator, 55, 66, 116
HTODemux, 117, 119
HTOHeatmap, 118, 118
HVFInfo, 120
HVFInfo.SCTAssay, 120

ICAPlot (DimPlot), 52
IFeaturePlot, 120
image, 52
ImageDimPlot, 121
ImageFeaturePlot, 123
IntegrateData, 15, 82, 84, 125, 130
IntegrateEmbeddings, 62, 76–79, 101, 128, 153, 154
IntegrateLayers, 130
IntegrationAnchorSet
(IntegrationAnchorSet-class), 131
IntegrationAnchorSet-class, 131
IntegrationData
(IntegrationData-class), 131
IntegrationData-class, 131
Intensity (contrast-theory), 45
ISpatialDimPlot, 132, 249
ISpatialFeaturePlot, 133, 249

JackStraw, 133, 167
JackStrawData-class, 135
JackStrawPlot, 135, 232
JointPCAIntegration, 136

L2CCA, 138
L2Dim, 138
LabelClusters, 139
Labeler (LabelPoints), 140
LabelPoints, 140
layout, 116, 117
levels.SCTAssay (SCTAssay-class), 232
levels<-.SCTAssay (SCTAssay-class), 232
LeverageScore, 141
LinkedDimPlot (LinkedPlots), 143
LinkedFeaturePlot (LinkedPlots), 143
LinkedPlot (LinkedPlots), 143
LinkedPlots, 143
Load10X_Spatial, 144, 189
LoadAkoya (ReadAkoya), 189
LoadAnnoyIndex, 145
LoadCurioSeeker, 146
LoadHuBMAPCODEX (ReadVitessce), 196
Loadings, 254
LoadNanostring (ReadNanostring), 192
LoadSTARmap, 146
LoadVizgen (ReadVizgen), 197
LoadXenium, 147
LocalStruct, 148
LogNormalize, 149
LogVMR, 150
Luminance (contrast-theory), 45

make.names, 19
MappingScore, 151
MapQuery, 75, 76, 78, 79, 101, 153
MeanVarPlot (VariableFeaturePlot), 262
merge.SCTAssay, 154
merge.Seurat, 207
MetaFeature, 155
MinMax, 156
MixingMetric, 157
MixscapeHeatmap, 158
MixscapeLDA, 160
ModalityWeights, 92
ModalityWeights
(ModalityWeights-class), 161
ModalityWeights-class, 161
MULTIseqDemux, 162

Neighbor, 95, 255
Neighbor-class, 163
Neighbors, 95
NNPlot, 163
NNtoGraph, 164
NoAxes (SeuratTheme), 243
NoGrid (SeuratTheme), 243
NoLegend (SeuratTheme), 243
NormalizeData, 14, 165

patchwork, 43, 52, 54, 57, 66, 68, 125, 164, 202, 266
PCAPlot (DimPlot), 52
PCASigGenes, 166
PHeatmap (DimHeatmap), 51
PercentAbove, 167
PercentageFeatureSet, 168
plan, 176, 194, 199
PlotClusterTree, 169
PlotPerturbScore, 169
PolyDimPlot, 170, 243
PolyFeaturePlot, 171, 243
PredictAssay, 172
PrepareBridgeReference, 77, 78, 173
PrepLDA, 175
PrepSCTFindMarkers, 176
PrepSCTIntegration, 177
ProjectData, 179
ProjectDim, 167, 180
ProjectDimReduc, 181
ProjectIntegration, 182
ProjectUMAP, 153, 154, 183
PseudobulkExpression, 185
PurpleAndYellow (BlackAndWhite), 24
Radius.SlideSeq, 186
Radius.STARmap (Radius.SlideSeq), 186
Radius.VisiumV1 (Radius.SlideSeq), 186
RANN, 100, 126, 257
RcppAnnoy, 100
Read10X, 186
Read10X_h5, 145, 188
Read10X_image, 188
Read10X_probe_metadata, 189
ReadAkoya, 189
ReadMtx, 191, 194, 195
ReadNanostring, 192
ReadParseBio, 194
readPNG, 263
ReadSlideSeq, 195
ReadSTARsojo, 195
ReadVitescce, 196
ReadVizgen, 197
ReadXenium (LoadXenium), 147
RegroupIdenets, 199
RelativeCounts, 200
RenameCells.SCTAssay, 201
RenameCells.SlideSeq
  (RenameCells.SCTAssay), 201
RenameCells.STARmap
  (RenameCells.SCTAssay), 201
RenameCells.VisiumV1
  (RenameCells.SCTAssay), 201
RestoreLegend (SeuratTheme), 243
RidgePlot, 201
RotatedAxis (SeuratTheme), 243
RPCAIntegration, 203
RunCCA, 205
RunGraphLaplacian, 207
RunICA, 208
RunLDA, 210
RunMarkVario, 97, 211
RunMixscape, 212
RunMoransI, 97, 213
RunPCA, 62, 214
RunSLSI, 216
RunSPCA, 217
RunSVD, 100
RunTFIDF, 100
RunTSNE, 219
RunUMAP, 9, 185, 221
SampleUMI, 226
SaveAnnoyIndex, 227
ScaleData, 14, 22, 228
ScaleFactors, 230
scalefactors, 263
scalefactors (ScaleFactors), 230
ScaleFactors.VisiumV1 (ScaleFactors), 230
ScoreJackStraw, 136, 231
SCTAssay, 120, 237
SCTAssay (SCTAssay-class), 232
SCTAssay-class, 232
SCTModel (SCTAssay-class), 232
SCTransform, 177, 178, 232, 233, 234
SCTResults, 237, 237
SCTResults<- (SCTResults), 237
SelectIntegrationFeatures, 63, 83, 84, 177, 178, 238
SelectIntegrationFeatures5, 240
SelectSCTIntegrationFeatures, 240
SetIntegrationData, 241
SetQuantile, 242
Seurat, 10, 11, 14, 22, 50, 63, 82, 83, 95, 99, 122, 123, 131, 138, 146–148, 155, 177, 178, 190, 194, 197, 199, 241
Seurat (Seurat-package), 8
Seurat-class, 242
Seurat-package, 8
SeuratAxes (SeuratTheme), 243
SeuratCommand-class, 243