Package ‘Seurat’

July 20, 2018

Version 2.3.4
Date 2018-07-16
Title Tools for Single Cell Genomics


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

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

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

Depends R (>= 3.4.0), ggplot2, cowplot, Matrix (>= 1.2.14),

Imports methods, ROCR, mixtools, lars, ica, tsne, Rtsne, fpc, ape, pbapply, igraph, RANN, dplyr, RColorBrewer, MASS, iriba, reshape2, gplots, Rcpp (>= 0.11.0), dtw, SDMTools, ploly, Hmisc, htr, tidyr, ggridges, metap, lmatch, cluster, fitdistplus, png, doSNOW, reticulate, foreach, hdf5r

LinkingTo Rcpp (>= 0.11.0), RcppEigen, RcppProgress

License GPL-3 file LICENSE

LazyData true

Collate 'RcppExports.R' 'alignment.R' 'seurat.R' 'conversion.R' 'as.R'
'cluster_determination.R' 'cluster_determination_internal.R'
'cluster_validation.R' 'data.R' 'deprecated_functions.R'
'differential_expression.R'
'differential_expression_internal.R' 'dimensional_reduction.R'
'dimensional_reduction_internal.R'
'dimensional_reduction_utilities.R' 'interaction.R'
'jackstraw.R' 'jackstraw_internal.R' 'multi_modal.R'
'not_used_yet.R' 'plotting.R' 'plotting_internal.R'


R topics documented:

'Seurat' 'plotting_utilities.R' 'preprocessing.R'
'Seurat' 'preprocessing_internal.R' 'printing_utilities.R' 'scoring.R'
'Seurat' 'spatial.R' 'spatial_internal.R' 'tSNE_project.R'
'Seurat' 'utilities.R' 'utilities_internal.R'

RoxygenNote 6.0.1

Suggests gdata, VGAM, tclust, testthat, caret, ranger, loomR, phateR,
S4Vectors, made4, SummarizedExperiment, SingleCellExperiment,
MAST, DESeq2, destiny

NeedsCompilation yes

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

Maintainer Paul Hoffman <seuratpackage@gmail.com>

Repository CRAN

Date/Publication 2018-07-20 10:14:52

R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddImputedScore</td>
<td>6</td>
</tr>
<tr>
<td>AddMetaData</td>
<td>7</td>
</tr>
<tr>
<td>AddModuleScore</td>
<td>7</td>
</tr>
<tr>
<td>AddSamples</td>
<td>9</td>
</tr>
<tr>
<td>AddSmoothedScore</td>
<td>10</td>
</tr>
<tr>
<td>AlignSubspace</td>
<td>11</td>
</tr>
<tr>
<td>AssessNodes</td>
<td>12</td>
</tr>
<tr>
<td>AssessSplit</td>
<td>13</td>
</tr>
<tr>
<td>AugmentPlot</td>
<td>14</td>
</tr>
<tr>
<td>AverageDetectionRate</td>
<td>14</td>
</tr>
<tr>
<td>AverageExpression</td>
<td>15</td>
</tr>
<tr>
<td>AveragePCA</td>
<td>16</td>
</tr>
<tr>
<td>BatchGene</td>
<td>16</td>
</tr>
<tr>
<td>BlackAndWhite</td>
<td>17</td>
</tr>
<tr>
<td>BuildClusterTree</td>
<td>17</td>
</tr>
<tr>
<td>BuildRFCClassifier</td>
<td>18</td>
</tr>
<tr>
<td>BuildSNN</td>
<td>19</td>
</tr>
<tr>
<td>CalcAlignmentMetric</td>
<td>20</td>
</tr>
<tr>
<td>CalcVarExpRatio</td>
<td>22</td>
</tr>
<tr>
<td>CaseMatch</td>
<td>23</td>
</tr>
<tr>
<td>cc.genes</td>
<td>23</td>
</tr>
<tr>
<td>CellCycleScoring</td>
<td>24</td>
</tr>
<tr>
<td>CellPlot</td>
<td>25</td>
</tr>
<tr>
<td>ClassifyCells</td>
<td>26</td>
</tr>
</tbody>
</table>
topics documented:

CollapseSpeciesExpressionMatrix .................................................. 27
ColorTSNESplit ........................................................................... 28
CombineIdent ............................................................................... 29
Convert ......................................................................................... 29
CreateSeuratObject ........................................................................ 31
CustomDistance ............................................................................ 32
CustomPalette ............................................................................... 33
DarkTheme ...................................................................................... 34
DBClustDimension .......................................................................... 34
DESeq2DETest ................................................................................ 35
DiffExpTest .................................................................................... 36
DiffTTest ......................................................................................... 37
DimElbowPlot ................................................................................ 38
DimHeatmap .................................................................................... 38
DimPlot .......................................................................................... 40
DimTopCells ................................................................................... 42
DimTopGenes ................................................................................ 42
DMEmbed ......................................................................................... 43
DMPlot ............................................................................................. 44
DoHeatmap ....................................................................................... 45
DoKMeans ......................................................................................... 46
DotPlot ............................................................................................. 47
DotPlotOld ...................................................................................... 48
ExpMean .......................................................................................... 49
ExpSD .............................................................................................. 50
ExpVar ............................................................................................. 50
ExtractField ..................................................................................... 51
FastWhichCells .............................................................................. 52
FeatureHeatmap .............................................................................. 52
FeatureLocator ............................................................................... 54
FeaturePlot ..................................................................................... 55
FetchData ......................................................................................... 56
FilterCells ....................................................................................... 57
FindAllMarkers .............................................................................. 58
FindAllMarkersNode ....................................................................... 59
FindClusters ................................................................................... 61
FindConservedMarkers .................................................................. 63
FindGeneTerms .............................................................................. 64
FindMarkers .................................................................................. 64
FindMarkersNode .......................................................................... 66
FindVariableGenes ........................................................................ 67
FitGeneK ........................................................................................ 69
GenePlot ........................................................................................ 70
GenesInCluster ............................................................................... 71
GetAssayData .................................................................................. 72
GetCellEmbeddings ....................................................................... 72
GetCentroids ................................................................................. 73
GetClusters ..................................................................................... 74
R topics documented:

GetDimReduction ................................................. 75
GetGeneLoadings ................................................. 75
GetIdent ............................................................... 76
HoverLocator ...................................................... 77
HTODemux ......................................................... 77
HTOHeatmap ....................................................... 79
ICAEmbed ............................................................. 80
ICALoad ............................................................... 80
ICAPlot ............................................................... 81
ICHeatmap ........................................................... 82
ICTopCells ........................................................... 83
ICTopGenes .......................................................... 83
InitialMapping ..................................................... 84
JackStraw ............................................................. 85
JackStrawPlot ....................................................... 86
KClustDimension ................................................... 87
KMeansHeatmap .................................................... 88
LogNormalize ....................................................... 89
LogVMR ............................................................... 89
MakeSparse .......................................................... 90
MarkerTest ........................................................... 91
MASTDETest ........................................................ 92
MatrixRowShuffle ............................................... 93
MergeNode ........................................................... 93
MergeSeurat ........................................................ 94
MetageneBicorPlot ............................................... 95
MinMax ............................................................... 96
MultiModal_CCA .................................................. 96
MultiModal_CIA .................................................... 97
NegBinomDETest .................................................. 98
NegBinomRegDETest ............................................. 99
NormalizeData .................................................... 100
NumberClusters .................................................. 100
OldDoHeatmap ..................................................... 101
pbmc_small .......................................................... 102
PCAEmbed ........................................................... 103
PCALoad .............................................................. 104
PCAPlot .............................................................. 104
PCASigGenes ........................................................ 105
PCElbowPlot ........................................................ 106
PCHeatmap ........................................................... 106
PCTopCells .......................................................... 107
PCTopGenes ........................................................ 108
PlotClusterTree ..................................................... 109
PoissonDETest ..................................................... 109
PrintAlignSubspaceParams ..................................... 110
PrintCalcParams ................................................... 111
PrintCalcVarExpRatioParams ................................. 112
R topics documented:

PrintCCAParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
PrintDim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
PrintDMParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
PrintFindClustersParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
PrintICA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
PrintICAParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
PrintPCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
PrintPCAParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
PrintSNNParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118
PrintTSNEParams . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
ProjectDim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
ProjectPCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
PurpleAndYellow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121
Read10X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122
Read10X_h5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
RefinedMapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123
RemoveFromTable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124
RenameCells . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
RenameIdent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
ReorderIdent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
RidgePlot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 127
RunCCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 128
RunDiffusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130
RunICA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
RunMultiCCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
RunPCA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
RunPHATE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
RunTSNE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
RunUMAP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138
SampleUMI . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139
SaveClusters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 140
ScaleData . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141
ScaleDataR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142
SetAllIdent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143
SetAssayData . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144
SetClusters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
SetDimReduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
SetIdent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
seurat . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147
Seurat-deprecated . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
Shuffle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
SplitDotPlotGG . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 151
SplitObject . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
StashIdent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
SubsetByPredicate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
SubsetColumn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
SubsetData . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
SubsetRow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
AddImputedScore

Description

Uses L1-constrained linear models (LASSO) to impute single cell gene expression values.

Usage

AddImputedScore(object, genes.use = NULL, genes.fit = NULL, s.use = 20,
                do.print = FALSE, gram = TRUE)

Arguments

object               Seurat object
genes.use            A vector of genes (predictors) that can be used for building the LASSO models.
genes.fit            A vector of genes to impute values for
s.use                Maximum number of steps taken by the algorithm (lower values indicate a greater degree of smoothing)
do.print             Print progress (output the name of each gene after it has been imputed).
gram                 The use.gram argument passed to lars

Value

Returns a Seurat object where the imputed values have been added to object@imputed

Examples

pbmc_small <- AddImputedScore(object = pbmc_small, genes.fit = "MS4A1")
AddMetaData

**Description**

Adds additional data for single cells to the Seurat object. Can be any piece of information associated with a cell (examples include read depth, alignment rate, experimental batch, or subpopulation identity). The advantage of adding it to the Seurat object is so that it can be analyzed/visualized using FetchData, VlnPlot, GenePlot, SubsetData, etc.

**Usage**

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

**Arguments**

- **object**
  - Seurat object

- **metadata**
  - Data frame where the row names are cell names (note: these must correspond exactly to the items in object@cell.names), and the columns are additional metadata items.

- **col.name**
  - Name for metadata if passing in single vector of information

**Value**

Seurat object where the additional metadata has been added as columns in object@meta.data

**Examples**

```r
cluster_letters <- LETTERS[pbmc_small@ident]
pbmc_small <- AddMetaData(
  object = pbmc_small,
  metadata = cluster_letters,
  col.name = 'letter.idents'
)
head(x = pbmc_small@meta.data)
```

---

AddModuleScore

**Calculate module scores for gene expression programs in single cells**

**Description**

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

AddModuleScore(object, genes.list = NULL, genes.pool = NULL, n.bin = 25,
seed.use = 1, ctrl.size = 100, use.k = FALSE, enrich.name = "Cluster",
random.seed = 1)

Arguments

  object       Seurat object
  genes.list   Gene expression programs in list
  genes.pool   List of genes to check expression levels against, defaults to rownames(x = ob-
               ject@data)
  n.bin        Number of bins of aggregate expression levels for all analyzed genes
  seed.use     Random seed for sampling
  ctrl.size    Number of control genes selected from the same bin per analyzed gene
  use.k        Use gene clusters returned from DoKMeans()
  enrich.name  Name for the expression programs
  random.seed  Set a random seed

Value

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

References

Tirosh et al, Science (2016)

Examples

  cd_genes <- list(c(
    'CD79B',
    'CD79A',
    'CD19',
    'CD180',
    'CD200',
    'CD3D',
    'CD2',
    'CD3E',
    'CD7',
    'CD8A',
    'CD14',
    'CD1C',
    'CD68',
    'CD9',
    'CD247'
  ))
  pbmc_small <- AddModuleScore(
    object = pbmc_small,
    genes.list = cd_genes,
    genes.pool = NULL,
    n.bin = 25,
    seed.use = 1,
    ctrl.size = 100,
    use.k = FALSE,
    enrich.name = "Cluster",
    random.seed = 1)
**AddSamples**

Add samples into existing Seurat object.

**Description**

Add samples into existing Seurat object.

**Usage**

```r
AddSamples(object, new.data, project = NULL, min.cells = 0, min.genes = 0,
            is.expr = 0, do.normalize = TRUE, scale.factor = 10000,
            do.scale = FALSE, do.center = FALSE, names.field = 1,
            names.delim = " ", meta.data = NULL, add.cell.id = NULL)
```

**Arguments**

- **object**: Seurat object
- **new.data**: Data matrix for samples to be added
- **project**: Project name (string)
- **min.cells**: Include genes with detected expression in at least this many cells
- **min.genes**: Include cells where at least this many genes are detected
- **is.expr**: Expression threshold for 'detected' gene
- **do.normalize**: Normalize the data after merging. Default is TRUE. If set, will perform the same normalization strategy as stored in the object
- **scale.factor**: scale factor in the log normalization
- **do.scale**: In object@scale.data, perform row-scaling (gene-based z-score)
- **do.center**: In object@scale.data, perform row-centering (gene-based centering)
- **names.field**: For the initial identity class for each cell, choose this field from the cell’s column name
- **names.delim**: For the initial identity class for each cell, choose this delimiter from the cell’s column name
- **meta.data**: Additional metadata to add to the Seurat object. Should be a data frame where the rows are cell names, and the columns are additional metadata fields
- **add.cell.id**: String to be appended to the names of all cells in new.data. E.g. if add.cell.id = "rep1", "cell1" becomes "cell1.rep1"
**AddSmoothedScore**  
Calculate smoothed expression values

**Description**  
Smothes expression values across the k-nearest neighbors based on dimensional reduction

**Usage**  
```
AddSmoothedScore(object, genes.fit = NULL, dim.1 = 1, dim.2 = 2,  
reduction.use = "tsne", k = 30, do.log = FALSE, do.print = FALSE,  
nn.pts = 0)
```

**Arguments**  
- **object**  
  Seurat object
- **genes.fit**  
  Genes to calculate smoothed values for
- **dim.1**  
  Dimension 1 to use for dimensional reduction
- **dim.2**  
  Dimension 2 to use for dimensional reduction
- **reduction.use**  
  Dimensional reduction to use
- **k**  
  k-param for k-nearest neighbor calculation. 30 by default
- **do.log**  
  Whether to perform smoothing in log space. Default is false.
- **do.print**  
  Print progress (output the name of each gene after it has been imputed).
- **nn.pts**  
  Error bound when performing nearest neighbor search using RANN; default of 0.0 implies exact nearest neighbor search

**Examples**
```
pbmc_small <- AddSmoothedScore(object = pbmc_small, genes.fit = "MS4A1", reduction.use = "tsne")
```
Description

Aligns subspaces across a given grouping variable.

Usage

AlignSubspace(object, reduction.type = "cca", grouping.var, dims.align, 
num.possible.genes = 2000, num.genes = 30, show.plots = FALSE, 
verbose = TRUE, ...)

Arguments

- **object**: Seurat object
- **reduction.type**: Reduction to align scores for. Default is "cca".
- **grouping.var**: Name of the grouping variable for which to align the scores
- **dims.align**: Dims to align, default is all
- **num.possible.genes**: Number of possible genes to search when choosing genes for the metagene. Set to 2000 by default. Lowering will decrease runtime but may result in metagenes constructed on fewer than num.genes genes.
- **num.genes**: Number of genes to use in construction of "metagene" (default is 30).
- **show.plots**: Show debugging plots
- **verbose**: Displays progress and other output
- ... Additional parameters to ScaleData

Details

Following is a description for the two group case but this can be extended to arbitrarily many groups which works by performing pairwise alignment to a reference group (the largest group). First, we identify genes that are driving variation in both datasets by looking at the correlation of gene expression with each projection vector (e.g. CC1) in both datasets. For this we use the biweight midcorrelation (bicor) and choose the top num.genes with the strongest bicor to construct a 'metagene' for each dataset. We then scale each metagene to match its 95% reference range and linearly shift them by the minimum difference between the two metagenes over the 10-90 quantile range. We then map each cell in the smaller dataset to a cell in the larger dataset using dynamic time warping (DTW) and apply the same map to the projection vectors (CC vectors) to place both datasets on a common aligned scale. We apply this procedure to each pair (group) of vectors individually for all specified in dims.align. For a full description of the method, see Butler et al 2017.

Value

Returns Seurat object with the dims aligned, stored in object@dr$reduction.type.aligned
Examples

```r
## Not run:
pbmc_small
# Requires CCA to have previously been run
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta.data$group <- "group1"
pbmc2@meta.data$group <- "group2"
pbmc_cca <- runCCA(pbmc1, pbmc2)
pbmc_cca <- AlignSubspace(pbmc_cca, reduction.type = "cca", grouping.var = "group", dims.align = 1:2)

## End(Not run)
```

---

### Assess Internal Nodes

#### Description

Method for automating assessment of tree splits over all internal nodes, or a provided list of internal nodes. Uses AssessSplit() for calculation of Out of Bag error (proxy for confidence in split).

#### Usage

```r
AssessNodes(object, node.list, all.below = FALSE, genes.training = NULL)
```

#### Arguments

- `object`: Seurat object
- `node.list`: List of internal nodes to assess and return
- `all.below`: If single node provided in node.list, assess all splits below (and including) provided node
- `genes.training`: A vector of genes to use to train the classifier, defaults to `rownames(x = object@data)`.

#### Value

Returns the Out of Bag error for a random forest classifiers trained on each internal node split or each split provided in the node list.

#### Examples

```r
## Not run:
pbmc_small
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca",
                           dims.use = 1:10, resolution = 1.1, save.SNN = TRUE)
pbmc_small <- BuildClusterTree(pbmc_small, reorder.numeric = TRUE, do.reorder = TRUE)
```
AssessSplit

AssessNodes(pbmc_small)

## End(Not run)

---

### Assess Cluster Split

**Description**

Method for determining confidence in specific bifurcations in the cluster tree. Use the Out of Bag (OOB) error of a random forest classifier to judge confidence.

**Usage**

```r
AssessSplit(object, node, cluster1, cluster2, genes.training = NULL, print.output = TRUE, ...)
```

**Arguments**

- `object` Seurat object
- `node` Node in the cluster tree in question
- `cluster1` First cluster to compare
- `cluster2` Second cluster to compare
- `genes.training` A vector of genes to use to train the classifier, defaults to `rownames(x = object@data)`
- `print.output` Print the OOB error for the classifier
- `...` Arguments passed on to `buildRFCclassifier`
  - `training.genes` Vector of genes to build the classifier on
  - `training.classes` Vector of classes to build the classifier on
  - `verbose` Additional progress print statements

**Value**

Returns the Out of Bag error for a random forest classifier trained on the split from the given node

**Examples**

```r
pbmc_small
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca",
                           dims.use = 1:10, resolution = 1.1, save.SNN = TRUE)
pbmc_small <- BuildClusterTree(pbmc_small, reorder.numeric = TRUE, do.reorder = TRUE)
# Assess based on a given node
AssessSplit(pbmc_small, node = 11)
# Assess based on two given clusters (or vectors of clusters)
AssessSplit(pbmc_small, cluster1 = 5, cluster2 = 6)
AssessSplit(pbmc_small, cluster1 = 4, cluster2 = c(5, 6))
```
AugmentPlot

*Augments ggplot2 scatterplot with a PNG image.*

**Description**

Used in to creating vector friendly plots. Exported as it may be useful to others more broadly.

**Usage**

AugmentPlot(plot1, imgFile)

**Arguments**

- `plot1`: ggplot2 scatterplot. Typically will have only labeled axes and no points
- `imgFile`: location of a PNG file that contains the points to overlay onto the scatterplot.

**Value**

ggplot2 scatterplot that includes the original axes but also the PNG file

**Examples**

```r
## Not run:  
data("pbmc_small")  
p <- PCAPlot(pbmc_small, do.return = TRUE)  
ggsave(filename = 'pcaplot.png', plot = p, device = png)  
pmod <- AugmentPlot(plot1 = p, imgFile = 'pcaplot.png')  
pmod  
## End(Not run)
```

-----

AverageDetectionRate

*Probability of detection by identity class*

**Description**

For each gene, calculates the probability of detection for each identity class.

**Usage**

AverageDetectionRate(object, thresh.min = 0)

**Arguments**

- `object`: Seurat object
- `thresh.min`: Minimum threshold to define 'detected' (log-scale)
AverageExpression

Value

Returns a matrix with genes as rows, identity classes as columns.

Examples

```r
head(AverageDetectionRate(object = pbmc_small))
```

---

### AverageExpression

**Averaged gene expression by identity class**

**Description**

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

**Usage**

```r
AverageExpression(object, genes.use = NULL, return.seurat = FALSE,
      add.ident = NULL, use.scale = FALSE, use.raw = FALSE,
      show.progress = TRUE, ...)
```

**Arguments**

- **object**: Seurat object
- **genes.use**: Genes to analyze. Default is all genes.
- **return.seurat**: Whether to return the data as a Seurat object. Default is false.
- **add.ident**: Place an additional label on each cell prior to averaging (very useful if you want to observe cluster averages, separated by replicate, for example).
- **use.scale**: Use scaled values for gene expression
- **use.raw**: Use raw values for gene expression
- **show.progress**: Show progress bar (default is T)
- **...**: Arguments to be passed to methods such as Seurat

**Details**

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

**Value**

Returns a matrix with genes as rows, identity classes as columns.

**Examples**

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

\textit{Average PCA scores by identity class}

\textbf{Description}

Returns the PCA scores for an 'average' single cell in each identity class

\textbf{Usage}

\texttt{AveragePCA(object)}

\textbf{Arguments}

- \texttt{object}: Seurat object

\textbf{Value}

Returns a matrix with genes as rows, identity classes as columns

\textbf{Examples}

\texttt{head(AveragePCA(object = pbmc_small))}

\textbf{BatchGene}

\textit{Identify potential genes associated with batch effects}

\textbf{Description}

Test for genes whose expression value is strongly predictive of batch (based on ROC classification). Important note: Assumes that the 'batch' of each cell is assigned to the cell’s identity class (will be improved in a future release)

\textbf{Usage}

\texttt{BatchGene(object, idents.use, genes.use = NULL, auc.cutoff = 0.6, thresh.use = 0)}

\textbf{Arguments}

- \texttt{object}: Seurat object
- \texttt{idents.use}: Batch names to test
- \texttt{genes.use}: Gene list to test
- \texttt{auc.cutoff}: Minimum AUC needed to qualify as a 'batch gene'
- \texttt{thresh.use}: Limit testing to genes which show, on average, at least X-fold difference (log-scale) in any one batch
**BlackAndWhite**

**Value**

Returns a list of genes that are strongly correlated with batch.

---

**A black and white color palette**

**Description**

A black and white color palette

**Usage**

BlackAndWhite(...)

**Arguments**

... Extra parameters to CustomPalette

**Value**

A color palette

**See Also**

CustomPalette

**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())

---

**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, genes.use = NULL, pcs.use = NULL, SNN.use = NULL, do.plot = TRUE, do.reorder = FALSE, reorder.numeric = FALSE, show.progress = TRUE)
Arguments

- **object**: Seurat object
- **genes.use**: Genes to use for the analysis. Default is the set of variable genes (object@var.genes). Assumes pcs.use=NULL (tree calculated in gene expression space)
- **pcs.use**: If set, tree is calculated in PCA space.
- **SNN.use**: If SNN is passed, build tree based on SNN graph connectivity between clusters
- **do.plot**: Plot the resulting phylogenetic tree
- **do.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)
- **show.progress**: 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 is stored in object@cluster.tree[[1]]

Examples

```r
pbmc_small
pbmc_small <- BuildClusterTree(pbmc_small, do.plot = FALSE)
```

BuildRFCClassifier  Build Random Forest Classifier

Description

Train the random forest classifier

Usage

```r
BuildRFClassifier(object, training.genes = NULL, training.classes = NULL, verbose = TRUE, ...)
```
Arguments

- **object**: Seurat object on which to train the classifier.
- **training.genes**: Vector of genes to build the classifier on.
- **training.classes**: Vector of classes to build the classifier on.
- **verbose**: Additional progress print statements.
- **...**: Additional parameters passed to ranger.

Value

Returns the random forest classifier.

Examples

```r
pbmc_small
# Builds the random forest classifier to be used with ClassifyCells
# Useful if you want to use the same classifier with several sets of new data
classifier <- BuildRFCClassifier(pbmc_small, training.classes = pbmc_small@ident)
```

BuildSNN  
*SNN Graph Construction*

Description

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

Usage

```r
BuildSNN(object, genes.use = NULL, reduction.type = "pca",
         dims.use = NULL, k.param = 10, plot.SNN = FALSE, prune.SNN = 1/15,
         print.output = TRUE, distance.matrix = NULL, force.recalc = FALSE,
         filename = NULL, save.SNN = TRUE, nn.eps = 0)
```

Arguments

- **object**: Seurat object.
- **genes.use**: A vector of gene names to use in construction of SNN graph if building directly based on expression data rather than a dimensionally reduced representation (i.e. PCs).
- **reduction.type**: Name of dimensional reduction technique to use in construction of SNN graph. (e.g. "pca", "ica")
- **dims.use**: A vector of the dimensions to use in construction of the SNN graph (e.g. To use the first 10 PCs, pass 1:10)
CalcAlignmentMetric

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

plot.SNN
- Plot the SNN 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).

print.output
- Whether or not to print output to the console

distance.matrix
- Build SNN from distance matrix (experimental)

force.recalc
- Force recalculation of SNN.

filename
- Write SNN directly to file named here as an edge list compatible with FindClusters

save.SNN
- Default behavior is to store the SNN in object@snn. Setting to FALSE can be used together with a provided filename to only write the SNN out as an edge file to disk.

nn.eps
- Error bound when performing nearest neighbor search using RANN; default of 0.0 implies exact nearest neighbor search

Value

Returns the object with object@snn filled

Examples

pbmc_small
- # Compute an SNN on the gene expression level
  pbmc_small <- BuildSNN(pbmc_small, genes.use = pbmc_small@var.genes)

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

  pbmc_small <- BuildSNN(pbmc_small, reduction.type = "pca", dims.use = 1:10)

CalcAlignmentMetric

Calculate an alignment score

Description

Calculates an alignment score to determine how well aligned two (or more) groups have been aligned. We first split the data into groups based on the grouping.var provided and randomly downsample all groups to have as many cells as in the smallest group. We then construct a nearest neighbor graph and ask for each cell, how many of its neighbors have the same group identity as it does. We then take the average over all cells, compare it to the expected value for perfectly mixed neighborhoods, and scale it to range from 0 to 1.
CalcAlignmentMetric

Usage

CalcAlignmentMetric(object, reduction.use = "cca.align\ded", dims.use,
grouping.var, nn, nn.eps = 0)

Arguments

object Seurat object
reduction.use Stored dimensional reduction on which to build NN graph. Usually going to be
cca.align\ded.
dims.use Dimensions to use in building the NN graph
grouping.var Grouping variable used in the alignment.
nn Number of neighbors to calculate in the NN graph construction
nn.eps Error bound when performing nearest neighbor search using RANN; default of
0.0 implies exact nearest neighbor search

Details

xbar is the average number of neighbors belonging to any cells' same group, N is the number of
groups in the given grouping.var, k is the number of neighbors in the KNN graph.

\[
1 - \frac{x - k}{N - k}
\]

Examples

## Not run:
pbmc_small
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta.data$group <- "group1"
pbmc2@meta.data$group <- "group2"
pbmc_cca <- RunCCA(pbmc1, pbmc2)
pbmc_cca <- AlignSubspace(pbmc_cca, reduction.type = "cca",
                        grouping.var = "group", dims.align = 1:5)
CalcAlignmentMetric(pbmc_cca, reduction.use = "cca.align\ded",
                   dims.use = 1:5, grouping.var = "group")

## End(Not run)
CalcVarExpRatio

*Calculate the ratio of variance explained by ICA or PCA to CCA*

**Description**

Calculate the ratio of variance explained by ICA or PCA to CCA

**Usage**

```r
CalcVarExpRatio(object, reduction.type = "pca", grouping.var, dims.use, verbose = TRUE)
```

**Arguments**

- `object`: Seurat object
- `reduction.type`: type of dimensional reduction to compare to CCA (pca, pcafast, ica)
- `grouping.var`: variable to group by
- `dims.use`: Vector of dimensions to project onto (default is the 1:number stored for cca)
- `verbose`: Display progress and other output

**Value**

Returns Seurat object with ratio of variance explained stored in object@meta.data$var.ratio

**Examples**

```r
pbmc_small
# Requires CCA to have previously been run
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmcsmall,cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmcsmall,cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta.data$group <- "group1"
pbmc2@meta.data$group <- "group2"
pbmc_cca <- RunCCA(pbmcsmall)
pbmc_cca <- CalcVarExpRatio(pbmcsmall,reduction.type = "pca", grouping.var = "group", dims.use = 1:5)
```
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**

```r
cd_genes <- c('Cd79b', 'Cd19', 'Cd200')
CaseMatch(search = cd_genes, match = rownames(x = pbmc_small@raw.data))
```

cc.genes

Cell cycle genes

**Description**

A list of genes used in cell-cycle regression

**Usage**

cc.genes

**Format**

A list of two vectors

- `s.genes` Genes associated with S-phase
- `g2m.genes` Genes associated with G2M-phase

**Source**

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

**Score cell cycle phases**

#### Description

Score cell cycle phases

#### Usage

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

#### Arguments

- `object`: A Seurat object
- `g2m.genes`: A vector of genes associated with G2M phase
- `s.genes`: A vector of genes associated with S phases
- `set.ident`: If true, sets identity to phase assignments Stashes old identities in 'old.ident'

#### Value

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

#### See Also

`addmodulescore`

#### Examples

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

## End(Not run)
```
**CellPlot**

**Cell-cell scatter plot**

**Description**

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

**Usage**

```r
CellPlot(object, cell1, cell2, gene.ids = NULL, col.use = "black",
        nrpoints.use = Inf, pch.use = 16, cex.use = 0.5, do.hover = FALSE,
        do.identify = FALSE, ...)
```

**Arguments**

- `object`: Seurat object
- `cell1`: Cell 1 name (can also be a number, representing the position in object@cell.names)
- `cell2`: Cell 2 name (can also be a number, representing the position in object@cell.names)
- `gene.ids`: Genes to plot (default, all genes)
- `col.use`: Colors to use for the points
- `nrpoints.use`: Parameter for smoothScatter
- `pch.use`: Point symbol to use
- `cex.use`: Point size
- `do.hover`: Enable hovering over points to view information
- `do.identify`: Opens a locator session to identify clusters of cells. points to reveal gene names (hit ESC to stop)
- `...`: Additional arguments to pass to smoothScatter

**Value**

No return value (plots a scatter plot)

**Examples**

```r
CellPlot(object = pbmc_small, cell1 = 'ATAGGAGAAACAGA', cell2 = 'CATCAGGATGCACA')
```
Classify new data based on the cluster information of the provided object. Random Forests are used as the basis of the classification.

### Usage

```r
ClassifyCells(object, classifier, training.genes = NULL,
               training.classes = NULL, new.data = NULL, ...)
```

### Arguments

- **object**: Seurat object on which to train the classifier
- **classifier**: Random Forest classifier from `BuildRFClassifier`. If not provided, it will be built from the training data provided.
- **training.genes**: Vector of genes to build the classifier on
- **training.classes**: Vector of classes to build the classifier on
- **new.data**: New data to classify
- **...**: additional parameters passed to `ranger`

### Value

Vector of cluster ids

### Examples

```r
pbmc_small
# take the first 10 cells as test data and train on the remaining 70 cells
test.pbmc <- SubsetData(object = pbmc_small, cells.use = pbmc_small@cell.names[1:10])
train.pbmc <- SubsetData(object = pbmc_small, cells.use = pbmc_small@cell.names[11:80])
predicted.classes <- ClassifyCells(
    object = train.pbmc,
    training.classes = train.pbmc@ident,
    new.data = test.pbmc@data
)```
**CollapseSpeciesExpressionMatrix**

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

**Description**

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

**Usage**

```r
CollapseSpeciesExpressionMatrix(data.matrix, prefix.1 = "HUMAN_", prefix.controls = "MOUSE_", features.controls.toKeep = 100)
```

**Arguments**

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

**Value**

A UMI count matrix. Rownames that started with `prefix.1` have this prefix discarded. For rownames starting with `prefix.2`, only the 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)
```
**ColorTSNESplit**  
*Color tSNE Plot Based on Split*

**Description**

Returns a tSNE plot 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**

```r
ColorTSNESplit(object, node, color1 = "red", color2 = "blue", color3 = "gray", ...)
```

**Arguments**

- `object` Seurat object
- `node` Node in cluster tree on which to base the split
- `color1` Color for the left side of the split
- `color2` Color for the right side of the split
- `color3` Color for all other cells
- `...` Arguments passed on to `tsneplot`
  - `do.label` FALSE by default. If TRUE, plots an alternate view where the center of each cluster is labeled
  - `pt.size` Set the point size
  - `label.size` Set the size of the text labels
  - `cells.use` Vector of cell names to use in the plot.
  - `colors.use` Manually set the color palette to use for the points

**Value**

Returns a tSNE plot

**Examples**

```
pbmc_small
PlotClusterTree(pbmc_small)
ColorTSNESplit(pbmc_small, node = 6)
```
### CombineIdent

**Sets identity class information to be a combination of two object attributes**

**Description**

Combined two attributes to define identity classes. Very useful if, for example, you have multiple cell types and multiple replicates, and you want to group cells based on combinations of both.

**Usage**

```r
CombineIdent(object, attribute.1 = "ident", attribute.2 = "orig.ident")
```

**Arguments**

- **object**: Seurat object
- **attribute.1**: First attribute for combination. Default is "ident"
- **attribute.2**: Second attribute for combination. Default is "orig.ident"

**Value**

A Seurat object where object@ident has been appropriately modified

**Examples**

```r
groups <- sample(c("group1", "group2", "group3"), size = 80, replace = TRUE)
celltype <- sample(c("celltype1", "celltype2", "celltype3"), size = 80, replace = TRUE)
new.metadata <- data.frame(groups = groups, celltype = celltype)
rownames(new.metadata) <- pbmc_small@cell.names
pbmc_small <- AddMetadata(object = pbmc_small, metadata = new.metadata)
pbmc_small <- CombineIdent(object = pbmc_small, attribute.1 = "celltype", attribute.2 = "groups")
pbmc_small@ident
```

### Convert

**Convert Seurat objects to other classes and vice versa**

**Description**

Usage

Convert(from, ...)

## S3 method for class 'seurat'
Convert(from, to, filename, chunk.dims = "auto",
    chunk.size = 1000, overwrite = FALSE, display.progress = TRUE,
    anndata.raw = "raw.data", anndata.X = "data", ...)

## S3 method for class 'SingleCellExperiment'
Convert(from, to, raw.data.slot = "counts",
    data.slot = "logcounts", ...)

## S3 method for class 'anndata.base.AnnData'
Convert(from, to, X.slot = "scale.data",
    raw.slot = "data", ...)

as.seurat(from)

## S3 method for class 'SingleCellExperiment'
as.seurat(from)

as.SingleCellExperiment(from)

## S3 method for class 'seurat'
as.SingleCellExperiment(from)

Arguments

from Object to convert from
...
Arguments passed to and from other methods
to Class of object to convert to
filename Filename for writing files
chunk.dims Internal HDF5 chunk size
chunk.size Number of cells to stream to loom file at a time
overwrite Overwrite existing file at filename?
display.progress Display a progress bar
anndata.raw Name of matrix (raw.data, data) to put in the anndata raw slot
anndata.X Name of matrix (data, scale.data) to put in the anndata X slot
raw.data.slot name of the SingleCellExperiment assay to slot into @raw.data
data.slot name of the SingleCellExperiment assay to slot into @data
X.slot Seurat slot to transfer anndata X into. Default is scale.data
raw.slot Seurat slot to transfer anndata raw into. Default is data
CreateSeuratObject

Value

An object of class to

Methods (by class)

- `seurat`: Convert a Seurat object
- `SingleCellExperiment`: Convert from SingleCellExperiment to a Seurat object
- `anndata.base.AnData`: from Anndata file to a Seurat object

Description

Initialize and setup the Seurat object

Usage

CreateSeuratObject(raw.data, project = "SeuratProject", min.cells = 0, min.genes = 0, is.expr = 0, normalization.method = NULL, scale.factor = 10000, do.scale = FALSE, do.center = FALSE, names.field = 1, names.delim = ",", meta.data = NULL, display.progress = TRUE, ...)

Arguments

- `raw.data`: Raw input data
- `project`: Project name (string)
- `min.cells`: Include genes with detected expression in at least this many cells. Will subset the raw.data matrix as well. To reintroduce excluded genes, create a new object with a lower cutoff.
- `min.genes`: Include cells where at least this many genes are detected.
- `is.expr`: Expression threshold for 'detected' gene. For most datasets, particularly UMI datasets, will be set to 0 (default). If not, when initializing, this should be set to a level based on pre-normalized counts (i.e. require at least 5 counts to be treated as expressing) All values less than this will be set to 0 (though maintained in object@raw.data).
- `normalization.method`: Method for cell normalization. Default is no normalization. In this case, run NormalizeData later in the workflow. As a shortcut, you can specify a normalization method (i.e. LogNormalize) here directly.
- `scale.factor`: If normalizing on the cell level, this sets the scale factor.
- `do.scale`: In object@scale.data, perform row-scaling (gene-based z-score). FALSE by default. In this case, run ScaleData later in the workflow. As a shortcut, you can specify do.scale = TRUE (and do.center = TRUE) here.
do.center  In object@scale.data, perform row-centering (gene-based centering)
names.field For the initial identity class for each cell, choose this field from the cell’s column name
names.delim For the initial identity class for each cell, choose this delimiter from the cell’s column name
meta.data  Additional metadata to add to the Seurat object. Should be a data frame where the rows are cell names, and the columns are additional metadata fields
display.progress display progress bar for normalization and/or scaling procedure.
... Ignored

Value
Returns a Seurat object with the raw data stored in object@raw.data. object@data, object@meta.data, object@ident, also initialized.

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

---

**CustomDistance**

_Run a custom distance function on an input data matrix_

**Description**

Run a custom distance function on an input data matrix

**Usage**

```r
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
CustomPalette

Author(s)
Jean Fan

Examples

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

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

custompalette

Create a custom color palette

Description

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

Usage

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

Arguments

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

Value

A color palette for plotting

Examples

myPalette <- CustomPalette()
myPalette
DarkTheme  

Description
Add a dark theme to ggplot objects

Usage
DarkTheme(...)

Arguments
... Extra parameters to be passed to theme()

Value
A ggplot2 theme object

See Also
theme

Examples
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')

DBClustDimension  

Description
Perform spectral density clustering on single cells

Usage
DBClustDimension(object, dim.1 = 1, dim.2 = 2, reduction.use = "tsne", G.use = NULL, set.ident = TRUE, seed.use = 1, ...)

Description
Find point clouds single cells in a two-dimensional space using density clustering (DBSCAN).
DESeq2DETest

Arguments

- object: Seurat object
- dim.1: First dimension to use
- dim.2: Second dimension to use
- reduction.use: Which dimensional reduction to use (either 'pca' or 'ica')
- G.use: Parameter for the density clustering. Lower value to get more fine-scale clustering
- set.ident: TRUE by default. Set identity class to the results of the density clustering. Unassigned cells (cells that cannot be assigned a cluster) are placed in cluster 1, if there are any.
- seed.use: Random seed for the dbscan function
- ...: Additional arguments to be passed to the dbscan function

Examples

```r
pbmc_small
# Density based clustering on the first two tSNE dimensions
pbmc_small <- DBClustDimension(pbmc_small)
```

DESeq2DETest

Differential expression using DESeq2

Description

Identifies differentially expressed genes between two groups of cells using DESeq2

Usage

```
DESeq2DETest(object, cells.1, cells.2, genes.use = NULL, assay.type = "RNA", ...)
```

Arguments

- object: Seurat object
- cells.1: Group 1 cells
- cells.2: Group 2 cells
- genes.use: Genes to use for test
- assay.type: Type of assay to fetch data for (default is RNA)
- ...: Extra parameters to pass to DESeq2::results
Details

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

Value

Returns a p-value ranked matrix of putative differentially expressed genes.

References


Examples

```R
## Not run:
pbm_small
DESeq2DETest(pbm_small, cells.1 = WhichCells(object = pbm_small, ident = 1),
cells.2 = WhichCells(object = pbm_small, ident = 2))

## End(Not run)
```

DiffExpTest

Likelihood ratio test for zero-inflated data

Description

Identifies differentially expressed genes between two groups of cells using the LRT model proposed in McDavid et al, Bioinformatics, 2013

Usage

```R
DiffExpTest(object, cells.1, cells.2, assay.type = "RNA", genes.use = NULL,
print.bar = TRUE)
```

Arguments

- `object` Seurat object
- `cells.1` Group 1 cells
- `cells.2` Group 2 cells
- `assay.type` Type of assay to fetch data for (default is RNA)
- `genes.use` Genes to test. Default is to use all genes
- `print.bar` Print a progress bar once expression testing begins (uses pbapply to do this)
Value

Returns a p-value ranked matrix of putative differentially expressed genes.

Examples

```r
pbmc_small
DiffExpTest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
            cells.2 = WhichCells(object = pbmc_small, ident = 2))
```

---

**DiffTTest**

*Differential expression testing using Student’s t-test*

Description

Identify differentially expressed genes between two groups of cells using the Student’s t-test

Usage

```r
DiffTTest(object, cells.1, cells.2, genes.use = NULL, print.bar = TRUE,
       assay.type = "RNA")
```

Arguments

- **object**: Seurat object
- **cells.1**: Group 1 cells
- **cells.2**: Group 2 cells
- **genes.use**: Genes to test. Default is to use all genes
- **print.bar**: Print a progress bar once expression testing begins (uses pbapply to do this)
- **assay.type**: Type of assay to fetch data for (default is RNA)

Value

Returns a p-value ranked matrix of putative differentially expressed genes.

Examples

```r
pbmc_small
DiffTTest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
           cells.2 = WhichCells(object = pbmc_small, ident = 2))
```
DimElbowPlot  

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

DimElbowPlot(object, reduction.type = "pca", dims.plot = 20, xlab = ",", ylab = ",", title = ",")

Arguments

- object: Seurat object
- reduction.type: Type of dimensional reduction to plot data for
- dims.plot: Number of dimensions to plot sd for
- xlab: X axis label
- ylab: Y axis label
- title: Plot title

Value

Returns ggplot object

Examples

DimElbowPlot(object = pbmc_small)

DimHeatmap  

Dimensional reduction heatmap

Description

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

DimHeatmap(object, assay.use = "RNA", reduction.type = "pca", dim.use = 1,
cells.use = NULL, num.genes = 30, use.full = FALSE, disp.min = -2.5,
disp.max = 2.5, do.return = FALSE, col.use = PurpleAndYellow(),
use.scale = TRUE, do.balanced = FALSE, remove.key = FALSE,
label.columns = NULL, check.plot = TRUE, ...)

Arguments

object Seurat object.
assay.use Assay to pull from - default is RNA
reduction.type Which dimmensional reduction t use
dim.use Dimensions to plot
cells.use A list of cells to plot. If numeric, just plots the top cells.
num.genes NUmber of genes to plot
use.full Use the full PCA (projected PCA). Default is FALSE
disp.min Minimum display value (all values below are clipped)
disp.max Maximum display value (all values above are clipped)
do.return If TRUE, returns plot object, otherwise plots plot object
col.use Color to plot.
use.scale Default is TRUE: plot scaled data. If FALSE, plot raw data on the heatmap.
do.balanced Plot an equal number of genes with both + and - scores.
remove.key Removes the color key from the plot.
label.columns Labels for columns
check.plot Check that plotting will finish in a reasonable amount of time
... Extra parameters for heatmap plotting.

Value

If do.return==TRUE, a matrix of scaled values which would be passed to heatmap.2. Otherwise, no return value, only a graphical output

Examples

DimHeatmap(object = pbmc_small)
**DimPlot**

**Dimensional reduction plot**

**Description**

Graphs the output of a dimensional reduction technique (PCA by default). Cells are colored by their identity class.

**Usage**

```r
DimPlot(object, reduction.use = "pca", dim.1 = 1, dim.2 = 2,
        cells.use = NULL, pt.size = 1, do.return = FALSE, do.bare = FALSE,
        cols.use = NULL, group.by = "ident", pt.shape = NULL,
        do.hover = FALSE, data.hover = "ident", do.identify = FALSE,
        do.label = FALSE, label.size = 4, no.legend = FALSE,
        coord.fixed = FALSE, no.axes = FALSE, dark.theme = FALSE,
        plot.order = NULL, cells.highlight = NULL, cols.highlight = "red",
        sizes.highlight = 1, plot.title = NULL, vector.friendly = FALSE,
        png.file = NULL, png.arguments = c(10, 10, 100), na.value = "grey50",
        ...)```

**Arguments**

- **object**: Seurat object
- **reduction.use**: Which dimensionality reduction to use. Default is "pca", can also be "tsne", or "ica", assuming these are precomputed.
- **dim.1**: Dimension for x-axis (default 1)
- **dim.2**: Dimension for y-axis (default 2)
- **cells.use**: Vector of cells to plot (default is all cells)
- **pt.size**: Adjust point size for plotting
- **do.return**: Return a ggplot2 object (default : FALSE)
- **do.bare**: Do only minimal formatting (default : FALSE)
- **cols.use**: Vector of colors, each color corresponds to an identity class. By default, ggplot assigns colors.
- **group.by**: Group (color) cells in different ways (for example, orig.ident)
- **pt.shape**: 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.
- **do.hover**: Enable hovering over points to view information
- **data.hover**: Data to add to the hover, pass a character vector of features to add. Defaults to cell name and ident. Pass ‘NULL’ to clear extra information.
- **do.identify**: Opens a locator session to identify clusters of cells.
- **do.label**: Whether to label the clusters
label.size  Sets size of labels
no.legend   Setting to TRUE will remove the legend
coord.fixed Use a fixed scale coordinate system (for spatial coordinates). Default is FALSE.
no.axes     Setting to TRUE will remove the axes
dark.theme  Use a dark theme for the plot
plot.order  Specify the order of plotting for the idsents. This can be useful for crowded plots
            if points of interest are being buried. Provide either a full list of valid idsents or a
            subset to be plotted last (on top).
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
plot.title   Title for plot
vector.friendly FALSE by default. If TRUE, points are flattened into a PNG, while axes/labels
               retain full vector resolution. Useful for producing AI-friendly plots with large
               numbers of cells.
png.file     Used only if vector.friendly is TRUE. Location for temporary PNG file.
png.arguments Used only if vector.friendly is TRUE. Vector of three elements (PNG width,
               PNG height, PNG DPI) to be used for temporary PNG. Default is c(10,10,100)
na.value     Color value for NA points when using custom scale.
...          Extra parameters to FeatureLocator for do.identify = TRUE

Value

If do.return==TRUE, returns a ggplot2 object. Otherwise, only graphical output.

See Also

FeatureLocator

Examples

DimPlot(object = pbmc_small)
**DimTopCells**

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

**Description**

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

**Usage**

```r
DimTopCells(object, dim.use = 1, reduction.type = "pca", num.cells = NULL, 
do.balanced = FALSE)
```

**Arguments**

- `object`: Seurat object
- `dim.use`: Components to use
- `reduction.type`: Dimensional reduction to find the highest score for
- `num.cells`: Number of cells to return
- `do.balanced`: Return an equal number of cells with both + and - scores.

**Value**

Returns a vector of cells

**Examples**

```r
pbmc_small
head(DimTopCells(object = pbmc_small, reduction.type = "pca"))
# Can specify which dimension and how many cells to return
DimTopCells(object = pbmc_small, reduction.type = "pca", dim.use = 2, num.cells = 5)
```

---

**DimTopGenes**

*Find genes with highest scores for a given dimensional reduction technique*

**Description**

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

**Usage**

```r
DimTopGenes(object, dim.use = 1, reduction.type = "pca", num.genes = 30, 
use.full = FALSE, do.balanced = FALSE)
```
**DMEmbed**

**Arguments**

- **object**: Seurat object
- **dim.use**: Dimension to use
- **reduction.type**: Dimensional reduction to find the highest score for
- **num.genes**: Number of genes to return
- **use.full**: Use the full PCA (projected PCA). Default is FALSE
- **do.balanced**: Return an equal number of genes with both + and - scores.

**Value**

Returns a vector of genes

**Examples**

```r
pbmc_small
DimTopGenes(object = pbmc_small, dim.use = 1, reduction.type = "pca")
# After projection:
DimTopGenes(object = pbmc_small, dim.use = 1, reduction.type = "pca", use.full = TRUE)
```

---

**DMEmbed**

*Diffusion Maps Cell Embeddings Accessor Function*

**Description**

Pull Diffusion maps cell embedding matrix

**Usage**

```r
DMEmbed(object, dims.use = NULL, cells.use = NULL)
```

**Arguments**

- **object**: Seurat object
- **dims.use**: Dimensions to include (default is all stored dims)
- **cells.use**: Cells to include (default is all cells)

**Value**

Diffusion maps embedding matrix for given cells and DMs
DMPlot

Plot Diffusion map

Description

Graphs the output of a Diffusion map analysis. Cells are colored by their identity class.

Usage

DMPlot(object, ...)

Arguments

- `object` - Seurat object
- `...` - Additional parameters to DimPlot, for example, which dimensions to plot.

Details

This function is a wrapper for DimPlot. See ?DimPlot for a full list of possible arguments which can be passed in here.

Examples

```r
## Not run:
pbmc_small <- RunDiffusion(pbmce_small, genes.use = pbmc_small@var.genes)
head(DMEMbed(object = pbmc_small))

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

Draws a heatmap of single cell gene expression using ggplot2.

**Usage**

```r
DoHeatmap(object, data.use = NULL, use.scaled = TRUE, cells.use = NULL, 
genes.use = NULL, disp.min = -2.5, disp.max = 2.5, group.by = "ident", 
group.order = NULL, draw.line = TRUE, col.low = "#FF00FF", 
col.mid = "#000000", col.high = "#FFFF00", slim.col.label = FALSE, 
remove.key = FALSE, rotate.key = FALSE, title = NULL, cex.col = 10, 
cex.row = 10, group.label.loc = "bottom", group.label.rot = FALSE, 
group.cex = 15, group.spacing = 0.15, assay.type = "RNA", 
do.plot = TRUE)
```

**Arguments**

- **object**: Seurat object
- **data.use**: Option to pass in data to use in the heatmap. Default will pick from either object@data or object@scale.data depending on use.scaled parameter. Should have cells as columns and genes as rows.
- **use.scaled**: Whether to use the data or scaled data if data.use is NULL
- **cells.use**: Cells to include in the heatmap (default is all cells)
- **genes.use**: Genes to include in the heatmap (ordered)
- **disp.min**: Minimum display value (all values below are clipped)
- **disp.max**: Maximum display value (all values above are clipped)
- **group.by**: Groups cells by this variable. Default is object@ident
- **group.order**: Order of groups from left to right in heatmap.
- **draw.line**: Draw vertical lines delineating different groups
- **col.low**: Color for lowest expression value
- **col.mid**: Color for mid expression value
- **col.high**: Color for highest expression value
- **slim.col.label**: display only the identity class name once for each group
- **remove.key**: Removes the color key from the plot.
- **rotate.key**: Rotate color scale horizontally
- **title**: Title for plot
- **cex.col**: Controls size of column labels (cells)
- **cex.row**: Controls size of row labels (genes)
DoKMeans

Perform k-means clustering on both genes and single cells

Usage

DoKMeans(object, genes.use = NULL, k.genes = NULL, k.cells = 0, k.seed = 1, do.plot = FALSE, data.cut = 2.5, k.cols = PurpleAndYellow(), set.ident = TRUE, do.constrained = FALSE, assay.type = "RNA", ...

Arguments

- object: Seurat object
- genes.use: Genes to use for clustering
- k.genes: K value to use for clustering genes
- k.cells: K value to use for clustering cells (default is NULL, cells are not clustered)
- k.seed: Random seed
- do.plot: Draw heatmap of clustered genes/cells (default is FALSE).
- data.cut: Clip all z-scores to have an absolute value below this. Reduces the effect of huge outliers in the data.
- k.cols: Color palette for heatmap
DotPlot

set.ident  If clustering cells (so k.cells > 0), set the cell identity class to its K-means cluster (default is TRUE)
do.constrained  FALSE by default. If TRUE, use the constrained K-means function implemented in the tclust package.
assay.type  Type of data to normalize for (default is RNA), but can be changed for multimodal analyses.
...  Additional parameters passed to kmeans (or tkmeans)

Details

K-means and heatmap are calculated on object@scale.data

Value

Seurat object where the k-means results for genes is stored in object@kmeans.obj[[1]], and the k-means results for cells is stored in object@kmeans.col[[1]]. The cluster for each cell is stored in object@meta.data[,"kmeans.ident"] and also object@ident (if set.ident=TRUE)

Examples

pbmc_small
# Cluster on genes only
pbmc_small <- DoKMeans(pbmc_small, k.genes = 3)
# Cluster on genes and cell
pbmc_small <- DoKMeans(pbmc_small, k.genes = 3, k.cells = 3)

DotPlot

Description

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

Usage

DotPlot(object, genes.plot, cols.use = c("lightgrey", "blue"),
        col.min = -2.5, col.max = 2.5, dot.min = 0, dot.scale = 6,
        scale.by = "radius", scale.min = NA, scale.max = NA, group.by,
        plot.legend = FALSE, do.return = FALSE, x.lab.rot = FALSE)
Arguments

- **object**: Seurat object
- **genes.plot**: Input vector of genes
- **cols.use**: Colors to plot, can pass a single character giving the name of a palette from `RColorBrewer::brewer.pal.info`
- **col.min**: Minimum scaled average expression threshold (everything smaller will be set to this)
- **col.max**: Maximum scaled average expression threshold (everything larger will be set to this)
- **dot.min**: The fraction of cells at which to draw the smallest dot (default is 0). All cell groups with less than this expressing the given gene will have no dot drawn.
- **dot.scale**: Scale the size of the points, similar to `cex`
- **scale.by**: Scale the size of the points by 'size' or 'radius'
- **scale.min**: Set lower limit for scaling, use NA for default
- **scale.max**: Set upper limit for scaling, use NA for default
- **group.by**: Factor to group the cells by
- **plot.legend**: plots the legends
- **do.return**: Return `ggplot2` object
- **x.lab.rot**: Rotate x-axis labels

Value

default, no return, only graphical output. If `do.return=TRUE`, returns a `ggplot2` object

See Also

`RColorBrewer::brewer.pal.info`

Examples

```r
cd_genes <- c("CD247", "CD3E", "CD9")
DotPlot(object = pbmc_small, genes.plot = cd_genes)
```

---

**DotPlotOld**

*Old Dot plot visualization (pre-ggplot implementation)* Intuitive way of visualizing how gene expression changes across different identity classes (clusters). The size of the dot encodes the percentage of cells within a class, while the color encodes the AverageExpression level of ‘expressing’ cells (green is high).
Description

Old Dot plot visualization (pre-ggplot implementation) Intuitive way of visualizing how gene expression changes across different identity classes (clusters). The size of the dot encodes the percentage of cells within a class, while the color encodes the AverageExpression level of 'expressing' cells (green is high).

Usage

DotPlotOld(object, genes.plot, cex.use = 2, cols.use = NULL, thresh.col = 2.5, dot.min = 0.05, group.by = NULL)

Arguments

- object: Seurat object
- genes.plot: Input vector of genes
- cex.use: Scaling factor for the dots (scales all dot sizes)
- cols.use: colors to plot
- thresh.col: The raw data value which corresponds to a red dot (lowest expression)
- dot.min: The fraction of cells at which to draw the smallest dot (default is 0.05)
- group.by: Factor to group the cells by

Value

Only graphical output

Examples

```R
cd_genes <- c("CD247", "CD3E", "CD9")
DotPlotOld(object = pbmc_small, genes.plot = cd_genes)
```

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
ExpVar

**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)

**Arguments**

x  
A vector of values

**Value**

Returns the standard deviation in log-space

**Examples**

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

---

ExpVar

*Calculate the variance of logged values*

**Description**

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

**Usage**

ExpVar(x)

**Arguments**

x  
A vector of values
ExtractField

Value

Returns the variance in log-space

Examples

\( \text{ExpVar}(x = c(1, 2, 3)) \)

ExtractField

Extract delimiter information from a string.

Description

Parses a string (usually a cell name) and extracts fields based on a delimiter

Usage

\( \text{ExtractField}(\text{string}, \text{field} = 1, \text{delim} = "_") \)

Arguments

- **string**: String to parse.
- **field**: Integer(s) indicating which field(s) to extract. Can be a vector multiple numbers.
- **delim**: Delimiter to use, set to underscore by default.

Value

A new string, that parses out the requested fields, and (if multiple), rejoins them with the same delimiter

Examples

\( \text{ExtractField}(\text{string} = 'Hello World', \text{field} = 1, \text{delim} = '_') \)
**FastWhichCells**

*Identify cells matching certain criteria (limited to character values)*

**Description**

FastWhichCells Identify cells matching certain criteria (limited to character values)

**Usage**

```r
FastWhichCells(object, group.by, subset.value, invert = FALSE)
```

**Arguments**

- `object`: Seurat object
- `group.by`: Group cells in different ways (for example, `orig.ident`). Should be a column name in `object@meta.data`
- `subset.value`: Return cells matching this value
- `invert`: invert cells to return. FALSE by default

**Examples**

```r
FastWhichCells(object = pbmc_small, group.by = 'res.1', subset.value = 1)
```

---

**FeatureHeatmap**

*Visualization of multiple features*

**Description**

Similar to FeaturePlot, however, also splits the plot by visualizing each identity class separately.

**Usage**

```r
FeatureHeatmap(object, features.plot, dim.1 = 1, dim.2 = 2, 
idents.use = NULL, pt.size = 2, cols.use = c("grey", "red"), 
pch.use = 16, reduction.use = "tsne", group.by = NULL, 
data.use = "data", sep.scale = FALSE, do.return = FALSE, 
min.exp = -Inf, max.exp = Inf, rotate.key = FALSE, plot.horiz = FALSE, 
key.position = "right")
```
FeatureHeatmap

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>features.plot</td>
<td>Vector of features to plot</td>
</tr>
<tr>
<td>dim.1</td>
<td>Dimension for x-axis (default 1)</td>
</tr>
<tr>
<td>dim.2</td>
<td>Dimension for y-axis (default 2)</td>
</tr>
<tr>
<td>idents.use</td>
<td>Which identity classes to display (default is all identity classes)</td>
</tr>
<tr>
<td>pt.size</td>
<td>Adjust point size for plotting</td>
</tr>
<tr>
<td>cols.use</td>
<td>Ordered vector of colors to use for plotting. Default is heat.colors(10).</td>
</tr>
<tr>
<td>pch.use</td>
<td>Pch for plotting</td>
</tr>
<tr>
<td>reduction.use</td>
<td>Which dimensionality reduction to use. Default is &quot;tsne&quot;, can also be &quot;pca&quot;, or &quot;ica&quot;, assuming these are precomputed.</td>
</tr>
<tr>
<td>group.by</td>
<td>Group cells in different ways (for example, orig.ident)</td>
</tr>
<tr>
<td>data.use</td>
<td>Dataset to use for plotting, choose from 'data', 'scale.data', or 'imputed'</td>
</tr>
<tr>
<td>sep.scale</td>
<td>Scale each group separately. Default is FALSE.</td>
</tr>
<tr>
<td>do.return</td>
<td>Return the ggplot2 object</td>
</tr>
<tr>
<td>min.exp</td>
<td>Min cutoff for scaled expression value, supports quantiles in the form of 'q#' (see FeaturePlot)</td>
</tr>
<tr>
<td>max.exp</td>
<td>Max cutoff for scaled expression value, supports quantiles in the form of 'q#' (see FeaturePlot)</td>
</tr>
<tr>
<td>rotate.key</td>
<td>rotate the legend</td>
</tr>
<tr>
<td>plot.horiz</td>
<td>rotate the plot such that the features are columns, groups are the rows</td>
</tr>
<tr>
<td>key.position</td>
<td>position of the legend (&quot;top&quot;, &quot;right&quot;, &quot;bottom&quot;, &quot;left&quot;)</td>
</tr>
</tbody>
</table>

Details

Particularly useful for seeing if the same groups of cells co-exhibit a common feature (i.e. co-express a gene), even within an identity class. Best understood by example.

Value

No return value, only a graphical output

See Also

FeaturePlot

Examples

```r
pbmc_small
FeatureHeatmap(object = pbmc_small, features.plot = "PC1")
```
### FeatureLocator

#### Description

Select points on a scatterplot and get information about them

#### Usage

```r
FeatureLocator(plot, data.plot, ...)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot</code></td>
<td>A ggplot2 plot</td>
</tr>
<tr>
<td><code>data.plot</code></td>
<td>The original data that went into the ggplot2 plot</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Extra parameters, such as dark.theme, recolor, or smooth for using a dark theme, recoloring based on selected cells, or using a smooth scatterplot, respectively</td>
</tr>
</tbody>
</table>

#### Value

The names of the points selected

#### See Also

- `locator`
- `ggplotR::ggplot_build`

#### Examples

```r
## Not run:
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'))
FeatureLocator(plot = p, data.plot = df)

## End(Not run)
```
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.plot, min.cutoff = NA, max.cutoff = NA, 
  dim.1 = 1, dim.2 = 2, cells.use = NULL, pt.size = 1, 
  cols.use = c(“yellow", “red"), pch.use = 16, overlay = FALSE, 
  do.hover = FALSE, data.hover = "ident", do.identify = FALSE, 
  reduction.use = "tsne", use.imputed = FALSE, nCol = NULL, 
  no.axes = FALSE, no.legend = TRUE, coord.fixed = FALSE, 
  dark.theme = FALSE, do.return = FALSE, vector.friendly = FALSE, 
  png.file = NULL, png.arguments = c(10, 10, 100))

Arguments

object
  Seurat object

features.plot
  Vector of features to plot

min.cutoff
  Vector of minimum cutoff values for each feature, may specify quantile in the form of ‘q##’ where ‘##’ is the quantile (eg, 1, 10)

max.cutoff
  Vector of maximum cutoff values for each feature, may specify quantile in the form of ‘q##’ where ‘##’ is the quantile (eg, 1, 10)

dim.1
  Dimension for x-axis (default 1)

dim.2
  Dimension for y-axis (default 2)

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

pt.size
  Adjust point size for plotting

cols.use
  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.

pch.use
  Pch for plotting

overlay
  Plot two features overlayed one on top of the other

do.hover
  Enable hovering over points to view information

data.hover
  Data to add to the hover, pass a character vector of features to add. Defaults to cell name and identity. Pass ‘NULL’ to remove extra data.

do.identify
  Opens a locator session to identify clusters of cells

reduction.use
  Which dimensionality reduction to use. Default is "tsne", can also be "pca", or "ica", assuming these are precomputed.
FetchData

use.imputed  Use imputed values for gene expression (default is FALSE)
ncol        Number of columns to use when plotting multiple features.
nom.axes    Remove axis labels
no.legend   Remove legend from the graph. Default is TRUE.
coord.fixed Use a fixed scale coordinate system (for spatial coordinates). Default is FALSE.
dark.theme  Plot in a dark theme
do.return   return the ggplot2 object
vector.friendly FALSE by default. If TRUE, points are flattened into a PNG, while axes/labels retain full vector resolution. Useful for producing AI-friendly plots with large numbers of cells.
png.file    Use specific name for temporary png file
png.arguments Set width, height, and DPI for ggsave

Value
No return value, only a graphical output

Examples
FeaturePlot(object = pbmc_small, features.plot = 'PC1')
Value

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

Examples

```r
pc1 <- fetchdata(object = pbmc_small, vars.all = 'PC1')
head(x = pc1)
```

Description

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

Usage

```r
FilterCells(object, subset.names, low.thresholds, high.thresholds, cells.use = NULL)
```

Arguments

- `object`: Seurat object
- `subset.names`: Parameters to subset on. Eg, the name of a gene, PC1, a column name in object@meta.data, etc. Any argument that can be retrieved using `FetchData`
- `low.thresholds`: Low cutoffs for the parameters (default is -Inf)
- `high.thresholds`: High cutoffs for the parameters (default is Inf)
- `cells.use`: A vector of cell names to use as a subset

Value

Returns a Seurat object containing only the relevant subset of cells

Examples

```r
head(x = fetchdata(object = pbmc_small, vars.all = 'LTB'))
pbmc_filtered <- FilterCells(
    object = pbmc_small,
    subset.names = 'LTB',
    high.thresholds = 6
)
head(x = fetchdata(object = pbmc_filtered, vars.all = 'LTB'))
```
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, genes.use = NULL, logfc.threshold = 0.25,
    test.use = "wilcox", min.pct = 0.1, min.diff.pct = -Inf,
    print.bar = TRUE, only.pos = FALSE, max.cells.per.ident = Inf,
    return.thresh = 0.01, do.print = FALSE, random.seed = 1,
    min.cells.gene = 3, min.cells.group = 3, latent.vars = NULL,
    assay.type = "RNA", ...)
```

**Arguments**

- `object` Seurat object
- `genes.use` Genes to test. Default is to use all genes
- `logfc.threshold` Limit testing to genes which show, on average, at least X-fold difference (log-scale) between the two groups of cells. Default is 0.25. Increasing logfc.threshold speeds up the function, but can miss weaker signals.
- `test.use` Denotes which test to use. Available options are:
  - "wilcox": Wilcoxon rank sum test (default)
  - "bimod": Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
  - "roc": Standard AUC classifier
  - "t": Student's t-test
  - "tobit": Tobit-test for differential gene expression (Trapnell et al., Nature Biotech, 2014)
  - "poisson": Likelihood ratio test assuming an underlying poisson distribution. Use only for UMI-based datasets
  - "negbinom": Likelihood ratio test assuming an underlying negative binomial distribution. Use only for UMI-based datasets
  - "MAST": GLM-framework that treats cellular detection rate as a covariate (Finak et al, Genome Biology, 2015)
  - "DESeq2": DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014)
- `min.pct` only test genes that are detected in a minimum fraction of min.pct cells in either of the two populations. Meant to speed up the function by not testing genes that are very infrequently expressed. Default is 0.1
**FindAllMarkersNode**

- **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.
- **print.bar**: Print a progress bar once expression testing begins (uses pbapply to do this).
- **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.
- **return.thresh**: Only return markers that have a p-value < return.thresh, or a power > return.thresh (if the test is ROC).
- **do.print**: FALSE by default. If TRUE, outputs updates on progress.
- **random.seed**: Random seed for downsampling.
- **min.cells.gene**: Minimum number of cells expressing the gene 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.
- **latent.vars**: Remove the effects of these variables, used only when test.use is one of 'neg-binom', 'poisson', or 'MAST'.
- **assay.type**: Type of assay to perform DE for (default is RNA).
- **...**: Additional parameters to pass to specific DE functions.

**Value**

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

**Examples**

```r
call_markers <- FindAllMarkers(object = pbmc_small)
head(x = call_markers)
```

**Description**

This function finds markers for all splits at or below the specified node.

**Usage**

```r
FindAllMarkersNode(object, node = NULL, genes.use = NULL,
                   logfc.threshold = 0.25, test.use = "wilcox", min.pct = 0.1,
                   min.diff.pct = 0.05, print.bar = TRUE, only.pos = FALSE,
                   max.cells.per.ident = Inf, return.thresh = 0.01, do.print = FALSE,
                   random.seed = 1, min.cells.gene = 3, min.cells.group = 3,
                   assay.type = "RNA", ...)
```
Arguments

object
Seurat object. Must have object@cluster.tree slot filled. Use BuildClusterTree() if not.

node
Node from which to start identifying split markers, default is top node.

genes.use
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.

test.use
Denotes which test to use. Seurat currently implements "bimod" (likelihood-ratio test for single cell gene expression, McDavid et al., Bioinformatics, 2013, default), "roc" (standard AUC classifier), "t" (Students t-test), and "tobit" (Tobit-test for differential gene expression, as in Trapnell et al., Nature Biotech, 2014), 'poisson', and 'negbinom'. The latter two options should only be used on UMI datasets, and assume an underlying poisson or negative-binomial distribution.

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

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

print.bar
Print a progress bar once expression testing begins (uses pbapply to do this)

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.

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

do.print
Print status updates

random.seed
Random seed for downsampling

min.cells.gene
Minimum number of cells expressing the gene 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

assay.type
Type of assay to fetch data for (default is RNA)

Additional parameters to pass to specific DE functions

Value

Returns a dataframe with a ranked list of putative markers for each node and associated statistics

Examples

pbmc_small

FindAllMarkersNode(pbmcsmall)
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, genes.use = NULL, reduction.type = "pca",
dims.use = NULL, k.param = 30, plot.SNN = FALSE, prune.SNN = 1/15,
print.output = TRUE, distance.matrix = NULL, save.SNN = FALSE,
reuse.SNN = FALSE, force.recalc = FALSE, nn.eps = 0,
modularity.fxn = 1, resolution = 0.8, algorithm = 1, n.start = 100,
n.iter = 10, random.seed = 0, temp.file.location = NULL,
edge.file.name = NULL)

Arguments

object Seurat object
genes.use A vector of gene names to use in construction of SNN graph if building directly based on expression data rather than a dimensionally reduced representation (i.e. PCs).
reduction.type Name of dimensional reduction technique to use in construction of SNN graph. (e.g. "pca", "ica")
dims.use A vector of the dimensions to use in construction of the SNN graph (e.g. To use the first 10 PCs, pass 1:10)
k.param Defines k for the k-nearest neighbor algorithm
plot.SNN Plot the SNN 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).
print.output Whether or not to print output to the console
distance.matrix Build SNN from distance matrix (experimental)
save.SNN Saves the SNN matrix associated with the calculation in object@snn
reuse.SNN Force utilization of stored SNN. If none store, this will throw an error.
force.recalc Force recalculation of SNN.
nn.eps Error bound when performing nearest neighbor search using RANN; default of 0.0 implies exact nearest neighbor search

modularity.fn Modularity function (1 = standard; 2 = alternative).

resolution Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.

algorithm Algorithm for modularity optimization (1 = original Louvain algorithm; 2 = Louvain algorithm with multilevel refinement; 3 = SLM algorithm).

n.start Number of random starts.

n.iter Maximal number of iterations per random start.

random.seed Seed of the random number generator.

temp.file.location Directory where intermediate files will be written. Specify the ABSOLUTE path.

edge.file.name Edge file to use as input for modularity optimizer jar.

Value

Returns a Seurat object and optionally the SNN matrix, object@ident has been updated with new cluster info

Examples

## Not run:

pbmc_small

pbmc_small <- FindClusters(
  object = pbmc_small,
  reduction.type = "pca",
  dims.use = 1:10,
  save.SNN = TRUE
)

# To explore a range of clustering options, pass a vector of values to the resolution parameter

pbmc_small <- FindClusters(
  object = pbmc_small,
  reduction.type = "pca",
  resolution = c(0.4, 0.8, 1.2),
  dims.use = 1:10,
  save.SNN = TRUE
)

## End(Not run)
FindConservedMarkers

Finds markers that are conserved between the two groups

Description

Finds markers that are conserved between the two groups

Usage

FindConservedMarkers(object, ident.1, ident.2 = NULL, grouping.var,
assay.type = "RNA", meta.method = minump, ...)

Arguments

object Seurat object
ident.1 Identity class to define markers for
ident.2 A second identity class for comparison. If NULL (default) - use all other cells
       for comparison.
grouping.var grouping variable
assay.type Type of assay to fetch data for (default is RNA)
meta.method method for combining p-values. Should be a function from the metap package
              (NOTE: pass the function, not a string)
... parameters to pass to FindMarkers

Value

Matrix containing a ranked list of putative conserved markers, and associated statistics (p-values
within each group and a combined p-value (such as Fishers combined p-value or others from the
MetaDE package), percentage of cells expressing the marker, average differences)

Examples

## Not run:
pbmc_small
# Create a simulated grouping variable
pbmc_small@meta.data$groups <- sample(
  x = c("g1", "g2"),
  size = length(x = pbmc_small@cell.names),
  replace = TRUE
)
FindConservedMarkers(pbmc_small, ident.1 = 0, ident.2 = 1, grouping.var = "groups")

## End(Not run)
FindGeneTerms  
*Find gene terms from Enrichr*

**Description**

Fing gene terms from Enrichr and return the XML information

**Usage**

```r
FindGeneTerms(QueryGene = NULL)
```

**Arguments**

- **querygene**  
  A gene to query on Enrichr

**Value**

An XML document with information on the queried gene

---

FindMarkers  
*Gene expression markers of identity classes*

**Description**

Finds markers (differentially expressed genes) for identity classes

**Usage**

```r
FindMarkers(object, ident.1, ident.2 = NULL, genes.use = NULL, logfc.threshold = 0.25, test.use = "wilcox", min.pct = 0.1, min.diff.pct = -Inf, print.bar = TRUE, only.pos = FALSE, max.cells.per.ident = Inf, random.seed = 1, latent.vars = NULL, min.cells.gene = 3, min.cells.group = 3, pseudocount.use = 1, assay.type = "RNA", ...)
```

**Arguments**

- **object**  
  Seurat 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.

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

test.use
Denotes which test to use. Available options are:
- "wilcox": Wilcoxon rank sum test (default)
- "bimod": Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc": Standard AUC classifier
- "t": Student’s t-test
- "tobit": Tobit-test for differential gene expression (Trapnell et al., Nature Biotech, 2014)
- "poisson": Likelihood ratio test assuming an underlying poisson distribution. Use only for UMI-based datasets
- "negbinom": Likelihood ratio test assuming an underlying negative binomial distribution. Use only for UMI-based datasets
- "MAST": GLM-framework that treats cellular detection rate as a covariate (Finak et al, Genome Biology, 2015)
- "DESeq2": DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014)

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

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

print.bar
Print a progress bar once expression testing begins (uses pbapply to do this)

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 'negbinom', 'poisson', or 'MAST'

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

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

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

assay.type
Type of assay to fetch data for (default is RNA)

... Additional parameters to pass to specific DE functions
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

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

See Also

MASTDETest, and DESeq2DETest for more information on these methods

NegBinomDETest

Examples

markers <- FindMarkers(object = pbmc_small, ident.1 = 3)
head(markers)

FindMarkersNode  Gene expression markers of identity classes defined by a phylogenetic clade

Description

Finds markers (differentially expressed genes) based on a branching point (node) in the phylogenetic tree. Markers that define clusters in the left branch are positive markers. Markers that define the right branch are negative markers.

Usage

FindMarkersNode(object, node, tree.use = NULL, genes.use = NULL, logfc.threshold = 0.25, test.use = "wilcox", assay.type = "RNA", ...)

Arguments

  object  Seurat object
  node    The node in the phylogenetic tree to use as a branch point
  tree.use Can optionally pass the tree to be used. Default uses the tree in object@cluster.tree
  genes.use Genes to test. Default is to use all genes
logfc.threshold

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

test.use

Denotes which test to use. Available options are:

- "wilcox" : Wilcoxon rank sum test (default)
- "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc" : Standard AUC classifier
- "t" : Student’s t-test
- "tobit" : Tobit-test for differential gene expression (Trapnell et al., Nature Biotech, 2014)
- "poisson" : Likelihood ratio test assuming an underlying poisson distribution. Use only for UMI-based datasets
- "negbinom" : Likelihood ratio test assuming an underlying negative binomial distribution. Use only for UMI-based datasets
- "MAST" : GLM-framework that treats cellular detection rate as a covariate (Finak et al, Genome Biology, 2015)
- "DESeq2" : DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014)

assay.type

Type of assay to fetch data for (default is RNA)

Additional arguments passed to FindMarkers

Value

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

Examples

FindMarkersNode(pbmcc_small, 5)
FindVariableGenes

Usage

FindVariableGenes(object, mean.function = ExpMean,
    dispersion.function = LogVMR, do.plot = TRUE, set.var.genes = TRUE,
    x.low.cutoff = 0.1, x.high.cutoff = 8, y.cutoff = 1,
    y.high.cutoff = Inf, num.bin = 20, binning.method = "equal_width",
    selection.method = "mean.var.plot", top.genes = 1000, do.recalc = TRUE,
    sort.results = TRUE, do.cpp = TRUE, display.progress = TRUE, ...)  

Arguments

object Seurat object
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/
do.plot Plot the average/dispersion relationship
set.var.genes Set object@var.genes to the identified variable genes (default is TRUE)
x.low.cutoff Bottom cutoff on x-axis for identifying variable genes
x.high.cutoff Top cutoff on x-axis for identifying variable genes
y.cutoff Bottom cutoff on y-axis for identifying variable genes
y.high.cutoff Top cutoff on y-axis for identifying variable genes
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 genes (can increase
        statistical power to detect overdispersed genes at high expression values, at
        the cost of reduced resolution along the x-axis)
selection.method Specifies how to select the genes to store in @var.genes.
    • mean.var.plot: Default method, placing cutoffs on the mean variability plot
    • dispersion: Choose the top.genes with the highest dispersion
top.genes Selects the genes with the highest value according to the selection method.
do.recalc TRUE by default. If FALSE, plots and selects variable genes without recalculating statistics for each gene.
sort.results If TRUE (by default), sort results in object@hvg.info in decreasing order of
dispersion
do.cpp Run c++ version of mean.function and dispersion.function if they exist.
display.progress show progress bar for calculations
... Extra parameters to VariableGenePlot
Details

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 genes 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.

Value

Returns a Seurat object, placing variable genes in object@var.genes. The result of all analysis is stored in object@hvg.info

See Also

VariableGenePlot

Examples

```r
pbmc_small <- FindVariableGenes(object = pbmc_small, do.plot = FALSE)
pbmc_small@var.genes
```

---

**FitGeneK**

*Build mixture models of gene expression*

Description

Models the imputed gene expression values as a mixture of gaussian distributions. For a two-state model, estimates the probability that a given cell is in the 'on' or 'off' state for any gene. Followed by a greedy k-means step where cells are allowed to flip states based on the overall structure of the data (see Manuscript for details)

Usage

```r
FitGeneK(object, gene, do.k = 2, num.iter = 1, do.plot = FALSE, 
genes.use = NULL, start.pct = NULL)
```

Arguments

- `object`: Seurat object
- `gene`: Gene to fit
- `do.k`: Number of modes for the mixture model (default is 2)
- `num.iter`: Number of 'greedy k-means' iterations (default is 1)
- `do.plot`: Plot mixture model results
- `genes.use`: Genes to use in the greedy k-means step (See manuscript for details)
- `start.pct`: Initial estimates of the percentage of cells in the 'on' state (usually estimated from the in situ map)
Value

A Seurat object, where the posterior of each cell being in the 'on' or 'off' state for each gene is stored in object@spatial@mix.probs

Examples

## Not run:

# Note that the PBMC test example object does not contain spatially restricted
# examples below are only demonstrate code

pmbc_small <- FitGeneK(object = pmbc_small, gene = "MS4A1")

## End(Not run)

---

GenePlot  
Scatter plot of single cell data

Description

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

Usage

GenePlot(object, gene1, gene2, cell.ids = NULL, col.use = NULL, pch.use = 16, cex.use = 1.5, use.imputed = FALSE, use.scaled = FALSE, use.raw = FALSE, do.hover = FALSE, data.hover = "ident", do.identify = FALSE, dark.theme = FALSE, do.spline = FALSE, spline.span = 0.75, ...)

Arguments

object  
Seurat object

gene1  
First feature to plot. Typically gene expression but can also be metrics, PC scores, etc. - anything that can be retrieved with FetchData

gene2  
Second feature to plot.

cell.ids  
Cells to include on the scatter plot.

col.use  
Colors to use for identity class plotting.

cpch.use  
Pch argument for plotting

cex.use  
Cex argument for plotting

use.imputed  
Use imputed values for gene expression (Default is FALSE)

use.scaled  
Use scaled data

use.raw  
Use raw data
GenesInCluster

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>do.hover</td>
<td>Enable hovering over points to view information</td>
</tr>
<tr>
<td>data.hover</td>
<td>Data to add to the hover, pass a character vector of features to add. Defaults to cell name and ident. Pass 'NULL' to clear extra information.</td>
</tr>
<tr>
<td>do.identify</td>
<td>Opens a locator session to identify clusters of cells.</td>
</tr>
<tr>
<td>dark.theme</td>
<td>Use a dark theme for the plot</td>
</tr>
<tr>
<td>do.spline</td>
<td>Add a spline (currently hardwired to df=4, to be improved)</td>
</tr>
<tr>
<td>spline.span</td>
<td>Spline span in loess function call</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to be passed to plot.</td>
</tr>
</tbody>
</table>

**Value**

No return, only graphical output

**Examples**

```r
geneplot(object = pbmc_small, gene1 = 'CD9', gene2 = 'CD3E')
```

---

**Description**

After k-means analysis, previously run with DoKMeans, returns a set of genes associated with each cluster.

**Usage**

```r
GenesInCluster(object, cluster.num, max.genes = 1e+06)
```

**Arguments**

- **object**: Seurat object. Assumes DoKMeans has already been run
- **cluster.num**: K-means cluster(s) to return genes for
- **max.genes**: Max number of genes to return

**Value**

A vector of genes who are members in the cluster.num k-means cluster(s)

**Examples**

```r
pbmc_small <- DoKMeans(object = pbmc_small, k.genes = 3)
pbmc_small <- GenesInCluster(object = pbmc_small, cluster.num = 1)
```
GetAssayData  

*Accessor function for multimodal data*

**Description**

Pull information for specified stored dimensional reduction analysis

**Usage**

```
GetAssayData(object, assay.type = "RNA", slot = "data")
```

**Arguments**

- `object` Seurat object
- `assay.type` Type of assay to fetch data for (default is RNA)
- `slot` Specific information to pull (i.e. raw.data, data, scale.data,...). Default is data

**Value**

Returns assay data

**Examples**

```r
# Simulate CITE-Seq results
df <- t(x = data.frame(  
  x = round(x = rnorm(n = 80, mean = 20, sd = 2)),  
  y = round(x = rbinom(n = 80, size = 100, prob = 0.2)),  
  row.names = pbmc_small@cell.names  
))
pbmc_small <- SetAssayData(  
  object = pbmc_small,  
  assay.type = 'CITE',  
  new.data = df,  
  slot = 'data'  
)
GetAssayData(object = pbmc_small, assay.type = 'CITE', slot = 'data')
```

GetCellEmbeddings  

*Dimensional Reduction Cell Embeddings Accessor Function*

**Description**

Pull cell embeddings matrix for specified stored dimensional reduction analysis
GetCentroids

Usage

GetCellEmbeddings(object, reduction.type = "pca", dims.use = NULL, cells.use = NULL)

Arguments

- object: Seurat object
- reduction.type: Type of dimensional reduction to fetch (default is PCA)
- dims.use: Dimensions to include (default is all stored dims)
- cells.use: Cells to include (default is all cells)

Value

Cell embedding matrix for given reduction, cells, and dimensions

Examples

pbmc_small

# Examine the head of the first 5 PC cell embeddings
head(GetCellEmbeddings(object = pbmc_small, reduction.type = "pca", dims.use = 1:5))

GetCellEmbeddings

Get cell centroids

Description

Calculate the spatial mapping centroids for each cell, based on previously calculated mapping probabilities for each bin.

Usage

GetCentroids(object, cells.use = NULL, get.exact = TRUE)

Arguments

- object: Seurat object
- cells.use: Cells to calculate centroids for (default is all cells)
- get.exact: Get exact centroid (Default is TRUE). If FALSE, identify the single closest bin.

Details

Currently, Seurat assumes that the tissue of interest has an 8x8 bin structure. This will be broadened in a future release.
GetClusters

**Value**

Data frame containing the x and y coordinates for each cell centroid.

**Examples**

```r
## Not run:
# Note that the PBMC test example object does not contain spatially restricted
# examples below are only demonstrate code
pbmc_small <- GetCentroids(pbmc_small, cells.use=pbmc_small@cell.names)

## End(Not run)
```

---

### Description

Retrieve cluster IDs as a dataframe. First column will be the cell name, second column will be the current cluster identity (pulled from object@ident).

### Usage

```r
GetClusters(object)
```

### Arguments

- `object` Seurat object with cluster assignments

### Value

Returns a dataframe with cell names and cluster assignments

### Examples

```r
pbmc_small
clusters <- GetClusters(object = pbmc_small)
head(clusters)
```
GetDimReduction

**Dimensional Reduction Accessor Function**

### Description

General accessor function for dimensional reduction objects. Pulls slot contents for specified stored dimensional reduction analysis.

### Usage

```r
GetDimReduction(object, reduction.type = "pca", slot = "gene.loadings")
```

### Arguments

- **object**: Seurat object
- **reduction.type**: Type of dimensional reduction to fetch (default is PCA)
- **slot**: Specific information to pull (must be one of the following: "cell.embeddings", "gene.loadings", "gene.loadings.full", "sdev", "key", "misc")

### Value

Returns specified slot results from given reduction technique

### Examples

```r
# pbmc_small
pbmc_small
# Get the PCA cell embeddings and print the top left corner
GetDimReduction(object = pbmc_small, reduction.type = "pca",
                 slot = "cell.embeddings")[1:5, 1:5]
# Get the standard deviation of each PC
GetDimReduction(object = pbmc_small, reduction.type = "pca", slot = "sdev")
```

---

GetGeneLoadings

**Dimensional Reduction Gene Loadings Accessor Function**

### Description

Pull gene loadings matrix for specified stored dimensional reduction analysis.

### Usage

```r
GetGeneLoadings(object, reduction.type = "pca", dims.use = NULL,
                 genes.use = NULL, use.full = FALSE)
```
Arguments

object Seurat object
reduction.type Type of dimensional reduction to fetch (default is PCA)
dims.use Dimensions to include (default is all stored dims)
genuses.use Genes to include (default is all genes)
use.full Return projected gene loadings (default is FALSE)

Value

Gene loading matrix for given reduction, cells, and genes

Examples

```r
pbmc_small
# Examine the head of the first 5 PC gene loadings
head(GetGeneLoadings(object = pbmc_small, reduction.type = "pca", dims.use = 1:5))
```

GetIdent

Get identity of cells

Description

Get identity of cells

Usage

GetIdent(object, uniq = TRUE, cells.use = NULL)

Arguments

object Seurat object
uniq logic, indicating whether to return unique ident values or ident of all cells
cells.use A vector of cell names. If specified, only the identity of these cells will be returned.

Value

Return the cell identities of this Seurat object

Examples

```r
GetIdent(pbm...
```
### HoverLocator

**Description**

Get quick information from a scatterplot by hovering over points

**Usage**

```r
HoverLocator(plot, data.plot, features.info = NULL, dark.theme = FALSE, ...)
```

**Arguments**

- `plot`: A ggplot2 plot
- `data.plot`: The oridinal data that went into the ggplot2 plot
- `features.info`: An optional dataframe or matrix of extra information to be displayed on hover
- `dark.theme`: Plot using a dark theme?
- `...`: Extra parameters to be passed to plotly::layout

**See Also**

- `plotly::layout`
- `ggplot2::ggplot_build`

**Examples**

```r
## Not run:
df <- data.frame(x = rnorm(n = 1000, mean = 20, sd = 2), y = rbinom(n = 1000, size = 100, prob = 0.2))
p <- ggplot(data = df, mapping = aes(x = x, y = y)) + geom_point(mapping = aes(color = 'red'))
HoverLocator(plot = p, data.plot = df)

## End(Not run)
```

---

### HTODemux

**Description**

Demultiplex samples based on data from cell 'hashing'

**Usage**

```r
HTODemux(object, assay.type = "HTO", positive_quantile = 0.99,
          init.centers = NULL, cluster_nstarts = 100, k_function = "clara",
          nsamples = 100, print.output = TRUE)
```
Arguments

- **object**: Seurat object. Assumes that the hash tag oligo (HTO) data has been added and normalized.
- **assay.type**: 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_centers**: Initial number of clusters for hashtags. Default is the # of hashtag oligo names + 1 (to account for negatives)
- **cluster_nstarts**: nstarts value for k-means clustering (for k_function = "kmeans"). 100 by default
- **k_function**: 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 k_function = "clara"
- **print.output**: 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
- **hto_classification**: Classification result, with doublets/multiplets named by the top two highest hashtags
- **hto_classification_global**: Global classification result (singlet, doublet or negative)
- **hash_ID**: Classification result where doublet IDs are collapsed

Examples

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

## End(Not run)
```
HTOHeatmap

Hashtag oligo heatmap

Description

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

Usage

```r
HTOHeatmap(object, hto.classification = "hto.classification",
            global.classification = "hto.classification_global", assay.type = "HTO",
            num.cells = 5000, singlet.names = NULL, ...)
```

Arguments

- **object** Seurat object. Assumes that the hash tag oligo (HTO) data has been added and normalized, and demultiplexing has been run with HTODemux().
- **hto.classification** The naming for object@meta.data slot with classification result from HTODemux().
- **global.classification** The slot for object@meta.data slot specifying a cell as singlet/doublet/negative.
- **assay.type** Hashtag assay name.
- **num.cells** 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.
- **...** Additional arguments for DoHeatmap().

Value

Returns a ggplot2 plot object.

Examples

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

## End(Not run)
```
ICAEmbed

ICA Cell Embeddings Accessor Function

Description

Pull ICA cell embeddings matrix

Usage

ICAEmbed(object, dims.use = NULL, cells.use = NULL)

Arguments

object Seurat object
dims.use Dimensions to include (default is all stored dims)
cells.use Cells to include (default is all cells)

Value

ICA cell embeddings matrix for given cells and ICs

Examples

pbmc_small
pbmc_small <- RunICA(pbmc_small, ics.compute = 10, ics.print = 0)
head(ICAEmbed(pbmc_small))
# Optionally, you can specify subsets of dims or cells to use
ICAEmbed(pbmc_small, dims.use = 1:5, cells.use = pbmc_small@cell.names[1:5])

ICALoad

ICA Gene Loadings Accessor Function

Description

Pull the ICA gene loadings matrix

Usage

ICALoad(object, dims.use = NULL, genes.use = NULL, use.full = FALSE)

Arguments

object Seurat object
dims.use Dimensions to include (default is all stored dims)
genes.use Genes to include (default is all)
use.full Return projected gene loadings (default is FALSE)
ICAPlot

Value

ICA gene loading matrix for given genes and ICs

Examples

pbmc_small <- RunICA(pbmc_small, ics.compute = 10, ics.print = 0)
head(ICALoad(pbmc_small))
# Optionally, you can specify subsets of dims or cells to use
ICALoad(pbmc_small, dims.use = 1:5, genes.use = pbmc_small@var.genes[1:5])

ICAPlot

Plot ICA map

Description

Graphs the output of a ICA analysis. Cells are colored by their identity class.

Usage

ICAPlot(object, ...)

Arguments

object

Seurat object

...  

Additional parameters to DimPlot, for example, which dimensions to plot.

Details

This function is a wrapper for DimPlot. See ?DimPlot for a full list of possible arguments which can be passed in here.

Examples

pbmc_small <- RunICA(object = pbmc_small, ics.compute = 25, print.results = FALSE)
ICAPlot(object = pbmc_small)
ICHeatmap

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

ICHeatmap(object, ic.use = 1, cells.use = NULL, num.genes = 30,
disp.min = -2.5, disp.max = 2.5, do.return = FALSE,
col.use = PurpleAndYellow(), use.scale = TRUE, do.balanced = FALSE,
remove.key = FALSE, label.columns = NULL, ...)

Arguments

object Seurat object
ic.use Components to use
cells.use A list of cells to plot. If numeric, just plots the top cells.
num.genes Number of genes to plot
disp.min Minimum display value (all values below are clipped)
disp.max Maximum display value (all values above are clipped)
do.return If TRUE, returns plot object, otherwise plots plot object
col.use Colors to plot.
use.scale Default is TRUE: plot scaled data. If FALSE, plot raw data on the heatmap.
do.balanced Plot an equal number of genes with both + and - scores.
remove.key Removes the color key from the plot.
label.columns Labels for columns
... Extra parameters passed to DimHeatmap

Value

If do.return==TRUE, a matrix of scaled values which would be passed to heatmap.2. Otherwise, no return value, only a graphical output

Examples

pbmc_small <- RunICA(object = pbmc_small, ics.compute = 25, print.results = FALSE)
ICHeatmap(object = pbmc_small)
ICTopCells

Find cells with highest ICA scores

Description

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

Usage

ICTopCells(object, ic.use = 1, num.cells = NULL, do.balanced = FALSE)

Arguments

- **object**: Seurat object
- **ic.use**: Independent component to use
- **num.cells**: Number of cells to return
- **do.balanced**: Return an equal number of cells with both + and - PC scores.

Value

Returns a vector of cells

Examples

```r
pbmc_small <- RunICA(object = pbmc_small, ics.compute = 10, ics.print = 0)
pbmc_small <- ProjectDim(object = pbmc_small, reduction.type = "ica", do.print = FALSE)
ICTopCells(object = pbmc_small)
# Can specify which dimension and how many cells to return
ICTopCells(object = pbmc_small, ic.use = 2, num.cells = 5)
```

ICTopGenes

Find genes with highest ICA scores

Description

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

Usage

ICTopGenes(object, ic.use = 1, num.genes = 30, use.full = FALSE, do.balanced = FALSE)
Arguments

object Seurat object
ic.use Independent components to use
num.genes Number of genes to return
use.full Use the full ICA (projected ICA), default is FALSE
do.balanced Return an equal number of genes with both + and - IC scores.

Value

Returns a vector of genes

Examples

pbmc_small
pbmc_small <- RunICA(object = pbmc_small, ics.compute = 10, ics.print = 0)
pbmc_small <- ProjectDim(object = pbmc_small, reduction.type = "ica", do.print = FALSE)
ICTopGenes(object = pbmc_small, ic.use = 1)
# After projection:
ICTopGenes(object = pbmc_small, ic.use = 1, use.full = TRUE)
JackStraw

Examples

## Not run:
# Note that the PBMC test example object does not contain spatially restricted
# examples below are only demonstrate code
pmbc_small <- InitialMapping(pbmce_small)

## End(Not run)

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, num.pc = 20, num.replicate = 100, prop.freq = 0.01,
          display.progress = TRUE, do.par = FALSE, num.cores = 1, maxit = 1000)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>num.pc</td>
<td>Number of PCs to compute significance for</td>
</tr>
<tr>
<td>num.replicate</td>
<td>Number of replicate samplings to perform</td>
</tr>
<tr>
<td>prop.freq</td>
<td>Proportion of the data to randomly permute for each replicate</td>
</tr>
<tr>
<td>display.progress</td>
<td>Print progress bar showing the number of replicates that have been processed.</td>
</tr>
<tr>
<td>do.par</td>
<td>use parallel processing for regressing out variables faster. If set to TRUE, will use half of the machines available cores (FALSE by default)</td>
</tr>
<tr>
<td>num.cores</td>
<td>If do.par = TRUE, specify the number of cores to use. Note that for higher number of cores, larger free memory is needed. If num.cores = 1 and do.par = TRUE, num.cores will be set to half of all available cores on the machine.</td>
</tr>
<tr>
<td>maxit</td>
<td>maximum number of iterations to be performed by the irlba function of RunPCA</td>
</tr>
</tbody>
</table>

Value

Returns a Seurat object where object@dr$pca@jackstraw@emperical.p.value represents p-values for each gene in the PCA analysis. If ProjectPCA is subsequently run, object@dr$pca@jackstraw@emperical.p.value.full then represents p-values for all genes.
References

Inspired by Chung et al, Bioinformatics (2014)

Examples

```r
## Not run:
pbmc_small = suppressWarnings(JackStraw(pbmc_small))
head(pbmc_small@dr$pca@jackstraw@emperical.p.value)

## End(Not run)
```

---

### 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

```r
JackStrawPlot(object, PCs = 1:5, nCol = 3, score.thresh = 1e-05,
plot.x.lim = 0.1, plot.y.lim = 0.3)
```

#### Arguments

- `object` Seurat plot
- `PCs` Which PCs to examine
- `nCol` Number of columns
- `score.thresh` Threshold to use for the proportion test of PC significance (see Details)
- `plot.x.lim` X-axis maximum on each QQ plot.
- `plot.y.lim` Y-axis maximum on each QQ plot.

#### Details

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

#### Value

Returns a Seurat object where `object@dr$pca@jackstraw@overall.p.values` represents p-values for each PC and `object@dr$pca@misc$jackstraw.plot` stores the ggplot2 plot.
KClustDimension

Author(s)

Thanks to Omri Wurtzel for integrating with ggplot

Examples

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

---

**KClustDimension**

**Perform spectral k-means clustering on single cells**

Description

Find point clouds of single cells in a low-dimensional space using k-means clustering. Can be useful for smaller datasets, where graph-based clustering can perform poorly.

Usage

```r
KClustDimension(object, dims.use = c(1, 2), reduction.use = "tsne", k.use = 5, set.ident = TRUE, seed.use = 1)
```

Arguments

- `object`: A Seurat object
- `dims.use`: Dimensions to use for clustering
- `reduction.use`: Dimensional Reduction to use for k-means clustering
- `k.use`: Number of clusters
- `set.ident`: Set identity of Seurat object
- `seed.use`: Random seed to use

Value

Object with clustering information

Examples

```r
pbmc_small
# K-means clustering on the first two tSNE dimensions
pbmc_small <- KClustDimension(pbmc_small)
```
**KMeansHeatmap**  

Plot k-means clusters

### Description

Plot k-means clusters

### Usage

```
KMeansHeatmap(object, cells.use = object@cell.names, genes.cluster = NULL,  
max.genes = 1e+06, slim.col.label = TRUE, remove.key = TRUE,  
row.lines = TRUE, ...)
```

### Arguments

- **object**  
  A Seurat object

- **cells.use**  
  Cells to include in the heatmap

- **genes.cluster**  
  Clusters to include in heatmap

- **max.genes**  
  Maximum number of genes to include in the heatmap

- **slim.col.label**  
  Instead of displaying every cell name on the heatmap, display only the identity  
  class name once for each group

- **remove.key**  
  Removes the color key from the plot

- **row.lines**  
  Color separations of clusters

- **...**  
  Extra parameters to DoHeatmap

### See Also

DoHeatmap

### Examples

```
pbmc_small <- DoKMeans(object = pbmc_small, k.genes = 3)  
KMeansHeatmap(object = pbmc_small)
```
LogNormalize

Normalize raw data

Description
Normalize count data per cell and transform to log scale

Usage
LogNormalize(data, scale.factor = 1e4, display.progress = TRUE)

Arguments
- data: Matrix with the raw count data
- scale.factor: Scale the data. Default is 1e4
- display.progress: Print progress

Value
Returns a matrix with the normalize and log transformed data

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

LogVMR

Calculate the variance to mean ratio of logged values

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

Usage
LogVMR(x)

Arguments
- x: A vector of values
**Value**

Returns the VMR in log-space

**Examples**

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

---

**MakeSparse**  
*Make object sparse*

**Description**

Converts stored data matrices to sparse matrices to save space. Converts object@raw.data and object@data to sparse matrices.

**Usage**

`MakeSparse(object)`

**Arguments**

- `object`  
  Seurat object

**Value**

Returns a seurat object with data converted to sparse matrices.

**Examples**

```r
pbmc_raw <- read.table(
  file = system.file('extdata', 'pbmc_raw.txt', package = 'Seurat'),
  as.is = TRUE
)
pbmc_small <- CreateSeuratObject(raw.data = pbmc_raw)
class(x = pbmc_small@raw.data)
pbmc_small <- MakeSparse(object = pbmc_small)
class(x = pbmc_small@raw.data)
```
MarkerTest

ROC-based marker discovery

Description

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.

Usage

```r
MarkerTest(object, cells.1, cells.2, genes.use = NULL, print.bar = TRUE,
assay.type = "RNA")
```

Arguments

- `object`: Seurat object
- `cells.1`: Group 1 cells
- `cells.2`: Group 2 cells
- `genes.use`: Genes to test. Default is to use all genes
- `print.bar`: Print a progress bar once expression testing begins (uses pbapply to do this)
- `assay.type`: Type of assay to fetch data for (default is RNA)

Details

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.

Value

Returns a 'predictive power' (abs(AUC-0.5)) ranked matrix of putative differentially expressed genes.

Examples

```r
pbmc_small
MarkerTest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
            cells.2 = WhichCells(object = pbmc_small, ident = 2))
```
MASTDETest

**Differential expression using MAST**

**Description**

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.

**Usage**

```
MASTDETest(object, cells.1, cells.2, genes.use = NULL, latent.vars = NULL,
           assay.type = "RNA", ...)
```

**Arguments**

- `object` Seurat object
- `cells.1` Group 1 cells
- `cells.2` Group 2 cells
- `genes.use` Genes to use for test
- `latent.vars` Confounding variables to adjust for in DE test. Default is "nUMI", which adjusts for cellular depth (i.e. cellular detection rate). For non-UMI based data, set to nGene instead.
- `assay.type` Type of assay to fetch data for (default is RNA)
- `...` Additional parameters to zero-inflated regression (zlm) function in MAST

**Details**

To use this method, please install MAST, using instructions at https://github.com/RGLab/MAST/

**Value**

Returns a p-value ranked matrix of putative differentially expressed genes.

**References**


**Examples**

```
## Not run:
pbmc_small
MASTDETest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
           cells.2 = WhichCells(object = pbmc_small, ident = 2))
## End(Not run)
```
MatrixRowShuffle  Independently shuffle values within each row of a matrix

Description

Creates a matrix where correlation structure has been removed, but overall values are the same.

Usage

MatrixRowShuffle(x)

Arguments

x  Matrix to shuffle

Value

Returns a scrambled matrix, where each row is shuffled independently.

Examples

```r
mat <- matrix(data = rbinom(n = 25, size = 20, prob = 0.2), nrow = 5)
mat
MatrixRowShuffle(x = mat)
```

MergeNode  Merge children of a node

Description

Merge the children of a node into a single identity class.

Usage

MergeNode(object, node.use, rebuild.tree = FALSE, ...)

Arguments

object  Seurat object
node.use  Merge children of this node
rebuild.tree  Rebuild cluster tree after the merge?
...  Extra parameters to BuildClusterTree, used only if rebuild.tree = TRUE
See Also
BuildClusterTree

Examples

PlotClusterTree(object = pbmc_small)
pbmc_small <- MergeNode(object = pbmc_small, node.use = 7, rebuild.tree = TRUE)
PlotClusterTree(object = pbmc_small)

mergeSeurat

Description

Merge two Seurat objects

Usage

mergeSeurat(object1, object2, project = NULL, min.cells = 0,
min.genes = 0, is.expr = 0, do.normalize = TRUE, scale.factor = 10000,
do.scale = FALSE, do.center = FALSE, names.field = 1,
names.delim = " ", add.cell.id1 = NULL, add.cell.id2 = NULL)

Arguments

object1 First Seurat object to merge
object2 Second Seurat object to merge
project Project name (string)
min.cells Include genes with detected expression in at least this many cells
min.genes Include cells where at least this many genes are detected
is.expr Expression threshold for 'detected' gene
do.normalize Normalize the data after merging. Default is TRUE. If set, will perform the same
normalization strategy as stored for the first object
scale.factor If normalizing on the cell level, this sets the scale factor.
do.scale In object@scale.data, perform row-scaling (gene-based z-score). FALSE by default,
so run ScaleData after merging.
do.center In object@scale.data, perform row-centering (gene-based centering). FALSE
by default
names.field For the initial identity class for each cell, choose this field from the cell’s column
name
names.delim For the initial identity class for each cell, choose this delimiter from the cell’s
column name
add.cell.id1 String passed to RenameCells for object1
add.cell.id2 String passed to RenameCells for object1
Value

Merged Seurat object

Examples

# Split pbmc_small for this example
pbmc1 <- SubsetData(object = pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc1
pbmc2 <- SubsetData(object = pbmc_small, cells.use = pbmc_small@cell.names[41:80])
pbmc2
# Merge pbmc1 and pbmc2 into one Seurat object
pbmc_merged <- MergeSeurat(object1 = pbmc1, object2 = pbmc2)
pbmc_merged

MetageneBicorPlot  Plot CC bicor saturation plot

Description

The function provides a useful plot for evaluating the number of CCs to proceed with in the Seurat alignment workflow. Here we look at the biweight midcorrelation (bicor) of the Xth gene ranked by minimum bicor across the specified CCs for each group in the grouping.var. For alignment of more than two groups, we average the bicor results for the reference group across the pairwise alignments.

Usage

MetageneBicorPlot(object, bicor.data, grouping.var, dims.eval, gene.num = 30, num.possible.genes = 2000, return.mat = FALSE, smooth = TRUE, display.progress = TRUE)

Arguments

object  A Seurat object
bicor.data  Optionally provide data.frame returned by function to avoid recalculation
grouping.var  Grouping variable specified in alignment procedure
dims.eval  dimensions to evaluate the bicor for
gene.num  Xth gene to look at bicor for
num.possible.genes  Number of possible genes to search when choosing genes for the metagene. Set to 2000 by default. Lowering will decrease runtime but may result in metagenes constructed on fewer than num.genes genes.
return.mat  Return data.matrix instead of ggplot2 object
smooth  Smooth curves
display.progress  Show progress bar
MultiModal_CCA

Examples

pbmc_small <- DoKMeans(object = pbmc_small, k.genes = 3)
KMeansHeatmap(object = 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)

MultiModal_CCA

Run Canonical Correlation Analysis (CCA) on multimodal data

Description

CCA finds a shared correlation structure between two different datasets, enabling integrated down-steam analysis

Usage

MultiModal_CCA(object, assay.1 = "RNA", assay.2 = "CITE",
features.1 = NULL, features.2 = NULL, num.cc = 20,
normalize.variance = TRUE)
MultiModal_CIA

Arguments

object Seurat object
assay.1 First assay for multimodal analysis. Default is RNA
assay.2 Second assay for multimodal analysis. Default is CITE for CITE-Seq analysis.
features.1 Features of assay 1 to consider (default is variable genes)
features.2 Features of assay 2 to consider (default is all features, i.e. for CITE-Seq, all antibodies)
um.cc Number of canonical correlations to compute and store. Default is 20, but will calculate less if either assay has <20 features.

normalize.variance Z-score the embedding of each CC to 1, so each CC contributes equally in downstream analysis (default is T)

Value

Returns object after CCA, with results stored in dimensional reduction cca.assay1 (i.e. cca.RNA) and cca.assay2. For example, results can be visualized using DimPlot(object.reduction.use="cca.RNA")

Description

CIA finds a shared correlation structure between two different datasets, enabling integrated downstream analysis

Usage

multiModal_CIA(object, assay.1 = "RNA", assay.2 = "CITE",
features.1 = NULL, features.2 = NULL, num.axes = 20,
normalize.variance = TRUE)

Arguments

object Seurat object
assay.1 First assay for multimodal analysis. Default is RNA
assay.2 Second assay for multimodal analysis. Default is CITE for CITE-Seq analysis.
features.1 Features of assay 1 to consider (default is variable genes)
features.2 Features of assay 2 to consider (default is all features, i.e. for CITE-Seq, all antibodies)
um.axes Number of principal axes to compute and store. Default is 20, but will calculate less if either assay has <20 features.
normalize.variance Return the normalized row scales, so each axis contributes equally in downstream analysis (default is T)
NegBinomDETest

Value

Returns object after CIA, with results stored in dimensional reduction cia.assay1 (ie. cia.RNA) and cia.assay2. For example, results can be visualized using DimPlot(object, reduction.use="cia.RNA")

---

NegBinomDETest  Negative binomial test for UMI-count based data

Description

Identifies differentially expressed genes between two groups of cells using a negative binomial generalized linear model

Usage

NegBinomDETest(object, cells.1, cells.2, genes.use = NULL, latent.vars = NULL, print.bar = TRUE, min.cells = 3, assay.type = "RNA")

Arguments

- object: Seurat object
- cells.1: Group 1 cells
- cells.2: Group 2 cells
- genes.use: Genes to use for test
- latent.vars: Latent variables to test
- print.bar: Print progress bar
- min.cells: Minimum number of cells threshold
- assay.type: Type of assay to fetch data for (default is RNA)

Value

Returns a p-value ranked matrix of putative differentially expressed genes.

Examples

```r
pbmc_small
# Note, not recommended for particularly small datasets - expect warnings
NegBinomDETest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1), cells.2 = WhichCells(object = pbmc_small, ident = 2))
```
NegBinomRegDETest

Negative binomial test for UMI-count based data (regularized version)

Description

Identifies differentially expressed genes between two groups of cells using a likelihood ratio test of negative binomial generalized linear models where the overdispersion parameter theta is determined by pooling information across genes.

Usage

NegBinomRegDETest(object, cells.1, cells.2, genes.use = NULL, latent.vars = NULL, print.bar = TRUE, min.cells = 3, assay.type = "RNA")

Arguments

object Seurat object

cells.1 Group 1 cells

cells.2 Group 2 cells

genes.use Genes to use for test

latent.vars Latent variables to test

print.bar Print progress bar

min.cells Minimum number of cells threshold

assay.type Type of assay to fetch data for (default is RNA)

Value

Returns a p-value ranked data frame of test results.

Examples

# Note, not recommended for particularly small datasets - expect warnings
NegBinomDETest(
  object = pbmc_small,
  cells.1 = WhichCells(object = pbmc_small, ident = 1),
  cells.2 = WhichCells(object = pbmc_small, ident = 2)
)
**NormalizeData**

*Normalize Assay Data*

**Description**

Normalize data for a given assay.

**Usage**

```r
NormalizeData(object, assay.type = "RNA",
               normalization.method = "LogNormalize", scale.factor = 10000,
               display.progress = TRUE)
```

**Arguments**

- **object**: Seurat object
- **assay.type**: Type of assay to normalize for (default is RNA), but can be changed for multi-modal analyses.
- **normalization.method**: Method for normalization. Default is log-normalization (LogNormalize). More methods to be added very shortly.
- **scale.factor**: Sets the scale factor for cell-level normalization.
- **display.progress**: display progress bar for scaling procedure.

**Value**

Returns object after normalization. Normalized data is stored in data slot.

**Examples**

```r
pbmc_small
pmbc_small <- NormalizeData(object = pbmc_small)
```

---

**NumberClusters**

*Convert the cluster labels to a numeric representation*

**Description**

Convert the cluster labels to a numeric representation.

**Usage**

```r
NumberClusters(object)
```
Arguments

object Seurat object

Value

Returns a Seurat object with the identities relabeled numerically starting from 1.

Examples

# Append "Cluster_" to cluster IDs to demonstrate numerical conversion
new.cluster.labels <- paste0("Cluster_", pbmc_small@ident)
pbm_small <- SetIdent(
  object = pbm_small,
  cells.use = pbm_small@cell.names,
  ident.use = new.cluster.labels
)
unique(pbm_small@ident)
# Now relabel the IDs numerically starting from 1
pbm_small <- NumberClusters(pbm_small)
unique(pbm_small@ident)

OldDoHeatmap  Gene expression heatmap

Description

Draws a heatmap of single cell gene expression using the heatmap.2 function. Has been replaced by the ggplot2 version (now in DoHeatmap), but kept for legacy

Usage

OldDoHeatmap(object, cells.use = NULL, genes.use = NULL, disp.min = NULL, disp.max = NULL, draw.line = TRUE, do.return = FALSE, order.by.ident = TRUE, col.use = PurpleAndYellow(), slim.col.label = FALSE, group.by = NULL, remove.key = FALSE, cex.col = NULL, do.scale = TRUE, ...)

Arguments

object Seurat object
cells.use Cells to include in the heatmap (default is all cells)
genuses.use Genes to include in the heatmap (ordered)
disp.min Minimum display value (all values below are clipped)
disp.max Maximum display value (all values above are clipped)
draw.line Draw vertical lines delineating cells in different identity classes.
do.return  Default is FALSE. If TRUE, return a matrix of scaled values which would be passed to heatmap.2
order.by.ident  Order cells in the heatmap by identity class (default is TRUE). If FALSE, cells are ordered based on their order in cells.use
col.use  Color palette to use
slim.col.label  if (order.by.ident==TRUE) then instead of displaying every cell name on the heatmap, display only the identity class name once for each group
group.by  If (order.by.ident==TRUE) default, you can group cells in different ways (for example, orig.ident)
remove.key  Removes the color key from the plot.
cex.col  positive numbers, used as cex.axis in for the column axis labeling. The defaults currently only use number of columns
do.scale  whether to use the data or scaled data
...  Additional parameters to heatmap.2. Common examples are cexRow and cex-Col, which set row and column text sizes

Value
If do.return==TRUE, a matrix of scaled values which would be passed to heatmap.2. Otherwise, no return value, only a graphical output

Examples
pbmc_small
doHeatmap(object = pbmc_small, genes.use = pbmc_small@var.genes)

pbmc_small  A small example version of the PBMC dataset

Description
A subsetted version of 10X Genomics’ 3k PBMC dataset

Usage
pbmc_small

Format
A Seurat object with the following slots filled

raw.data  Raw expression data
data  Normalized expression data
scale.data  Scaled expression data
PCAEmbed

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

Source

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

PCAEmbed

PCA Cell Embeddings Accessor Function

Description

Pull PCA cell embedding matrix

Usage

PCAEmbed(object, dims.use = NULL, cells.use = NULL)

Arguments

object Seurat object
dims.use Dimensions to include (default is all stored dims)
cells.use Cells to include (default is all cells)

Value

PCA cell embedding matrix for given cells and PCs

Examples

pbmc_small
head(PCAEmbed(pbmcsmall))
# Optionally, you can specify subsets of dims or cells to use
PCAEmbed(pbmcsmall, dims.use = 1:5, cells.use = pbmc_small@cell.names[1:5])
PCALoad \hspace{1cm} PCA Gene Loadings Accessor Function

Description

Pull the PCA gene loadings matrix

Usage

\texttt{PCALoad(object, dims\_use = NULL, genes\_use = NULL, use\_full = FALSE)}

Arguments

- \texttt{object} \hspace{1cm} Seurat object
- \texttt{dims\_use} \hspace{1cm} Dimensions to include (default is all stored dims)
- \texttt{genes\_use} \hspace{1cm} Genes to include (default is all genes)
- \texttt{use\_full} \hspace{1cm} Return projected gene loadings (default is FALSE)

Value

PCA gene loading matrix for given genes and PCs

Examples

\begin{verbatim}
pbm\_small
head(PCALoad(pbm\_small))
# Optionally, you can specify subsets of dims or genes to use
PCALoad(pbm\_small, dims\_use = 1:5, genes\_use = pbm\_small\@var\_genes[1:5])
\end{verbatim}

PCAPlot \hspace{1cm} Plot PCA map

Description

Graphs the output of a PCA analysis. Cells are colored by their identity class.

Usage

\texttt{PCAPlot(object, ...)}

Arguments

- \texttt{object} \hspace{1cm} Seurat object
- \texttt{...} \hspace{1cm} Additional parameters to DimPlot, for example, which dimensions to plot.
Details

This function is a wrapper for DimPlot. See ?DimPlot for a full list of possible arguments which can be passed in here.

Examples

PCAPlot(object = pbmc_small)

Description

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

Usage

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

Arguments

- object: Seurat object
- pcs.use: PCS to use.
- pval.cut: P-value cutoff
- use.full: Use the full list of genes (from the projected PCA). Assumes that ProjectPCA 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.

Examples

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

*Quickly Pick Relevant PCs*

**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 PCs and is much faster to run.

**Usage**

```r
PCElbowPlot(object, num.pc = 20)
```

**Arguments**

- `object`: Seurat object
- `num.pc`: Number of PCs to plot

**Value**

Returns ggplot object

**Examples**

```r
PCElbowPlot(object = pbmc_small)
```

---

**PCHeatmap**

*Principal component 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
PCHeatmap(object, pc.use = 1, cells.use = NULL, num.genes = 30,
          use.full = FALSE, disp.min = -2.5, disp.max = 2.5, do.return = FALSE,
          col.use = PurpleAndYellow(), use.scale = TRUE, do.balanced = FALSE,
          remove.key = FALSE, label.columns = NULL, ...)```

Arguments

object  Seurat object.
pc.use  PCs to plot
cells.use  A list of cells to plot. If numeric, just plots the top cells.
num.genes  Number of genes to plot
use.full  Use the full PCA (projected PCA). Default is FALSE
disp.min  Minimum display value (all values below are clipped)
disp.max  Maximum display value (all values above are clipped)
do.return  If TRUE, returns plot object, otherwise plots plot object
col.use  Color to plot.
use.scale  Default is TRUE: plot scaled data. If FALSE, plot raw data on the heatmap.
do.balanced  Plot an equal number of genes with both + and - scores.
remove.key  Removes the color key from the plot.
label.columns  Whether to label the columns. Default is TRUE for 1 PC, FALSE for > 1 PC
...  Extra parameters for DimHeatmap

Value

If do.return==TRUE, a matrix of scaled values which would be passed to heatmap.2. Otherwise, no return value, only a graphical output

Examples

PCHeatmap(object = pbmc_small)

PCTopCells  Find cells with highest PCA scores

Description

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

Usage

PCTopCells(object, pc.use = 1, num.cells = NULL, do.balanced = FALSE)

Arguments

object  Seurat object
pc.use  Principal component to use
num.cells  Number of cells to return
do.balanced  Return an equal number of cells with both + and - PC scores.
PCTopGenes

Find genes with highest PCA scores

Description

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

Usage

PCTopGenes(object, pc.use = 1, num.genes = 30, use.full = FALSE, do.balanced = FALSE)

Arguments

  object     Seurat object
  pc.use     Principal components to use
  num.genes  Number of genes to return
  use.full   Use the full PCA (projected PCA). Default is FALSE
  do.balanced Return an equal number of genes with both + and - PC scores.

Value

Returns a vector of genes

Examples

pbmc_small
  PCTopGenes(object = pbmc_small, pc.use = 1)
  # After projection:
  PCTopGenes(object = pbmc_small, pc.use = 1, use.full = TRUE)
**PlotClusterTree**

*Plot phylogenetic tree*

**Description**

Plots previously computed phylogenetic tree (from `BuildClusterTree`)

**Usage**

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

**Arguments**

- `object`: Seurat object
- `...`: Additional arguments for plotting the phylogeny

**Value**

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

**Examples**

```r
PlotClusterTree(object = pbmc_small)
```

---

**PoissonDETest**

*Poisson test for UMI-count based data*

**Description**

Identifies differentially expressed genes between two groups of cells using a poisson generalized linear model

**Usage**

```r
PoissonDETest(object, cells.1, cells.2, min.cells = 3, genes.use = NULL, latent.vars = NULL, print.bar = TRUE, assay.type = "RNA")
```
Arguments

- **object**: Seurat object
- **cells.1**: Group 1 cells
- **cells.2**: Group 2 cells
- **min.cells**: Minimum number of cells expressing the gene in at least one of the two groups
- **genes.use**: Genes to use for test
- **latent.vars**: Latent variables to test
- **print.bar**: Print progress bar
- **assay.type**: Type of assay to fetch data for (default is RNA)

Value

Returns a p-value ranked matrix of putative differentially expressed genes.

Examples

```r
# Note, expect warnings with example dataset due to min.cells threshold.
PoissonDETest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
              cells.2 = WhichCells(object = pbmc_small, ident = 2))
```

---

Description

Print the parameters chosen for the latest AlignSubspace calculation for each stored aligned subspace.

Usage

```r
PrintAlignSubspaceParams(object, raw = FALSE)
```

Arguments

- **object**: Seurat object
- **raw**: Print the entire contents of the calculation settings slot (calc.params) for the AlignSubspace calculation. Default (FALSE) will print a nicely formatted summary.

Value

No return value. Only prints to console.
Examples

```r
## Not run:
# Requires CCA to have previously been run
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbm<outside-left-delimiter>small,cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbm<outside-left-delimiter>small,cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta$data$group <- "group1"
pbmc2@meta$data$group <- "group2"
pbm<outside-left-delimiter>c<outside-left-delimiter> <<outside-left-delimiter> RunCCA(pbmc1,pbmc2)
pbm<outside-left-delimiter>c<outside-left-delimiter> <- AlignSubspace(pbm<outside-left-delimiter>c<outside-left-delimiter>ca,red<outside-left-delimiter>uction.type = "cca", grouping.var = "group", dims.align = 1:2)
PrintAlignSubspaceParams(object = pbmc<outside-left-delimiter>small)
```

## End(Not run)

---

PrintCalcParams | Print the calculation

Description

Print entire contents of calculation settings slot (calc.params) for given calculation.

Usage

```r
PrintCalcParams(object, calculation, raw = FALSE, return.list = FALSE)
```

Arguments

- `object`: Seurat object
- `calculation`: Name of calculation (function name) to check parameters for
- `raw`: Print the entire contents of the calculation settings slot (calc.params) for the RunPCA calculation.
- `return.list`: Return the calculation parameters as a list

Value

Prints the calculation settings and optionally returns them as a list

Examples

```r
PrintCalcParams(object = pbmc<outside-left-delimiter>small, calculation = 'RunPCA')
PrintCalcParams(object = pbmc<outside-left-delimiter>small, calculation = 'RunPCA', raw = TRUE)
```
PrintCalcVarExpRatioParams

*Print Parameters Associated with CalcVarExpRatio*

**Description**

Print the parameters chosen for CalcVarExpRatio.

**Usage**

`PrintCalcVarExpRatioParams(object, raw = FALSE)`

**Arguments**

- **object**: Seurat object
- **raw**: Print the entire contents of the calculation settings slot (calc.params) for CalcVarExpRatio. Default (FALSE) will print a nicely formatted summary.

**Value**

No return value. Only prints to console.

**Examples**

```r
## Not run:
# Requires CCA to have previously been run
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta$data$group <- "group1"
pbmc2@meta$data$group <- "group2"
pbmc_cca <- RunCCA(pbmc1, pbmc2)
pbmc_cca <- CalcVarExpRatio(pbmc_cca, reduction.type = "pca", grouping.var = "group", dims.use = 1:5)
PrintCalcVarExpRatioParams(object = pbmc_cca)

## End(Not run)
```

PrintCCAParams

*Print CCA Calculation Parameters*

**Description**

Print the parameters chosen for the latest stored CCA calculation.
PrintCCAParams(object, raw = FALSE)

Arguments

object Seurat object
raw Print the entire contents of the calculation settings slot (calc.params) for the RunCCA calculation. Default (FALSE) will print a nicely formatted summary.

Value

No return value. Only prints to console.

Examples

# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmc_small, cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta.data$group <- "group1"
pbmc2@meta.data$group <- "group2"
pbmccca <- RunCCA(pbmc1, pbmc2)
PrintCCAParams(object = pbmccca)

PrintDim

Print the results of a dimensional reduction analysis

Description

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

Usage

PrintDim(object, reduction.type = "pca", dims.print = 1:5,
genesis.print = 30, use.full = FALSE)

Arguments

object Seurat object
reduction.type Reduction technique to print results for
dims.print Number of dimensions to display
genesis.print Number of genes to display
use.full Use full PCA (i.e. the projected PCA, by default FALSE)

Value

Set of genes defining the components
Examples

```r
pbmc_small
PrintDim(object = pbmc_small, reduction.type = "pca")
# Options for how many dimensions and how many genes to print
PrintDim(object = pbmc_small, reduction.type = "pca", dims.print = 1:2, genes.print = 5)
# Can also print for the projected PCA
PrintDim(object = pbmc_small, reduction.type = "pca", use.full = TRUE)
```

---

**Description**

Print the parameters chosen for the latest stored diffusion map calculation.

**Usage**

```r
PrintDMParams(object, raw = FALSE)
```

**Arguments**

- `object` Seurat object
- `raw` Print the entire contents of the calculation settings slot (calc.params) for the RunDiffusion calculation. Default (FALSE) will print a nicely formatted summary.

**Value**

No return value. Only prints to console.

**Examples**

```r
## Not run:
# Run Diffusion on variable genes
pbmc_small <- RunDiffusion(pbmc_small, genes.use = pbmc_small@var.genes)
PrintDMParams(object = pbmc_small)

## End(Not run)
```
PrintFindClustersParams

Print FindClusters Calculation Parameters

Description
Print the parameters chosen for the latest FindClusters calculation for each stored resolution.

Usage
PrintFindClustersParams(object, resolution, raw = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>resolution</td>
<td>Optionally specify only a subset of resolutions to print parameters for.</td>
</tr>
<tr>
<td>raw</td>
<td>Print the entire contents of the calculation settings slot (calc.params) for the FindClusters calculation. Default (FALSE) will print a nicely formatted summary.</td>
</tr>
</tbody>
</table>

Value
No return value. Only prints to console.

Examples
PrintFindClustersParams(object = pbmc_small, raw = TRUE)

PrintICA

Print the results of a ICA analysis

Description
Prints a set of genes that most strongly define a set of independent components

Usage
PrintICA(object, ics.print = 1:5, genes.print = 30, use.full = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>ics.print</td>
<td>Set of ICs to print genes for</td>
</tr>
<tr>
<td>genes.print</td>
<td>Number of genes to print for each PC</td>
</tr>
<tr>
<td>use.full</td>
<td>Use full PCA (i.e. the projected PCA, by default FALSE)</td>
</tr>
</tbody>
</table>
PrintICAParams

Description

Print the parameters chosen for the latest stored ICA calculation.

Usage

PrintICAParams(object, raw = FALSE)

Arguments

- object: Seurat object
- raw: Print the entire contents of the calculation settings slot (calc.params) for the ICA calculation. Default (FALSE) will print a nicely formatted summary.

Value

No return value. Only prints to console.

Examples

pbmc_small <- RunICA(object = pbmc_small, ics.compute = 10, ics.print = 0)
pbmcmath <- ProjectDim(object = pbmc_small, reduction.type = "ica", do.print = FALSE)
PrintICA(object = pbmc_small)
  # Options for how many dimensions and how many genes to print
PrintICA(object = pbmc_small, ics.print = 1:2, genes.print = 5)
  # Can also print for the projected PCA
PrintICA(object = pbmc_small, use.full = TRUE)
**PrintPCA**

*Print the results of a PCA analysis*

**Description**

Prints a set of genes that most strongly define a set of principal components.

**Usage**

\[\text{PrintPCA}(\text{object, pcs.print = 1:5, genes.print = 30, use.full = FALSE})\]

**Arguments**

- **object**: Seurat object
- **pcs.print**: Set of PCs to print genes for
- **genes.print**: Number of genes to print for each PC
- **use.full**: Use full PCA (i.e. the projected PCA, by default FALSE)

**Value**

Only text output

**Examples**

```r
pbmc_small
PrintPCA(object = pbmc_small)
```

# Options for how many dimensions and how many genes to print
```
PrintPCA(object = pbmc_small, pcs.print = 1:2, genes.print = 5)
```

# Can also print for the projected PCA
```
PrintPCA(object = pbmc_small, use.full = TRUE)
```

**PrintPCAParams**

*Print PCA Calculation Parameters*

**Description**

Print the parameters chosen for the latest stored PCA calculation.

**Usage**

\[\text{PrintPCAParams}(\text{object, raw = FALSE})\]
PrintSNNParams

Argument

- `object`: Seurat object
- `raw`: Print the entire contents of the calculation settings slot (calc.params) for the RunPCA calculation. Default (FALSE) will print a nicely formatted summary.

Value

No return value. Only prints to console.

Examples

```r
printpcaparams(object = pbmc_small)
```

---

PrintSNNParams

`Print SNN Construction Calculation Parameters`

Description

Print the parameters chosen for the latest stored SNN calculation (via BuildSNN or FindClusters).

Usage

```r
PrintSNNParams(object, raw = FALSE)
```

Arguments

- `object`: Seurat object
- `raw`: Print the entire contents of the calculation settings slot (calc.params) for the BuildSNN calculation. Default (FALSE) will print a nicely formatted summary.

Value

No return value. Only prints to console.

Examples

```r
pbmc_small <- BuildSNN(object = pbmc_small)
PrintSNNParams(object = pbmc_small)
```
**PrintTSNEParams**

**Print TSNE Calculation Parameters**

**Description**

Print the parameters chosen for the latest stored TSNE calculation.

**Usage**

`PrintTSNEParams(object, raw = FALSE)`

**Arguments**

- `object` Seurat object
- `raw` Print the entire contents of the calculation settings slot (calc.params) for the `RunTSNE` calculation. Default (FALSE) will print a nicely formatted summary.

**Value**

No return value. Only prints to console.

**Examples**

```r
pbmc_small <- RunTSNE(pbmcl_small, perplexity = 10)
PrintTSNEParams(object = pbmc_small)
```

**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.type = "pca", dims.print = 1:5, dims.store = 30, genes.print = 30, replace.dim = FALSE, do.center = FALSE, do.print = TRUE, assay.type = "RNA")`
**ProjectPCA**

Project Principal Components Analysis onto full dataset

**Description**

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

**Usage**

ProjectPCA(object, do.print = TRUE, pcs.print = 1:5, pcs.store = 30, genes.print = 30, replace.pc = FALSE, do.center = FALSE)
Arguments

object Seurat object
do.print Print top genes associated with the projected PCs
pcs.print Number of PCs to print genes for
pcs.store Number of PCs to store (default is 30)
genes.print Number of genes with highest/lowest loadings to print for each PC
replace.pc Replace the existing PCA (overwrite object@dr$pca@gene.loadings), not done by default.
do.center Center the dataset prior to projection (should be set to TRUE)

Value

Returns Seurat object with the projected PCA values in object@dr$pca@gene.loadings.full

Examples

```
pbmc_small
pbmc_small <- ProjectPCA(pbmc_small)
# Visualize top projected genes in heatmap
PCHeatmap(pbmc_small,pc.use = 1,use.full = TRUE,do.balanced = TRUE)
```

PurpleAndYellow (PurpleAndYellow)

Description

A purple and yellow color palette

Usage

PurpleAndYellow(...)

Arguments

... Extra parameters to CustomPalette

Value

A color palette

See Also

CustomPalette
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())

Description

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

Usage

Read10X(data.dir = NULL)

Arguments

data.dir Directory containing the matrix.mtx, genes.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.

Value

Returns a sparse matrix with rows and columns labeled

Examples

## Not run:
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(raw.data = expression_matrix)

## End(Not run)
Read10X_h5  

**Read 10X hdf5 file**

**Description**

Read gene expression matrix from 10X CellRanger hdf5 file

**Usage**

`Read10X_h5(filename, ensg.names = FALSE)`

**Arguments**

- **filename**: Path to h5 file
- **ensg.names**: Label row names with ENSG names rather than unique gene names

**Value**

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

RefinedMapping  

**Quantitative refinement of spatial inferences**

**Description**

Refines the initial mapping with more complex models that allow gene expression to vary quantitatively across bins (instead of 'on' or 'off'), and that also considers the covariance structure between genes.

**Usage**

`RefinedMapping(object, genes.use)`

**Arguments**

- **object**: Seurat object
- **genes.use**: Genes to use to drive the refinement procedure.

**Details**

Full details given in spatial mapping manuscript.

**Value**

Seurat object, where mapping probabilities for each bin are stored in object@final.prob
Examples

## Not run:
# Note that the PBMC test example object does not contain spatially restricted
# examples below are only demonstrate code
pmbc_small <- RefinedMapping(pmbc_small, genes.use=pmbc_small@var.genes)

## End(Not run)

---

### RemoveFromTable

**Remove data from a table**

**Description**

This function will remove any rows from a data frame or matrix that contain certain values

**Usage**

```r
RemoveFromTable(to.remove, data)
```

**Arguments**

- `to.remove`: A vector of values that indicate removal
- `data`: A data frame or matrix

**Value**

A data frame or matrix with values removed by row

**Examples**

```r
df <- data.frame(
    x = rnorm(n = 100, mean = 20, sd = 2),
    y = rbinom(n = 100, size = 100, prob = 0.2)
)
nrow(x = df)
nrow(x = RemoveFromTable(to.remove = 20, data = df))
```
**Description**

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

**Usage**

```r
RenameCells(object, add.cell.id = NULL, new.names = NULL, for.merge = FALSE)
```

**Arguments**

- `object`: Seurat object
- `add.cell.id`: prefix to add cell names
- `new.names`: vector of new cell names
- `for.merge`: Only rename slots needed for merging Seurat objects. Currently only renames the raw.data and meta.data slots.

**Details**

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

**Value**

Seurat object with new cell names

**Examples**

```r
head(pbmcsmall@cell.names)
pbmcsmall <- RenameCells(pbmcsmall, add.cell.id = "Test")
head(pbmcsmall@cell.names)
```
**RenameIdent**

*Rename one identity class to another*

**Description**

Can also be used to join identity classes together (for example, to merge clusters).

**Usage**

```
RenameIdent(object, old.ident.name = NULL, new.ident.name = NULL)
```

**Arguments**

- `object` Seurat object
- `old.ident.name` The old identity class (to be renamed)
- `new.ident.name` The new name to apply

**Value**

A Seurat object where object@ident has been appropriately modified

**Examples**

```
head(x = pbmc_small@ident)
pbmc_small <- RenameIdent(
  object = pbmc_small,
  old.ident.name = 0,
  new.ident.name = 'cluster_0'
)
head(x = pbmc_small@ident)
```

---

**ReorderIdent**

*Reorder identity classes*

**Description**

Re-assigns the identity classes according to the average expression of a particular feature (i.e, gene expression, or PC score) Very useful after clustering, to re-order cells, for example, based on PC scores

**Usage**

```
ReorderIdent(object, feature = "PC1", rev = FALSE, aggregate.fxn = mean,
  reorder.numeric = FALSE, ...)
```
RidgePlot

Arguments

- object: Seurat object
- feature: Feature to reorder on. Default is PC1
- rev: Reverse ordering (default is FALSE)
- aggregate.fxn: Function to evaluate each identity class based on (default is mean)
- reorder.numeric: Rename all identity classes to be increasing numbers starting from 1 (default is FALSE)
- ... additional arguments (i.e. use.imputed=TRUE)

Value

A seurat object where the identity have been re-ordered based on the average.

Examples

```r
head(x = pbmc_small@ident)
pbmc_small <- ReorderIdent(object = pbmc_small)
head(x = pbmc_small@ident)
```

RidgePlot

Single cell ridge plot

Description

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

Usage

```r
RidgePlot(object, features.plot, ident.include = NULL, nCol = NULL,
do.sort = FALSE, y.max = NULL, same.y.lims = FALSE, size.x.use = 16,
size.y.use = 16, size.title.use = 20, cols.use = NULL,
group.by = NULL, y.log = FALSE, x.lab.rot = FALSE, y.lab.rot = FALSE,
legend.position = "right", single.legend = TRUE, remove.legend = FALSE,
do.return = FALSE, return.plotlist = FALSE, ...)
```

Arguments

- object: Seurat object
- features.plot: Features to plot (gene expression, metrics, PC scores, anything that can be retrieved by FetchData)
- ident.include: Which classes to include in the plot (default is all)
- nCol: Number of columns if multiple plots are displayed
do.sort Sort identity classes (on the x-axis) by the average expression of the attribute being potted
y.max Maximum y axis value
same.y.lims Set all the y-axis limits to the same values
size.x.use X axis title font size
size.y.use Y axis title font size
size.title.use Main title font size
cols.use Colors to use for plotting
group.by Group (color) cells in different ways (for example, orig.ident)
y.log plot Y axis on log scale
x.lab.rot Rotate x-axis labels
y.lab.rot Rotate y-axis labels
legend.position Position the legend for the plot
single.legend Consolidate legend the legend for all plots
remove.legend Remove the legend from the plot
do.return Return a ggplot2 object (default: FALSE)
return.plotlist Return the list of individual plots instead of compiled plot.
... additional parameters to pass to FetchData (for example, use.imputed, use.scaled, use.raw)

Value
By default, no return, only graphical output. If do.return=TRUE, returns a list of ggplot objects.

Examples
RidgePlot(object = pbmc_small, features.plot = 'PC1')

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(object, object2, group1, group2, group.by, num.cc = 20, genes.use, scale.data = TRUE, rescale.groups = FALSE, ...)


RunCCA

Arguments

object Seurat object
object2 Optional second object. If object2 is passed, object1 will be considered as group1 and object2 as group2.
group1 First set of cells (or IDs) for CCA
group2 Second set of cells (or IDs) for CCA
group.by Factor to group by (column vector stored in object@meta.data)
num.cc Number of canonical vectors to calculate
genes.use Set of genes to use in CCA. Default is object@var.genes. If two objects are given, the default is the union of both variable gene sets that are also present in both objects.
scale.data Use the scaled data from the object
rescale.groups Rescale each set of cells independently
... 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)

Value

Returns Seurat object with the CCA stored in the @dr$cca slot. If one object is passed, the same object is returned. If two are passed, a combined object is returned.

See Also

MergeSeurat

Examples

pbmc_small
# As CCA requires two datasets, we will split our test object into two just for this example
pbmc1 <- SubsetData(pbmc_small,cells.use = pbmc_small@cell.names[1:40])
pbmc2 <- SubsetData(pbmc_small,cells.use = pbmc_small@cell.names[41:80])
pbmc1@meta.data$group <- "group1"
pbmc2@meta.data$group <- "group2"
pbmc_cca <- RunCCA(pbmc1,pbmc2)
# Print results
PrintDim(pbmc_cca,reduction.type = 'cca')
RunDiffusion

**Run diffusion map**

**Description**

NOTE: Prior to v2.3.4, this function used the R package diffusionMap to compute the diffusion map components. This package was being archived and thus RunDiffusion now uses the destiny package for the diffusion computations. Please be aware that this will result in different default values as the two underlying package implementations are different.

**Usage**

```r
RunDiffusion(object, cells.use = NULL, dims.use = 1:5, genes.use = NULL, reduction.use = "pca", q.use = 0.01, max.dim = 2, scale.clip = 10, reduction.name = "dm", reduction.key = "DM", ...)```

**Arguments**

- `object` Seurat object
- `cells.use` Which cells to analyze (default, all cells)
- `dims.use` Which dimensions to use as input features
- `genes.use` If set, run the diffusion map procedure on this subset of genes (instead of running on a set of reduced dimensions). Not set (NULL) by default
- `reduction.use` Which dimensional reduction (PCA or ICA) to use for the diffusion map input. Default is PCA
- `q.use` Quantile to clip diffusion map components at. This addresses an issue where 1-2 cells will have extreme values that obscure all other points. 0.01 by default
- `max.dim` Max dimension to keep from diffusion calculation
- `scale.clip` Max/min value for scaled data. Default is 3
- `reduction.name` Dimensional reduction name, specifies the position in the object$dr list. dm by default
- `reduction.key` Dimensional reduction key, specifies the string before the number for the dimension names. DM by default
- `...` Additional arguments to the DiffusionMap call

**Value**

Returns a Seurat object with a diffusion map
RunICA

Run Independent Component Analysis on gene expression

Examples

```r
## Not run:
pbmc_small
# Run Diffusion on variable genes
pbmc_small <- RunDiffusion(pbmc_small, genes.use = pbmc_small@var.genes)
# Run Diffusion map on first 10 PCs
pbmc_small <- RunDiffusion(pbmc_small, genes.use = pbmc_small@var.genes)
# Plot results
DMPlot(pbmc_small)

## End(Not run)
```

Description

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

Usage

```r
RunICA(object, ic.genes = NULL, ics.compute = 50, use.imputed = FALSE, 
rev.ica = FALSE, print.results = TRUE, ics.print = 1:5, 
genomes.print = 50, ica.function = "icafast", seed.use = 1, 
reduction.name = "ica", reduction.key = "IC", ...)
```

Arguments

- `object` Seurat object
- `ic.genes` Genes to use as input for ICA. Default is `object@var.genes`
- `ics.compute` Number of ICs to compute
- `use.imputed` Run ICA on imputed values (FALSE by default)
- `rev.ica` By default, computes the dimensional reduction on the cell x gene matrix. Setting to true will compute it on the transpose (gene x cell matrix).
- `print.results` Print the top genes associated with each dimension
- `ics.print` ICs to print genes for
- `genes.print` Number of genes to print for each IC
- `ica.function` ICA function from ica package to run (options: icafast, icaimax, icajade)
- `seed.use` Random seed to use for fastica
- `reduction.name` dimensional reduction name, specifies the position in the `object$dr` list. ica by default
- `reduction.key` dimensional reduction key, specifies the string before the number for the dimension names. IC by default
- `...` Additional arguments to be passed to fastica
Value

Returns Seurat object with an ICA calculation stored in object@dr$ica

Examples

```r
pbmc_small
# Run ICA on variable genes (default)
pbmc_small <- RunICA(pbmc_small, ics.compute=5)
# Run ICA on different gene set (in this case all genes)
pbmc_small <- RunICA(pbmc_small, ic.genes = rownames(pbmc_small@data))
# Plot results
ICAPlot(pbmc_small)
```

Description

Runs a canonical correlation analysis

Usage

```r
RunMultiCCA(object.list, genes.use, add.cell.ids = NULL, niter = 25, 
num.ccs = 1, standardize = TRUE)
```

Arguments

- `object.list`: List of Seurat objects
- `genes.use`: Genes to use in mCCA.
- `add.cell.ids`: Vector of strings to pass to `RenameCells` to give unique cell names
- `niter`: Number of iterations to perform. Set by default to 25.
- `num.ccs`: Number of canonical vectors to calculate
- `standardize`: Standardize scale.data matrices to be centered (mean zero) and scaled to have a standard deviation of 1.

Value

Returns a combined Seurat object with the CCA stored in the @dr$cca slot.
Examples

```r
## Not run:
pbmc_small
# As multi-set CCA requires more than two datasets, we will split our test object into
# three just for this example
pbmc1 <- SubsetData(pbmc_small, cells.use = pbmc_small@names[1:30])
pbmc2 <- SubsetData(pbmc_small, cells.use = pbmc_small@names[31:60])
pbmc3 <- SubsetData(pbmc_small, cells.use = pbmc_small@names[61:80])
pbmc1@meta$data$group <- "group1"
pbmc2@meta$data$group <- "group2"
pbmc3@meta$data$group <- "group3"
pbmc.list <- list(pbmc1, pbmc2, pbmc3)
pbmc_cca <- RunMultiCCA(object.list = pbmc.list, genes.use = pbmc_small@var.genes, num.ccs = 3)
# Print results
PrintDim(pbmcc_cca, reduction.type = 'cca')

## End(Not run)
```

---

RunPCA  
Run Principal Component Analysis on gene expression using IRLBA

Description

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

Usage

```r
RunPCA(object, pc.genes = NULL, pcs.compute = 20, use.imputed = FALSE,
       rev.pca = FALSE, weight.by.var = TRUE, do.print = TRUE,
       pcs.print = 1:5, genes.print = 30, reduction.name = "pca",
       reduction.key = "PC", assay.type = "RNA", seed.use = 42, ...)
```

Arguments

- **object**: Seurat object
- **pc.genes**: Genes to use as input for PCA. Default is `object@var.genes`
- **pcs.compute**: Total Number of PCs to compute and store (20 by default)
- **use.imputed**: Run PCA on imputed values (FALSE 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)
- **do.print**: Print the top genes associated with high/low loadings for the PCs
- **pcs.print**: PCs to print genes for
genes.print  Number of genes to print for each PC
reduction.name dimensional reduction name, specifies the position in the object$dr list. pca by default
reduction.key dimensional reduction key, specifies the string before the number for the dimension names. PC by default
assay.type Data type, RNA by default. Can be changed for multimodal
seed.use Set a random seed. By default, sets the seed to 42. Setting NULL will not set a seed.
... Additional arguments to be passed to IRLBA

Value
Returns Seurat object with the PCA calculation stored in object@dr$pca.

Examples
pbmc_small
# Run PCA on variable genes (default)
pbmc_small <- RunPCA(pbmc_small)
# Run PCA on different gene set (in this case all genes)
pbmc_small=RunPCA(pbmc_small,pc.genes = rownames(pbmc_small@data))
# Run PCA but compute more than 20 dimensions
pbmc_small=RunPCA(pbmc_small,pcs.compute=30)
# Plot results
PCAPlot(pbmc_small)

RunPHATE

Description
PHATE is a data reduction method specifically designed for visualizing **high** dimensional data in **low** dimensional spaces. To run, you must first install the 'phate' python package (e.g. via pip install phate). Details on this package can be found here: https://github.com/KrishnaswamyLab/PHATE. For help, visit https://krishnaswamylab.org/get-help. For a more in depth discussion of the mathematics underlying PHATE, see the bioRxiv paper here: https://www.biorxiv.org/content/early/2017/12/01/120378.

Usage
RunPHATE(object, cells.use = NULL, genes.use = NULL, assay.type = "RNA",
max.dim = 2L, k = 5, alpha = 15, n.landmark = 2000, gamma = 1,
t = "auto", knn.dist.method = "euclidean", mds.method = "metric",
mds.dist.method = "euclidean", t.max = 100, npca = 100,
plot.optimal.t = FALSE, verbose = 1, n.jobs = 1, seed.use = NA,
reduction.name = "phate", reduction.key = "PHATE", ...)

Run PHATE
**Arguments**

object Seurat object

cells.use Which cells to analyze (default, all cells)

genes.use If set, run PHATE on this subset of genes. Not set (NULL) by default

assay.type Assay to pull data for (default: 'RNA')

max.dim Total number of dimensions to embed in PHATE.

k int, optional (default: 5) number of nearest neighbors on which to build kernel

alpha int, optional (default: 15) sets decay rate of kernel tails. If NA, alpha decaying kernel is not used

n.landmark int, optional (default: 2000) number of landmarks to use in fast PHATE

gamma float, optional (default: 1) Informational distance constant between -1 and 1. 'gamma=1' gives the PHATE log potential, 'gamma=0' gives a square root potential.

t int, optional (default: 'auto') power to which the diffusion operator is powered

knn.dist.method string, optional (default: 'euclidean') The desired distance function for calculating pairwise distances on the data. If 'precomputed', 'data' is treated as a (n_samples, n_samples) distance or affinity matrix

mds.method string, optional (default: 'metric') choose from 'classic', 'metric', and 'non-metric' which MDS algorithm is used for dimensionality reduction

mds.dist.method string, optional (default: 'euclidean') recommended values: 'euclidean' and 'cosine'

t.max int, optional (default: 100) Maximum value of t to test for automatic t selection.

npca int, optional (default: 100) Number of principal components to use for calculating neighborhoods. For extremely large datasets, using n_pca < 20 allows neighborhoods to be calculated in log(n_samples) time.

plot.optimal.t boolean, optional (default: FALSE) If TRUE, produce a plot showing the Von Neumann Entropy curve for automatic t selection.

verbose 'int' or 'boolean', optional (default : 1) If 'TRUE' or '> 0', print verbose updates.

n.jobs 'int', optional (default: 1) The number of jobs to use for the computation. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all, which is useful for debugging. For n_jobs below -1, (n cpus + 1 + n.jobs) are used. Thus for n_jobs = -2, all CPUs but one are used

seed.use int or 'NA'; random state (default: 'NA')

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

reduction.key dimensional reduction key, specifies the string before the number for the dimension names. PHATE by default

... Additional arguments for 'phateR::phate'
Value

Returns a Seurat object containing a PHATE representation

References


Examples

```r
if (reticulate::py_module_available("phate")) {

  # Load data
  pbmc_small

  # Run PHATE with default parameters
  pbmc_small <- RunPHATE(object = pbmc_small)
  # Plot results
  DimPlot(object = pbmc_small, reduction.use = 'phate')

  # Try smaller \( k \) for a small dataset, and larger \( t \) for a noisy embedding
  pbmc_small <- RunPHATE(object = pbmc_small, k = 4, t = 12)
  # Plot results
  DimPlot(object = pbmc_small, reduction.use = 'phate')

  # For increased emphasis on local structure, use sqrt potential (\( \gamma = 0 \))
  pbmc_small <- RunPHATE(object = pbmc_small, \gamma = 0)
  # Plot results
  DimPlot(object = pbmc_small, reduction.use = 'phate')
}
```

---

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

```r
RunTSNE(object, reduction.use = "pca", cells.use = NULL, dims.use = 1:5, 
  genes.use = NULL, seed.use = 1, tsne.method = "Rtsne", add.iter = 0, 
  dim.embed = 2, distance.matrix = NULL, reduction.name = "tsne", 
  reduction.key = "tSNE_", ...)
```
RunTSNE

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>reduction.use</td>
<td>Which dimensional reduction (e.g. PCA, ICA) to use for the tSNE. Default is</td>
</tr>
<tr>
<td></td>
<td>PCA</td>
</tr>
<tr>
<td>cells.use</td>
<td>Which cells to analyze (default, all cells)</td>
</tr>
<tr>
<td>dims.use</td>
<td>Which dimensions to use as input features</td>
</tr>
<tr>
<td>genes.use</td>
<td>If set, run the tSNE on this subset of genes (instead of running on a set of</td>
</tr>
<tr>
<td></td>
<td>reduced dimensions). Not set (NULL) by default</td>
</tr>
<tr>
<td>seed.use</td>
<td>Random seed for the t-SNE</td>
</tr>
<tr>
<td>tsne.method</td>
<td>Select the method to use to compute the tSNE. Available methods are:</td>
</tr>
<tr>
<td></td>
<td>• Rtsne: Use the Rtsne package Barnes-Hut implementation of tSNE (default)</td>
</tr>
<tr>
<td></td>
<td>• tsne: standard tsne - not recommended for large datasets</td>
</tr>
<tr>
<td></td>
<td>• FIt-SNE: Use the FFT-accelerated Interpolation-based t-SNE. Based on</td>
</tr>
<tr>
<td></td>
<td>Kluger Lab code found here: <a href="https://github.com/KlugerLab/FIt-SNE">https://github.com/KlugerLab/FIt-SNE</a></td>
</tr>
<tr>
<td>add.iter</td>
<td>If an existing tSNE has already been computed, uses the current tSNE to</td>
</tr>
<tr>
<td></td>
<td>seed the algorithm and then adds additional iterations on top of this</td>
</tr>
<tr>
<td>dim.embed</td>
<td>The dimensional space of the resulting tSNE embedding (default is 2). For</td>
</tr>
<tr>
<td></td>
<td>example, set to 3 for a 3d tSNE</td>
</tr>
<tr>
<td>distance.matrix</td>
<td>If set, runs tSNE on the given distance matrix instead of data matrix</td>
</tr>
<tr>
<td></td>
<td>(experimental)</td>
</tr>
<tr>
<td>reduction.name</td>
<td>Dimensional reduction name, specifies the position in the object$dr list.</td>
</tr>
<tr>
<td></td>
<td>tsne by default</td>
</tr>
<tr>
<td>reduction.key</td>
<td>Dimensional reduction key, specifies the string before the number for the</td>
</tr>
<tr>
<td></td>
<td>dimension names. tsNE_ by default</td>
</tr>
<tr>
<td></td>
<td>Additional arguments to the tSNE call. Most commonly used is perplexity (ex-</td>
</tr>
<tr>
<td></td>
<td>pected number of neighbors default is 30)</td>
</tr>
</tbody>
</table>

Value

Returns a Seurat object with a tSNE embedding in object@dr$tsne@cell.embeddings

Examples

```r
pbmc_small
# Run tSNE on first five PCs, note that for test dataset (only 80 cells)
# we can't use default perplexity of 30
pbmc_small <- RunTSNE(pbmc_small, reduction.use = "pca", dims.use = 1:5, perplexity=10)
# Run tSNE on first five independent components from ICA
pbmc_small <- RunICA(pbmc_small, ics.compute=5)
pbmc_small <- RunTSNE(pbmc_small, reduction.use = "ica", dims.use = 1:5, perplexity=10)
# Plot results
TSNEPlot(pbmc_small)
```
RunUMAP

Description

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

Usage

RunUMAP(object, cells.use = NULL, dims.use = 1:5, reduction.use = "pca",
genes.use = NULL, assay.use = "RNA", max.dim = 2L,
reduction.name = "umap", reduction.key = "UMAP", n_neighbors = 30L,
min_dist = 0.3, metric = "correlation", 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>cells.use</td>
<td>Which cells to analyze (default, all cells)</td>
</tr>
<tr>
<td>dims.use</td>
<td>Which dimensions to use as input features, used only if genes.use is NULL</td>
</tr>
<tr>
<td>reduction.use</td>
<td>Which dimensional reduction (PCA or ICA) to use for the UMAP input. Default is PCA</td>
</tr>
<tr>
<td>genes.use</td>
<td>If set, run UMAP on this subset of genes (instead of running on a set of reduced dimensions). Not set (NULL) by default</td>
</tr>
<tr>
<td>assay.use</td>
<td>Assay to pull data for when using genes.use</td>
</tr>
<tr>
<td>max.dim</td>
<td>Max dimension to keep from UMAP procedure.</td>
</tr>
<tr>
<td>reduction.name</td>
<td>dimensional reduction name, specifies the position in the object$dr list. umap by default</td>
</tr>
<tr>
<td>reduction.key</td>
<td>dimensional reduction key, specifies the string before the number for the dimension names. UMAP by default</td>
</tr>
<tr>
<td>n_neighbors</td>
<td>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.</td>
</tr>
<tr>
<td>min_dist</td>
<td>min_dist: This controls how tightly the embedding is allowed compress points together. Larger values ensure embedded points are more evenly distributed, while smaller values allow the algorithm to optimise more accurately with regard to local structure. Sensible values are in the range 0.001 to 0.5.</td>
</tr>
<tr>
<td>metric</td>
<td>metric: This determines the choice of metric used to measure distance in the input space. A wide variety of metrics are already coded, and a user defined function can be passed as long as it has been JITd by numba.</td>
</tr>
</tbody>
</table>
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, progress.bar = FALSE)

Arguments

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

Value

Matrix with downsampled data

References

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

Examples

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

## End(Not run)
SaveClusters

Save cluster assignments to a TSV file

Description

Save cluster assignments to a TSV file

Usage

SaveClusters(object, file)

Arguments

object Seurat object with cluster assignments
file Path to file to write cluster assignments to

Value

No return value. Writes clusters assignments to specified file.

Examples

## Not run:
pbmc_small
file.loc <- "~/Desktop/cluster_assignments.tsv"
SaveClusters(object = pbmc_small, file = file.loc)

## End(Not run)
ScaleData

Scale and center the data.

Description

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

Usage

ScaleData(object, genes.use = NULL, data.use = NULL, vars.to.regress, model.use = "linear", use.umi = FALSE, do.scale = TRUE, do.center = TRUE, scale.max = 10, block.size = 1000, min.cells.to.block = 3000, display.progress = TRUE, assay.type = "RNA", do.cpp = TRUE, check.for.norm = TRUE, do.par = FALSE, num.cores = 1)

Arguments

- object: Seurat object
- genes.use: Vector of gene names to scale/center. Default is all genes in object@data.
- data.use: Can optionally pass a matrix of data to scale, default is object@data[genes.use, ]
- vars.to.regress: Variables to regress out (previously latent.vars in RegressOut). For example, nUMI, or percent.mito.
- 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 genes 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 genes 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.
- display.progress: Displays a progress bar for scaling procedure.
- assay.type: Assay to scale data for. Default is RNA. Can be changed for multimodal analyses.
do.cpp  By default (TRUE), most of the heavy lifting is done in c++. We’ve maintained support for our previous implementation in R for reproducibility (set this to FALSE) as results can change slightly due to differences in numerical precision which could affect downstream calculations.

check.for.norm  Check to see if data has been normalized, if not, output a warning (TRUE by default)

do.par  use parallel processing for regressing out variables faster. If set to TRUE, will use half of the machines available cores (FALSE by default)

num.cores  If do.par = TRUE, specify the number of cores to use.

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

Value

Returns a seurat object with object@scale.data updated with scaled and/or centered data.

Examples

pbmc_small <- ScaleData(object = pbmc_small)
## Not run:
# To regress out certain effects
pbmc_small = ScaleData(object = pbmc_small, vars.to.regress = effects_list)
## End(Not run)
SetAllIdent

Arguments

| object  | Seurat object |
| genes.use | Vector of gene names to scale/center. Default is all genes in object@data. |
| data.use | Can optionally pass a matrix of data to scale, default is object@data[genes.use,] |
| do.scale | Whether to scale the data. |
| do.center | Whether to center the data. |
| scale.max | Max value to accept for scaled data. The default is 10. Setting this can help reduce the effects of genes that are only expressed in a very small number of cells. |

Value

Returns a seurat object with object@scale.data updated with scaled and/or centered data.

Examples

```r
# Not run:
pbmc_small <- ScaleDataR(object = pbmc_small)

# End(Not run)
```

Description

Switch identity class definition to another variable

Usage

```r
SetAllIdent(object, id = NULL)
```

Arguments

| object  | Seurat object |
| id | Variable to switch identity class to (for example, 'DBclust.ident', the output of density clustering) Default is orig.ident - the original annotation pulled from the cell name. |

Value

A Seurat object where object@ident has been appropriately modified
Examples

```r
head(x = pbmc_small@ident)
pbmc_small <- SetAllIdent(object = pbmc_small, id = 'orig.ident')
head(x = pbmc_small@ident)
```
**SetClusters**

*Set Cluster Assignments*

**Description**

Easily set the cluster assignments using the output of GetClusters() — a dataframe with cell names as the first column and cluster assignments as the second.

**Usage**

```r
SetClusters(object, clusters = NULL)
```

**Arguments**

- **object**: Seurat object
- **clusters**: A dataframe containing the cell names and cluster assignments to set for the object.

**Value**

Returns a Seurat object with the identities set to the cluster assignments that were passed.

**Examples**

```r
pbmc_small
# Get clusters as a dataframe with GetClusters.
clusters <- GetClusters(object = pbmc_small)
# Use SetClusters to set cluster IDs
pbmc_small <- SetClusters(object = pbmc_small, clusters = clusters)
```

---

**SetDimReduction**

*Dimensional Reduction Mutator Function*

**Description**

Set information for specified stored dimensional reduction analysis.

**Usage**

```r
SetDimReduction(object, reduction.type, slot, new.data)
```
Arguments

object: Seurat object
reduction.type: Type of dimensional reduction to set
slot: Specific information to set (must be one of the following: "cell.embeddings", "gene.loadings", "gene.loadings.full", "sdev", "key", "misc")
new.data: New data to set

Value

Seurat object with updated slot

Examples

pbmc_small

# Simulate adding a new dimensional reduction
new.cell.embeddings <- GetCellEmbeddings(object = pbmc_small, reduction.type = "pca")
new.gene.loadings <- GetGeneLoadings(object = pbmc_small, reduction.type = "pca")
SetDimReduction(
  object = pbmc_small,
  reduction.type = "new.pca",
  slot = "cell.embeddings",
  new.data = new.cell.embeddings
)
SetDimReduction(
  object = pbmc_small,
  reduction.type = "new.pca",
  slot = "gene.loadings",
  new.data = new.gene.loadings
)

SetIdent

Set identity class information

Description

Sets the identity class value for a subset (or all) cells

Usage

SetIdent(object, cells.use = NULL, ident.use = NULL)

Arguments

object: Seurat object
cells.use: Vector of cells to set identity class info for (default is all cells)
ident.use: Vector of identity class values to assign (character vector)
Value

A Seurat object where object@ident has been appropriately modified

Examples

```r
cluster2 <- WhichCells(object = pbmc_small, ident = 2)
pbmc_small@ident[cluster2]
pbmc_small <- SetIdent(
  object = pbmc_small,
  cells.use = cluster2,
  ident.use = 'cluster_2'
)
pbmc_small@ident[cluster2]
```

seurat

The Seurat Class

Description

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

Details

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

Slots

- `raw.data`: The raw project data
- `data`: The normalized expression matrix (log-scale)
- `scale.data`: scaled (default is z-scoring each gene) expression matrix; used for dimensional reduction and heatmap visualization
- `var.genes`: Vector of genes exhibiting high variance across single cells
- `is.expr`: Expression threshold to determine if a gene is expressed (0 by default)
- `ident`: The 'identity class' for each cell
- `meta.data`: Contains meta-information about each cell, starting with number of genes detected (nGene) and the original identity class (orig.ident); more information is added using `AddMetaData`
- `project.name`: Name of the project (for record keeping)
- `dr`: List of stored dimensional reductions; named by technique
- `assay`: List of additional assays for multimodal analysis; named by technique
- `hvg.info`: The output of the mean/variability analysis for all genes
imputed  Matrix of imputed gene scores
cell.names  Names of all single cells (column names of the expression matrix)
cluster.tree  List where the first element is a phylo object containing the phylogenetic tree relating different identity classes
snn  Sparse matrix object representation of the SNN graph
calc.params  Named list to store all calculation-related parameter choices
kmeans  Stores output of gene-based clustering from DoKMeans
spatial  Stores internal data and calculations for spatial mapping of single cells
misc  Miscellaneous spot to store any data alongside the object (for example, gene lists)
version  Version of package used in object creation

Seurat-deprecated  Deprecated function(s) in the Seurat package

Description
These functions are provided for compatibility with older version of the Seurat package. They may eventually be completely removed.

Usage
vlnplot(...)  Parameters to be passed to the modern version of the function

Arguments
...  Parameters to be passed to the modern version of the function

Details
vlnplot  now a synonym for VlnPlot
subsetData  now a synonym for SubsetData
pca  now a synonym for RunPCA
PCA  now a synonym for PCA
project.pca  now a synonym for ProjectPCA
viz.pca  now a synonym for VizPCA
set.ident  now a synonym for SetIdent
pca.plot  now a synonym for PCAPlot
pcHeatmap  now a synonym for PHeatmap
jackStraw  now a synonym for JackStraw
jackStrawPlot  now a synonym for JackStrawPlot
run_tsne  now a synonym for RunTSNE
tsne.plot  now a synonym for TSNEPlot
find.markers  now a synonym for FindMarkers
find_all_markers  now a synonym for FindAllMarkers
genePlot  now a synonym for GenePlot
Seurat-deprecated

feature.plot now a synonym for FeaturePlot
buildClusterTree now a synonym for BuildClusterTree
plotClusterTree now a synonym for PlotClusterTree
plotNoiseModel has been removed and may be replaced at a later date
add_samples now a synonym for AddSamples
subsetCells now deleted
project.samples has been removed and may be replaced at a later date
run_diffusion now a synonym for RunDiffusion
ica now a synonym for RunICA
    ICA now a synonym for RunICA
cluster.alpha now a synonym for AverageDetectionRate
average.pca now a synonym for AveragePCA
average.expression now a synonym for AverageExpression
icTopGenes now a synonym for ICTopGenes
pcTopGenes now a synonym for PCTopGenes
pcTopCells now a synonym for PCTopCells
fetch.data now a synonym for FetchData
viz.ica now a synonym for VizIca
regulatorScore now deleted
find.markers.node now a synonym for FindMarkersNode
diffExp.test now a synonym for DiffExpTest
tobit.test now a synonym for TobitTest
batch.gene has been removed and may be restored at a later date
marker.test now a synonym for MarkerTest
which.cells now a synonym for WhichCells
set.all.ident now a synonym for SetAllIdent
rename.ident now a synonym for RenameIdent
posterior.plot now a synonym for PosteriorPlot
    map.cell has been deprecated
get.centroids now a synonym for GetCentroids
    refined.mapping now a synonym for RefinedMapping
initial.mapping now a synonym for InitialMapping
    calc.insitu now a synonym for CalcInsitu
fit.gene.k now a synonym for FitGeneK
fit.gene.mix now a synonym for FitGeneMix
addSmoothedScore now a synonym for AddSmoothedScore
addImputedScore now a synonym for AddImputedScore
getNewScore has been removed without replacement
calcNoiseModels has been removed and may be replaced at a later date
feature.plot.keynote has been removed without replacement
feature.heatmap now a synonym for FeatureHeatmap
    ica.plot now a synonym for ICAPlot
    spatial.de has been removed and may be replaced at a later date
    DBclust_dimension now a synonym for DBClustDimension
    Kclust_dimension now a synonym for KClustDimension
    pca.sig.genes now a synonym for PCASigGenes
doHeatMap now a synonym for DoHeatMap
icHeatmap now a synonym for ICHeatmap
Shuffle

Shuffle a vector

Description
Shuffle a vector

Usage
Shuffle(x)

Arguments
x A vector

Value
A vector with the same values of x, just in random order
**Examples**

```r
v <- seq(10)
v2 <- shuffle(x = v)
v2
```

---

**SplitDotPlotGG**  
**Split Dot plot visualization**

**Description**

Intuitive way of visualizing how gene expression changes across different identity classes (clusters). The size of the dot encodes the percentage of cells within a class, while the color encodes the AverageExpression level of 'expressing' cells. Splits the cells into groups based on a grouping variable. Still in BETA

**Usage**

```r
SplitDotPlotGG(object, grouping.var, genes.plot, gene.groups,  
cols.use = c("blue", "red"), col.min = -2.5, col.max = 2.5,  
dot.min = 0, dot.scale = 6, group.by, plot.legend = FALSE,  
do.return = FALSE, x.lab.rot = FALSE)
```

**Arguments**

- `object` Seurat object
- `grouping.var` Grouping variable for splitting the dataset
- `genes.plot` Input vector of genes
- `gene.groups` Add labeling bars to the top of the plot
- `cols.use` colors to plot
- `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 0.05).
- `dot.scale` Scale the size of the points, similar to cex
- `group.by` Factor to group the cells by
- `plot.legend` plots the legends
- `do.return` Return ggplot2 object
- `x.lab.rot` Rotate x-axis labels

**Value**

default, no return, only graphical output. If do.return=TRUE, returns a ggplot2 object
Examples

# Create a simulated grouping variable
pbmc_small@meta.data$groups <- sample(
  x = c("g1", "g2"),
  size = length(x = pbmc_small@cell.names),
  replace = TRUE
)

SplitDotPlotGG(pbmc_small, grouping.var = "groups", genes.plot = pbmc_small@var.genes[1:5])

---

SplitObject

Splits object into a list of subsetted objects.

Description

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

Usage

SplitObject(object, attribute.1 = "ident", ...)

Arguments

object

Seurat object

attribute.1

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

...

Additional parameters to pass to SubsetData

Value

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

Examples

# Assign the test object a three level attribute
groups <- sample(c("group1", "group2", "group3"), size = 80, replace = TRUE)

names(groups) <- pbmc_small@cell.names

pbmc_small <- AddMetaData(object = pbmc_small, metadata = groups, col.name = "group")

obj.list <- SplitObject(pbmc_small, attribute.1 = "group")
StashIdent  

**Set identity class information**

**Description**

Stashes the identity in `data.info` to be retrieved later. Useful if, for example, testing multiple clustering parameters.

**Usage**

```r
StashIdent(object, save.name = "oldIdent")
```

**Arguments**

- `object`: Seurat object
- `save.name`: Store current `object@ident` under this column name in `object@meta.data`. Can be easily retrieved with `SetAllIdent`.

**Value**

A Seurat object where `object@ident` has been appropriately modified.

**Examples**

```r
head(x = pbmc_small@meta.data)
pbmc_small <- StashIdent(object = pbmc_small, save.name = 'cluster.ident')
head(x = pbmc_small@meta.data)
```

---

SubsetByPredicate  

**Return a subset of the Seurat object.**

**Description**

Creates a Seurat object containing only a subset of the cells in the original object. Forms a dataframe by fetching the variables in `vars.use`, then subsets it using `base::subset` with `predicate` as the filter. Returns the corresponding subset of the Seurat object.

**Usage**

```r
SubsetByPredicate(object, vars.use, predicate)
```

**Arguments**

- `object`: Seurat object
- `vars.use`: Variables to fetch for use in `base::subset`. Character vector.
- `predicate`: String to be parsed into an R expression and evaluated as an input to `base::subset`. 
Examples

pbmc1 <- SubsetByPredicate(object = pbmc_small,
                          vars.use = c("nUMI", "res.1"),
                          predicate = "nUMI < 200 & res.1=='3'")

pbmc1

SubsetData

Return a subset of the Seurat object

Description

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

Usage

SubsetData(object, cells.use = NULL, subset.name = NULL, ident.use = NULL,
ident.remove = NULL, accept.low = -Inf, accept.high = Inf,
accept.value = NULL, do.center = FALSE, do.scale = FALSE,
max.cells.per.ident = Inf, random.seed = 1, do.clean = FALSE,
subset.raw, ...)

Arguments

object Seurat object
cells.use A vector of cell names to use as a subset. If NULL (default), then this list will be computed based on the next three arguments. Otherwise, will return an object consisting only of these cells
subset.name Parameter to subset on. Eg, the name of a gene, PC1, a column name in object@meta.data, etc. Any argument that can be retrieved using FetchData
ident.use Create a cell subset based on the provided identity classes
ident.remove Subtract out cells from these identity classes (used for filtration)
accept.low Low cutoff for the parameter (default is -Inf)
accept.high High cutoff for the parameter (default is Inf)
accept.value Returns cells with the subset name equal to this value
do.center Recenter the new object@scale.data
do.scale Rescale the new object@scale.data. FALSE by default
max.cells.per.ident Can be used to downsample the data to a certain max per cell ident. Default is INF.
random.seed Random seed for downsampling
do.clean Only keep object@raw.data and object@data. Cleans out most other slots. Can be useful if you want to start a fresh analysis on just a subset of the data. Also clears out stored clustering results in object@meta.data (any columns containing "res"). Will by default subset the raw.data slot.
subset.raw Also subset object@raw.data
...
Additional arguments to be passed to FetchData (for example, use.imputed=TRUE)

Value

Returns a Seurat object containing only the relevant subset of cells

Examples

pbmc1 <- SubsetData(object = pbmc_small, cells.use = pbmc_small@cell.names[1:40])
pbmc1
SubsetData, code, invert = FALSE)

Arguments
data Matrix or data frame with row names
code Pattern for matching within row names
invert Invert the search?

Value
Returns a subset of data. If invert = TRUE, returns data where rownames do not contain code, otherwise returns data where rownames contain code

Examples

cd_genes <- subsetRow(data = pbmc_small@raw.data, code = 'CD')
head(as.matrix(cd_genes)[, 1:4])

TobitTest

TobitTest(Differential expression testing using Tobit models

Description
Identifies differentially expressed genes between two groups of cells using Tobit models, as proposed in Trapnell et al., Nature Biotechnology, 2014

Usage
TobitTest(object, cells.1, cells.2, genes.use = NULL, print.bar = TRUE, assay.type = "RNA")
**TransferIdent**

**Arguments**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Seurat object</td>
</tr>
<tr>
<td>cells.1</td>
<td>Group 1 cells</td>
</tr>
<tr>
<td>cells.2</td>
<td>Group 2 cells</td>
</tr>
<tr>
<td>genes.use</td>
<td>Genes to test. Default is to use all genes</td>
</tr>
<tr>
<td>print.bar</td>
<td>Print a progress bar once expression testing begins (uses pbapply to do this)</td>
</tr>
<tr>
<td>assay.type</td>
<td>Type of assay to fetch data for (default is RNA)</td>
</tr>
</tbody>
</table>

**Value**

Returns a p-value ranked matrix of putative differentially expressed genes.

**Examples**

```r
pbmc_small
## Not run:
TobitTest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
          cells.2 = WhichCells(object = pbmc_small, ident = 2))
## End(Not run)
```

---

**Description**

Transfers identity class information (or meta data) from one object to another, assuming the same cell barcode names are in each. Can be very useful if you have multiple Seurat objects that share a subset of underlying data.

**Usage**

```r
TransferIdent(object.from, object.to, data.to.transfer = "ident",
              keep.existing = TRUE, add.cell.id1 = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object.from</td>
<td>Seurat object to transfer information from</td>
</tr>
<tr>
<td>object.to</td>
<td>Seurat object to transfer information onto</td>
</tr>
<tr>
<td>data.to.transfer</td>
<td>What data should be transferred over? Default is the identity class (&quot;ident&quot;), but can also include any column in <a href="mailto:object.from@meta.data">object.from@meta.data</a></td>
</tr>
<tr>
<td>keep.existing</td>
<td>For cells in object.to that are not present in object.from, keep existing data? TRUE by default. If FALSE, set to NA.</td>
</tr>
<tr>
<td>add.cell.id1</td>
<td>Prefix to add (followed by an underscore) to cells in object.from. NULL by default, in which case no prefix is added.</td>
</tr>
</tbody>
</table>

---
**Value**

A Seurat object where `object@ident` or `object@meta.data` has been appropriately modified

**Examples**

```r
# Duplicate the test object and assign random new ids to transfer
pbmc_small@ident
pbmc_small2 <- SetIdent(object = pbmc_small, cells.use = pbmc_small@cell.names,
                    ident.use = sample(pbmc_small@ident))
pbmc_small2@ident
```

```r
pbmc_small <- TransferIdent(object.from = pbmc_small2, object.to = pbmc_small)
pbmc_small@ident
```

---

**TSNEPlot**  
*Plot tSNE map*

**Description**

Graphs the output of a tSNE analysis. Cells are colored by their identity class.

**Usage**

```r
TSNEPlot(object, do.label = FALSE, pt.size = 1, label.size = 4,
        cells.use = NULL, colors.use = NULL, ...)
```

**Arguments**

- `object`  
  Seurat object

- `do.label`  
  FALSE by default. If TRUE, plots an alternate view where the center of each cluster is labeled

- `pt.size`  
  Set the point size

- `label.size`  
  Set the size of the text labels

- `cells.use`  
  Vector of cell names to use in the plot.

- `colors.use`  
  Manually set the color palette to use for the points

- `...`  
  Additional parameters to `DimPlot`, for example, which dimensions to plot.

**Details**

This function is a wrapper for `DimPlot`. See `?DimPlot` for a full list of possible arguments which can be passed in here.

**See Also**

`DimPlot`
UpdateSeuratObject

**Examples**

```r
TSNEplot(object = pbmc_small)
```

---

**Description**

Updates Seurat objects to new structure for storing data/calculations.

**Usage**

```r
UpdateSeuratObject(object)
```

**Arguments**

- `object` Seurat object

**Value**

Returns a Seurat object compatible with latest changes

**Examples**

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

## End(Not run)
```

---

ValidateClusters

**Cluster Validation**

**Description**

Methods for validating the legitimacy of clusters using classification. SVMs are used as the basis for the classification. Merging is done based on the connectivity from an SNN graph.

**Usage**

```r
ValidateClusters(object, pc.use = NULL, top.genes = 30, 
min.connectivity = 0.01, acc.cutoff = 0.9, verbose = TRUE)
```
ValidateSpecificClusters

Arguments

- **object**: Seurat object
- **pc_use**: Which PCs to use to define genes in model construction
- **top_genes**: Use the top X genes for each PC in model construction
- **min_connectivity**: Threshold of connectedness for comparison of two clusters
- **acc_cutoff**: Accuracy cutoff for classifier
- **verbose**: Controls whether to display progress and merging results

Value

Returns a Seurat object, object@ident has been updated with new cluster info

Examples

```
pbmc_small
# May throw warnings when cluster sizes are particularly small
## Not run:
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca",
                           dim.use = 1:10, resolution = 1.1, save.SNN = TRUE)
pbmc_small <- ValidateClusters(pbmc_small, pc_use = 1:10)

## End(Not run)
```

ValidateSpecificClusters

*Specific Cluster Validation*

Description

Methods for validating the legitimacy of two specific clusters using classification. SVMs are used as the basis for the classification. Merging is done based on the connectivity from an SNN graph.

Usage

```
ValidateSpecificClusters(object, cluster1 = NULL, cluster2 = 1,
                         pc.use = 2, top.genes = 30, acc.cutoff = 0.9)
```

Arguments

- **object**: Seurat object
- **cluster1**: First cluster to check classification
- **cluster2**: Second cluster to check with classification
- **pc.use**: Which PCs to use for model construction
- **top.genes**: Use the top X genes for model construction
- **acc.cutoff**: Accuracy cutoff for classifier
VariableGenePlot

Description

View variable genes

Usage

VariableGenePlot(object, do.text = TRUE, cex.use = 0.5, 
cex.text.use = 0.5, do.spike = FALSE, pch.use = 16, col.use = "black", 
spike.col.use = "red", plot.both = FALSE, do.contour = TRUE, 
contour.lwd = 3, contour.col = "white", contour.lty = 2, 
x.low.cutoff = 0.1, x.high.cutoff = 8, y.cutoff = 1, 
y.high.cutoff = Inf)

Arguments

object Seurat object
do.text Add text names of variable genes to plot (default is TRUE)
cex.use Point size
cex.text.use Text size
do.spike FALSE by default. If TRUE, color all genes starting with ^ERCC a different color
pch.use Pch value for points
col.use Color to use
spike.col.use if do.spike, color for spike-in genes
plot.both Plot both the scaled and non-scaled graphs.
do.contour Draw contour lines calculated based on all genes

Examples

## Not run:
pbmc_small
pbmc_small <- FindClusters(object = pbmc_small, reduction.type = "pca", 
dims.use = 1:10, resolution = 1.1, save.SNN = TRUE)
pbmc_small <- ValidateSpecificClusters(pbmc_small, cluster1 = 1, 
cluster2 = 2, pc.use = 1:10)

## End(Not run)
VizDimReduction

**Description**

Visualize top genes associated with reduction components

**Usage**

```
VizDimReduction(object, reduction.type = "pca", dims.use = 1:5,
    num.genes = 30, use.full = FALSE, font.size = 0.5, nCol = NULL,
    do.balanced = FALSE)
```

**Arguments**

- `object`: Seurat object
- `reduction.type`: Reduction technique to visualize results for
- `dims.use`: Number of dimensions to display
- `num.genes`: Number of genes to display
- `use.full`: Use reduction values for full dataset (i.e. projected dimensional reduction values)
- `font.size`: Font size
- `nCol`: Number of columns to display
- `do.balanced`: Return an equal number of genes with + and - scores. If FALSE (default), returns the top genes ranked by the scores absolute values

**Value**

Graphical, no return value

**Examples**

```
VizDimReduction(object = pbmc_small)
```
VizICA

Visualize ICA genes

Description
Visualization top genes associated with principal components

Usage
VizICA(object, ics.use = 1:5, num.genes = 30, use.full = FALSE, font.size = 0.5, nCol = NULL, do.balanced = FALSE)

Arguments
- object: Seurat object
- ics.use: Number of ICs to display
- num.genes: Number of genes to display
- use.full: Use full ICA (i.e. the projected ICA, by default FALSE)
- font.size: Font size
- nCol: Number of columns to display
- do.balanced: Return an equal number of genes with both + and - IC scores. If FALSE (by default), returns the top genes ranked by the score’s absolute values

Value
Graphical, no return value

Examples
pbmc_small <- RunICA(object = pbmc_small, ics.compute = 25, print.results = FALSE)
VizICA(object = pbmc_small)

VizPCA

Visualize PCA genes

Description
Visualization top genes associated with principal components

Usage
VizPCA(object, pcs.use = 1:5, num.genes = 30, use.full = FALSE, font.size = 0.5, nCol = NULL, do.balanced = FALSE)
Arguments

- **object**: Seurat object
- **pcs.use**: Number of PCs to display
- **num.genes**: Number of genes to display
- **use.full**: Use full PCA (i.e. the projected PCA, by default FALSE)
- **font.size**: Font size
- **nCol**: Number of columns to display
- **do.balanced**: Return an equal number of genes with both + and - PC scores. If FALSE (by default), returns the top genes ranked by the score’s absolute values

Value

Graphical, no return value

Examples

```R
VizPCA(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.plot, ident.include = NULL, nCol = NULL,
    do.sort = FALSE, y.max = NULL, same.y.lims = FALSE, size.x.use = 16,
    size.y.use = 16, size.title.use = 20, adjust.use = 1,
    point.size.use = 1, cols.use = NULL, group.by = NULL, y.log = FALSE,
    x.lab.rot = FALSE, y.lab.rot = FALSE, legend.position = "right",
    single.legend = TRUE, remove.legend = FALSE, do.return = FALSE,
    return.plotlist = FALSE, ...)
```

Arguments

- **object**: Seurat object
- **features.plot**: Features to plot (gene expression, metrics, PC scores, anything that can be retrieved by FetchData)
- **ident.include**: Which classes to include in the plot (default is all)
- **nCol**: Number of columns if multiple plots are displayed
whichCells

WhichCells

Identify cells matching certain criteria

Description

Returns a list of cells that match a particular set of criteria such as identity class, high/low values for particular PCs, etc.
Usage

WhichCells(object, ident = NULL, ident.remove = NULL, cells.use = NULL, subset.name = NULL, accept.low = -Inf, accept.high = Inf, accept.value = NULL, max.cells.per.ident = Inf, random.seed = 1, ...)

Arguments

object  Seurat object
ident   Identity classes to subset. Default is all identities.
ident.remove  Identity classes to remove. Default is NULL.
cells.use   Subset of cell names
subset.name  Parameter to subset on. Eg, the name of a gene, PC1, a column name in object@meta.data, etc. Any argument that can be retrieved using FetchData
accept.low  Low cutoff for the parameter (default is -Inf)
accept.high  High cutoff for the parameter (default is Inf)
accept.value Returns all cells with the subset name equal to this value
max.cells.per.ident  Can be used to downsample the data to a certain max per cell ident. Default is INF.
random.seed  Random seed for downsampling
...  Additional arguments to be passed to FetchData (for example, use.imputed=TRUE)

Value

A vector of cell names

Examples

WhichCells(object = pbmc_small, ident = 2)

WilcoxDTest

Differential expression using Wilcoxon Rank Sum

Description

Identifies differentially expressed genes between two groups of cells using a Wilcoxon Rank Sum test

Usage

WilcoxDTest(object, cells.1, cells.2, genes.use = NULL, print.bar = TRUE, assay.type = "RNA", ...)
WilcoxDDETest

Arguments

- object: Seurat object
- cells.1: Group 1 cells
- cells.2: Group 2 cells
- genes.use: Genes to use for test
- print.bar: Print a progress bar
- assay.type: Type of assay to perform DE for (default is RNA)
- ...: Extra parameters passed to wilcox.test

Value

Returns a p-value ranked matrix of putative differentially expressed genes.

Examples

```r
pbmc_small
WilcoxDDETest(pbmc_small, cells.1 = WhichCells(object = pbmc_small, ident = 1),
              cells.2 = WhichCells(object = pbmc_small, ident = 2))
```
Index

*Topic datasets
  cc.genes, 23
  pbmc_small, 102

add_samples (Seurat-deprecated), 148
AddImputedScore, 6
addImputedScore (Seurat-deprecated), 148
AddMetaData, 7
addMetaData (Seurat-deprecated), 148
AddSamples, 9
AddSmoothedScore, 10
addSmoothedScore (Seurat-deprecated), 148
AlignSubspace, 11
as.seurat (Convert), 29
as.SingleCellExperiment (Convert), 29
AssessNodes, 12
AssessSplit, 13
AugmentPlot, 14
average.expression (Seurat-deprecated), 148
average.pca (Seurat-deprecated), 148
AverageDetectionRate, 14
AverageExpression, 15
AveragePCA, 16
batch.gene (Seurat-deprecated), 148
BatchGene, 16
BlackAndWhite, 17
BuildClusterTree, 17
buildClusterTree (Seurat-deprecated), 148
BuildRFClassifier, 18
BuildSNNS, 19

calc.insitu (Seurat-deprecated), 148
calcAlignmentMetric, 20
calcNoiseModels (Seurat-deprecated), 148
calcVarExpRatio, 22
calinskiPlot (Seurat-deprecated), 148
CaseMatch, 23
cc.genes, 23
cell.cor.matrix (Seurat-deprecated), 148
CellCycleScoring, 24
CellPlot, 25
cellPlot (Seurat-deprecated), 148
ClassifyCells, 26
cluster.alpha (Seurat-deprecated), 148
CollapseSpeciesExpressionMatrix, 27
ColorTSNESplit, 28
CombineIdent, 29
Convert, 29
CreateSeuratObject, 31
CustomDistance, 32
CustomPalette, 33
DarkTheme, 34
DBclust_dimension (Seurat-deprecated), 148
DBClustDimension, 34
DESeq2DETest, 35, 66
diffExp.test (Seurat-deprecated), 148
DiffExpTest, 36
diffTTest, 37
DimElbowPlot, 38
DimHeatmap, 38
DimPlot, 40
DimTopCells, 42
DimTopGenes, 42
DMEnbed, 43
DMPPlot, 44
doHeatmap, 45
doHeatMap (Seurat-deprecated), 148
doKMeans, 46
doKMeans (Seurat-deprecated), 148
dot.plot (Seurat-deprecated), 148
DotPlot, 47
DotPlotOld, 48
ExpMean, 49
ExpSD, 50
ExpVar, 50
ExtractField, 51

FastWhichCells, 52
feature.heatmap (Seurat-deprecated), 148
feature.plot (Seurat-deprecated), 148
FeatureHeatmap, 52
FeatureLocator, 54
FeaturePlot, 53, 55
fetch.data (Seurat-deprecated), 148
FetchData, 56
FilterCells, 57
find.markers (Seurat-deprecated), 148
find_all.markers (Seurat-deprecated), 148
FindAllMarkers, 58
FindAllMarkersNode, 59
FindClusters, 61
FindConservedMarkers, 63
FindGeneTerms, 64
FindMarkers, 64
FindMarkersNode, 66
FindVariableGenes, 67
fit.gene.k (Seurat-deprecated), 148
fit.gene.mix (Seurat-deprecated), 148
FitGeneK, 69
gene.cor.matrix (Seurat-deprecated), 148
GenePlot, 70
genePlot (Seurat-deprecated), 148
genes.in.cluster (Seurat-deprecated), 148
geneScorePlot (Seurat-deprecated), 148
GenesInCluster, 71
get.centroids (Seurat-deprecated), 148
GetAssayData, 72
GetCellEmbeddings, 72
GetCentroids, 73
GetClusters, 74
GetDimReduction, 75
GetGeneLoadings, 75
GetIdent, 76
getNewScore (Seurat-deprecated), 148

HeatmapNode (Seurat-deprecated), 148
HoverLocator, 77
HTODemux, 77

HTOHeatmap, 79
ICA (Seurat-deprecated), 148
ica (Seurat-deprecated), 148
ICAEmbed, 80
ICALoad, 80
ICAPlot, 81
ICHeatmap, 82
icHeatmap (Seurat-deprecated), 148
ICTopCells, 83
ICTopGenes, 83
ictopGenes (Seurat-deprecated), 148
initial.mapping (Seurat-deprecated), 148
InitialMapping, 84
jackRandom (Seurat-deprecated), 148
JackStraw, 85
jackStraw (Seurat-deprecated), 148
jackStrawFull (Seurat-deprecated), 148
jackStrawMC (Seurat-deprecated), 148
JackStrawPlot, 86
jackStrawPlot (Seurat-deprecated), 148
JoyPlot (Seurat-deprecated), 148
Kclust_dimension (Seurat-deprecated), 148
KClustDimension, 87
KMeansHeatmap, 88
kMeansHeatmap (Seurat-deprecated), 148
LogNormalize, 89
LogVMR, 89

MakeSparse, 90
map.cell (Seurat-deprecated), 148
marker.test (Seurat-deprecated), 148
MarkerTest, 91
MASTDETest, 66, 92
MatrixRowShuffle, 93
MeanVarPlot (Seurat-deprecated), 148
MergeNode, 93
MergeSeurat, 94
MetageneBicorPlot, 95
MinMax, 96
minusc (Seurat-deprecated), 148
minusr (Seurat-deprecated), 148
MultiModal_CCA, 96
MultiModal_CIA, 97

NegBinomDETest, 66, 98
NegBinomRegDETest, 99
NormalizeData, 100
NumberClusters, 100
OldDoHeatmap, 101

pbmc_small, 102
PCA (Seurat-deprecated), 148
pca (Seurat-deprecated), 148
PCAEmbed, 103
PCALoad, 104
PCAPlot, 104
PCAsigGenes, 105
PCElbowPlot, 106
PHeatmap, 106
pcHeatmap (Seurat-deprecated), 148
PCTopCells, 107
pcTopCells (Seurat-deprecated), 148
pcTopGenes, 108
pcTopGenes (Seurat-deprecated), 148
PlotClusterTree, 109
plotClusterTree (Seurat-deprecated), 148
plotNoiseModel (Seurat-deprecated), 148
PoissonDETest, 109
posterior.plot (Seurat-deprecated), 148
PrintAlignSubspaceParams, 110
PrintCalcParams, 111
PrintCalcVarExpRatioParams, 112
PrintCCAParams, 112
PrintDim, 113
PrintDMParams, 114
PrintFindClustersParams, 115
PrintICA, 115
PrintICAParams, 116
PrintPCA, 117
PrintPCAParams, 117
PrintSNNParams, 118
PrintTSNEParams, 119
project.pca (Seurat-deprecated), 148
project.samples (Seurat-deprecated), 148
ProjectDim, 119
ProjectPCA, 120
PurpleAndYellow, 121

Read10X, 122
Read10X_h5, 123
refined.mapping (Seurat-deprecated), 148
RefinedMapping, 123
RegressOut (Seurat-deprecated), 148
RegressOut (Seurat-deprecated), 148
regularScore (Seurat-deprecated), 148
RemoveFromTable, 124
removePC (Seurat-deprecated), 148
rename.ident (Seurat-deprecated), 148
RenameCells, 94, 125, 132
RenameIdent, 126
ReorderIdent, 126
RidgePlot, 127
run_diffusion (Seurat-deprecated), 148
run_tsne (Seurat-deprecated), 148
RunCCA, 128
RunDiffusion, 130
RunICA, 131
RunMulticCA, 132
RunPCA, 133
RunPHATE, 134
RunTSNE, 136
RunUMAP, 138
SampleUMI, 139
SaveClusters, 140
ScaleData, 141
ScaleDataR, 142
set.all.ident (Seurat-deprecated), 148
set.ident (Seurat-deprecated), 148
SetAllIdent, 143
SetAssayData, 144
SetClusters, 145
SetDimReduction, 145
SetIdent, 146
seurat, 147
seurat-class, 147
Seurat-deprecated, 148
Shuffle, 150
spatial.de (Seurat-deprecated), 148
SplitDotPlotGG, 151
SplitObject, 152
StashIdent, 153
SubsetByPredicate, 153
subsetCells (Seurat-deprecated), 148
SubsetColumn, 154
SubsetData, 154
subsetData (Seurat-deprecated), 148
SubsetRow, 156

tnse.plot (Seurat-deprecated), 148
tobit.test (Seurat-deprecated), 148
TobitTest, 156
TransferIdent, 157
tsne.plot (Seurat-deprecated), 148
TSNEPlot, 158
UpdateSeuratObject, 159
ValidateClusters, 159
ValidateSpecificClusters, 160
VariableGenePlot, 161
viz.ica (Seurat-deprecated), 148
viz.pca (Seurat-deprecated), 148
VizClassification (Seurat-deprecated), 148
VizDimReduction, 162
VizICA, 163
VizPCA, 163
VlnPlot, 164
vlnPlot (Seurat-deprecated), 148
which.cells (Seurat-deprecated), 148
WhichCells, 165
WilcoxDETest, 166
writ.table (Seurat-deprecated), 148