

Package ‘SuperCell’

May 7, 2026

Type Package

Title Simplification of scRNA-Seq Data by Merging Together Similar Cells

Version 1.1

Description Aggregates large single-cell data into metacell dataset by merging together gene expression of very similar cells. 'SuperCell' uses 'velocity.R' <doi:10.1038/s41586-018-0414-6> <<https://github.com/velocity-team/velocity.R>> for RNA velocity and 'Weighted-Cluster' <doi:10.12682/lives.2296-1658.2013.24> <<https://mephisto.unige.ch/weightedcluster/>> for weighted clustering on metacells. We also recommend installing 'scater' Bioconductor package <doi:10.18129/B9.bioc.scater> <<https://bioconductor.org/packages/release/bioc/html/scater.html>>.

License GPL-3

BugReports <https://github.com/GfellerLab/SuperCell/issues>

Encoding UTF-8

LazyData true

LazyDataCompression xz

biocViews Software

Additional_repositories <https://mteleman.github.io/drat2>

Imports igraph, RANN, corpcor, weights, Hmisc, Matrix, matrixStats, plyr, irlba, grDevices, patchwork, ggplot2, umap, entropy, Rtsne, dbscan, scales, plotfunctions, proxy, methods, rlang,

RoxygenNote 7.3.3

Suggests SingleCellExperiment, SummarizedExperiment, cowplot, scater, Seurat, knitr, rmarkdown, remotes, bluster, velocity.R, WeightedCluster, testthat (>= 3.0.0)

Depends R (>= 4.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Mariia Bilous [aut],
 Matei Teleman [cre]
Maintainer Matei Teleman <matei.teleman@unil.ch>
Repository CRAN
Date/Publication 2025-12-16 12:20:07 UTC

Contents

anndata_2_supercell	3
build_knn_graph	3
build_knn_graph_nn2	5
cell_lines	6
knn_graph_from_dist	6
metacell2_anndata_2_supercell	7
SCimplify	7
SCimplify_for_velocity	10
SCimplify_from_embedding	10
sc_mixing_score	12
supercell_2_sce	13
supercell_2_Seurat	14
supercell_assign	15
supercell_cluster	16
supercell_DimPlot	17
supercell_estimate_velocity	18
supercell_FindAllMarkers	19
supercell_FindMarkers	20
supercell_GE	21
supercell_GeneGenePlot	22
supercell_GeneGenePlot_single	23
supercell_GE_idx	24
supercell_merge	25
supercell_mergeGE	26
supercell_plot	27
supercell_plot_GE	29
supercell_plot_tSNE	30
supercell_plot_UMAP	30
supercell_prcomp	31
supercell_purity	32
supercell_rescale	33
supercell_silhouette	33
supercell_tSNE	34
supercell_UMAP	34
supercell_VlnPlot	35
supercell_VlnPlot_single	36

Index

38

anndata_2_supercell	<i>Convert Anndata metacell object (Metacell-2 or SEACells) to Super-cell like object</i>
---------------------	---

Description

Convert Anndata metacell object (Metacell-2 or SEACells) to Super-cell like object

Usage

```
anndata_2_supercell(adata, simplification.algo = "unknown")
```

Arguments

adata	anndata object of metacells (for example, the output of <code>collect_metacells()</code> for Metacells or the output of <code>SEACells.core.summarize_by_SEACell</code>) Please, **make sure** , adata has 'uns['sc.obs']' field containing observation information of single-cell data, in particular, a column 'membership' (single-cell assignemnt to metacells)
simplification.algo	metacell construction algorithm (i.e., Metacell2 or SEACells)

Value

a list of super-cell like object (similar to the output of `SCimplify`)

build_knn_graph	<i>Build kNN graph</i>
-----------------	------------------------

Description

Build kNN graph either from distance (from == "dist") or from coordinates (from == "coordinates")

Usage

```
build_knn_graph(
  X,
  k = 5,
  from = c("dist", "coordinates"),
  use.nn2 = TRUE,
  return_neighbors_order = FALSE,
  dist_method = "euclidean",
  cor_method = "pearson",
  p = 2,
  directed = FALSE,
```

```

DoSNN = FALSE,
which.snn = c("bluster", "dbscan"),
pruning = NULL,
kmin = 0,
...
)

```

Arguments

<code>X</code>	either distance or matrix of coordinates (rows are samples and cols are coordinates)
<code>k</code>	kNN parameter
<code>from</code>	from which data type to build kNN network: "dist" if X is a distance (dissimilarity) or "coordinates" if X is a matrix with coordinates as cols and cells as rows
<code>use.nn2</code>	whether use nn2 method to build kNN network faster (available only for "coordinates" option)
<code>return_neighbors_order</code>	whether return order of neighbors (not available for nn2 option)
<code>dist_method</code>	method to compute dist (if X is a matrix of coordinates) available: c("cor", "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski")
<code>cor_method</code>	if distance is computed as correlation (dist_method == "cor"), which type of correlation to use (available: "pearson", "kendall", "spearman")
<code>p</code>	p param in "dist" function
<code>directed</code>	whether to build a directed graph
<code>DoSNN</code>	whether to apply shared nearest neighbors (default is FALSE)
<code>which.snn</code>	whether to use neighborsToSNNGraph or sNN for sNN graph construction
<code>pruning</code>	quantile to perform edge pruning (default is NULL - no pruning applied) based on PCA distance distribution
<code>kmin</code>	keep at least kmin edges in single-cell graph when pruning applied (idnored if <code>is.null(pruning)</code>)
<code>...</code>	other parameters of neighborsToSNNGraph or sNN

Value

a list with components

- `graph.knn` - igraph object
- `order` - Nxk matrix with indices of k nearest neighbors ordered by relevance (from 1st to k-th)

build_knn_graph_nn2 *Build kNN graph using RANN::nn2 (used in "build_knn_graph")*

Description

Build kNN graph using RANN::nn2 (used in "build_knn_graph")

Usage

```
build_knn_graph_nn2(
  X,
  k = min(5, ncol(X)),
  mode = "all",
  DoSNN = FALSE,
  which.snn = c("bluster", "dbscan"),
  pruning = NULL,
  kmin = 0,
  ...
)
```

Arguments

X	matrix of coordinates (rows are samples and cols are coordinates)
k	kNN parameter
mode	mode of graph_from_adj_list ('all' – undirected graph, 'out' – directed graph)
DoSNN	whether to apply shared nearest neighbors (default is FALSE)
which.snn	whether to use neighborsToSNNGraph or sNN for sNN graph construction
pruning	quantile to perform edge pruning (default is NULL - no pruning applied) based on PCA distance distribution
kmin	keep at least kmin edges in single-cell graph when pruning applied (idnored if <code>is.null(pruning)</code>)
...	other parameters of neighborsToSNNGraph or sNN

Value

a list with components

- graph.knn - igraph object

cell_lines	<i>Cancer cell lines dataset</i>
------------	----------------------------------

Description

ScRNA-seq data of 5 cancer cell lines from [Tian et al., 2019](<https://doi.org/10.1038/s41592-019-0425-8>).

Usage

```
cell_lines
```

Format

A list with gene expression (i.e., log-normalized counts) (GE), and metadata data (meta):

GE gene expression (log-normalized counts) matrix

meta cells metadata (cell line annotation)

Details

Data available at authors' [GitHub](https://github.com/LuyiTian/sc_mixology/blob/master/data/) under file name *sincell_with_class_5cl.Rdata*.

Source

[doi:10.1038/s4159201904258](https://doi.org/10.1038/s4159201904258)

knn_graph_from_dist	<i>Build kNN graph from distance (used in "build_knn_graph")</i>
---------------------	--

Description

Build kNN graph from distance (used in "build_knn_graph")

Usage

```
knn_graph_from_dist(D, k = 5, return_neighbors_order = TRUE, mode = "all")
```

Arguments

D	dist matrix or dist object (preferentially)
k	kNN parameter
return_neighbors_order	whether return order of neighbors (not available for nn2 option)
mode	mode of graph_from_adj_list ('all' – undirected graph, 'out' – directed graph)

Value

a list with components

- graph.knn - igraph object
- order - Nxk matrix with indices of k nearest neighbors ordered by relevance (from 1st to k-th)

metacell2_anndata_2_supercell

Convert Metacells (Metacell-2) to Super-cell like object

Description

Convert Metacells (Metacell-2) to Super-cell like object

Usage

```
metacell2_anndata_2_supercell(adata, obs.sc)
```

Arguments

adata	anndata object of metacells (the output of <code>collect_metacells()</code>)
obs.sc	a dataframe of the single-cell anndata object used to compute metacells (anndata after applying <code>divide_and_conquer_pipeline()</code> function)

Value

a list of super-cell like object (similar to the output of `SCimplify`)

SCimplify

Detection of metacells with the SuperCell approach

Description

This function detects metacells (former super-cells) from single-cell gene expression matrix

Usage

```
SCimplify(
  X,
  genes.use = NULL,
  genes.exclude = NULL,
  cell.annotation = NULL,
  cell.split.condition = NULL,
  n.var.genes = min(1000, nrow(X)),
  gamma = 10,
```

```

k.knn = 5,
do.scale = TRUE,
n.pc = 10,
fast.pca = TRUE,
do.approx = FALSE,
approx.N = 20000,
block.size = 10000,
seed = 12345,
igraph.clustering = c("walktrap", "louvain"),
return.singlecell.NW = TRUE,
return.hierarchical.structure = TRUE,
...
)

```

Arguments

<code>X</code>	log-normalized gene expression matrix with rows to be genes and cols to be cells
<code>genes.use</code>	a vector of genes used to compute PCA
<code>genes.exclude</code>	a vector of genes to be excluded when computing PCA
<code>cell.annotation</code>	a vector of cell type annotation, if provided, metacells that contain single cells of different cell type annotation will be split in multiple pure metacell (may result in slightly larger number of metacells than expected with a given gamma)
<code>cell.split.condition</code>	a vector of cell conditions that must not be mixed in one metacell. If provided, metacells will be split in condition-pure metacell (may result in significantly(!) larger number of metacells than expected)
<code>n.var.genes</code>	if "genes.use" is not provided, "n.var.genes" genes with the largest variation are used
<code>gamma</code>	graining level of data (proportion of number of single cells in the initial dataset to the number of metacells in the final dataset)
<code>k.knn</code>	parameter to compute single-cell kNN network
<code>do.scale</code>	whether to scale gene expression matrix when computing PCA
<code>n.pc</code>	number of principal components to use for construction of single-cell kNN network
<code>fast.pca</code>	use irlba as a faster version of <code>prcomp</code> (one used in Seurat package)
<code>do.approx</code>	compute approximate kNN in case of a large dataset (>50'000)
<code>approx.N</code>	number of cells to subsample for an approximate approach
<code>block.size</code>	number of cells to map to the nearest metacell at the time (for approx coarse-graining)
<code>seed</code>	seed to use to subsample cells for an approximate approach
<code>igraph.clustering</code>	clustering method to identify metacells (available methods "walktrap" (default) and "louvain" (not recommended, gamma is ignored)).


```

return.singlecell.NW
    whether return single-cell network (which consists of approx.N if "do.approx"
    or all cells otherwise)
return.hierarchical.structure
    whether return hierarchical structure of metacell
...
    other parameters of build\_knn\_graph function

```

Value

a list with components

- `graph.supercells` - igraph object of a simplified network (number of nodes corresponds to number of metacells)
- `membership` - assignment of each single cell to a particular metacell
- `graph.singlecells` - igraph object (kNN network) of single-cell data
- `supercell_size` - size of metacells (former super-cells)
- `gamma` - requested graining level
- `N.SC` - number of obtained metacells
- `genes.use` - used genes
- `do.approx` - whether approximate coarse-graining was performed
- `n.pc` - number of principal components used for metacells construction
- `k.knn` - number of neighbors to build single-cell graph
- `sc.cell.annotation.` - single-cell cell type annotation (if provided)
- `sc.cell.split.condition.` - single-cell split condition (if provided)
- `SC.cell.annotation.` - super-cell cell type annotation (if was provided for single cells)
- `SC.cell.split.condition.` - super-cell split condition (if was provided for single cells)

Examples

```

data(cell_lines) # list with GE - gene expression matrix (logcounts), meta - cell meta data
GE <- cell_lines$GE

```

```

SC <- SCimplify(GE, # log-normalized gene expression matrix
               gamma = 20, # graining level
               n.var.genes = 1000,
               k.knn = 5, # k for kNN algorithm
               n.pc = 10, # number of principal components to use
               do.approx = TRUE) #

```

SCimplify_for_velocity

Construct super-cells from spliced and un-spliced matrices

Description

Construct super-cells from spliced and un-spliced matrices

Usage

```
SCimplify_for_velocity(emat, nmat, gamma = NULL, membership = NULL, ...)
```

Arguments

emat	spliced (exonic) count matrix
nmat	unspliced (nascent) count matrix
gamma	graining level of data (proportion of number of single cells in the initial dataset to the number of super-cells in the final dataset)
membership	metacell membership vector (if provided, will be used for emat, nmat metacell matrices averaging)
...	other parameters from SCimplify

Value

list containing vector of membership, spliced count and un-spliced count matrices

SCimplify_from_embedding

Detection of metacells with the SuperCell approach from low dim representation

Description

This function detects metacells (former super-cells) from single-cell gene expression matrix

Usage

```
SCimplify_from_embedding(
  X,
  cell.annotation = NULL,
  cell.split.condition = NULL,
  gamma = 10,
  k.knn = 5,
  n.pc = 10,
```

```

do.approx = FALSE,
approx.N = 20000,
block.size = 10000,
seed = 12345,
igraph.clustering = c("walktrap", "louvain"),
return.singlecell.NW = TRUE,
return.hierarchical.structure = TRUE,
...
)

```

Arguments

<code>X</code>	low dimensional embedding matrix with rows to be cells and cols to be low-dim components
<code>cell.annotation</code>	a vector of cell type annotation, if provided, metacells that contain single cells of different cell type annotation will be split in multiple pure metacell (may result in slightly larger numbe of metacells than expected with a given gamma)
<code>cell.split.condition</code>	a vector of cell conditions that must not be mixed in one metacell. If provided, metacells will be split in condition-pure metacell (may result in significantly(!) larger number of metacells than expected)
<code>gamma</code>	graining level of data (proportion of number of single cells in the initial dataset to the number of metacells in the final dataset)
<code>k.knn</code>	parameter to compute single-cell kNN network
<code>n.pc</code>	number of principal components to use for construction of single-cell kNN network
<code>do.approx</code>	compute approximate kNN in case of a large dataset (>50'000)
<code>approx.N</code>	number of cells to subsample for an approximate approach
<code>block.size</code>	number of cells to map to the nearest metacell at the time (for approx coarse-graining)
<code>seed</code>	seed to use to subsample cells for an approximate approach
<code>igraph.clustering</code>	clustering method to identify metacells (available methods "walktrap" (default) and "louvain" (not recommended, gamma is ignored)).
<code>return.singlecell.NW</code>	whether return single-cell network (which consists of approx.N if "do.approx" or all cells otherwise)
<code>return.hierarchical.structure</code>	whether return hierarchical structure of metacell
<code>...</code>	other parameters of build_knn_graph function

Value

a list with components

- graph.supercells - igraph object of a simplified network (number of nodes corresponds to number of metacells)
- membership - assignment of each single cell to a particular metacell
- graph.singlecells - igraph object (kNN network) of single-cell data
- supercell_size - size of metacells (former super-cells)
- gamma - requested graining level
- N.SC - number of obtained metacells
- genes.use - used genes (NA due to low-dim representation)
- do.approx - whether approximate coarse-graining was performed
- n.pc - number of principal components used for metacells construction
- k.knn - number of neighbors to build single-cell graph
- sc.cell.annotation. - single-cell cell type annotation (if provided)
- sc.cell.split.condition. - single-cell split condition (if provided)
- SC.cell.annotation. - super-cell cell type annotation (if was provided for single cells)
- SC.cell.split.condition. - super-cell split condition (if was provided for single cells)

sc_mixing_score

Compute mixing of single-cells within supercell

Description

Compute mixing of single-cells within supercell

Usage

```
sc_mixing_score(SC, clusters)
```

Arguments

SC	super-cell object (output of SCimplify function)
clusters	vector of clustering assignment (reference assignment)

Value

a vector of single-cell mixing within super-cell it belongs to, which is defined as: 1 - proportion of cells of the same annotation (e.g., cell type) within the same super-cell With 0 meaning that super-cell consists of single cells from one cluster (reference assignment) and higher values correspond to higher cell type mixing within super-cell

supercell_2_sce *Super-cells to SingleCellExperiment object*

Description

This function transforms super-cell gene expression and super-cell partition into [SingleCellExperiment](#) object

Usage

```
supercell_2_sce(
  SC.GE,
  SC,
  fields = c(),
  var.genes = NULL,
  do.preproc = TRUE,
  is.log.normalized = TRUE,
  do.center = TRUE,
  do.scale = TRUE,
  ncomponents = 50
)
```

Arguments

SC.GE	gene expression matrix with genes as rows and cells as columns
SC	super-cell (output of SCimplify function)
fields	which fields of SC to use as cell metadata
var.genes	set of genes used as a set of variable features of SingleCellExperiment (by default is the set of genes used to generate super-cells)
do.preproc	whether to do preprocessing, including data normalization, scaling, HVG, PCA, nearest neighbors, TRUE by default, change to FALSE to speed up conversion
is.log.normalized	whether SC.GE is log-normalized counts. If yes, then SingleCellExperiment field assay name = 'logcounts' else assay name = 'counts'
do.center	whether to center gene expression matrix to compute PCA
do.scale	whether to scale gene expression matrix to compute PCA
ncomponents	number of principal components to compute

Value

[SingleCellExperiment](#) object

Examples

```

data(cell_lines)
SC      <- SCimplify(cell_lines$GE, gamma = 20)
SC$idnt <- supercell_assign(clusters = cell_lines$meta, supercell_membership = SC$membership)
SC.GE   <- supercell_GE(cell_lines$GE, SC$membership)
sce     <- supercell_2_sce(SC.GE = SC.GE, SC = SC, fields = c("idnt"))

```

supercell_2_Seurat *Super-cells to Seurat object*

Description

This function transforms super-cell gene expression and super-cell partition into [Seurat](#) object

Usage

```

supercell_2_Seurat(
  SC.GE,
  SC,
  fields = c(),
  var.genes = NULL,
  do.preproc = TRUE,
  is.log.normalized = TRUE,
  do.center = TRUE,
  do.scale = TRUE,
  N.comp = NULL,
  output.assay.version = "v4"
)

```

Arguments

SC.GE	gene expression matrix with genes as rows and cells as columns
SC	super-cell (output of SCimplify function)
fields	which fields of SC to use as cell metadata
var.genes	set of genes used as a set of variable features of Seurat (by default is the set of genes used to generate super-cells), ignored if !do.preproc
do.preproc	whether to do preprocessing, including data normalization, scaling, HVG, PCA, nearest neighbors, TRUE by default, change to FALSE to speed up conversion
is.log.normalized	whether SC.GE is log-normalized counts. If yes, then Seurat field data is replaced with counts after normalization (see 'Details' section), ignored if !do.preproc
do.center	whether to center gene expression matrix to compute PCA, ignored if !do.preproc
do.scale	whether to scale gene expression matrix to compute PCA, ignored if !do.preproc

`N.comp` number of principal components to use for construction of single-cell kNN network, ignored if `!do.preproc`

`output.assay.version` version of the seurat assay in output, `"v4"` by default, `"v5"` requires [Seurat v5](#) installed.

Details

Since the input of [CreateSeuratObject](#) should be unnormalized count matrix (UMIs or TPMs, see [CreateSeuratObject](#)). Thus, we manually set field ``assays$RNA@data`` to `SC.GE` if `is.log.normalized == TRUE`. Avoid running [NormalizeData](#) for the obtained Seurat object, otherwise this will overwrite field ``assays$RNA@data``. If you have run [NormalizeData](#), then make sure to replace ``assays$RNA@data`` with correct matrix by running ``your_seurat@assays$RNA@data <- your_seurat@assays$RNA@counts``.

Since super-cells have different size (consist of different number of single cells), we use sample-weighted algorithms for all possible steps of the downstream analysis, including scaling and dimensionality reduction. Thus, generated Seurat object comes with the results of sample-wighted scaling (available as ``your_seurat@assays$RNA@scale.data`` or ``your_seurat@assays$RNA@misc[["scale.data.weighted"]`` to reproduce if the first one has been overwritten) and PCA (available as ``your_seurat@reductions$pca`` or ``your_seurat@reductions$pca_weighted`` to reproduce if the first one has been overwritten).

Value

[Seurat](#) object

Examples

```
data(cell_lines)
SC <- SCimplify(X=cell_lines$GE, gamma = 20)
SC$ident <- supercell_assign(clusters = cell_lines$meta, supercell_membership = SC$membership)
SC.GE <- supercell_GE(cell_lines$GE, SC$membership)
m.seurat <- supercell_2_Seurat(SC.GE = SC.GE, SC = SC, fields = c("ident"))
```

<code>supercell_assign</code>	<i>Assign super-cells to the most abundant cluster</i>
-------------------------------	--

Description

Assign super-cells to the most abundant cluster

Usage

```
supercell_assign(
  clusters,
  supercell_membership,
  method = c("jaccard", "relative", "absolute")
)
```

Arguments

- clusters a vector of clustering assignment
- supercell_membership a vector of assignment of single-cell data to super-cells (membership field of [SCimplify](#) function output)
- method method to define the most abundant cell cluster within super-cells. Available: "jaccard" (default), "relative", "absolute".
- jaccard - assigns super-cell to cluster with the maximum jaccard coefficient (recommended)
 - relative - assigns super-cell to cluster with the maximum relative abundance (normalized by cluster size), may result in assignment of super-cells to poorly represented (small) cluster due to normalization
 - absolute - assigns super-cell to cluster with the maximum absolute abundance within super-cell, may result in disappearance of poorly represented (small) clusters

Value

a vector of super-cell assignment to clusters

supercell_cluster *Cluster super-cell data*

Description

Cluster super-cell data

Usage

```
supercell_cluster(
  D,
  k = 5,
  supercell_size = NULL,
  algorithm = c("hclust", "PAM"),
  method = NULL,
  return.hcl = TRUE
)
```

Arguments

- D a dissimilarity matrix or a dist object
- k number of clusters
- supercell_size a vector with supercell size (ordered the same way as in D)
- algorithm which algorithm to use to compute clustering: "hclust" (default) or "PAM" (see [wcKMedoids](#))

method	which method of algorithm to use: <ul style="list-style-type: none"> • for "hclust": "ward.D", "ward.D2" (default), "single", "complete", "average", "mcquitty", "median" or "centroid", (see hclust) • for "PAM": "KMedoids", "PAM" or "PAMonce" (default), (see wcKMedoids)
return.hcl	whether to return a result of "hclust" (only for "hclust" algorithm)

Value

a list with components

- clustering - vector of clustering assignment of super-cells
- algo - the algorithm used
- method - method used with an algorithm
- hlc - [hclust](#) result (only for "hclust" algorithm when return.hcl is TRUE)

supercell_DimPlot *Plot metacell 2D plot (PCA, UMAP, tSNE etc)*

Description

Plots 2d representation of metacells

Usage

```
supercell_DimPlot(
  SC,
  groups = NULL,
  dim.name = "PCA",
  dim.1 = 1,
  dim.2 = 2,
  color.use = NULL,
  asp = 1,
  alpha = 0.7,
  title = NULL,
  do.sqtr.rescale = FALSE
)
```

Arguments

SC	SuperCell computed metacell object (the output of SCimplify)
groups	an assignment of metacells to any group (for plotting in different colors)
dim.name	name of the dimensionality reduction to plot (must be a field in SC)
dim.1	dimension to plot on X-axis
dim.2	dimension to plot on Y-axis

color.use	colros to use for groups, if NULL, an automatic palette of colors will be applied
asp	aspect ratio
alpha	a rotation of the layout (either provided or computed)
title	a title of a plot
do.sqtr.rescale	whether to sqrt-scale node size (to balance plot if some metacells are large and covers smaller metacells)

Value

`ggplot`

supercell_estimate_velocity

Run RNavelocity for super-cells (slightly modified from <https://github.com/velocyto-team/velocyto.R>) Not yet adjusted for super-cell size (not sample-weighted)

Description

Run RNavelocity for super-cells (slightly modified from <https://github.com/velocyto-team/velocyto.R>) Not yet adjusted for super-cell size (not sample-weighted)

Usage

```
supercell_estimate_velocity(
  emat,
  nmat,
  smat = NULL,
  membership = NULL,
  supercell_size = NULL,
  do.run.avegaring = (ncol(emat) == length(membership)),
  kCells = 10,
  ...
)
```

Arguments

emat	spliced (exonic) count matrix (see https://github.com/velocyto-team/velocyto.R)
nmat	unspliced (nascent) count matrix (https://github.com/velocyto-team/velocyto.R)
smat	optional spanning read matrix (used in offset calculations) (https://github.com/velocyto-team/velocyto.R)
membership	supercell membership ('membership' field of SCimplify)

`supercell_size` a vector with supercell size (if `emat` and `nmat` provided at super-cell level)
`do.run.avegaring` whether to run averaging of `emat` & `nmat` (if `nmat` provided at a single-cell level)
`kCells` number of k nearest neighbors (NN) to use in slope calculation smoothing (see <https://github.com/velocyto-team/velocyto.R>)
`...` other parameters from <https://github.com/velocyto-team/velocyto.R>

Value

results of <https://github.com/velocyto-team/velocyto.R> plus metacell size vector

`supercell_FindAllMarkers`

Differential expression analysis of supep-cell data. Most of the parameters are the same as in Seurat [FindAllMarkers](#) (for simplicity)

Description

Differential expression analysis of supep-cell data. Most of the parameters are the same as in Seurat [FindAllMarkers](#) (for simplicity)

Usage

```

supercell_FindAllMarkers(
  ge,
  clusters,
  supercell_size = NULL,
  genes.use = NULL,
  logfc.threshold = 0.25,
  min.expr = 0,
  min.pct = 0.1,
  seed = 12345,
  only.pos = FALSE,
  return.extra.info = FALSE,
  do.bootstrapping = FALSE
)

```

Arguments

`ge` gene expression matrix for super-cells (rows - genes, cols - super-cells)
`clusters` a vector with clustering information (ordered the same way as in `ge`)
`supercell_size` a vector with supercell size (ordered the same way as in `ge`)
`genes.use` set of genes to test. Default – all genes in `ge`
`logfc.threshold` log fold change threshold for genes to be considered in the further analysis

min.expr	minimal expression (default 0)
min.pct	remove genes with lower percentage of detection from the set of genes which will be tested
seed	random seed to use
only.pos	whether to compute only positive (upregulated) markers
return.extra.info	whether to return extra information about test and its statistics. Default is FALSE.
do.bootstrapping	whether to perform bootstrapping when computing standard error and p-value in wtd.t.test

Value

list of results of [supercell_FindMarkers](#)

supercell_FindMarkers *Differential expression analysis of supep-cell data. Most of the parameters are the same as in Seurat [FindMarkers](#) (for simplicity)*

Description

Differential expression analysis of supep-cell data. Most of the parameters are the same as in Seurat [FindMarkers](#) (for simplicity)

Usage

```
supercell_FindMarkers(
  ge,
  supercell_size = NULL,
  clusters,
  ident.1,
  ident.2 = NULL,
  genes.use = NULL,
  logfc.threshold = 0.25,
  min.expr = 0,
  min.pct = 0.1,
  seed = 12345,
  only.pos = FALSE,
  return.extra.info = FALSE,
  do.bootstrapping = FALSE
)
```

Arguments

<code>ge</code>	gene expression matrix for super-cells (rows - genes, cols - super-cells)
<code>supercell_size</code>	a vector with supercell size (ordered the same way as in <code>ge</code>)
<code>clusters</code>	a vector with clustering information (ordered the same way as in <code>ge</code>)
<code>ident.1</code>	name(s) of cluster for which markers are computed
<code>ident.2</code>	name(s) of clusters for comparison. If NULL (default), then all the other clusters used
<code>genes.use</code>	set of genes to test. Default – all genes in <code>ge</code>
<code>logfc.threshold</code>	log fold change threshold for genes to be considered in the further analysis
<code>min.expr</code>	minimal expression (default 0)
<code>min.pct</code>	remove genes with lower percentage of detection from the set of genes which will be tested
<code>seed</code>	random seed to use
<code>only.pos</code>	whether to compute only positive (upregulated) markers
<code>return.extra.info</code>	whether to return extra information about test and its statistics. Default is FALSE.
<code>do.bootstrapping</code>	whether to perform bootstrapping when computing standard error and p-value in wtd.t.test

Value

a matrix with a test name (t-test), statistics, adjusted p-values, logFC, percentage of detection in each ident and mean expression

supercell_GE	<i>Simplification of scRNA-seq dataset</i>
--------------	--

Description

This function converts (i.e., averages or sums up) gene-expression matrix of single-cell data into a gene expression matrix of metacells

Usage

```
supercell_GE(
  ge,
  groups,
  mode = c("average", "sum"),
  weights = NULL,
  do.median.norm = FALSE
)
```

Arguments

ge	gene expression matrix (or any coordinate matrix) with genes as rows and cells as cols
groups	vector of membership (assignment of single-cell to metacells)
mode	string indicating whether to average or sum up 'ge' within metacells
weights	vector of a cell weight (NULL by default), used for computing average gene expression withing cluster of metaells
do.median.norm	whether to normalize by median value (FALSE by default)

Value

a matrix of simplified (averaged withing groups) data with ncol equal to number of groups and nrows as in the initial dataset

supercell_GeneGenePlot

Gene-gene correlation plot

Description

Plots gene-gene expression and computes their correaltion

Usage

```
supercell_GeneGenePlot(
  ge,
  gene_x,
  gene_y,
  supercell_size = NULL,
  clusters = NULL,
  color.use = NULL,
  idents = NULL,
  pt.size = 1,
  alpha = 0.9,
  x.max = NULL,
  y.max = NULL,
  same.x.lims = FALSE,
  same.y.lims = FALSE,
  ncol = NULL,
  combine = TRUE,
  sort.by.corr = TRUE
)
```

Arguments

<code>ge</code>	a gene expression matrix of super-cells (ncol same as number of super-cells)
<code>gene_x</code>	gene or vector of genes (if vector, has to be the same length as <code>gene_y</code>)
<code>gene_y</code>	gene or vector of genes (if vector, has to be the same length as <code>gene_x</code>)
<code>supercell_size</code>	a vector with supercell size (ordered the same way as in <code>ge</code>)
<code>clusters</code>	a vector with clustering information (ordered the same way as in <code>ge</code>)
<code>color.use</code>	colors for idents
<code>idents</code>	idents (clusters) to plot (default all)
<code>pt.size</code>	point size (if supercells have identical sizes)
<code>alpha</code>	transparency
<code>x.max</code>	max of x axis
<code>y.max</code>	max of y axis
<code>same.x.lims</code>	same x axis for all plots
<code>same.y.lims</code>	same y axis for all plots
<code>ncol</code>	number of columns in combined plot
<code>combine</code>	combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot
<code>sort.by.corr</code>	whether to sort plots by absolute value of correlation (first plot genes with largest (anti-)correlation)

Value

a list with components

- `p` - is a combined ggplot or list of ggplots if `combine = TRUE`
- `w.cor` - weighted correlation between genes

a list, where

`supercell_GeneGenePlot_single`

Plot Gene-gene correlation plot for 1 feature

Description

Used for [supercell_GeneGenePlot](#)

Usage

```

supercell_GeneGenePlot_single(
  ge_x,
  ge_y,
  gene_x_name,
  gene_y_name,
  supercell_size = NULL,
  clusters = NULL,
  color.use = NULL,
  x.max = NULL,
  y.max = NULL,
  pt.size = 1,
  alpha = 0.9
)

```

Arguments

ge_x	first gene expression vector (same length as number of super-cells)
ge_y	second gene expression vector (same length as number of super-cells)
gene_x_name	name of gene x
gene_y_name	name of gene y
supercell_size	a vector with supercell size (ordered the same way as in ge)
clusters	a vector with clustering information (ordered the same way as in ge)
color.use	colors for idents
x.max	max of x axis
y.max	max of y axis
pt.size	point size (0 by default)
alpha	transparency of dots

supercell_GE_idx	<i>Simplification of scRNA-seq dataset (old version, not used since 12.02.2021)</i>
------------------	---

Description

This function converts gene-expression matrix of single-cell data into a gene expression matrix of super-cells

Usage

```
supercell_GE_idx(ge, groups, weights = NULL, do.median.norm = FALSE)
```


Arguments

ge	gene expression matrix (or any coordinate matrix) with genes as rows and cells as cols
groups	vector of membership (assignment of single-cell to super-cells)
weights	vector of a cell weight (NULL by default), used for computing average gene expression withing cluster of super-cells
do.median.norm	whether to normalize by median value (FALSE by default)

Value

a matrix of simplified (averaged withing groups) data with ncol equal to number of groups and nrows as in the initial dataset

supercell_merge	<i>Merging independent SuperCell objects</i>
-----------------	--

Description

This function merges independent SuperCell objects

Usage

```
supercell_merge(SCs, fields = c())
```

Arguments

SCs	list of SuperCell objects (results of SCimplify)
fields	which additional fields (e.g., metadata) of the the SuperCell objects to keep when merging

Value

a list with components

- membership - assignment of each single cell to a particular metacell
- cell.ids - the original ids of single-cells
- supercell_size - size of metacells (former super-cells)
- gamma - graining level of the merged object (estimated as an average size of metacells as the independent SuperCell objects might have different graining levels)
- N.SC - number of obtained metacells

Examples

```

data(cell_lines) # list with GE - gene expression matrix (logcounts), meta - cell meta data
GE <- cell_lines$GE
cell.meta <- cell_lines$meta

cell.idx.HCC827 <- which(cell.meta == "HCC827")
cell.idx.H838 <- which(cell.meta == "H838")

SC.HCC827 <- SCsimplify(GE[,cell.idx.HCC827], # log-normalized gene expression matrix
  gamma = 20, # graining level
  n.var.genes = 1000,
  k.knn = 5, # k for kNN algorithm
  n.pc = 10) # number of principal components to use
SC.HCC827$cell.line <- supercell_assign(
  cell.meta[cell.idx.HCC827],
  supercell_membership = SC.HCC827$membership)

SC.H838 <- SCsimplify(GE[,cell.idx.H838], # log-normalized gene expression matrix
  gamma = 30, # graining level
  n.var.genes = 1000, # number of top var genes to use for the dim reduction
  k.knn = 5, # k for kNN algorithm
  n.pc = 15) # number of prncipal components to use
SC.H838$cell.line <- supercell_assign(
  cell.meta[cell.idx.H838],
  supercell_membership = SC.H838$membership)

SC.merged <- supercell_merge(list(SC.HCC827, SC.H838), fields = c("cell.line"))

# compute metacell gene expression for SC.HCC827
SC.GE.HCC827 <- supercell_GE(GE[, cell.idx.HCC827], groups = SC.HCC827$membership)
# compute metacell gene expression for SC.H838
SC.GE.H838 <- supercell_GE(GE[, cell.idx.H838], groups = SC.H838$membership)
# merge GE matrices
SC.GE.merged <- supercell_mergeGE(list(SC.GE.HCC827, SC.GE.H838))

```

supercell_mergeGE	<i>Merging metacell gene expression matrices from several independent SuperCell objects</i>
-------------------	---

Description

This function merges independent SuperCell objects

Usage

```
supercell_mergeGE(SC.GEs)
```

Arguments

SC.GEs list of metacell gene expression matrices (result of [supercell_GE](#)), make sure the order of the gene expression matrices is the same as in the call of [supercell_merge](#)

Value

a merged matrix of gene expression

Examples

```
# see examples in \link{supercell_merge}
```

supercell_plot	<i>Plot metacell NW</i>
----------------	-------------------------

Description

Plot metacell NW

Usage

```
supercell_plot(  
  SC.nw,  
  group = NULL,  
  color.use = NULL,  
  lay.method = c("nicely", "fr", "components", "drl", "graphopt"),  
  lay = NULL,  
  alpha = 0,  
  seed = 12345,  
  main = NA,  
  do.frames = TRUE,  
  do.extra.log.rescale = FALSE,  
  do.directed = FALSE,  
  log.base = 2,  
  do.extra.sqtr.rescale = FALSE,  
  frame.color = "black",  
  weights = NULL,  
  min.cell.size = 0,  
  return.meta = FALSE  
)
```

Arguments

SC.nw	a super-cell (metacell) network (a field supercell_network of the output of SCimplify)
group	an assignment of metacells to any group (for plotting in different colors)
color.use	colros to use for groups, if NULL, an automatic palette of colors will be applied
lay.method	method to compute layout of the network (for the moment there several available: "nicely" for layout_nicely and "fr" for layout_with_fr , "components" for layout_components , "drl" for layout_with_drl , "graphopt" for layout_with_graphopt). If your dataset has clear clusters, use "components"
lay	a particular layout of a graph to plot (in is not NULL, lay.method is ignored and new layout is not computed)
alpha	a rotation of the layout (either provided or computed)
seed	a random seed used to compute graph layout
main	a title of a plot
do.frames	whether to keep vertex.frames in the plot
do.extra.log.rescale	whether to log-scale node size (to balance plot if some metacells are large and covers smaller metacells)
do.directed	whether to plot edge direction
log.base	base with thich to log-scale node size
do.extra.sqtr.rescale	whether to sqrt-scale node size (to balance plot if some metacells are large and covers smaller metacells)
frame.color	color of node frames, black by default
weights	edge weights used for some layout algorithms
min.cell.size	do not plot cells with smaller size
return.meta	whether to return all the meta data

Value

plot of a super-cell network

Examples

```
data(cell_lines) # list with GE - gene expression matrix (logcounts), meta - cell meta data
GE <- cell_lines$GE
cell.meta <- cell_lines$meta

SC <- SCimplify(GE, # gene expression matrix
               gamma = 20) # graining level

# Assign metacell to a cell line
SC2cellline <- supercell_assign(
  clusters = cell.meta, # single-cell assignment to cell lines
```

```

    supercell_membership = SC$membership) # single-cell assignment to metacells

# Plot metacell network colored by cell line
supercell_plot(SC$graph.supercells, # network
               group = SC2cellline, # group assignment
               main = "Metacell colored by cell line assignment",
               lay.method = 'nicely')

```

supercell_plot_GE *Plot super-cell NW colored by an expression of a gene (gradient color)*

Description

Plot super-cell NW colored by an expression of a gene (gradient color)

Usage

```

supercell_plot_GE(
  SC.nw,
  ge,
  color.use = c("gray", "blue"),
  n.color.gradient = 10,
  main = NA,
  legend.side = 4,
  gene.name = NULL,
  ...
)

```

Arguments

SC.nw	a super-cell network (a field <code>supercell_network</code> of the output of SCimplify)
ge	a gene expression vector (same length as number of super-cells)
color.use	colors of gradient
n.color.gradient	number of bins of the gradient, default is 10
main	plot title
legend.side	a side parameter of gradientLegend function (default is 4)
gene.name	name of gene of for which gene expression is plotted
...	rest of the parameters of supercell_plot function

Value

plot of a super-cell network with color representing an expression level

supercell_plot_tSNE *Plot super-cell tSNE (Use [supercell_DimPlot](#) instead) Plots super-cell tSNE (result of [supercell_tSNE](#))*

Description

Plot super-cell tSNE (Use [supercell_DimPlot](#) instead) Plots super-cell tSNE (result of [supercell_tSNE](#))

Usage

```
supercell_plot_tSNE(  
  SC,  
  groups,  
  tSNE_name = "SC_tSNE",  
  color.use = NULL,  
  asp = 1,  
  alpha = 0.7,  
  title = NULL  
)
```

Arguments

SC	super-cell structure (output of SCimplify) with a field tSNE_name containing tSNE result
groups	coloring metacells by groups
tSNE_name	the name of the field containing tSNE result
color.use	colors of groups
asp	plot aspect ratio
alpha	transparency of
title	title of the plot

Value

[ggplot](#)

supercell_plot_UMAP *Plot super-cell UMAP (Use [supercell_DimPlot](#) instead) Plots super-cell UMAP (result of [supercell_UMAP](#))*

Description

Plot super-cell UMAP (Use [supercell_DimPlot](#) instead) Plots super-cell UMAP (result of [supercell_UMAP](#))

Usage

```
supercell_plot_UMAP(
  SC,
  groups,
  UMAP_name = "SC_UMAP",
  color.use = NULL,
  asp = 1,
  alpha = 0.7,
  title = NULL
)
```

Arguments

SC	super-cell structure (output of SCimplify) with a field UMAP_name containing UMAP result
groups	coloring metacells by groups
UMAP_name	the name of the field containing UMAP result
color.use	colors of groups
asp	plot aspect ratio
alpha	transparency of
title	title of the plot

Value

[ggplot](#)

supercell_prcomp	<i>compute PCA for super-cell data (sample-weighted data)</i>
------------------	---

Description

compute PCA for super-cell data (sample-weighted data)

Usage

```
supercell_prcomp(
  X,
  genes.use = NULL,
  genes.exclude = NULL,
  supercell_size = NULL,
  k = 20,
  do.scale = TRUE,
  do.center = TRUE,
  fast.pca = TRUE,
  seed = 12345
)
```

Arguments

X	super-cell transposed gene expression matrix (! where rows represent super-cells and cols represent genes)
genes.use	genes to use for dimensionality reduction
genes.exclude	genes to exclude from dimensionaloty reduction
supercell_size	a vector with supercell sizes (ordered the same way as in X)
k	number of components to compute
do.scale	scale data before PCA
do.center	center data before PCA
fast.pca	whether to run fast PCA (works for datasets with super-cells > 50)
seed	a seed to use for set.seed

Value

the same object as [prcomp](#) result

supercell_purity	<i>Compute purity of super-cells</i>
------------------	--------------------------------------

Description

Compute purity of super-cells

Usage

```
supercell_purity(
  clusters,
  supercell_membership,
  method = c("max_proportion", "entropy")[1]
)
```

Arguments

clusters	vector of clustering assignment (reference assignment)
supercell_membership	vector of assignment of single-cell data to super-cells (membership field of SCimplify function output)
method	method to compute super-cell purity. "max_proportion" if the purity is defined as a proportion of the most abundant cluster (cell type) within super-cell or "entropy" if the purity is defined as the Shanon entropy of the cell types super-cell consists of.

Value

a vector of super-cell purity, which is defined as: - proportion of the most abundant cluster within super-cell for method = "max_proportion" or - Shanon entropy for method = "entropy". With 1 meaning that super-cell consists of single cells from one cluster (reference assignment)

supercell_rescale *Rescale supercell object*

Description

This function recomputes super-cell structure at a different graining level (gamma) or for a specific number of super-cells (N.SC)

Usage

```
supercell_rescale(SC.object, gamma = NULL, N.SC = NULL)
```

Arguments

SC.object	super-cell object (an output from SCimplify function)
gamma	new grainig level (provide either gamma or N.SC)
N.SC	new number of super-cells (provide either gamma or N.SC)

Value

the same object as [SCimplify](#) at a new graining level

supercell_silhouette *Compute Silhouette index accounting for samlpe size (super cells size) ###*

Description

Compute Silhouette index accounting for samlpe size (super cells size) ###

Usage

```
supercell_silhouette(x, dist, supercell_size = NULL)
```

Arguments

x	- clustering
dist	- distance among super-cells
supercell_size	- super-cell size

Value

silhouette result

supercell_tSNE *Compute tSNE of super-cells*

Description

Computes tSNE of super-cells

Usage

```
supercell_tSNE(
  SC,
  PCA_name = "SC_PCA",
  n.comp = NULL,
  perplexity = 30,
  seed = 12345,
  ...
)
```

Arguments

SC	super-cell structure (output of SCimplify) with a field PCA_name containig PCA result
PCA_name	name of SC field containing result of supercell_pcomp
n.comp	number of vector of principal components to use for computing tSNE
perplexity	perplexity parameter (parameter of Rtsne)
seed	random seed
...	other parameters of Rtsne

Value

[Rtsne](#) result

supercell_UMAP *Compute UMAP of super-cells*

Description

Computes UMAP of super-cells

Usage

```
supercell_UMAP(SC, PCA_name = "SC_PCA", n.comp = NULL, n_neighbors = 15, ...)
```

Arguments

SC	super-cell structure (output of SCimplify) with a field PCA_name containing PCA result
PCA_name	name of SC field containing result of supercell_prcomp
n.comp	number of vector of principal components to use for computing UMAP
n_neighbors	number of neighbors (parameter of umap)
...	other parameters of umap

Value

[umap](#) result

supercell_VlnPlot *Violin plots*

Description

Violin plots (similar to [VlnPlot](#) with some changes for super-cells)

Usage

```
supercell_VlnPlot(
  ge,
  supercell_size = NULL,
  clusters,
  features = NULL,
  idents = NULL,
  color.use = NULL,
  pt.size = 0,
  pch = "o",
  y.max = NULL,
  y.min = NULL,
  same.y.lims = FALSE,
  adjust = 1,
  ncol = NULL,
  combine = TRUE,
  angle.text.y = 90,
  angle.text.x = 45
)
```

Arguments

ge	a gene expression matrix (ncol same as number of super-cells)
supercell_size	a vector with supercell size (ordered the same way as in ge)
clusters	a vector with clustering information (ordered the same way as in ge)

features	name of genes of for which gene expression is plotted
idents	idents (clusters) to plot (default all)
color.use	colors for idents
pt.size	point size (0 by default)
pch	shape of jitter dots
y.max	max of y axis
y.min	min of y axis
same.y.lims	same y axis for all plots
adjust	param of geom_violin
ncol	number of columns in combined plot
combine	combine plots into a single patchworked ggplot object. If FALSE, return a list of ggplot
angle.text.y	rotation of y text
angle.text.x	rotation of x text

Value

combined ggplot or list of ggplots if combine = TRUE

supercell_VlnPlot_single
Plot Violin plot for 1 feature

Description

Used for supercell_VlnPlot

Usage

```
supercell_VlnPlot_single(
  ge1,
  supercell_size = NULL,
  clusters,
  feature = NULL,
  color.use = NULL,
  pt.size = 0,
  pch = "o",
  y.max = NULL,
  y.min = NULL,
  adjust = 1,
  angle.text.y = 90,
  angle.text.x = 45
)
```

Arguments

<code>ge1</code>	a gene expression vector (same length as number of super-cells)
<code>supercell_size</code>	a vector with supercell size (ordered the same way as in <code>ge</code>)
<code>clusters</code>	a vector with clustering information (ordered the same way as in <code>ge</code>)
<code>feature</code>	gene to plot
<code>color.use</code>	colors for idents
<code>pt.size</code>	point size (0 by default)
<code>pch</code>	shape of jitter dots
<code>y.max</code>	max of y axis
<code>y.min</code>	min of y axis
<code>adjust</code>	param of <code>geom_violin</code>
<code>angle.text.y</code>	rotation of y text
<code>angle.text.x</code>	rotation of x text

Index

- * **datasets**
 - cell_lines, 6
- anndata_2_supercell, 3
- build_knn_graph, 3, 9, 11
- build_knn_graph_nn2, 5
- cell_lines, 6
- CreateSeuratObject, 15
- FindAllMarkers, 19
- FindMarkers, 20
- ggplot, 18, 30, 31
- gradientLegend, 29
- graph_from_adj_list, 5, 6
- hclust, 17
- irlba, 8
- knn_graph_from_dist, 6
- layout_components, 28
- layout_nicely, 28
- layout_with_dr1, 28
- layout_with_fr, 28
- layout_with_graphopt, 28
- metacell2_anndata_2_supercell, 7
- neighborsToSNNGraph, 4, 5
- nn2, 4
- NormalizeData, 15
- patchwork, 23, 36
- prcomp, 32
- Rtsne, 34
- sc_mixing_score, 12
- SCsimplify, 3, 7, 7, 10, 12–14, 16–18, 25, 28–35
- SCsimplify_for_velocity, 10
- SCsimplify_from_embedding, 10
- Seurat, 14, 15
- SingleCellExperiment, 13
- sNN, 4, 5
- supercell_2_sce, 13
- supercell_2_Seurat, 14
- supercell_assign, 15
- supercell_cluster, 16
- supercell_DimPlot, 17, 30
- supercell_estimate_velocity, 18
- supercell_FindAllMarkers, 19
- supercell_FindMarkers, 20, 20
- supercell_GE, 21, 27
- supercell_GE_idx, 24
- supercell_GeneGenePlot, 22, 23
- supercell_GeneGenePlot_single, 23
- supercell_merge, 25, 27
- supercell_mergeGE, 26
- supercell_plot, 27, 29
- supercell_plot_GE, 29
- supercell_plot_tSNE, 30
- supercell_plot_UMAP, 30
- supercell_prcomp, 31, 34, 35
- supercell_purity, 32
- supercell_rescale, 33
- supercell_silhouette, 33
- supercell_tSNE, 30, 34
- supercell_UMAP, 30, 34
- supercell_VlnPlot, 35
- supercell_VlnPlot_single, 36
- umap, 35
- VlnPlot, 35
- wcKMedoids, 16, 17
- wtd.t.test, 20, 21