Package: monocle3 (via r-universe)

August 30, 2024

Title Clustering, Differential Expression, and Trajectory Analysis for Single-Cell RNA-Seq

Version 1.3.7

Description Monocle 3 performs clustering, differential expression and trajectory analysis for single-cell expression experiments. It orders individual cells according to progress through a biological process, without knowing ahead of time which genes define progress through that process. Monocle 3 also performs differential expression analysis, clustering, visualization, and other useful tasks on single-cell expression data. It is designed to work with RNA-Seq data, but could be used with other types as well.

biocViews Software, SingleCell, RNASeq, ATACSeq, Normalization, Preprocessing, DimensionReduction, Visualization, QualityControl, Clustering, Classification, Annotation, GeneExpression, DifferentialExpression

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

LinkingTo Rcpp

Depends R (>= 4.0.0), Biobase, SingleCellExperiment

Imports assertthat (>= 0.2.1), batchelor, BiocGenerics (>= 0.28.0), BiocParallel, DelayedArray (>= 0.8.0), DelayedMatrixStats (>= 1.4.0), digest (>= 0.6.28), dplyr (>= 0.8.0.1), future (>= 1.23.0), ggplot2 (>= 3.1.1), ggrastr, ggrepel (>= 0.8.1), grr, HDF5Array, igraph (>= 1.5.0), irlba (>= 2.3.3), leidenbase (>= 0.1.25), limma (>= 3.38.3), lme4 (>= 1.1-27), lmtest (>= 0.9-36), MASS (>= 7.3-51.4), Matrix (>= 1.2-17), methods, openssl, pbapply (>= 1.4-0), pbmcapply (>= 1.4.1), pheatmap, plotly (>= 4.9.0), plyr (>= 1.8.4), proxy (>= 0.4-23), pscl (>= 1.5.2), purrr (>= 0.3.2), RANN (>= 2.6.1), RColorBrewer, Rcpp

(>= 1.0.1), reshape2 (>= 1.4.3), rsample (>= 0.0.5),
RhpcBLASctl, RcppAnnoy, RcppHNSW (>= 0.3.0), Rtsne (>= 0.15),
S4Vectors, sf, shiny, slam (>= 0.1-45), spdep (>= 1.1-2),
speedglm (>= 0.3-2), stats, stringr (>= 1.4.0),
SummarizedExperiment (>= 1.11.5), utils, uwot (>= 0.1.8),
tibble (>= 2.1.1), tidyr (>= 0.8.3), viridis (>= 0.5.1)

Suggests testthat (>= 2.1.0), pryr (>= 0.1.4), knitr, rmarkdown,
spelling, scran

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Maintainer Brent Ewing <bge@uw.edu>

Repository https://blaserlab.r-universe.dev

RemoteUrl https://github.com/cole-trapnell-lab/monocle3

RemoteRef HEAD

RemoteSha 98402ed0c10cac020524bebbb9300614a799f6d1

Contents

`aggregate_gene_expression` *Creates a matrix with aggregated expression values for arbitrary groups of genes*

Description

Creates a matrix with aggregated expression values for arbitrary groups of genes

Usage

```
aggregate_gene_expression(
  cds,
  gene_group_df = NULL,
  cell_group_df = NULL,
  norm_method = c("log", "binary", "size_only"),
  pseudocount = 1,
  scale_agg_values = TRUE,
  max_agg_value = 3,
  min_agg_value = -3,
  exclude.na = TRUE,
  gene_agg_fun = "sum",
  cell_agg_fun = "mean"
)
```

Arguments

cds	The cell_data_set on which this function operates
gene_group_df	A dataframe in which the first column contains gene ids or short gene names and the second contains groups. If NULL, genes are not grouped.
cell_group_df	A dataframe in which the first column contains cell ids and the second contains groups. If NULL, cells are not grouped.
norm_method	How to transform gene expression values before aggregating them. If "log", a pseudocount is added. If "size_only", values are divided by cell size factors prior to aggregation.
pseudocount	Value to add to expression prior to log transformation and aggregation.
scale_agg_values	Whether to center and scale aggregated groups of genes.
max_agg_value	If scale_agg_values is TRUE, the maximum value the resulting Z scores can take. Higher values are capped at this threshold.
min_agg_value	If scale_agg_values is TRUE, the minimum value the resulting Z scores can take. Lower values are capped at this threshold.
exclude.na	Logical indicating whether or not to exclude NA values from the aggregated matrix.
gene_agg_fun	Function used for gene aggregation. This can be either sum or mean. Default is sum.
cell_agg_fun	Function used for cell aggregation. Default is mean.

Value

A matrix of dimension NxM, where N is the number of gene groups and M is the number of cell groups.

Examples

```
## Not run:
   expression_matrix <- readRDS(system.file('extdata',
                                            'worm_l2/worm_l2_expression_matrix.rds',
                                            package='monocle3'))
   cell_metadata <- readRDS(system.file('extdata',
                            'worm_l2/worm_l2_coldata.rds',
                             package='monocle3'))
   gene_metadata <- readRDS(system.file('extdata',
                            'worm_l2/worm_l2_rowdata.rds',
                            package='monocle3'))

   cds <- new_cell_data_set(expression_data=expression_matrix,
                            cell_metadata=cell_metadata,
                            gene_metadata=gene_metadata)

  cds <- preprocess_cds(cds, num_dim = 100)
  cds <- reduce_dimension(cds)
  cds <- cluster_cells(cds, resolution=1e-5)
```

```
  colData(cds)$assigned_cell_type <- as.character(partitions(cds))
  colData(cds)$assigned_cell_type <- dplyr::recode(colData(cds)$assigned_cell_type,
                                                   "1"="Germline",
                                                   "2"="Body wall muscle",
                                                   "3"="Unclassified neurons",
                                                   "4"="Vulval precursors",
                                                   "5"="Failed QC",
                                                   "6"="Seam cells",
                                                   "7"="Pharyngeal epithelia",
                                                   "8"="Coelomocytes",
                                                   "9"="Am/PH sheath cells",
                                                   "10"="Failed QC",
                                                   "11"="Touch receptor neurons",
                                                   "12"="Intestinal/rectal muscle",
                                                   "13"="Pharyngeal neurons",
                                                   "14"="NA",
                                                   "15"="flp-1(+) interneurons",
                                                   "16"="Canal associated neurons",
                                                   "17"="Ciliated sensory neurons",
                                                   "18"="Other interneurons",
                                                   "19"="Pharyngeal gland",
                                                   "20"="Failed QC",
                                                   "21"="Ciliated sensory neurons",
                                                   "22"="Oxygen sensory neurons",
                                                   "23"="Ciliated sensory neurons",
                                                   "24"="Ciliated sensory neurons",
                                                   "25"="Ciliated sensory neurons",
                                                   "26"="Ciliated sensory neurons",
                                                   "27"="Oxygen sensory neurons",
                                                   "28"="Ciliated sensory neurons",
                                                   "29"="Unclassified neurons",
                                                   "30"="Socket cells",
                                                   "31"="Failed QC",
                                                   "32"="Pharyngeal gland",
                                                   "33"="Ciliated sensory neurons",
                                                   "34"="Ciliated sensory neurons",
                                                   "35"="Ciliated sensory neurons",
                                                   "36"="Failed QC",
                                                   "37"="Ciliated sensory neurons",
                                                   "38"="Pharyngeal muscle")
  neurons_cds <- cds[,grepl("neurons", colData(cds)$assigned_cell_type, ignore.case=TRUE)]
  pr_graph_test_res <- graph_test(neurons_cds, neighbor_graph="knn")
  pr_deg_ids <- row.names(subset(pr_graph_test_res, q_value < 0.05))
  gene_module_df <- find_gene_modules(neurons_cds[pr_deg_ids,], resolution=1e-2)
  cell_group_df <- tibble::tibble(cell=row.names(colData(neurons_cds)),
                                  cell_group=partitions(cds)[colnames(neurons_cds)])
  agg_mat <- aggregate_gene_expression(neurons_cds, gene_module_df, cell_group_df)

## End(Not run)
```

`align_cds` *Align cells from different groups within a cds*

Description

Data sets that contain cells from different groups often benefit from alignment to subtract differences between them. Alignment can be used to remove batch effects, subtract the effects of treatments, or even potentially compare across species. `align_cds` executes alignment and stores these adjusted coordinates.

This function can be used to subtract both continuous and discrete batch effects. For continuous effects, `align_cds` fits a linear model to the cells' PCA or LSI coordinates and subtracts them using Limma. For discrete effects, you must provide a grouping of the cells, and then these groups are aligned using Batchelor, a "mutual nearest neighbor" algorithm described in:

Haghverdi L, Lun ATL, Morgan MD, Marioni JC (2018). "Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors." Nat. Biotechnol., 36(5), 421-427. doi: 10.1038/nbt.4091

Usage

```
align_cds(
  cds,
  preprocess_method = c("PCA", "LSI"),
  alignment_group = NULL,
  alignment_k = 20,
  residual_model_formula_str = NULL,
  verbose = FALSE,
  build_nn_index = FALSE,
  nn_control = list(),
  ...
)
```

Arguments

`cds`	the cell_data_set upon which to perform this operation
`preprocess_method`	a string specifying the low-dimensional space in which to perform alignment, currently either PCA or LSI. Default is "PCA".
`alignment_group`	String specifying a column of colData to use for aligning groups of cells. The column specified must be a factor. Alignment can be used to subtract batch effects in a non-linear way. For correcting continuous effects, use residual_model_formula_str. Default is NULL.
`alignment_k`	The value of k used in mutual nearest neighbor alignment

residual_model_formula_str
: NULL or a string model formula specifying any effects to subtract from the data before dimensionality reduction. Uses a linear model to subtract effects. For non-linear effects, use alignment_group. Default is NULL.

verbose
: Whether to emit verbose output during dimensionality reduction

build_nn_index
: logical When this argument is set to TRUE, align_cds builds the nearest neighbor index from the aligned reduced matrix for later use. Default is FALSE.

nn_control
: An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information.

...
: additional arguments to pass to limma::lmFit if residual_model_formula is not NULL

Value

an updated cell_data_set object

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
```

---

align_transform *Apply an alignment transform model to a cell_data_set.*

Description

align_transform is not supported. Co-embed your data sets if you need batch correction.

Usage

```
align_transform(cds, reduction_method = c("Aligned"))
```

Arguments

cds	a cell_data_set to be transformed.
reduction_method	a previously loaded transform model that is used to reduce the dimensions of the count matrix in the cell_data_set. The "Aligned" transform is not supported.

Value

The cds is returned without processing.

---

calc_principal_graph *Function to automatically learn the structure of data by either using L1-graph or the spanning-tree formulization*

Description

Function to automatically learn the structure of data by either using L1-graph or the spanning-tree formulization

Usage

```
calc_principal_graph(
  X,
  C0,
  maxiter = 10,
  eps = 1e-05,
  L1.gamma = 0.5,
  L1.sigma = 0.01,
  verbose = TRUE
)
```

Arguments

X	the input data DxN
C0	the initialization of centroids
maxiter	maximum number of iteration
eps	relative objective difference
L1.gamma	regularization parameter for k-means (the prefix of 'param' is used to avoid name collision with gamma)
L1.sigma	bandwidth parameter
verbose	emit results from iteration

Value

a list of X, C, W, P, objs X is the input data C is the centers for principal graph W is the principal graph matrix P is the cluster assignment matrix objs is the objective value for the function

cell_data_set *The cell_data_set class*

Description

The main class used by Monocle3 to hold single-cell expression data. cell_data_set extends the Bioconductor SingleCellExperiment class.

Details

This class is initialized from a matrix of expression values along with cell and feature metadata.

Fields

`reduce_dim_aux` SimpleList, auxiliary information from reduced dimension.

`principal_graph_aux` SimpleList, auxiliary information from principal graph construction

`principal_graph` SimpleList of igraph objects containing principal graphs for different dimensionality reduction.

`clusters` SimpleList of cluster information for different dimensionality reduction.

cell_data_set-methods *Methods for the cell_data_set class*

Description

Methods for the cell_data_set class

Arguments

`object` The cell_data_set object

choose_cells *Choose cells interactively to subset a cds*

Description

Choose cells interactively to subset a cds

Usage

```
choose_cells(
  cds,
  reduction_method = c("UMAP", "tSNE", "PCA", "Aligned"),
  clear_cds = FALSE,
  return_list = FALSE
)
```

Arguments

cds	CDS object to subset
reduction_method	The reduction method to plot while choosing cells.
clear_cds	Logical, clear CDS slots before returning. After clearing the cds, re-run processing from preprocess_cds(), ... Default is FALSE.
return_list	Logical, return a list of cells instead of a subsetted CDS object.

Value

A subset CDS object. If return_list = FALSE, a list of cell names.

choose_graph_segments *Choose cells along the path of a principal graph*

Description

Choose cells along the path of a principal graph

Usage

```
choose_graph_segments(
  cds,
  reduction_method = "UMAP",
  starting_pr_node = NULL,
  ending_pr_nodes = NULL,
  return_list = FALSE,
  clear_cds = TRUE
)
```

Arguments

cds	CDS object to be subsetted.
reduction_method	The reduction method to plot while choosing cells. Currently only "UMAP" is supported.
starting_pr_node	NULL, or a string with the name of the starting principal node to be used. You can see the principal nodes in your dataset by using plot_cells with label_principal_points = TRUE.
ending_pr_nodes	NULL, or one or more strings with the name(s) of the ending principal node(s) to be used. You can see the principal nodes in your dataset by using plot_cells with label_principal_points = TRUE.
return_list	Logical, return a list of cells instead of a subsetted CDS object.
clear_cds	Logical, clear CDS slots before returning. After clearing the cds, re-run processing from preprocess_cds(), ... Default is TRUE.

Value

A subset CDS object. If return_list = FALSE, a list of cell and graph node names.

---

clear_cds_slots *Clear CDS slots*

Description

Function to clear all CDS slots besides colData, rowData and expression data.

Usage

```
clear_cds_slots(cds)
```

Arguments

cds	cell_data_set to be cleared

Value

A cell_data_set with only expression, rowData and colData present.

clusters *Generic to extract clusters from CDS object*

Description

Generic to extract clusters from CDS object

Usage

```
clusters(x, reduction_method = "UMAP")
```

Arguments

x	A cell_data_set object.
reduction_method	Reduced dimension to extract clusters for.

Value

Clusters.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))
cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
clusters_factors <- clusters(cds, "UMAP")
```

clusters,cell_data_set-method *Method to extract clusters from CDS object*

Description

Method to extract clusters from CDS object

Usage

```
## S4 method for signature 'cell_data_set'
clusters(x, reduction_method = "UMAP")
```

Arguments

x	A cell_data_set object.
reduction_method	Reduced dimension to extract clusters for.

Value

Clusters.

cluster_cells *Cluster cells using Louvain/Leiden community detection*

Description

Unsupervised clustering of cells is a common step in many single-cell expression workflows. In an experiment containing a mixture of cell types, each cluster might correspond to a different cell type. This function takes a cell_data_set as input, clusters the cells using Louvain/Leiden community detection, and returns a cell_data_set with internally stored cluster assignments. In addition to clusters this function calculates partitions, which represent superclusters of the Louvain/Leiden communities that are found using a kNN pruning method. Cluster assignments can be accessed using the clusters function and partition assignments can be accessed using the partitions function.

Usage

```
cluster_cells(
  cds,
  reduction_method = c("UMAP", "tSNE", "PCA", "LSI", "Aligned"),
  k = 20,
  cluster_method = c("leiden", "louvain"),
  num_iter = 2,
  partition_qval = 0.05,
```

```
  weight = FALSE,
  resolution = NULL,
  random_seed = 42,
  verbose = FALSE,
  nn_control = list(),
  ...
)
```

Arguments

cds	The cell_data_set upon which to perform clustering.
reduction_method	The dimensionality reduction method upon which to base clustering. Options are "UMAP", "tSNE", "PCA" and "LSI".
k	Integer number of nearest neighbors to use when creating the k nearest neighbor graph for Louvain/Leiden clustering. k is related to the resolution of the clustering result, a bigger k will result in lower resolution and vice versa. Default is 20.
cluster_method	String indicating the clustering method to use. Options are "louvain" or "leiden". Default is "leiden". Resolution parameter is ignored if set to "louvain".
num_iter	Integer number of iterations used for Louvain/Leiden clustering. The clustering result giving the largest modularity score will be used as the final clustering result. Default is 1. Note that if num_iter is greater than 1, the random_seed argument will be ignored for the louvain method.
partition_qval	Numeric, the q-value cutoff to determine when to partition. Default is 0.05.
weight	A logical argument to determine whether or not to use Jaccard coefficients for two nearest neighbors (based on the overlapping of their kNN) as the weight used for Louvain clustering. Default is FALSE.
resolution	Parameter that controls the resolution of clustering. If NULL (Default), the parameter is determined automatically.
random_seed	The seed used by the random number generator in louvain-igraph package. This argument will be ignored if num_iter is larger than 1.
verbose	A logic flag to determine whether or not we should print the run details.
nn_control	An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information. The default metric is cosine for reduction_methods PCA, LSI, and Aligned, and is euclidean for reduction_methods tSNE and UMAP.
...	Additional arguments passed to the leidenbase package.

Value

an updated cell_data_set object, with cluster and partition information stored internally and accessible using `clusters` and `partitions`

References

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. Science, 344(6191), 1492-1496. doi:10.1126/science.1242072

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre: Fast unfolding of communities in large networks. J. Stat. Mech. (2008) P10008

V. A. Traag and L. Waltman and N. J. van Eck: From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports, 9(1) (2019). doi: 10.1038/s41598-019-41695-z.

Jacob H. Levine and et. al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. Cell, 2015.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                         'worm_embryo/worm_embryo_rowdata.rds',
                         package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                             'worm_embryo/worm_embryo_expression_matrix.rds',
                             package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
```

---

coefficient_table *Extract coefficient table from a fit_models result.*

Description

Extracts a table of coefficients from a tibble containing model objects. It tests whether each coefficient differs significantly from zero under the Wald test and adjusts the p-values for multiple hypothesis testing using the method of Benjamini and Hochberg, placing these adjusted values in the q-value column.

Usage

```
coefficient_table(model_tbl)
```

Arguments

model_tbl	A tibble of model objects, generally the output of fit_models.

Value

A table of coefficient data for each gene.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds, num_dim=50)
cds <- align_cds(cds, alignment_group = "batch",
                 residual_model_formula_str = "~ bg.300.loading + bg.400.loading +
                 bg.500.1.loading + bg.500.2.loading + bg.r17.loading +
                 bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds_subset <- cds[rowData(cds)$gene_short_name %in% ciliated_genes,]
gene_fits <- fit_models(cds_subset, model_formula_str = "~embryo.time")
fit_coefs <- coefficient_table(gene_fits)
```

---

combine_cds *Combine a list of cell_data_set objects*

Description

This function will combine a list of cell_data_set objects into a new cell_data_set object.

Usage

```
combine_cds(
  cds_list,
  keep_all_genes = TRUE,
  cell_names_unique = FALSE,
```

```
  sample_col_name = "sample",
  keep_reduced_dims = FALSE
)
```

Arguments

cds_list	List of cds objects to be combined.
keep_all_genes	Logical indicating what to do if there is a mismatch in the gene sets of the CDSs. If TRUE, all genes are kept and cells from CDSs missing a given gene will be filled in with zeroes. If FALSE, only the genes in common among all of the CDSs will be kept. Default is TRUE.
cell_names_unique	Logical indicating whether all of the cell IDs across all of the CDSs are unique. If FALSE, the CDS name is appended to each cell ID to prevent collisions. These cell IDs are used as count matrix column names and colData(cds) row names. Cell names stored in other cds locations are not modified so you will need to modify them manually for consistency. Default is FALSE.
sample_col_name	A string to be the column name for the colData column that indicates which original cds the cell derives from. Default is "sample".
keep_reduced_dims	Logical indicating whether to keep the reduced dimension matrices. Default is FALSE.

Value

A combined cell_data_set object.

---

compare_models *Compare goodness of fit of two models.*

Description

Compares goodness of fit for two ways of fitting a set of genes' expression using a likelihood ratio test. The likelihood ratio test helps one decide whether the improvement in fit is large enough to justify the additional complexity of extra terms in the full model in comparison to the reduced model.

Usage

```
compare_models(model_tbl_full, model_tbl_reduced)
```

Arguments

model_tbl_full	A tibble of model objects, generally output of fit_models, to be compared with model_tbl_reduced
model_tbl_reduced	A tibble of model objects, generally output of fit_models, to be compared with model_tbl_full.

Value

The result of a likelihood test by gene.

`detect_genes` *Detects genes above minimum threshold.*

Description

For each gene in a cell_data_set object, detect_genes counts how many cells are expressed above a minimum threshold. In addition, for each cell, detect_genes counts the number of genes above this threshold that are detectable. Results are added as columns num_cells_expressed and num_genes_expressed in the rowData and colData tables respectively.

Usage

```
detect_genes(cds, min_expr = 0)
```

Arguments

`cds`	Input cell_data_set object.
`min_expr`	Numeric indicating expression threshold

Value

Updated cell_data_set object

Examples

```
## Not run:
   cds <- detect_genes(cds, min_expr=0.1)

## End(Not run)
```

`estimate_size_factors` *Function to calculate size factors for single-cell RNA-seq data*

Description

Function to calculate size factors for single-cell RNA-seq data

Usage

```
estimate_size_factors(
  cds,
  round_exprs = TRUE,
  method = c("mean-geometric-mean-total", "mean-geometric-mean-log-total")
)
```

Arguments

cds	The cell_data_set
round_exprs	A logic flag to determine whether or not the expression value should be rounded
method	A string to specify the size factor calculation approach. Options are "mean-geometric-mean-total" (default), "mean-geometric-mean-log-total".

Value

Updated cell_data_set object with a new colData column called 'Size_Factor'.

Examples

```
cds <- load_a549()
colData(cds)[['Size_Factor']] <- NULL
cds <- estimate_size_factors(cds)
```

---

evaluate_fits *Evaluate fit of model objects.*

Description

Evaluate_fits takes a tibble created by the fit_models function and returns a table that assists with evaluating how well the model explains the gene expression data.

Usage

```
evaluate_fits(model_tbl)
```

Arguments

model_tbl	A tibble of model objects, generally output of fit_models.

Value

A table with fit information on each gene.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
```

```
                                          package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds, num_dim=50)
cds <- align_cds(cds, alignment_group = "batch",
                 residual_model_formula_str = "~ bg.300.loading + bg.400.loading +
              bg.500.1.loading + bg.500.2.loading + bg.r17.loading + bg.b01.loading +
                 bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds_subset <- cds[rowData(cds)$gene_short_name %in% ciliated_genes,]
gene_fits <- fit_models(cds_subset, model_formula_str = "~embryo.time")
evaluate_fits(gene_fits)
```

`exprs` *Generic to access cds count matrix*

Description

Generic to access cds count matrix

Usage

```
exprs(x)
```

Arguments

`x`	A cell_data_set object.

Value

Count matrix.

Examples

```
cds <- load_a549()
exprs(cds)
```

exprs,cell_data_set-method *Method to access cds count matrix*

Description

Method to access cds count matrix

Usage

```
## S4 method for signature 'cell_data_set'
exprs(x)
```

Arguments

x	A cell_data_set object.

Value

Count matrix.

fData *Generic to access cds rowData table*

Description

Generic to access cds rowData table

Usage

```
fData(x)
```

Arguments

x	A cell_data_set object.

Value

rowData table.

Examples

```
cds <- load_a549()
fData(cds)
```

fData,cell_data_set-method *Method to access cds rowData table*

Description

Method to access cds rowData table

Usage

```
## S4 method for signature 'cell_data_set'
fData(x)
```

Arguments

x	A cell_data_set object.

Value

rowData table.

Examples

```
cds <- load_a549()
fData(cds)
```

fData<- *Generic to set cds rowData table*

Description

Generic to set cds rowData table

Usage

```
fData(x) <- value
```

Arguments

x	A cell_data_set object.
value	A data frame to set to colData table.

Value

x.

Examples

```
cds <- load_a549()
fData(cds)[['row_index']] <- seq(nrow(fData(cds)))
```

`fData<-,cell_data_set-method` *Method to set cds rowData table*

Description

Method to set cds rowData table

Usage

```
## S4 replacement method for signature 'cell_data_set'
fData(x) <- value
```

Arguments

x	A cell_data_set object.
value	A data frame to set to colData table.

Value

x.

`find_gene_modules` *Cluster genes into modules that are co-expressed across cells.*

Description

Cluster genes into modules that are co-expressed across cells.

Usage

```
find_gene_modules(
  cds,
  reduction_method = c("UMAP"),
  max_components = 2,
  umap.metric = "cosine",
  umap.min_dist = 0.1,
  umap.n_neighbors = 15L,
  umap.fast_sgd = FALSE,
```

```
  umap.nn_method = "annoy",
  k = 20,
  leiden_iter = 1,
  partition_qval = 0.05,
  weight = FALSE,
  resolution = NULL,
  random_seed = 0L,
  cores = 1,
  verbose = FALSE,
  preprocess_method = c("PCA", "LSI"),
  nn_control = list(),
  ...
)
```

Arguments

cds	the cell_data_set upon which to perform this operation
reduction_method	The dimensionality reduction method used to generate the lower dimensional space in which genes will be clustered. Currently only UMAP is supported.
max_components	The number of dimensions in which to cluster genes into modules.
umap.metric	Metric used by UMAP for measuring similarity between genes .
umap.min_dist	Minimum distance parameter passed to UMAP.
umap.n_neighbors	Number of nearest neighbors used by UMAP.
umap.fast_sgd	Whether to allow UMAP to perform fast stochastic gradient descent. Defaults to TRUE. Setting FALSE will result in slower, but deterministic behavior (if cores=1).
umap.nn_method	The method used for nearest neighbor network construction during UMAP.
k	number of kNN used in creating the k nearest neighbor graph for Louvain clustering. The number of kNN is related to the resolution of the clustering result, bigger number of kNN gives low resolution and vice versa. Default to be 20
leiden_iter	Integer number of iterations used for Leiden clustering. The clustering result with the largest modularity score is used as the final clustering result. Default to be 1.
partition_qval	Significance threshold used in Louvain community graph partitioning.
weight	A logic argument to determine whether or not we will use Jaccard coefficient for two nearest neighbors (based on the overlapping of their kNN) as the weight used for Louvain clustering. Default to be FALSE.
resolution	Resolution parameter passed to Louvain. Can be a list. If so, this method will evaluate modularity at each resolution and use the one with the highest value.
random_seed	the seed used by the random number generator in Leiden.
cores	number of cores computer should use to execute function
verbose	Whether or not verbose output is printed.

preprocess_method
: a string specifying the low-dimensional space to use for gene loadings, currently either PCA or LSI. Default is "PCA".

nn_control
: An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information.

...
: Additional arguments passed to UMAP and Louvain analysis.

Value

A dataframe with genes and the modules to which they are assigned.

Examples

```
## Not run:
   expression_matrix <- readRDS(system.file('extdata',
                                            'worm_l2/worm_l2_expression_matrix.rds',
                                            package='monocle3'))
   cell_metadata <- readRDS(system.file('extdata',
                                        'worm_l2/worm_l2_coldata.rds',
                                        package='monocle3'))
   gene_metadata <- readRDS(system.file('extdata',
                                        'worm_l2/worm_l2_rowdata.rds',
                                        package='monocle3'))

   cds <- new_cell_data_set(expression_data=expression_matrix,
                            cell_metadata=cell_metadata,
                            gene_metadata=gene_metadata)

  cds <- preprocess_cds(cds, num_dim = 100)
  cds <- reduce_dimension(cds)
  cds <- cluster_cells(cds, resolution=1e-5)
  colData(cds)$assigned_cell_type <- as.character(partitions(cds))
  colData(cds)$assigned_cell_type <- dplyr::recode(colData(cds)$assigned_cell_type,
                                                   "1"="Germline",
                                                   "2"="Body wall muscle",
                                                   "3"="Unclassified neurons",
                                                   "4"="Vulval precursors",
                                                   "5"="Failed QC",
                                                   "6"="Seam cells",
                                                   "7"="Pharyngeal epithelia",
                                                   "8"="Coelomocytes",
                                                   "9"="Am/PH sheath cells",
                                                   "10"="Failed QC",
                                                   "11"="Touch receptor neurons",
                                                   "12"="Intestinal/rectal muscle",
                                                   "13"="Pharyngeal neurons",
                                                   "14"="NA",
                                                   "15"="flp-1(+) interneurons",
                                                   "16"="Canal associated neurons",
                                                   "17"="Ciliated sensory neurons",
                                                   "18"="Other interneurons",
                                                   "19"="Pharyngeal gland",
```

```
                                                    "20"="Failed QC",
                                                    "21"="Ciliated sensory neurons",
                                                    "22"="Oxygen sensory neurons",
                                                    "23"="Ciliated sensory neurons",
                                                    "24"="Ciliated sensory neurons",
                                                    "25"="Ciliated sensory neurons",
                                                    "26"="Ciliated sensory neurons",
                                                    "27"="Oxygen sensory neurons",
                                                    "28"="Ciliated sensory neurons",
                                                    "29"="Unclassified neurons",
                                                    "30"="Socket cells",
                                                    "31"="Failed QC",
                                                    "32"="Pharyngeal gland",
                                                    "33"="Ciliated sensory neurons",
                                                    "34"="Ciliated sensory neurons",
                                                    "35"="Ciliated sensory neurons",
                                                    "36"="Failed QC",
                                                    "37"="Ciliated sensory neurons",
                                                    "38"="Pharyngeal muscle")
  neurons_cds <- cds[,grepl("neurons", colData(cds)$assigned_cell_type, ignore.case=TRUE)]
   pr_graph_test_res <- graph_test(neurons_cds, neighbor_graph="knn")
   pr_deg_ids <- row.names(subset(pr_graph_test_res, q_value < 0.05))
   gene_module_df <- find_gene_modules(neurons_cds[pr_deg_ids,], resolution=1e-2)

## End(Not run)
```

`fit_models` *Fits a model for each gene in a cell_data_set object.*

Description

This function fits a generalized linear model for each gene in a cell_data_set. Formulae can be provided to account for additional covariates (e.g. day collected, genotype of cells, media conditions, etc).

Usage

```
fit_models(
  cds,
  model_formula_str,
  expression_family = "quasipoisson",
  reduction_method = "UMAP",
  cores = 1,
  clean_model = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

cds	The cell_data_set upon which to perform this operation.
model_formula_str	A formula string specifying the model to fit for the genes.
expression_family	Specifies the family function used for expression responses. Can be one of "quasipoisson", "negbinomial", "poisson", "binomial", "gaussian", "zipoisson", "zinegbinomial", or "mixed-negbinomial". Default is "quasipoisson".
reduction_method	Which method to use with clusters() and partitions(). Default is "UMAP".
cores	The number of processor cores to use during fitting.
clean_model	Logical indicating whether to clean the model. Default is TRUE.
verbose	Logical indicating whether to emit progress messages.
...	Additional arguments passed to model fitting functions.

Value

a tibble where the rows are genes and columns are

- id character vector from `rowData(cds)$id`
- gene_short_names character vector from `rowData(cds)$gene_short_names`
- num_cells_expressed int vector from `rowData(cds)$num_cells_expressed`
- gene_id character vector from row.names(rowData(cds))‘
- model GLM model list returned by speedglm
- model_summary model summary list returned by `summary(model)`
- status character vector of model fitting status: OK when model converged, otherwise FAIL

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds, num_dim=50)
cds <- align_cds(cds, alignment_group = "batch",
                 residual_model_formula_str = "~ bg.300.loading + bg.400.loading +
```

```
                bg.500.1.loading + bg.500.2.loading + bg.r17.loading + bg.b01.loading +
                   bg.b02.loading")
    cds <- reduce_dimension(cds)
    ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
    cds_subset <- cds[rowData(cds)$gene_short_name %in% ciliated_genes,]
    gene_fits <- fit_models(cds_subset, model_formula_str = "~embryo.time")
```

`fix_missing_cell_labels` *Replace NA cell column data values left after running transfer_cell_labels.*

Description

Try to replace NA values left in a query cell_data_set after running transfer_cell_labels.

Usage

```
fix_missing_cell_labels(
  cds,
  reduction_method = c("UMAP", "PCA", "LSI"),
  from_column_name,
  to_column_name = from_column_name,
  out_notna_models_dir = NULL,
  k = 10,
  nn_control = list(),
  top_frac_threshold = 0.5,
  top_next_ratio_threshold = 1.5,
  verbose = FALSE
)
```

Arguments

`cds`	the cell_data_set upon which to perform this operation
`reduction_method`	a string specifying the reduced dimension matrix to use for the label transfer. These are "PCA", "LSI", and "UMAP". Default is "UMAP".
`from_column_name`	a string giving the name of the query cds column with NA values to fix.
`to_column_name`	a string giving the name of the query cds column where the fixed column data will be stored. The default is from_column_name
`out_notna_models_dir`	a string with the name of the transform model directory where you want to save the not-NA transform models, which includes the nearest neighbor index. If NULL, the not-NA models are not saved. The default is NULL.

k an integer giving the number of reference nearest neighbors to find. This value must be large enough to find meaningful column value fractions. See the top_frac_threshold parameter below for additional information. The default is 10.

nn_control An optional list of parameters used to make the nearest neighbors index. See the set_nn_control help for additional details. The default metric is cosine for reduction_methods PCA and LSI and is euclidean for reduction_method UMAP.

top_frac_threshold
a numeric value. The top fraction of reference values must be greater than top_frac_threshold in order to be transferred to the query. The top fraction is the fraction of the k neighbors with the most frequent value. The default is 0.5.

top_next_ratio_threshold
a numeric value giving the minimum value of the ratio of the counts of the most frequent to the second most frequent reference values required for transferring the reference value to the query. The default is 1.5.

verbose a boolean controlling verbose output.

Details

fix_missing_cell_labels uses non-NA cell data values in the query cell_data_set to replace NAs in nearby cells. It partitions the cells into a set with NA and a set with non-NA column data values. It makes a nearest neighbor index using cells with non-NA values, and for each cell with NA, it tries to find an acceptable non-NA column data value as follows. If more than top_frac_threshold fraction of them have the same value, it replaces the NA with it. If not, it checks whether the ratio of the most frequent to the second most frequent values is at least top_next_ratio_threshold, in which case it copies the most frequent value. Otherwise, it leaves the NA.

Value

an updated cell_data_set object

Examples

```
## Not run:
   expression_matrix <- readRDS(system.file('extdata',
                                            'worm_l2/worm_l2_expression_matrix.rds',
                                            package='monocle3'))
   cell_metadata <- readRDS(system.file('extdata',
                                        package='monocle3'))
   gene_metadata <- readRDS(system.file('extdata',
                                        'worm_l2/worm_l2_rowdata.rds',
                                        package='monocle3'))

   cds <- new_cell_data_set(expression_data=expression_matrix,
                            cell_metadata=cell_metadata,
                            gene_metadata=gene_metadata)

  ncell <- nrow(colData(cds))
  cell_sample <- sample(seq(ncell), 2 * ncell / 3)
  cell_set <- seq(ncell) %in% cell_sample
  cds1 <- cds[,cell_set]
```

```
  cds1 <- preprocess_cds(cds1)
  cds1 <- reduce_dimension(cds1, build_nn_index=TRUE)
  save_transform_models(cds1, 'tm')

  cds2 <- cds[,!cell_set]
  cds2 <- load_transform_models(cds2, 'tm')
  cds2 <- preprocess_transform(cds2, 'PCA')
  cds2 <- reduce_dimension_transform(cds2)
cds2 <- transfer_cell_labels(cds2, 'UMAP', colData(cds1), 'cao_cell_type', 'transfer_cell_type')
 cds2 <- fix_missing_cell_labels(cds2, 'UMAP', 'transfer_cell_type', 'fixed_cell_type')

## End(Not run)
```

`generate_centers` *Function to reproduce the behavior of eye function in matlab*

Description

Function to reproduce the behavior of eye function in matlab

Usage

```
generate_centers(X, W, P, param.gamma)
```

Arguments

`X`	input data
`W`	the principal graph matrix
`P`	the cluster assignment matrix
`param.gamma`	regularization parameter for k-means (the prefix of 'param' is used to avoid name collision with gamma)

Value

A matrix C for the centers for principal graph

generate_garnett_marker_file *Generate a Garnett marker file from top_markers output.*

Description

Generate a Garnett marker file from top_markers output.

Usage

```
generate_garnett_marker_file(
  marker_test_res,
  file = "./marker_file.txt",
  max_genes_per_group = 10,
  remove_duplicate_genes = FALSE
)
```

Arguments

marker_test_res	Tibble of top markers, output of top_markers.
file	Path to the marker file to be generated. Default is "./marker_file.txt".
max_genes_per_group	Numeric, the maximum number of genes to output per cell type entry. Default is 10.
remove_duplicate_genes	Logical indicating whether marker genes that mark multiple cell groups should be excluded. Default is FALSE. When FALSE, a message will be emitted when duplicates are present.

Value

None, marker file is written to file parameter location.

get_citations *Access citations for methods used during analysis.*

Description

Access citations for methods used during analysis.

Usage

```
get_citations(cds)
```

Arguments

cds	The cds object to access citations from.

Value

A data frame with the methods used and the papers to be cited.

Examples

```
{
  ## Not run:
     get_citations(cds)

## End(Not run)
}
```

---

get_genome_in_matrix_path

Get a genome from Cell Ranger output

Description

Get a genome from Cell Ranger output

Usage

```
get_genome_in_matrix_path(matrix_path, genome = NULL)
```

Arguments

matrix_path	Path to a matrices directory produced by the Cell Ranger pipeline
genome	Genome to specifically check for, otherwise will check for whatever genome(s) exist there

Value

A string representing the genome found

graph_test *Test genes for differential expression based on the low dimensional embedding and the principal graph*

Description

We are often interested in finding genes that are differentially expressed across a single-cell trajectory. Monocle3 introduces a new approach for finding such genes that draws on a powerful technique in spatial correlation analysis, the Moran's I test. Moran's I is a measure of multi-directional and multi-dimensional spatial autocorrelation. The statistic tells you whether cells at nearby positions on a trajectory will have similar (or dissimilar) expression levels for the gene being tested. Although both Pearson correlation and Moran's I ranges from -1 to 1, the interpretation of Moran's I is slightly different: +1 means that nearby cells will have perfectly similar expression; 0 represents no correlation, and -1 means that neighboring cells will be *anti-correlated*.

Usage

```
graph_test(
  cds,
  neighbor_graph = c("knn", "principal_graph"),
  reduction_method = "UMAP",
  k = 25,
  method = c("Moran_I"),
  alternative = "greater",
  expression_family = "quasipoisson",
  cores = 1,
  verbose = FALSE,
  nn_control = list()
)
```

Arguments

cds	a cell_data_set object upon which to perform this operation
neighbor_graph	String indicating what neighbor graph to use. "principal_graph" and "knn" are supported. Default is "knn", but "principal_graph" is recommended for trajectory analysis.
reduction_method	character, the method used to reduce dimension. Currently only supported for "UMAP".
k	Number of nearest neighbors used for building the kNN graph which is passed to knn2nb function during the Moran's I (Geary's C) test procedure.
method	a character string specifying the method (currently only 'Moran_I' is supported) for detecting significant genes showing correlation along the principal graph embedded in the low dimensional space.
alternative	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.

expression_family
: a character string specifying the expression family function used for the test.

cores
: the number of cores to be used while testing each gene for differential expression.

verbose
: Whether to show spatial test (Moran's I) errors and warnings. Only valid for cores = 1.

nn_control
: An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information.

Value

a data frame containing the p values and q-values from the Moran's I test on the parallel arrays of models.

See Also

moran.test geary.test

Examples

```
expression_matrix <- readRDS(system.file('extdata',
                                         'worm_l2/worm_l2_expression_matrix.rds',
                                         package='monocle3'))
cell_metadata <- readRDS(system.file('extdata',
                         'worm_l2/worm_l2_coldata.rds',
                          package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                         'worm_l2/worm_l2_rowdata.rds',
                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds, num_dim = 100)
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds, resolution=1e-5)
colData(cds)$assigned_cell_type <- as.character(partitions(cds))
colData(cds)$assigned_cell_type <- dplyr::recode(colData(cds)$assigned_cell_type,
                                                 "1"="Germline",
                                                 "2"="Body wall muscle",
                                                 "3"="Unclassified neurons",
                                                 "4"="Vulval precursors",
                                                 "5"="Failed QC",
                                                 "6"="Seam cells",
                                                 "7"="Pharyngeal epithelia",
                                                 "8"="Coelomocytes",
                                                 "9"="Am/PH sheath cells",
                                                 "10"="Failed QC",
                                                 "11"="Touch receptor neurons",
                                                 "12"="Intestinal/rectal muscle",
```

```
                                                "13"="Pharyngeal neurons",
                                                "14"="NA",
                                                "15"="flp-1(+) interneurons",
                                                "16"="Canal associated neurons",
                                                "17"="Ciliated sensory neurons",
                                                "18"="Other interneurons",
                                                "19"="Pharyngeal gland",
                                                "20"="Failed QC",
                                                "21"="Ciliated sensory neurons",
                                                "22"="Oxygen sensory neurons",
                                                "23"="Ciliated sensory neurons",
                                                "24"="Ciliated sensory neurons",
                                                "25"="Ciliated sensory neurons",
                                                "26"="Ciliated sensory neurons",
                                                "27"="Oxygen sensory neurons",
                                                "28"="Ciliated sensory neurons",
                                                "29"="Unclassified neurons",
                                                "30"="Socket cells",
                                                "31"="Failed QC",
                                                "32"="Pharyngeal gland",
                                                "33"="Ciliated sensory neurons",
                                                "34"="Ciliated sensory neurons",
                                                "35"="Ciliated sensory neurons",
                                                "36"="Failed QC",
                                                "37"="Ciliated sensory neurons",
                                                "38"="Pharyngeal muscle")
neurons_cds <- cds[,grepl("neurons", colData(cds)$assigned_cell_type, ignore.case=TRUE)]
 pr_graph_test_res <- graph_test(cds, neighbor_graph="knn")
```

`identity_table` *Report matrix and model identity information.*

Description

Write the cell_data_set matrix and model identity information to stdout.

Usage

```
identity_table(cds)
```

Arguments

`cds`	the cell_data_set to use.

Details

A matrix identity is a checksum that is stored in the cell_data_set when a reduced dimension matrix is created and when certain functions read count matrices into the cell_data_set, such as load_mm_data(). At the same time, the same checksum is stored as the model identity in order to link the model to its matrix.

Additionally, Monocle3 stores the identity of the matrix from which the matrix was made. For example, in the case of a UMAP reduced dimension matrix made from a PCA reduced dimension matrix, the cell_data_set has the identities of both the UMAP and the PCA matrices. The UMAP identity is stored as 'matrix_id' and the PCA as 'prev_matrix_id'. Similarly, the model and the previous model identities are stored as 'model_id' and 'prev_model_id'. This allows one to trace a matrix to its origin, which may be helpful when a cell_data_set is partially reprocessed; for example, if preprocess_cds() is re-run but reduce_dimension() is not. Also, it may be helpful when transform models are loaded with the load_transform_models() function, in which case the matrix and model identities will differ.

The identity of the model used to transform a matrix is stored with the matrix identity information as 'model_id'. Ordinarily, the matrix 'matrix_id' and 'model_id' and the corresponding model 'model_id' will have the same string value. However, they differ when the preprocess_transform() and reduce_dim_transform() functions are used to transform a matrix.

Notes:

- Certain file and directory paths may be stored in the cell_data_set as identifiers.
- Checksums are calculated using the digest function in the digest package. The matrix dimensions are stored with the checksum.
- Matrix transformations such as subsetting and row and or column reordering do not affect the matrix identity.
- The matrix identity string is stored in the internal metadata slot of the cell_data_set and the model identity string is stored in the model object in the cds@reduce_dim_aux slot of the cell_data_set.

Value

Write identity information to stdout.

Examples

```
cds <- load_a549()
cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
identity_table(cds)
```

learn_graph *Learn principal graph from the reduced dimension space using reversed graph embedding*

Description

Monocle3 aims to learn how cells transition through a biological program of gene expression changes in an experiment. Each cell can be viewed as a point in a high-dimensional space, where each dimension describes the expression of a different gene. Identifying the program of gene expression changes is equivalent to learning a *trajectory* that the cells follow through this space. However, the more dimensions there are in the analysis, the harder the trajectory is to learn. Fortunately, many genes typically co-vary with one another, and so the dimensionality of the data can be reduced with a wide variety of different algorithms. Monocle3 provides two different algorithms for dimensionality reduction via `reduce_dimension` (UMAP and tSNE). Both take a cell_data_set object and a number of dimensions allowed for the reduced space. You can also provide a model formula indicating some variables (e.g. batch ID or other technical factors) to "subtract" from the data so it doesn't contribute to the trajectory. The function `learn_graph` is the fourth step in the trajectory building process after `preprocess_cds`, `reduce_dimension`, and `cluster_cells`. After `learn_graph`, `order_cells` is typically called.

Usage

```
learn_graph(
  cds,
  use_partition = TRUE,
  close_loop = TRUE,
  learn_graph_control = NULL,
  verbose = FALSE
)
```

Arguments

`cds`	the cell_data_set upon which to perform this operation
`use_partition`	logical parameter that determines whether to use partitions calculated during `cluster_cells` and therefore to learn disjoint graph in each partition. When `use_partition = FALSE`, a single graph is learned across all partitions. Default is TRUE.
`close_loop`	logical parameter that determines whether or not to perform an additional run of loop closing after estimating the principal graphs to identify potential loop structure in the data space. Default is TRUE.
`learn_graph_control`	NULL or a list of control parameters to be passed to the reversed graph embedding function. Default is NULL. A list of potential control parameters is provided in details.
`verbose`	Whether to emit verbose output during graph learning.

Value

an updated cell_data_set object

Optional `learn_graph_control` parameters

euclidean_distance_ratio: The maximal ratio between the euclidean distance of two tip nodes in the spanning tree and the maximum distance between any connecting points on the spanning tree allowed to be connected during the loop closure procedure. Default is 1.

geodesic_distance_ratio: The minimal ratio between the geodesic distance of two tip nodes in the spanning tree and the length of the diameter path on the spanning tree allowed to be connected during the loop closure procedure. (Both euclidean_distance_ratio and geodesic_distance_ratio need to be satisfied to introduce the edge for loop closure). Default is 1/3.

minimal_branch_len: The minimal length of the diameter path for a branch to be preserved during graph pruning procedure. Default is 10.

orthogonal_proj_tip: Whether to perform orthogonal projection for cells corresponding to the tip principal points. Default is FALSE.

prune_graph: Whether or not to perform an additional round of graph pruning to remove small insignificant branches. Default is TRUE.

scale:

ncenter:

nn.k: Maximum number of nearest neighbors to compute in the reversed graph embedding. Set k=NULL to let learn_graph estimate k. Default is 25.

rann.k: nn.k replaces rann.k but rann.k is available for compatibility with existing code.

maxiter:

eps:

L1.gamma:

L1.sigma:

nn.method: The method to use for finding nearest neighbors. nn.method can be one of 'nn2', 'annoy', or 'hnsw'.

nn.metric: The distance metric for the annoy or hnsw nearest neighbor index build. See help(set_nn_control) for more information.

nn.n_trees: The number of trees used to build the annoy nearest neighbor index. See help(set_nn_control) for more information.

nn.search_k: The number of nodes to search in an annoy index search. See help(set_nn_control) for more information.

nn.M: Related to internal dimensionality of HNSW index. See help(set_nn_control) for more information.

nn.ef_construction: Controls the HNSW index build speed/accuracy tradeoff.

nn.ef: Controls the HNSW index search speed/accuracy tradeoff. See help(set_nn_control) for more information.

nn.grain_size: Used by annoy and HNSW to set the minimum amount of work to do per thread. See help(set_nn_control) for more information.

nn.cores: Used by annoy and HNSW to control the number of threads used. See help(set_nn_control) for more information.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
```

---

load_a549 *Build a small cell_data_set.*

Description

Build a small cell_data_set.

Usage

```
load_a549()
```

Value

cds object

Examples

```
cds <- load_a549()
```

`load_cellranger_data` *Load data from the 10x Genomics Cell Ranger pipeline*

Description

Loads cellranger data into a cell_data_set object. Note that if your dataset is from version 3.0 and contains non-Gene-Expression data (e.g. Antibodies or CRISPR features), only the Gene Expression data is returned.

Usage

```
load_cellranger_data(
  pipestance_path = NULL,
  genome = NULL,
  barcode_filtered = TRUE,
  umi_cutoff = 100
)
```

Arguments

`pipestance_path`	Path to the output directory produced by Cell Ranger
`genome`	The desired genome (e.g., 'hg19' or 'mm10')
`barcode_filtered`	Load only the cell-containing barcodes
`umi_cutoff`	Numeric, desired cutoff to include a cell. Default is 100.

Details

- the pipestance_path argument takes the name of a Cell Ranger output directory, in which it looks for the required data files, for example, pipestance_path=10x_data
- for Cell Ranger version 2 data, load_cellranger_data expects to find the required files barcodes.tsv, genes.tsv, and matrix.mtx in the directories as
  - 10x_data/outs/filtered_gene_bc_matrices//barcodes.tsv
  - 10x_data/outs/filtered_gene_bc_matrices//genes.tsv
  - 10x_data/outs/filtered_gene_bc_matrices//matrix.mtx

  where is the name of a genome. load_cellranger_data expects to find either a single genome directory in 10x_data/outs/filtered_gene_bc_matrices or a genome directory with the name given with the genome argument.
- for Cell Ranger version 3 data, load_cellranger_data expects to find the required files barcodes.tsv.gz, features.tsv.gz, and matrix.mtx.gz in the directories as
  - 10x_data/outs/filtered_feature_bc_matrix/barcodes.tsv.gz
  - 10x_data/outs/filtered_feature_bc_matrix/features.tsv.gz
  - 10x_data/outs/filtered_feature_bc_matrix/matrix.mtx.gz
- if any of the files is not in the expected directory, load_cellranger_data will terminate with an error

Value

a new cell_data_set object

Examples

```
cell_ranger_data <- system.file("extdata", "cell_ranger_3", package = "monocle3")
gene_bc_matrix <- load_cellranger_data(cell_ranger_data)
```

`load_mm_data` *Load data from matrix market format files.*

Description

Load data from matrix market format files.

Usage

```
load_mm_data(
  mat_path,
  feature_anno_path,
  cell_anno_path,
  header = FALSE,
  feature_metadata_column_names = NULL,
  cell_metadata_column_names = NULL,
  umi_cutoff = 100,
  quote = "\"'",
  sep = "\t"
)
```

Arguments

`mat_path`	Path to the Matrix Market .mtx matrix file. The values are read and stored as a sparse matrix with nrows and ncols, as inferred from the file. Required.
`feature_anno_path`	Path to a feature annotation file. The feature_anno_path file must have nrows lines and at least one column. The values in the first column label the matrix rows and each must be distinct in the column. Values in additional columns are stored in the cell_data_set 'gene' metadata. For gene features, we urge use of official gene IDs for labels, such as Ensembl or Wormbase IDs. In this case, the second column has typically a 'short' gene name. Additional information such as gene_biotype may be stored in additional columns starting with column 3. Required.
`cell_anno_path`	Path to a cell annotation file. The cell_anno_path file must have ncols lines and at least one column. The values in the first column label the matrix columns and each must be distinct in the column. Values in additional columns are stored in the cell_data_set cells metadata. Required.

header Logical set to TRUE if both feature_anno_path and cell_anno_path files have column headers, or set to FALSE if both files do not have column headers (only these cases are supported). The files may have either ncols or ncols-1 header fields. In both cases, the first column is used as the matrix dimension names. The default is FALSE.

feature_metadata_column_names
A character vector of feature metadata column names. The number of names must be one less than the number of columns in the feature_anno_path file. These values will replace those read from the feature_anno_path file header, if present. The default is NULL.

cell_metadata_column_names
A character vector of cell metadata column names. The number of names must be one less than the number of columns in the cell_anno_path file. These values will replace those read from the cell_anno_path file header, if present. The default is NULL.

umi_cutoff UMI per cell cutoff. Columns (cells) with less than umi_cutoff total counts are removed from the matrix. The default is 100.

quote A character string specifying the quoting characters used in the feature_anno_path and cell_anno_path files. The default is "\"'".

sep field separator character in the annotation files. If sep = "", the separator is white space, that is, one or more spaces, tabs, newlines, or carriage returns. The default is the tab character for tab-separated-value files.

Value

cds object

Comments

- load_mm_data estimates size factors.

Examples

```
pmat<-system.file("extdata", "matrix.mtx.gz", package = "monocle3")
prow<-system.file("extdata", "features_c3h0.txt", package = "monocle3")
pcol<-system.file("extdata", "barcodes_c2h0.txt", package = "monocle3")
cds <- load_mm_data( pmat, prow, pcol,
                     feature_metadata_column_names =
                     c('gene_short_name', 'gene_biotype'), sep='' )

# In this example, the features_c3h0.txt file has three columns,
# separated by spaces. The first column has official gene names, the
# second has short gene names, and the third has gene biotypes.
```

load_monocle_objects *Load a full Monocle3 cell_data_set.*

Description

Load a full Monocle3 cell_data_set, which was saved using save_monocle_objects. For more information read the help information for save_monocle_objects.

Usage

```
load_monocle_objects(directory_path)
```

Arguments

directory_path	a string giving the name of the directory from which to read the saved cell_data_set files.

Value

a cell_data_set.

Examples

```
  ## Not run:
    cds <- load_a549()
    save_monocle_objects(cds, 'mo')
    cds1 <- load_monocle_objects('mo')

## End(Not run)
```

load_mtx_data *Load data from matrix market format*

Description

Load data from matrix market format

Usage

```
load_mtx_data(mat_path, gene_anno_path, cell_anno_path, umi_cutoff = 100)
```

Arguments

mat_path	Path to the .mtx matrix market file.
gene_anno_path	Path to gene annotation file.
cell_anno_path	Path to cell annotation file.
umi_cutoff	UMI per cell cutoff, default is 100.

Value

cds object

Examples

```
pmat<-system.file("extdata", "matrix.mtx.gz", package = "monocle3")
prow<-system.file("extdata", "features_c3h0.txt", package = "monocle3")
pcol<-system.file("extdata", "barcodes_c2h0.txt", package = "monocle3")
cds <- load_mtx_data( pmat, prow, pcol)
```

`load_transform_models` *Load transform models into a cell_data_set.*

Description

Load transform models, which were saved using save_transform_models, into a cell_data_set. This function over-writes existing models in the cell_data_set. For more information read the help information for save_transform_models.

Usage

```
load_transform_models(cds, directory_path)
```

Arguments

`cds`	a cell_data_set to be transformed using the models.
`directory_path`	a string giving the name of the directory from which to read the model files.

Value

a cell_data_set with the transform models loaded by load_transform_models.

Examples

```
  ## Not run:
    cds <- load_a549()
    cds <- preprocess_cds(cds)
    cds <- reduce_dimension(cds)
    save_transform_models(cds, 'tm')
    cds1 <- load_a549()
    cds1 <- load_transform_models(cds1, 'tm')

## End(Not run)
```

load_worm_embryo *Build a cell_data_set from C. elegans embryo data.*

Description

Build a cell_data_set from C. elegans embryo data.

Usage

```
load_worm_embryo()
```

Value

cds object

load_worm_l2 *Build a cell_data_set from C. elegans L2 data.*

Description

Build a cell_data_set from C. elegans L2 data.

Usage

```
load_worm_l2()
```

Value

cds object

make_cds_nn_index *Make and store a nearest neighbor index in the cell_data_set.*

Description

Make a nearest neighbor index from the specified reduction_method matrix in the cell_data_set using either the default nearest neighbor method or the method specified in the nn_control list parameter, and store the index in the cell_data_set. This function returns a cell_data_set.

Usage

```
make_cds_nn_index(
  cds,
  reduction_method = c("UMAP", "PCA", "LSI", "Aligned", "tSNE"),
  nn_control = list(),
  verbose = FALSE
)
```

Arguments

cds	a cell_data_set with the reduced dimension matrix from which to make the nearest neighbor index and with which the index is stored.
reduction_method	a string giving the reduced dimension matrix to use for making the nn_index nearest neighbor index. Note: distances in tSNE space reflect spatial differences poorly so using nearest neighbors with it may be meaningless.
nn_control	a list of parameters to use for making the nearest neighbor index. See the set_nn_control help for details.
verbose	a boolean indicating whether to emit verbose output.

Value

a cell_data_set with the stored index.

Examples

```
cds <- load_a549()
cds <- preprocess_cds(cds)
cds <- make_cds_nn_index(cds, 'PCA')
```

---

make_nn_index *Make a nearest neighbor index.*

Description

Make a nearest neighbor index from the subject_matrix using either the default nearest neighbor method or the method specified in the nn_control list parameter. The function returns the index.

Usage

```
make_nn_index(subject_matrix, nn_control = list(), verbose = FALSE)
```

Arguments

subject_matrix	the matrix used to build the index.
nn_control	a list of parameters used to make the nearest neighbor index. See the set_nn_control help for details.
verbose	a boolean indicating whether to emit verbose output.

Value

a nearest neighbor index.

Examples

```
cds <- load_a549()
cds <- preprocess_cds(cds)
nn_index <- make_nn_index(SingleCellExperiment::reducedDims(cds)[['PCA']])
```

---

mc_es_apply *Multicore apply-like function for cell_data_set*

Description

mc_es_apply computes the row-wise or column-wise results of FUN, just like esApply. Variables in colData from cds are available in FUN.

Usage

```
mc_es_apply(
  cds,
  MARGIN,
  FUN,
  required_packages,
  cores = 1,
  convert_to_dense = TRUE,
  reduction_method = "UMAP",
  ...
)
```

Arguments

cds	A cell_data_set object.
MARGIN	The margin to apply to, either 1 for rows (samples) or 2 for columns (features).
FUN	Any function.
required_packages	A list of packages FUN will need. Failing to provide packages needed by FUN will generate errors in worker threads.

cores The number of cores to use for evaluation.

convert_to_dense
Whether to force conversion of a sparse matrix to a dense one before calling FUN.

reduction_method
character, the method used to reduce dimension. Default "UMAP".

... Additional parameters for FUN.

Value

The result of with(colData(cds) apply(counts(cds)), MARGIN, FUN, ...))

---

model_predictions *Predict values of new data using fit_models model.*

Description

Predict new data values and return as a matrix

Usage

```
model_predictions(model_tbl, new_data, type = "response")
```

Arguments

model_tbl	A tibble of model objects, generally output of fit_models.
new_data	A data frame of new data to be passed to predict for prediction.
type	String of type to pass to predict. Default is "response".

Value

Prediction matrix.

---

new_cell_data_set *Create a new cell_data_set object.*

Description

Create a new cell_data_set object.

Usage

```
new_cell_data_set(expression_data, cell_metadata = NULL, gene_metadata = NULL)
```

Arguments

expression_data	expression data matrix for an experiment, can be a sparseMatrix.
cell_metadata	data frame containing attributes of individual cells, where row.names(cell_metadata) = colnames(expression_data).
gene_metadata	data frame containing attributes of features (e.g. genes), where row.names(gene_metadata) = row.names(expression_data).

Value

a new cell_data_set object

Examples

```
  small_a549_colData_df <- readRDS(system.file("extdata",
                                               "small_a549_dex_pdata.rda",
                                               package = "monocle3"))
  small_a549_rowData_df <- readRDS(system.file("extdata",
                                               "small_a549_dex_fdata.rda",
                                               package = "monocle3"))
  small_a549_exprs <- readRDS(system.file("extdata",
                                          "small_a549_dex_exprs.rda",
                                          package = "monocle3"))
  small_a549_exprs <- small_a549_exprs[,row.names(small_a549_colData_df)]

  cds <- new_cell_data_set(expression_data = small_a549_exprs,
                           cell_metadata = small_a549_colData_df,
                           gene_metadata = small_a549_rowData_df)
```

---

normalized_counts *Return a size-factor normalized and (optionally) log-transformed expression matrix*

Description

Return a size-factor normalized and (optionally) log-transformed expression matrix

Usage

```
normalized_counts(
  cds,
  norm_method = c("log", "binary", "size_only"),
  pseudocount = 1
)
```

Arguments

cds	A CDS object to calculate normalized expression matrix from.
norm_method	String indicating the normalization method. Options are "log" (Default), "binary" and "size_only".
pseudocount	A pseudocount to add before log transformation. Ignored if norm_method is not "log". Default is 1.

Value

Size-factor normalized, and optionally log-transformed, expression matrix.

Examples

```
cds <- load_a549()
normalized_matrix <- normalized_counts(cds)
```

---

order_cells *Orders cells according to pseudotime.*

Description

Assigns cells a pseudotime value based on their projection on the principal graph learned in the learn_graph function and the position of chosen root states. This function takes as input a cell_data_set and returns it with pseudotime information stored internally. order_cells() optionally takes "root" state(s) in the form of cell or principal graph node IDs, which you can use to specify the start of the trajectory. If you don't provide a root state, an plot will be generated where you can choose the root state(s) interactively. The trajectory will be composed of segments.

Usage

```
order_cells(
  cds,
  reduction_method = "UMAP",
  root_pr_nodes = NULL,
  root_cells = NULL,
  verbose = FALSE
)
```

Arguments

cds	the cell_data_set upon which to perform this operation
reduction_method	a string specifying the reduced dimension method to use when ordering cells. Currently only "UMAP" is supported.

root_pr_nodes	NULL or a vector of starting principal points. If provided, pseudotime will start (i.e. be zero) at these graph nodes. You can find the principal point names by running plot_cells with label_principal_points = TRUE. Both root_pr_nodes and root_cells cannot be provided.
root_cells	NULL or a vector of starting cells. If provided, pseudotime will start (i.e. be zero) at these cells. Both root_pr_nodes and root_cells cannot be provided.
verbose	Whether to show running information for order_cells

Value

an updated cell_data_set object.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
cds <- order_cells(cds, root_pr_nodes='Y_21')
```

---

partitions *Generic to extract partitions from CDS object*

Description

Generic to extract partitions from CDS object

Usage

```
partitions(x, reduction_method = "UMAP")
```

Arguments

x	A cell_data_set object.
reduction_method	Reduced dimension to partitions clusters for.

Value

Partitions.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                        package='monocle3'))
cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
partitions_factors <- partitions(cds, "UMAP")
```

---

partitions,cell_data_set-method

Method to extract partitions from CDS object

Description

Method to extract partitions from CDS object

Usage

```
## S4 method for signature 'cell_data_set'
partitions(x, reduction_method = "UMAP")
```

Arguments

x	A cell_data_set object.
reduction_method	Reduced dimension to partitions clusters for.

Value

Partitions.

`pData` *Generic to access cds colData table*

Description

Generic to access cds colData table

Usage

```
pData(x)
```

Arguments

x	A cell_data_set object.

Value

colData.

Examples

```
cds <- load_a549()
pData(cds)
```

`pData,cell_data_set-method` *Method to access cds colData table*

Description

Method to access cds colData table

Usage

```
## S4 method for signature 'cell_data_set'
pData(x)
```

Arguments

x	A cell_data_set object.

Value

colData.

pData<- *Generic to set cds colData table*

Description

Generic to set cds colData table

Usage

```
pData(x) <- value
```

Arguments

x	A cell_data_set object.
value	A data frame to set to colData table.

Value

x.

Examples

```
cds <- load_a549()
pData(cds)[['row_index']] <- seq(nrow(pData(cds)))
```

pData<-,cell_data_set-method *Method to set cds colData table*

Description

Method to set cds colData table

Usage

```
## S4 replacement method for signature 'cell_data_set'
pData(x) <- value
```

Arguments

x	A cell_data_set object.
value	A data frame to set to colData table.

Value

x.

Examples

```
cds <- load_a549()
pData(cds)[['row_index']] <- seq(nrow(pData(cds)))
```

---

plot_cells *Plots the cells along with their trajectories.*

Description

Plots the cells along with their trajectories.

Usage

```
plot_cells(
  cds,
  x = 1,
  y = 2,
  reduction_method = c("UMAP", "tSNE", "PCA", "LSI", "Aligned"),
  color_cells_by = "cluster",
  group_cells_by = "cluster",
  genes = NULL,
  show_trajectory_graph = TRUE,
  trajectory_graph_color = "grey28",
  trajectory_graph_segment_size = 0.75,
  norm_method = c("log", "size_only"),
  label_cell_groups = TRUE,
  label_groups_by_cluster = TRUE,
  group_label_size = 2,
  labels_per_group = 1,
  label_branch_points = TRUE,
  label_roots = TRUE,
  label_leaves = TRUE,
  graph_label_size = 2,
  cell_size = 0.35,
  cell_stroke = I(cell_size/2),
  alpha = 1,
  min_expr = 0.1,
  rasterize = FALSE,
  scale_to_range = TRUE,
  label_principal_points = FALSE
)
```

Arguments

cds	cell_data_set for the experiment
x	the column of SingleCellExperiment::reducedDims(cds) to plot on the horizontal axis
y	the column of SingleCellExperiment::reducedDims(cds) to plot on the vertical axis
reduction_method	The lower dimensional space in which to plot cells. Must be one of "UMAP", "tSNE", "PCA" and "LSI".
color_cells_by	What to use for coloring the cells. Must be either the name of a column of colData(cds), or one of "clusters", "partitions", or "pseudotime".
group_cells_by	How to group cells when labeling them. Must be either the name of a column of colData(cds), or one of "clusters" or "partitions". If a column in colData(cds), must be a categorical variable.
genes	Facet the plot, showing the expression of each gene in a facet panel. Must be either a list of gene ids (or short names), or a dataframe with two columns that groups the genes into modules that will be aggregated prior to plotting. If the latter, the first column must be gene ids, and the second must the group for each gene.
show_trajectory_graph	Whether to render the principal graph for the trajectory. Requires that learn_graph() has been called on cds.
trajectory_graph_color	The color to be used for plotting the trajectory graph.
trajectory_graph_segment_size	The size of the line segments used for plotting the trajectory graph.
norm_method	How to normalize gene expression scores prior to plotting them. Must be one of "log" or "size_only".
label_cell_groups	Whether to label cells in each group (as specified by group_cells_by) according to the most frequently occurring label(s) (as specified by color_cells_by) in the group. If false, plot_cells() simply adds a traditional color legend.
label_groups_by_cluster	Instead of labeling each cluster of cells, place each label once, at the centroid of all cells carrying that label.
group_label_size	Font size to be used for cell group labels.
labels_per_group	How many labels to plot for each group of cells. Defaults to 1, which plots only the most frequent label per group.
label_branch_points	Whether to plot a label for each branch point in the principal graph.
label_roots	Whether to plot a label for each root in the principal graph.
label_leaves	Whether to plot a label for each leaf node in the principal graph.

graph_label_size
: How large to make the branch, root, and leaf labels.

cell_size
: The size of the point for each cell

cell_stroke
: The stroke used for plotting each cell - default is 1/2 of the cell_size

alpha
: Alpha for the cells. Useful for reducing overplotting.

min_expr
: Minimum expression threshold for plotting genes

rasterize
: Whether to plot cells as a rastered bitmap. Requires the ggrastr package.

scale_to_range
: Logical indicating whether to scale expression to percent of maximum expression.

label_principal_points
: Logical indicating whether to label roots, leaves, and branch points with principal point names. This is useful for order_cells and choose_graph_segments in non-interactive mode.

Value

a ggplot2 plot object

Examples

```
  ## Not run:
    lung <- load_A549()
    plot_cells(lung)
    plot_cells(lung, color_cells_by="log_dose")
    plot_cells(lung, markers="GDF15")

## End(Not run)
```

---

plot_cells_3d *Plot a dataset and trajectory in 3 dimensions*

Description

Plot a dataset and trajectory in 3 dimensions

Usage

```
plot_cells_3d(
  cds,
  dims = c(1, 2, 3),
  reduction_method = c("UMAP", "tSNE", "PCA", "LSI", "Aligned"),
  color_cells_by = "cluster",
  genes = NULL,
  show_trajectory_graph = TRUE,
  trajectory_graph_color = "black",
```

```
  trajectory_graph_segment_size = 5,
  norm_method = c("log", "size_only"),
  color_palette = NULL,
  color_scale = "Viridis",
  cell_size = 25,
  alpha = 1,
  min_expr = 0.1
)
```

Arguments

cds	cell_data_set to plot
dims	numeric vector that indicates the dimensions used to create the 3D plot, by default it is the first three dimensions.
reduction_method	string indicating the reduction method to plot.
color_cells_by	the cell attribute (e.g. the column of colData(cds)) to map to each cell's color. Default is cluster.
genes	a gene name or gene id to color the plot by.
show_trajectory_graph	a logical used to indicate whether to graph the principal graph backbone. Default is TRUE.
trajectory_graph_color	the color of graph backbone. Default is black.
trajectory_graph_segment_size	numeric indicating the width of the graph backbone. Default is 5.
norm_method	string indicating the method used to transform gene expression when gene markers are provided. Default is "log". "size_only" is also supported.
color_palette	List of colors to pass to plotly for coloring cells by categorical variables. Default is NULL. When NULL, plotly uses default colors.
color_scale	The name of the color scale passed to plotly for coloring cells by numeric scale. Default is "Viridis".
cell_size	numeric indicating the size of the point to be plotted. Default is 25.
alpha	numeric indicating the alpha value of the plotted cells. Default is 1.
min_expr	numeric indicating the minimum marker gene value to be colored. Default is 0.1.

Value

a plotly plot object

Examples

```
## Not run:
plot_cells_3d(cds, markers=c("Rbfox3, Neurod1", "Sox2"))

## End(Not run)
```

plot_genes_by_group *Create a dot plot to visualize the mean gene expression and percentage of expressed cells in each group of cells*

Description

Create a dot plot to visualize the mean gene expression and percentage of expressed cells in each group of cells

Usage

```
plot_genes_by_group(
  cds,
  markers,
  group_cells_by = "cluster",
  reduction_method = "UMAP",
  norm_method = c("log", "size_only"),
  lower_threshold = 0,
  max.size = 10,
  ordering_type = c("cluster_row_col", "maximal_on_diag", "none"),
  axis_order = c("group_marker", "marker_group"),
  flip_percentage_mean = FALSE,
  pseudocount = 1,
  scale_max = 3,
  scale_min = -3,
  color_by_group = FALSE
)
```

Arguments

cds	A cell_data_set for plotting.
markers	A list of gene ids (or short names) to show in the plot
group_cells_by	How to group cells when labeling them. Must be either the name of a column of colData(cds), or one of "clusters" or "partitions". If a column in colData(cds), must be a categorical variable.
reduction_method	The dimensionality reduction method used for clusters and partitions.
norm_method	Determines how to transform expression values prior to plotting. Options are "log" and "size_only". Default is "log".
lower_threshold	The lowest gene expressed treated as expressed. By default, zero.
max.size	The maximum size of the dot. By default, it is 10.
ordering_type	How to order the genes / groups on the dot plot. Only accepts 'cluster_row_col' (use biclustering to cluster the rows and columns), 'maximal_on_diag' (position each column so that the maximal color shown on each column on the diagonal, if

	the current maximal is used in earlier columns, the next largest one is position), and 'none' (preserve the ordering from the input gene or alphabetical ordering of groups). Default is 'cluster_row_col'.
axis_order	Whether to put groups on x-axis, genes on y-axis (option 'group_marker') or the reverse order (option 'marker_group'). Default is "group_marker".
flip_percentage_mean	Logical indicating whether to use color of the dot to represent the percentage (by setting flip_percentage_mean = FALSE, default) and size of the dot the mean expression, or the opposite (by setting flip_percentage_mean = TRUE).
pseudocount	A pseudo-count added to the average gene expression.
scale_max	The maximum value (in standard deviations) to show in the heatmap. Values larger than this are set to the max.
scale_min	The minimum value (in standard deviations) to show in the heatmap. Values smaller than this are set to the min.
color_by_group	Color cells by the group to which they belong.

Value

a ggplot2 plot object

`plot_genes_in_pseudotime` *Plots expression for one or more genes as a function of pseudotime*

Description

Plots expression for one or more genes as a function of pseudotime

Usage

```
plot_genes_in_pseudotime(
  cds_subset,
  min_expr = NULL,
  cell_size = 0.75,
  nrow = NULL,
  ncol = 1,
  panel_order = NULL,
  color_cells_by = "pseudotime",
  trend_formula = "~ splines::ns(pseudotime, df=3)",
  label_by_short_name = TRUE,
  vertical_jitter = NULL,
  horizontal_jitter = NULL
)
```

Arguments

cds_subset	subset cell_data_set including only the genes to be plotted.
min_expr	the minimum (untransformed) expression level to plot.
cell_size	the size (in points) of each cell used in the plot.
nrow	the number of rows used when laying out the panels for each gene's expression.
ncol	the number of columns used when laying out the panels for each gene's expression
panel_order	vector of gene names indicating the order in which genes should be laid out (left-to-right, top-to-bottom). If label_by_short_name = TRUE, use gene_short_name values, otherwise use feature IDs.
color_cells_by	the cell attribute (e.g. the column of colData(cds)) to be used to color each cell.
trend_formula	the model formula to be used for fitting the expression trend over pseudotime.
label_by_short_name	label figure panels by gene_short_name (TRUE) or feature ID (FALSE).
vertical_jitter	A value passed to ggplot to jitter the points in the vertical dimension. Prevents overplotting, and is particularly helpful for rounded transcript count data.
horizontal_jitter	A value passed to ggplot to jitter the points in the horizontal dimension. Prevents overplotting, and is particularly helpful for rounded transcript count data.

Value

a ggplot2 plot object

---

plot_genes_violin *Plot expression for one or more genes as a violin plot*

Description

Accepts a subset of a cell_data_set and an attribute to group cells by, and produces a ggplot2 object that plots the level of expression for each group of cells.

Usage

```
plot_genes_violin(
  cds_subset,
  group_cells_by = NULL,
  min_expr = 0,
  nrow = NULL,
  ncol = 1,
  panel_order = NULL,
  label_by_short_name = TRUE,
  normalize = TRUE,
  log_scale = TRUE,
  pseudocount = 0
)
```

Arguments

cds_subset	Subset cell_data_set to be plotted.
group_cells_by	NULL of the cell attribute (e.g. the column of colData(cds)) to group cells by on the horizontal axis. If NULL, all cells are plotted together.
min_expr	the minimum (untransformed) expression level to be plotted. Default is 0.
nrow	the number of panels per row in the figure.
ncol	the number of panels per column in the figure.
panel_order	the order in which genes should be laid out (left-to-right, top-to-bottom). Should be gene_short_name if label_by_short_name = TRUE or feature ID if label_by_short_name = FALSE.
label_by_short_name	label figure panels by gene_short_name (TRUE) or feature id (FALSE). Default is TRUE.
normalize	Logical, whether or not to normalize expression by size factor. Default is TRUE.
log_scale	Logical, whether or not to scale data logarithmically. Default is TRUE.
pseudocount	A pseudo-count added to the gene expression. Default is 0.

Value

a ggplot2 plot object

Examples

```
cds <- load_a549()
cds_subset <- cds[row.names(subset(rowData(cds),
                gene_short_name %in% c("ACTA1", "ID1", "CCNB2"))),]
plot_genes_violin(cds_subset, group_cells_by="culture_plate", ncol=2,
                 min_expr=0.1)
```

---

plot_pc_variance_explained

Plots the fraction of variance explained by the each component based on PCA from the normalized expression data determined using preprocess_cds. This is the fraction of the component variance relative to the variance of the components retained in the PCA; not the total variance.

Description

Plots the fraction of variance explained by the each component based on PCA from the normalized expression data determined using preprocess_cds. This is the fraction of the component variance relative to the variance of the components retained in the PCA; not the total variance.

Usage

```
plot_pc_variance_explained(cds)
```

Arguments

cds	cell_data_set of the experiment.

Value

ggplot object.

Examples

```
cds <- load_a549()
cds <- preprocess_cds(cds)
plot_pc_variance_explained(cds)
```

---

plot_percent_cells_positive

Plots the number of cells expressing one or more genes above a given value as a barplot

---

Description

@description Accepts a subset cell_data_set and the parameter `group_cells_by`, used for dividing cells into groups. Returns one or more bar graphs (one graph for each gene in the cell_data_set). Each graph shows the percentage (or number) of cells that express a gene in each sub-group in the cell_data_set.

Usage

```
plot_percent_cells_positive(
  cds_subset,
  group_cells_by = NULL,
  min_expr = 0,
  nrow = NULL,
  ncol = 1,
  panel_order = NULL,
  plot_as_count = FALSE,
  label_by_short_name = TRUE,
  normalize = TRUE,
  plot_limits = NULL,
  bootstrap_samples = 100,
  conf_int_alpha = 0.95
)
```

Arguments

cds_subset	Subset cell_data_set to be plotted.
group_cells_by	the cell attribute (e.g. the column of colData(cds)) to group cells by on the horizontal axis. If NULL, all cells plotted as one group.
min_expr	the minimum (untransformed) expression level to consider the gene 'expressed'. Default is 0.
nrow	the number of panels per row in the figure.
ncol	the number of panels per column in the figure.
panel_order	the order in which genes should be laid out (left-to-right, top-to-bottom). Should be gene_short_name if label_by_short_name = TRUE or feature ID if label_by_short_name = FALSE.
plot_as_count	Logical, whether to plot as a count of cells rather than a percent. Default is FALSE.
label_by_short_name	label figure panels by gene_short_name (TRUE) or feature id (FALSE). Default is TRUE.
normalize	Logical, whether or not to normalize expression by size factor. Default is TRUE.
plot_limits	A pair of number specifying the limits of the y axis. If NULL, scale to the range of the data. Example c(0,100).
bootstrap_samples	The number of bootstrap replicates to generate when plotting error bars. Default is 100.
conf_int_alpha	The size of the confidence interval to use when plotting error bars. Default is 0.95.

Value

a ggplot2 plot object

Examples

```
cds <- load_a549()
cds_subset <- cds[row.names(subset(rowData(cds),
                                   gene_short_name %in% c("NDRG4", "HBG2"))),]
plot_percent_cells_positive(cds_subset, group_cells_by="culture_plate")
```

`preprocess_cds` *Preprocess a cds to prepare for trajectory inference*

Description

Most analyses (including trajectory inference, and clustering) in Monocle3, require various normalization and preprocessing steps. `preprocess_cds` executes and stores these preprocessing steps.

Specifically, depending on the options selected, `preprocess_cds` first normalizes the data by log and size factor to address depth differences, or by size factor only. Next, `preprocess_cds` calculates a lower dimensional space that will be used as the input for further dimensionality reduction like tSNE and UMAP.

Usage

```
preprocess_cds(
  cds,
  method = c("PCA", "LSI"),
  num_dim = 50,
  norm_method = c("log", "size_only", "none"),
  use_genes = NULL,
  pseudo_count = NULL,
  scaling = TRUE,
  verbose = FALSE,
  build_nn_index = FALSE,
  nn_control = list()
)
```

Arguments

`cds`	the cell_data_set upon which to perform this operation
`method`	a string specifying the initial dimension method to use, currently either "PCA" or "LSI". For "LSI" (latent semantic indexing), it converts the (sparse) expression matrix into a tf-idf matrix and then performs SVD to decompose the gene expression / cells into certain modules / topics. Default is "PCA".
`num_dim`	the dimensionality of the reduced space.
`norm_method`	Determines how to transform expression values prior to reducing dimensionality. Options are "log", "size_only", and "none". Default is "log". Users should only use "none" if they are confident that their data is already normalized.
`use_genes`	NULL or a list of gene IDs. If a list of gene IDs, only this subset of genes is used for dimensionality reduction. Default is NULL.
`pseudo_count`	NULL or the amount to increase expression values before normalization and dimensionality reduction. If NULL (default), a pseudo_count of 1 is added for log normalization and 0 is added for size factor only normalization.
`scaling`	When this argument is set to TRUE (default), it will scale each gene before running trajectory reconstruction. Relevant for method = PCA only.

verbose	Whether to emit verbose output during dimensionality reduction
build_nn_index	logical When this argument is set to TRUE, preprocess_cds builds and stores the nearest neighbor index from the reduced dimension matrix for later use. Default is FALSE.
nn_control	An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information.

Value

an updated cell_data_set object

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))
cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)
cds <- preprocess_cds(cds)
```

---

preprocess_transform *Apply a preprocess transform model to a cell_data_set.*

Description

Applies a previously calculated preprocess transform model to a new count matrix. For more information read the help information for save_transform_models.

Usage

```
preprocess_transform(
  cds,
  reduction_method = c("PCA", "LSI"),
  block_size = NULL,
  cores = 1
)
```

Arguments

cds	a cell_data_set to be transformed.
reduction_method	a previously loaded transform model that is used to reduce the dimensions of the count matrix in the cell_data_set. The "PCA" and "LSI" transforms are supported. The default is "PCA".
block_size	a numeric value for the DelayedArray block size used only in this function. Default is NULL, which does not affect the current block size.
cores	the number of cores to use for the matrix multiplication. The default is 1.

Value

a cell_data_set with a preprocess reduced count matrix.

Preprocessing notes

Filtering: apply the same filters to the query and reference data set. For example, use the same UMI cutoff value for both data sets. You can check the cutoff value by finding the range of UMI values before applying normalization using range(counts(cds)).

Size factors: use the same method and round_exprs parameters to calculate the Size_Factor values for both data sets. See the estimate_size_factors() help for additional information.

Troubleshooting: if the projection fails, try comparing histograms of various values of the reference and query data sets. For example, in order to examine the size factor values use hist(colData(cds)[['Size_Factor']], breaks=100).

Examples

```
## Not run:
  cell_metadata <- readRDS(system.file('extdata',
                                       'worm_embryo/worm_embryo_coldata.rds',
                                       package='monocle3'))
  gene_metadata <- readRDS(system.file('extdata',
                                       'worm_embryo/worm_embryo_rowdata.rds',
                                       package='monocle3'))
  expression_matrix <- readRDS(system.file('extdata',
                                      'worm_embryo/worm_embryo_expression_matrix.rds',
                                           package='monocle3'))

  cds <- new_cell_data_set(expression_data=expression_matrix,
                           cell_metadata=cell_metadata,
                           gene_metadata=gene_metadata)

  ncell <- nrow(colData(cds))
  cell_sample <- sample(seq(ncell), 2 * ncell / 3)
  cell_set <- seq(ncell) %in% cell_sample
  cds1 <- cds[,cell_set]
  cds1 <- preprocess_cds(cds1)
  save_transform_models(cds1, 'tm')
  cds2 <- cds[,!cell_set]
```

```
    cds2 <- load_transform_models(cds2, 'tm')
    cds2 <- preprocess_transform(cds2, 'PCA')

## End(Not run)
```

`principal_graph` *Generic to extract principal graph from CDS*

Description

Generic to extract principal graph from CDS

Usage

```
principal_graph(x)
```

Arguments

x	A cell_data_set object.

Value

Principle graph.

Examples

```
    cell_metadata <- readRDS(system.file('extdata',
                                         'worm_embryo/worm_embryo_coldata.rds',
                                         package='monocle3'))
    gene_metadata <- readRDS(system.file('extdata',
                                         'worm_embryo/worm_embryo_rowdata.rds',
                                         package='monocle3'))
    expression_matrix <- readRDS(system.file('extdata',
                                        'worm_embryo/worm_embryo_expression_matrix.rds',
                                             package='monocle3'))

    cds <- new_cell_data_set(expression_data=expression_matrix,
                             cell_metadata=cell_metadata,
                             gene_metadata=gene_metadata)

    cds <- preprocess_cds(cds)
    cds <- align_cds(cds, alignment_group =
                    "batch", residual_model_formula_str = "~ bg.300.loading +
                     bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                     bg.r17.loading + bg.b01.loading + bg.b02.loading")
  cds <- reduce_dimension(cds)
  ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
  cds <- cluster_cells(cds)
  cds <- learn_graph(cds)
```

```
pr_gr <- principal_graph(cds)
```

`principal_graph,cell_data_set-method` *Method to extract principal graph from CDS*

Description

Method to extract principal graph from CDS

Usage

```
## S4 method for signature 'cell_data_set'
principal_graph(x)
```

Arguments

`x`	A cell_data_set object.

Value

Principle graph.

`principal_graph<-` *Generic to set principal graph to CDS*

Description

Generic to set principal graph to CDS

Usage

```
principal_graph(x) <- value
```

Arguments

`x`	A cell_data_set object.
`value`	A principal graph object.

Value

x.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
pr_gr <- principal_graph(cds)
principal_graph(cds) <- NULL
principal_graph(cds) <- pr_gr
```

---

principal_graph<-,cell_data_set-method

Generic to set principal graph to CDS

Description

Generic to set principal graph to CDS

Usage

```
## S4 replacement method for signature 'cell_data_set'
principal_graph(x) <- value
```

Arguments

x	A cell_data_set object.
value	A principal graph object.

Value

x.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
pr_gr <- principal_graph(cds)
principal_graph(cds) <- NULL
principal_graph(cds) <- pr_gr
```

---

principal_graph_aux *Generic to extract principal graph auxiliary information from CDS*

Description

Generic to extract principal graph auxiliary information from CDS

Usage

```
principal_graph_aux(x)
```

Arguments

x	A cell_data_set object.

Value

Principal graph auxiliary information.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
pr_gr_aux <- principal_graph_aux(cds)
```

---

principal_graph_aux,cell_data_set-method

Method to extract principal graph auxiliary information from CDS

Description

Method to extract principal graph auxiliary information from CDS

Usage

```
## S4 method for signature 'cell_data_set'
principal_graph_aux(x)
```

Arguments

x	A cell_data_set object.

Value

Principal graph auxiliary information.

`principal_graph_aux<-` *Generic to set principal graph auxiliary information into CDS*

Description

Generic to set principal graph auxiliary information into CDS

Usage

```
principal_graph_aux(x) <- value
```

Arguments

x	A cell_data_set object.
value	A S4Vectors::SimpleList of principal graph auxiliary information.

Value

x.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                "batch", residual_model_formula_str = "~ bg.300.loading +
                 bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                 bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
pr_gr_aux <- principal_graph_aux(cds)
```

```
principal_graph_aux(cds) <- NULL
principal_graph_aux(cds) <- pr_gr_aux
```

`principal_graph_aux<-,cell_data_set-method`
Method to set principal graph auxiliary information into CDS

Description

Method to set principal graph auxiliary information into CDS

Usage

```
## S4 replacement method for signature 'cell_data_set'
principal_graph_aux(x) <- value
```

Arguments

x	A cell_data_set object.
value	A S4Vectors::SimpleList of principal graph auxiliary information.

Value

x.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- align_cds(cds, alignment_group =
                 "batch", residual_model_formula_str = "~ bg.300.loading +
                  bg.400.loading + bg.500.1.loading + bg.500.2.loading +
                  bg.r17.loading + bg.b01.loading + bg.b02.loading")
cds <- reduce_dimension(cds)
```

```
ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
cds <- cluster_cells(cds)
cds <- learn_graph(cds)
pr_gr_aux <- principal_graph_aux(cds)
principal_graph_aux(cds) <- NULL
principal_graph_aux(cds) <- pr_gr_aux
```

`pseudotime` *Generic to extract pseudotime from CDS object*

Description

Generic to extract pseudotime from CDS object

Usage

```
pseudotime(x, reduction_method = "UMAP")
```

Arguments

`x`	A cell_data_set object.
`reduction_method`	Reduced dimension to extract pseudotime for.

Value

Pseudotime values.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                         'worm_embryo/worm_embryo_coldata.rds',
                         package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_expression_matrix.rds',
                                          package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds, num_dim=50)
cds <- align_cds(cds, alignment_group = "batch",
                 residual_model_formula_str = "~ bg.300.loading + bg.400.loading +
```

```
                bg.500.1.loading + bg.500.2.loading + bg.r17.loading + bg.b01.loading +
                  bg.b02.loading")
    cds <- reduce_dimension(cds)
    ciliated_genes <- c("che-1", "hlh-17", "nhr-6", "dmd-6", "ceh-36", "ham-1")
    cds <- cluster_cells(cds)
    cds <- learn_graph(cds)
    cds <- order_cells(cds,root_pr_nodes='Y_27')
    ps_tim <- pseudotime(cds)
```

`pseudotime,cell_data_set-method` *Method to extract pseudotime from CDS object*

Description

Method to extract pseudotime from CDS object

Usage

```
## S4 method for signature 'cell_data_set'
pseudotime(x, reduction_method = "UMAP")
```

Arguments

`x`	A cell_data_set object.
`reduction_method`	Reduced dimension to extract clusters for.

Value

Pseudotime values.

`reduce_dimension` *Compute a projection of a cell_data_set object into a lower dimensional space with non-linear dimension reduction methods*

Description

Monocle3 aims to learn how cells transition through a biological program of gene expression changes in an experiment. Each cell can be viewed as a point in a high-dimensional space, where each dimension describes the expression of a different gene. Identifying the program of gene expression changes is equivalent to learning a *trajectory* that the cells follow through this space. However, the more dimensions there are in the analysis, the harder the trajectory is to learn. Fortunately, many genes typically co-vary with one another, and so the dimensionality of the data can be reduced with a wide variety of different algorithms. Monocle3 provides two different algorithms for dimensionality reduction via `reduce_dimension` (UMAP and tSNE). The function `reduce_dimension` is the second step in the trajectory building process after `preprocess_cds`.

UMAP is implemented from the package uwot.

Usage

```
reduce_dimension(
  cds,
  max_components = 2,
  reduction_method = c("UMAP", "tSNE", "PCA", "LSI", "Aligned"),
  preprocess_method = NULL,
  umap.metric = "cosine",
  umap.min_dist = 0.1,
  umap.n_neighbors = 15L,
  umap.fast_sgd = FALSE,
  umap.nn_method = "annoy",
  verbose = FALSE,
  cores = 1,
  build_nn_index = FALSE,
  nn_control = list(),
  ...
)
```

Arguments

`cds`	the cell_data_set upon which to perform this operation.
`max_components`	the dimensionality of the reduced space. Default is 2.
`reduction_method`	A character string specifying the algorithm to use for dimensionality reduction. Currently "UMAP", "tSNE", "PCA", "LSI", and "Aligned" are supported.
`preprocess_method`	A string indicating the preprocessing method used on the data. Options are "PCA" and "LSI". Default is "LSI".
`umap.metric`	A string indicating the distance metric to be used when calculating UMAP. Default is "cosine". See uwot package's `umap` for details.
`umap.min_dist`	Numeric indicating the minimum distance to be passed to UMAP function. Default is 0.1.See uwot package's `umap` for details.

umap.n_neighbors
: Integer indicating the number of neighbors to use during kNN graph construction. Default is 15L. See uwot package's umap for details.

umap.fast_sgd
: Logical indicating whether to use fast SGD. Default is TRUE. See uwot package's umap for details.

umap.nn_method
: String indicating the nearest neighbor method to be used by UMAP. Default is "annoy". See uwot package's umap for details.

verbose
: Logical, whether to emit verbose output.

cores
: Number of cores to use for computing the UMAP.

build_nn_index
: logical When this argument is set to TRUE, preprocess_cds builds the nearest neighbor index from the reduced dimension matrix for later use. Default is FALSE.

nn_control
: An optional list of parameters used to make the nearest neighbor index. See the set_nn_control help for detailed information. The default metric is cosine for reduction_methods PCA, LSI, and Aligned, and is euclidean for reduction_methods tSNE and UMAP. Note: distances in tSNE space reflect spatial differences poorly so using nearest neighbors with it may be meaningless.

...
: additional arguments to pass to the dimensionality reduction function.

Value

an updated cell_data_set object

References

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

tSNE: Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. J. Mach. Learn. Res., 9(Nov):2579– 2605, 2008.

Examples

```
cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))
cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
```

`reduce_dimension_transform` *Apply a reduce_dimension transform model to a cell_data_set.*

Description

Applies a previously calculated reduce_dimension transform model to a new preprocess transformed matrix. For more information read the help information for save_transform_models.

Usage

```
reduce_dimension_transform(
  cds,
  preprocess_method = NULL,
  reduction_method = c("UMAP")
)
```

Arguments

`cds`	a cell_data_set to be transformed.
`preprocess_method`	the reduced dimension matrix to be transformed using the reduction_method transform model. The default is NULL, which uses the preprocess_method that was used when the reduce_dimension model was built.
`reduction_method`	a previously loaded reduce_dimension transform model that is used to reduce the dimensions of the preprocessed matrix in the cell_data_set. Only "UMAP" is supported.

Value

a cell_data_set with a transformed reduced count matrix.

Examples

```
## Not run:
  cell_metadata <- readRDS(system.file('extdata',
                                       'worm_embryo/worm_embryo_coldata.rds',
                                       package='monocle3'))
  gene_metadata <- readRDS(system.file('extdata',
                                       'worm_embryo/worm_embryo_rowdata.rds',
                                       package='monocle3'))
  expression_matrix <- readRDS(system.file('extdata',
                                      'worm_embryo/worm_embryo_expression_matrix.rds',
                                           package='monocle3'))
  cds <- new_cell_data_set(expression_data=expression_matrix,
                           cell_metadata=cell_metadata,
                           gene_metadata=gene_metadata)
```

```
    ncell <- nrow(colData(cds))
    cell_sample <- sample(seq(ncell), 2 * ncell / 3)
    cell_set <- seq(ncell) %in% cell_sample
    cds1 <- cds[,cell_set]
    cds1 <- preprocess_cds(cds1)
    cds1 <- reduce_dimension(cds1)
    save_transform_models(cds1, 'tm')
    cds2 <- cds[,!cell_set]
    cds2 <- load_transform_models(cds2, 'tm')
    cds2 <- preprocess_transform(cds2, 'PCA')
    cds2 <- reduce_dimension_transform(cds2)

## End(Not run)
```

`repmat` *function to reproduce the behavior of repmat function in matlab to replicate and tile an matrix*

Description

function to reproduce the behavior of repmat function in matlab to replicate and tile an matrix

Usage

```
repmat(X, m, n)
```

Arguments

`X`	matrix for tiling and replicate the data
`m`	a numeric value for tiling a matrix
`n`	a numeric value for tiling a matrix

Value

a matrix

`save_monocle_objects` *Save a Monocle3 full cell_data_set.*

Description

Save a Monocle3 full cell_data_set to a specified directory by writing the R objects to RDS files and the nearest neighbor indexes to index files. The assays objects are saved as HDF5Array files when hdf5_assays=TRUE or when the cell_data_set assays are HDF5Array objects. If any assay in the cell_data set is an HDF5 object, all assays must be. When save_monocle_objects is run with hdf5_assays=TRUE, the load_monocle_objects function loads the saved assays into HDF5Array objects in the resulting cell_data_set. Note: operations such as preprocess_cds that are run on assays stored as HDF5Arrays are much, much slower than the same operations run on assays stored as in-memory matrices. You may want to investigate parameters related to the Bioconductor DelayedArray and BiocParallel packages in this case.

Usage

```
save_monocle_objects(
  cds,
  directory_path,
  hdf5_assays = FALSE,
  comment = "",
  verbose = TRUE
)
```

Arguments

`cds`	a cell_data_set to save.
`directory_path`	a string giving the name of the directory in which to write the object files.
`hdf5_assays`	a boolean determining whether the non-HDF5Array assay objects are saved as HDF5 files. At this time cell_data_set HDF5Array assay objects are stored as HDF5Assay files regardless of the hdf5_assays parameter value.
`comment`	a string with optional notes that is saved with the objects.
`verbose`	a boolean determining whether to print information about the saved files.

Value

none.

Examples

```
## Not run:
  cds <- load_a549()
  save_monocle_objects(cds, 'mo')

## End(Not run)
```

save_transform_models *Save cell_data_set transform models.*

Description

Save the transform models in the cell_data_set to the specified directory by writing the R objects to RDS files and the nearest neighbor indexes to index files. save_transform_models saves transform models made by running the preprocess_cds and reduce_dimension functions on an initial cell_data_set. Subsequent cell_data_sets are transformed into the reduced dimension space of the initial cell_data_set by loading the new data into a new cell_data_set, loading the initial data set transform models into the new cell_data_set using the load_transform_models function, and applying those transform models to the new data set using the preprocess_transform and reduce_dimension_transform functions. In this case, do not run the preprocess_cds or reduce_dimension functions on the new cell_data_set. Additionally, save_transform_models saves nearest neighbor indexes when the preprocess_cds and reduce_dimension functions are run with the make_nn_index=TRUE parameter. These indexes are used to find matches between cells in the new processed cell_data_set and the initial cell_data_set using index search functions. For more information see the help for transfer_cell_labels. save_transform_models saves the models to a directory given by directory_path.

Usage

```
save_transform_models(cds, directory_path, comment = "", verbose = TRUE)
```

Arguments

cds	a cell_data_set with existing models.
directory_path	a string giving the name of the directory in which to write the model files.
comment	a string with optional notes that is saved with the objects.
verbose	a boolean determining whether to print information about the saved files.

Value

none.

Examples

```
  ## Not run:
    cds <- load_a549()
    cds <- preprocess_cds(cds)
    cds <- reduce_dimension(cds)
    save_transform_models(cds, 'tm')

## End(Not run)
```

`search_cds_nn_index` *Search a nearest neighbor index that is stored in the cds.*

Description

Search a nearest neighbor index for cells near those in the query_matrix.

Usage

```
search_cds_nn_index(
  query_matrix,
  cds,
  reduction_method = c("UMAP", "PCA", "LSI", "Aligned", "tSNE"),
  k = 25,
  nn_control = list(),
  verbose = FALSE
)
```

Arguments

`query_matrix`	a reduced dimension matrix used to find the nearest neighbors in the index nn_index.
`cds`	a cell_data_set in which the nearest neighbor index is stored.
`reduction_method`	a string giving the reduced dimension matrix used to make the nearest neighbor index, and determines where the index is stored in the cell_data_set. Note: distances in tSNE space reflect spatial differences poorly so using nearest neighbors with it may be meaningless.
`k`	an integer for the number of nearest neighbors to return for each cell. Default is 25.
`nn_control`	a list of parameters used to make and search the nearest neighbors indexes. See the set_nn_control help for additional details. Note that if nn_control[['search_k']] is not defined, transfer_cell_labels will try to use search_k <- 2 * n_trees * k where n_trees is the value used to build the index. The default metric is cosine for reduction_methods PCA, LSI, and Aligned, and is euclidean for reduction_methods tSNE and UMAP.
`verbose`	a boolean indicating whether to emit verbose output.

Value

a list list(nn.idx, nn.dists) where nn.idx is a matrix of nearest neighbor indices and nn.dists is a matrix of the distance between the index given by the row number and the index given in nn.idx. If the same reduced dim matrix is used to make the index and search the index, the index given by the row number should be in the row, usually in the first column.

Examples

```
  cds <- load_a549()
  cds <- preprocess_cds(cds)
  cds <- make_cds_nn_index(cds, 'PCA')
nn_res <- search_cds_nn_index(SingleCellExperiment::reducedDims(cds)[['PCA']], cds, 'PCA', 10)
```

`search_nn_index` *Search a nearest neighbor index.*

Description

Search a nearest neighbor index for cells near those in the query_matrix.

Usage

```
search_nn_index(
  query_matrix,
  nn_index,
  k = 25,
  nn_control = list(),
  verbose = FALSE
)
```

Arguments

`query_matrix`	a reduced dimension matrix used to find the nearest neighbors in the index nn_index.
`nn_index`	a nearest_neighbor index.
`k`	an integer for the number of nearest neighbors to return for each cell. Default is 25.
`nn_control`	a list of parameters used to search the nearest neighbor index. See the set_nn_control help for details. Note: the default annoy search_k parameter value is set to the default value of 2 * n_trees * k. It does not know the value of n_trees that was used to build the annoy index so if a non-default n_trees value was used to build the index, you may need to set search_k in nn_control list when you run search_nn_index.
`verbose`	a boolean indicating whether to emit verbose output.

Value

a list list(nn.idx, nn.dists) where nn.idx is a matrix of nearest neighbor indices and nn.dists is a matrix of the distance between the index given by the row number and the index given in nn.idx. If the same reduced dim matrix is used to make the index and search the index, the index given by the row number should be in the row, usually in the first column.

Examples

```
  cds <- load_a549()
  cds <- preprocess_cds(cds)
  nn_index <- make_nn_index(SingleCellExperiment::reducedDims(cds)[['PCA']])
nn_res <- search_nn_index(SingleCellExperiment::reducedDims(cds)[['PCA']], nn_index, 10)
```

`search_nn_matrix` *Search a subject matrix for nearest neighbors to a query_matrix.*

Description

Make a nearest neighbors index using the subject matrix and search it for nearest neighbors to the query_matrix.

Usage

```
search_nn_matrix(
  subject_matrix,
  query_matrix,
  k = 25,
  nn_control = list(),
  verbose = FALSE
)
```

Arguments

`subject_matrix`	a matrix used to build a nearest neighbor index.
`query_matrix`	a matrix used to search the subject_matrix nearest neighbor index.
`k`	an integer for the number of nearest neighbors to return for each cell. Default is 25.
`nn_control`	a list of parameters used to make and search the nearest neighbor index. See the set_nn_control help for details.
`verbose`	a boolean indicating whether to emit verbose output.

Value

a list list(nn.idx, nn.dists) where nn.idx is a matrix of nearest neighbor indices and nn.dists is a matrix of the distance between the index given by the row number and the index given in nn.idx. If the query_matrix is the same as the subject matrix, the index given by the row number should be in the row, usually in the first column.

set_cds_nn_index *Set a nearest neighor index in the cell_data_set.*

Description

Store the given nearest neighbor index in the cell_data_set. The reduction_method parameter tells set_cds_nn_index where in the cell_data_set to store the index.

Usage

```
set_cds_nn_index(
  cds,
  reduction_method = c("UMAP", "PCA", "LSI", "Aligned", "tSNE"),
  nn_index,
  verbose = FALSE
)
```

Arguments

cds	a cell_data_set in which to store the nearest neighbor index.
reduction_method	a string giving the reduced dimension matrix used to make the nn_index nearest neighbor index, and determines where the index is stored in the cell_data_set.
nn_index	a nearest neighbor index to store in cds.
verbose	a boolean indicating whether to emit verbose output.

Value

a cell_data_set with the stored index.

set_nn_control *Verify and set nearest neighbor parameter list.*

Description

Verifies the listed parameter values that will be passed to the nearest neighbor function given by nn_control[['method']]. Unspecified values are set to default values. To see the default values, call the function with nn_control=list(show_values=TRUE).

Usage

```
set_nn_control(
  mode,
  nn_control = list(),
  nn_control_default = list(),
  nn_index = NULL,
  k = NULL,
  verbose = FALSE
)
```

Arguments

mode	the nearest neighbor operation for which the nn_control list will be used. 1=make index, 2=search index, and 3=both make and search index. Required parameter.
nn_control	an optional list of parameters passed to the nearest neighbor function specified by nn_control[['method']]. If a value is not given in nn_control, the value in nn_control_default is used. If neither is given, a fallback default is assigned.
nn_control_default	an optional nn_control list to use when a parameter is not given in nn_control.
nn_index	an nn_index. This may be used to look up parameters that were used to make an index. For example, the default search_k parameter depends on the n_trees values used to make the index. The default is NULL.
k	integer the number of desired nearest neighbor points to return from a search. k is used to • set the annoy search_k parameter when search_k is not given in nn_control or nn_control_default where search_k is 2 * n_trees * k. • test the hnsw ef parameter, which must be at least as large as k. k is ignored for index builds and does not give the number of nearest neighbors to return for a search.
verbose	a boolean indicating whether to emit verbose output.

Value

an updated nn_control list.

Optional nn_control parameters

method The method used to find nearest neighbor points. The available methods are 'nn2', 'annoy', and 'hnsw'. Detailed information about each method can be found on the WWW sites: https://cran.r-project.org/web/packages/RANN/, https://cran.r-project.org/web/packages/RcppAnnoy/index.html, and https://cran.rstudio.com/web/packages/RcppHNSW/index.html.

metric The distance metric used by the nearest neighbor functions. Annoy accepts 'euclidean', 'cosine', 'manhattan', and 'hamming'. HNSW accepts 'euclidean', 'l2', 'cosine', and 'ip'. RANN uses 'euclidean'.

n_trees The annoy index build parameter that affects the build time and index size. Larger values give more accurate results, longer build times, and larger indexes.

search_k The annoy index search parameter that affects the search accuracy and time. Larger values give more accurate results and longer search times. Default is 2 * n_trees * k. In order to set search_k, the following conditions are tested and the first TRUE condition is used: nn_control[['search_k']] exists; mode=2 and nn_index and k are not NULL; nn_control[['n_trees']] exists; nn_control_default[['search_k']] exists; nn_control_default[['n_trees']] exists. If none of those is TRUE, the fallback default n_trees value is used. If the set_nn_control k parameter value is not NULL, it is used; otherwise, the default is used.

M The HNSW index build parameter that affects the search accuracy and memory requirements. Larger values give more accurate search results and increase the index memory use.

ef_construction The HNSW index build parameter that affects the search accuracy and index build time. Larger values give more accurate search results and longer build times. Default is 200.

ef The HNSW index search parameter that affects the search accuracy and search time. Larger values give more accurate results and longer search times. ef must be greater than or equal to k.

grain_size The HNSW parameter that gives the minimum amount of work to do per thread.

cores The annoy and HNSW parameter that gives the number of threads to use for the annoy index search and for the HNSW index build and search.

show_values A logical value used to show the nearest neighbor parameters to use, and then exit the function. When show_values=TRUE is the only nn_control value, the parameters are the defaults for the function. Each function that calls set_nn_control may have its own nn_control_default list.

`size_factors` *Get the size factors from a cds object.*

Description

A wrapper around `colData(cds)$Size_Factor`

Usage

```
size_factors(cds)
```

Arguments

`cds`	A cell_data_set object.

Value

An updated cell_data_set object

Examples

```
cds <- load_a549()
size_factors(cds)
```

size_factors<- *Set the size factor values in the cell_data_set*

Description

Set the size factor values in the cell_data_set

Usage

```
size_factors(cds) <- value
```

Arguments

cds	A cell_data_set object.
value	the size factor values.

Value

An updated cell_data_set object

soft_assignment *Function to calculate the third term in the objective function*

Description

Function to calculate the third term in the objective function

Usage

```
soft_assignment(X, C, sigma)
```

Arguments

X	input data
C	center of graph (D * K)
sigma	bandwidth parameter

Value

a matrix with diagonal element as 1 while other elements as zero (eye matrix)

`sparse_prcomp_irlba` *Principal Components Analysis*

Description

Efficient computation of a truncated principal components analysis of a given data matrix using an implicitly restarted Lanczos method from the `irlba` package.

Usage

```
sparse_prcomp_irlba(x, n = 3, retx = TRUE, center = TRUE, scale. = FALSE, ...)
```

Arguments

`x`	a numeric or complex matrix (or data frame) which provides the data for the principal components analysis.
`n`	integer number of principal component vectors to return, must be less than `min(dim(x))`.
`retx`	a logical value indicating whether the rotated variables should be returned.
`center`	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a centering vector of length equal the number of columns of `x` can be supplied.
`scale.`	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with S, but scaling is often advisable. Alternatively, a vector of length equal the number of columns of `x` can be supplied. The value of `scale` determines how column scaling is performed (after centering). If `scale` is a numeric vector with length equal to the number of columns of `x`, then each column of `x` is divided by the corresponding value from `scale`. If `scale` is `TRUE` then scaling is done by dividing the (centered) columns of `x` by their standard deviations if `center=TRUE`, and the root mean square otherwise. If `scale` is `FALSE`, no scaling is done. See `scale` for more details.
`...`	additional arguments passed to `irlba`.

Value

A list with class "prcomp" containing the following components:

- sdev the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix).
- rotation the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors).
- x if `retx` is `TRUE` the value of the rotated data (the centered (and scaled if requested) data multiplied by the `rotation` matrix) is returned. Hence, `cov(x)` is the diagonal matrix `diag(sdev^2)`.
- center, scale the centering and scaling used, or `FALSE`.

Note

The signs of the columns of the rotation matrix are arbitrary, and so may differ between different programs for PCA, and even between different builds of R.

NOTE DIFFERENCES WITH THE DEFAULT prcomp FUNCTION! The tol truncation argument found in prcomp is not supported. In place of the truncation tolerance in the original function, the prcomp_irlba function has the argument n explicitly giving the number of principal components to return. A warning is generated if the argument tol is used, which is interpreted differently between the two functions.

See Also

prcomp

Examples

```
## Not run:
  set.seed(1)
  x  <- matrix(rnorm(200), nrow=20)
  p1 <- irlba::prcomp_irlba(x, n=3)
  summary(p1)

  # Compare with
  p2 <- prcomp(x, tol=0.7)
  summary(p2)
## End(Not run)
```

---

top_markers *Identify the genes most specifically expressed in groups of cells*

Description

Identify the genes most specifically expressed in groups of cells

Usage

```
top_markers(
  cds,
  group_cells_by = "cluster",
  genes_to_test_per_group = 25,
  reduction_method = "UMAP",
  marker_sig_test = TRUE,
  reference_cells = NULL,
  speedglm.maxiter = 25,
  cores = 1,
  verbose = FALSE
)
```

Arguments

cds	A cell_data_set object to calculate top markers for.
group_cells_by	String indicating what to group cells by for comparison. Default is "cluster".
genes_to_test_per_group	Numeric, how many genes of the top ranked specific genes by Jenson-Shannon to do the more expensive regression test on.
reduction_method	String indicating the method used for dimensionality reduction. Currently only "UMAP" is supported.
marker_sig_test	A flag indicating whether to assess the discriminative power of each marker through logistic regression. Can be slow, consider disabling to speed up top_markers().
reference_cells	If provided, top_markers will perform the marker significance test against a "reference set" of cells. Must be either a list of cell ids from colnames(cds), or a positive integer. If the latter, top_markers() will randomly select the specified number of reference cells. Accelerates the marker significance test at some cost in sensitivity.
speedglm.maxiter	Maximum number of iterations allowed for fitting GLM models when testing markers for cell group.
cores	Number of cores to use.
verbose	Whether to print verbose progress output.

Value

a data.frame where the rows are genes and the columns are

- gene_id vector of gene names
- gene_short_name vector of gene short names
- cell_group character vector of the cell group to which the cell belongs
- marker_score numeric vector of marker scores as the fraction expressing scaled by the specificity. The value ranges from 0 to 1.
- mean_expression numeric vector of mean normalized expression of the gene in the cell group
- fraction_expressing numeric vector of fraction of cells expressing the gene within the cell group
- specificity numeric vector of a measure of how specific the gene's expression is to the cell group based on the Jensen-Shannon divergence. The value ranges from 0 to 1.
- pseudo_R2 numeric vector of pseudo R-squared values, a measure of how well the gene expression model fits the categorical data relative to the null model. The value ranges from 0 to 1.
- marker_test_p_value numeric vector of likelihood ratio p-values
- marker_test_q_value numeric vector of likelihood ratio q-values

Examples

```
library(dplyr)

cell_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_coldata.rds',
                                     package='monocle3'))
gene_metadata <- readRDS(system.file('extdata',
                                     'worm_embryo/worm_embryo_rowdata.rds',
                                     package='monocle3'))
expression_matrix <- readRDS(system.file('extdata',
                                    'worm_embryo/worm_embryo_expression_matrix.rds',
                                         package='monocle3'))

cds <- new_cell_data_set(expression_data=expression_matrix,
                         cell_metadata=cell_metadata,
                         gene_metadata=gene_metadata)

cds <- preprocess_cds(cds)
cds <- reduce_dimension(cds)
cds <- cluster_cells(cds)
marker_test_res <- top_markers(cds, group_cells_by="partition", reference_cells=1000)
top_specific_markers <- marker_test_res %>%
                         filter(fraction_expressing >= 0.10) %>%
                         group_by(cell_group) %>%
                         top_n(1, pseudo_R2)
top_specific_marker_ids <- unique(top_specific_markers %>% pull(gene_id))
```

---

transfer_cell_labels *Transfer cell column data from a reference to a query cell_data_set.*

Description

For each cell in a query cell_data_set, transfer_cell_labels finds sufficiently similar cell data in a reference cell_data_set and copies the value in the specified column to the query cell_data_set.

Usage

```
transfer_cell_labels(
  cds_query,
  reduction_method = c("UMAP", "PCA", "LSI"),
  ref_coldata,
  ref_column_name,
  query_column_name = ref_column_name,
  transform_models_dir = NULL,
  k = 10,
  nn_control = list(),
  top_frac_threshold = 0.5,
```

```
  top_next_ratio_threshold = 1.5,
  verbose = FALSE
)
```

Arguments

cds_query	the cell_data_set upon which to perform this operation
reduction_method	a string specifying the reduced dimension matrix to use for the label transfer. These are "PCA", "LSI", and "UMAP". Default is "UMAP".
ref_coldata	the reference cell_data_set colData data frame, which is obtained using the colData(cds_ref) function.
ref_column_name	a string giving the name of the reference cell_data_set column with the values to copy to the query cell_data_set.
query_column_name	a string giving the name of the query cell_data_set column to which you want the values copied. The default is ref_column_name.
transform_models_dir	a string giving the name of the transform model directory to load into the query cell_data_set. If it is NULL, use the transform models in the query cell_data_set, which requires that the reference transform models were loaded into the query cell_data_set before transfer_cell_labels is called. The default is NULL. transfer_cells_labels uses the nearest neighbor index, which must be stored in the transform model.
k	an integer giving the number of reference nearest neighbors to find. This value must be large enough to find meaningful column value fractions. See the top_frac_threshold parameter below for additional information. The default is 10.
nn_control	An optional list of parameters used to make and search the nearest neighbors indices. See the set_nn_control help for additional details. Note that if nn_control[['search_k']] is not defined, transfer_cell_labels will try to use search_k <- 2 * n_trees * k where n_trees is the value used to build the index. The default metric is cosine for reduction_methods PCA and LSI and is euclidean for reduction_method UMAP.
top_frac_threshold	a numeric value. The top fraction of reference values must be greater than top_frac_threshold in order to be transferred to the query. The top fraction is the fraction of the k neighbors with the most frequent value. The default is 0.5.
top_next_ratio_threshold	a numeric value giving the minimum value of the ratio of the counts of the most frequent to the second most frequent reference values required for transferring the reference value to the query. The default is 1.5.
verbose	a boolean controlling verbose output.

Details

transfer_cell_labels requires a nearest neighbor index made from a reference reduced dimension matrix, the reference cell data to transfer, and a query cell_data_set. The index can be made

from UMAP coordinates using the build_nn_index=TRUE option in the reduce_dimensions(..., build_nn_index=TRUE) function, for example. The query cell_data_set must have been processed with the preprocess_transform and reduce_dimension_transform functions using the models created when the reference cell_data_set was processed, rather than with preprocess_cds and reduce_dimension.

The models are made when the reference cell_data_set is processed and must be saved to disk at that time using save_transform_models. The load_transform_models function loads the models into the query cell_data_set where they can be used by preprocess_transform and reduce_dimension_transform. The cells in the reference and query cell_data_sets must be similar in the sense that they map to similar reduced dimension coordinates.

When the ref_column_name values are discrete, the sufficiently most frequent value is transferred. When the values are continuous the mean of the k nearest neighbors is transferred.

In the case of discrete values, transfer_cell_labels processes each query cell as follows. It finds the k nearest neighbor cells in the reference set, and if more than top_frac_threshold fraction of them have the same value, it copies that value to the query_column_name column in the query cell_data_set. If the fraction is at or below top_frac_threshold, it checks whether the ratio of the most frequent to the second most frequent value is at least top_next_ratio_threshold, in which case it copies the value; otherwise, it sets it to NA.

Notes:

- Monocle3 does not have an align_transform function to apply align_cds-related transforms at this time. If your data sets require batch correction, you need to co-embed them.
- transfer_cell_labels does not check that the reference nearest neighbor index is consistent with the query matrix.

Value

an updated cell_data_set object

Examples

```
## Not run:
   expression_matrix <- readRDS(system.file('extdata',
                                            'worm_l2/worm_l2_expression_matrix.rds',
                                            package='monocle3'))
   cell_metadata <- readRDS(system.file('extdata',
                            'worm_l2/worm_l2_coldata.rds',
                             package='monocle3'))
   gene_metadata <- readRDS(system.file('extdata',
                            'worm_l2/worm_l2_rowdata.rds',
                            package='monocle3'))

   cds <- new_cell_data_set(expression_data=expression_matrix,
                            cell_metadata=cell_metadata,
                            gene_metadata=gene_metadata)
  ncell <- nrow(colData(cds))
  cell_sample <- sample(seq(ncell), 2 * ncell / 3)
  cell_set <- seq(ncell) %in% cell_sample
  cds1 <- cds[,cell_set]
```

```
  cds1 <- preprocess_cds(cds1)
  cds1 <- reduce_dimension(cds1, build_nn_index=TRUE)
  save_transform_models(cds1, 'tm')

  cds2 <- cds[,!cell_set]
  cds2 <- load_transform_models(cds2, 'tm')
  cds2 <- preprocess_transform(cds2, 'PCA')
  cds2 <- reduce_dimension_transform(cds2)
cds2 <- transfer_cell_labels(cds2, 'UMAP', colData(cds1), 'cao_cell_type', 'transfer_cell_type')

## End(Not run)
```

Index