Package 'DecoFlex'

June 13, 2023

Type Package

Title DecoFlex is a deconvolution method for multi-omics data

Version 0.1.0

Maintainer Crhistian Cardona <crhisto@gmail>

Description DecoFlex stands as an innovative deconvolution solution designed for multi-omics data. Leveraging the power of the NMMFlex Python library, DecoFlex applies the Non-Negative Multiple Matrix Factorization (NMMF) technique to multiple inputs. This methodology allows it to proficiently handle intricate omics data, making it an invaluable tool in the world of bioinformatics and computational biology.

URL https://github.com/crhisto/DecoFlex

BugReports https://github.com/crhisto/DecoFlex/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Remotes crhisto/Biobase,
crhisto/xbioc

Depends R (>= 3.5),

Imports Biobase,
stats,
graphics (>= 3.6.3),
Matrix (>= 1.5-4),
reticulate (>= 1.28),
Seurat (>= 4.3.0),
RColorBrewer (>= 1.1-3),
gplots (>= 3.1.3),
ggplot2 (>= 3.4.2),
grDevices (>= 3.6.3),
plyr (>= 1.8.8)

Suggests knitr,
rmarkdown,
roxygen2 (>= 7.2.3),
testthat (>= 3.0.0)

SystemRequirements Python library NMMFlex >= 0.1.0 (https://github.com/crhisto/NMMFlex)

RoxygenNote 7.2.3

Config/testthat/edition 3

VignetteBuilder knitr

R topics documented:

`add_clustering_single_cell_dataset` *Add Clustering Information to Single Cell Dataset*

Description

This function enhances a single cell dataset by adding metacluster information to it. It achieves this by iterating through each unique value in the hierarchical fit, identifying the celltypes associated with that value, and tagging the corresponding cells in the single cell dataset with a new metacluster identifier.

Usage

```
add_clustering_single_cell_dataset(
  sc.eset,
  var_cluster_name = "cluster_normalized",
  fit_hierarchical
)
```

Arguments

`sc.eset`	A single cell ExpressionSet object. This dataset is modified in-place to include metacluster identifiers.
`var_cluster_name`	A string specifying the column name in sc.eset where the cluster identifiers are stored. Default is 'cluster_normalized'.
`fit_hierarchical`	A named vector where the names correspond to celltypes and the values represent the hierarchical clustering fit for each celltype.

Value

An ExpressionSet object (sc.eset) with additional metacluster information.

add_grouping_single_cell

Add Grouping to Single Cell Data

Description

The add_grouping_single_cell function iterates over a list of cell types and assigns labels to each single cell in a dataset based on its group at each level. The labels are in the format 'subcluster_level_#*leaf*#'.

Usage

```
add_grouping_single_cell(
  single_cell_data_exp,
  sub_clusters_var,
  level_group_name,
  hierarchy_level_groups
)
```

Arguments

single_cell_data_exp	A list containing the single cell data. The function will add new labels to this list.
sub_clusters_var	A string representing the variable used to indicate the subclusters in single_cell_data_exp.
level_group_name	A string representing the name of the level group variable in single_cell_data_exp. The function will add labels to this variable.
hierarchy_level_groups	A list of cell types that define the hierarchy levels. Each cell type should be a list itself with the elements celltype_list, tree_level, and leaf_number.

Value

The function returns the single_cell_data_exp list with added labels to the level_group_name variable.

aggregate_true_top_clustering_prop

Aggregate True Proportions for Top-Level Clustering

Description

aggregate_true_top_clustering_prop function calculates the true proportions of top-level clustersbased on the true subclustering proportions.

Usage

```
aggregate_true_top_clustering_prop(
  single_cell_data_exp,
  true_prop,
  top_clusters_var,
  top_clusters_list,
  sub_clusters_var
)
```

Arguments

`single_cell_data_exp`	A data frame or matrix containing the single cell data.
`true_prop`	A matrix or data frame containing the true proportions of the subclusters.
`top_clusters_var`	The variable name of the top clusters in the `single_cell_data_exp`.
`top_clusters_list`	A list containing the identifiers of the top clusters.
`sub_clusters_var`	The variable name of the subclusters in the `single_cell_data_exp`.

Details

The function iterates over the top-level clusters in `top_clusters_list`. For each top-level cluster, it identifies the corresponding subclusters and calculates the sum of their true proportions. The process continues until the aggregated true proportions for all top-level clusters have been calculated.

Value

A matrix that contains the aggregated true proportions of the top clusters.

`calculate_best_clustering_generic` *Calculate Optimal Clustering at Generic Level*

Description

This function recursively analyses a level of clustering by attempting to separate it into 2, 3, or 4 clusters. The aim is to minimize the correlation between the groups. For each subgroup, the function is called again until each cluster contains 2 or 3 cell types. This enables granular and optimized clustering across different levels of hierarchy.

Usage

```
calculate_best_clustering_generic(
  single_cell_data_exp,
  reference,
  var_cell_type,
  cell_types,
  var_sample,
```

```
  tree_level = 0,
  leaf_number = 1,
  number_clusters_one_celltype = 1,
  distance_method = "euclidean",
  hclust_method = "average",
  min_size_leaf = 3,
  max_k_tried_hier_clustering = 3,
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame or matrix containing single-cell expression data.
reference	A reference data set or object to guide the clustering process.
var_cell_type	A string or variable indicating the cell type for each cell.
cell_types	A vector of cell types present in the data.
var_sample	A string or variable indicating the sample for each cell.
tree_level	An integer indicating the current level of the tree hierarchy in the clustering process. Default is 0.
leaf_number	An integer indicating the number of leaf nodes in the tree hierarchy. Default is 1.
number_clusters_one_celltype	An integer indicating the number of clusters per cell type. Default is 1.
distance_method	A string indicating the distance measure to be used for clustering. Default is 'euclidean'.
hclust_method	A string indicating the method to be used for hierarchical clustering. Default is 'average'.
min_size_leaf	An integer indicating the minimum size of the leaf nodes. Default is 3.
max_k_tried_hier_clustering	An integer indicating the maximum number of clusters tried in the hierarchical clustering. Default is 3.
verbose	A logical value indicating whether to print additional output during the computation. Default is FALSE.

Value

A list containing the optimal clustering results.

---

calculate_best_cluster_global

Calculate the Best Global Cluster

Description

Creates the tree of cluster groups and levels. Decides the number of clusters that will minimize the correlation between them, given a list of cell types.

Usage

```
calculate_best_cluster_global(
  single_cell_data_exp,
  var_cell_type,
  subset_celltypes = NULL,
  var_sample,
  use_min_cor_strategy = TRUE,
  delete_shared_level_markers = FALSE,
  delete_shared_internal_markers = FALSE,
  number_clusters_one_celltype = 1,
  distance_method = "euclidean",
  hclust_method = "average",
  min_size_leaf = 3,
  max_k_tried_hier_clustering = 3,
  random_seed = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  filter_main_markers = TRUE,
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame or matrix containing single-cell expression data.
var_cell_type	A vector or factor indicating the cell type for each cell.
subset_celltypes	A vector of cell types to subset for the analysis. Default is NULL, implying all cell types are used.
var_sample	A vector or factor indicating the sample source for each cell.
use_min_cor_strategy	A logical value indicating whether to use the minimum correlation strategy. Default is TRUE.
delete_shared_level_markers	A logical value indicating whether to delete markers shared at level of the hierarchy. Default is FALSE.
delete_shared_internal_markers	A logical value indicating whether to delete markers shared internally. Default is FALSE.
number_clusters_one_celltype	An integer specifying the number of clusters for each cell type. Default is 1.
distance_method	A string specifying the method to compute the distance. Default is 'euclidean'.
hclust_method	A string specifying the method to perform hierarchical clustering. Default is 'average'.
min_size_leaf	An integer specifying the minimum size of the leaf in the hierarchical clustering. Default is 3.
max_k_tried_hier_clustering	An integer specifying the maximum number of clusters tried in hierarchical clustering. Default is 3.

`random_seed` An integer specifying the random seed for reproducibility. Default is NULL.

`param.logfc.threshold`
A numeric value indicating the log-fold-change threshold for marker selection. Default is 2.0.

`param.p_val_adj`
A numeric value indicating the adjusted p-value threshold for marker selection. Default is 0.05.

`filter_main_markers`
A logical value indicating whether to filter main markers. Default is TRUE.

`verbose` A logical value indicating whether to print additional output during the computation. Default is FALSE.

Value

A list containing results of the clustering process.

`calculate_joint_cor` *Calculate Joint Correlation*

Description

This function computes the correlation among distinct groups within the reference data based on a set of marker genes. The function then calculates the average joint correlation, which reflects the average correlation coefficient across all possible pairs of different groups.

Usage

```
calculate_joint_cor(reference, markers)
```

Arguments

`reference` A matrix or dataframe of reference data, with each column representing a sample and each row corresponding to a gene.

`markers` A vector of marker genes, which will be used to subset the reference dataset for the correlation computation.

@import stats

Details

The function works by initially calculating the correlation matrix of the reference data subsetted by the specified marker genes. It then computes the joint correlation by summing all correlation coefficients, excluding the self-correlations (i.e., correlation of a group with itself). This sum is divided by the number of unique pair combinations of different groups, yielding the average joint correlation. The result is a single numeric value representing the average degree of correlation between the different groups in the reference dataset based on the specified marker genes.

Value

A single numeric value representing the average joint correlation.

calculate_k_clustering *Optimal k Clustering for Hierarchical Clustering*

Description

This function calculates the optimal number (k) of clusters for a hierarchical clustering given a specific set of cell types. It can be customized with different distance measures and agglomeration methods.

Usage

```
calculate_k_clustering(
  single_cell_data_exp,
  reference,
  var_cell_type,
  cell_types,
  k_value,
  var_sample,
  distance_method = "euclidean",
  hclust_method = "average",
  extra_title = NULL,
  verbose = TRUE
)
```

Arguments

single_cell_data_exp	A data frame or matrix containing single-cell expression data.
reference	A reference data set or object to guide the clustering process.
var_cell_type	A string or variable indicating the cell type for each cell.
cell_types	A vector of cell types present in the data.
k_value	An integer indicating the number of clusters to be created.
var_sample	A string or variable indicating the sample for each cell.
distance_method	A string indicating the distance measure to be used for clustering. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given. Default is 'euclidean'.
hclust_method	A string indicating the method to be used for hierarchical clustering. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). Default is 'average'.
extra_title	A string used for the additional title in the plotting.
verbose	A logical value indicating whether to print additional output during the computation. Default is TRUE.

Value

A list containing the optimal k clustering results.

`calculate_markers` *Calculate Marker Genes*

Description

The `calculate_markers` function identifies marker genes for specific cell types based on given criteria and strategies. These strategies help ensure that the marker genesidentified are more unique to each cell type, providing better accuracy in downstream analysis.

Usage

```
calculate_markers(
  single_cell_data_exp,
  reference,
  group_clusters_var,
  use_min_cor_strategy = FALSE,
  delete_shared_level_markers = TRUE,
  shared_level_markers = NULL,
  delete_shared_internal_markers = TRUE,
  p_value_attribute = "p_val_adj",
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  test.use.value = "wilcox",
  filter_markers = NULL,
  marker_strategy = NULL,
  verbose = TRUE
)
```

Arguments

`single_cell_data_exp`
: A data frame or matrix containing the single cell data.

`reference`
: A data frame or matrix containing the reference data.

`group_clusters_var`
: The variable name of the group clusters in the `single_cell_data_exp`.

`use_min_cor_strategy`
: Logical value indicating whether to use the minimum correlation strategy. This strategy is used to identify marker genes that are least correlated with each other, thereby ensuring uniqueness. Default is FALSE.

`delete_shared_level_markers`
: Logical value indicating whether to delete shared level markers. This parameter enables the removal of marker genes shared across different levels to ensure uniqueness within each level. Default is TRUE.

`shared_level_markers`
: A vector containing shared level markers. NULL by default.

`delete_shared_internal_markers`
: Logical value indicating whether to delete shared internal markers. This parameter allows for the removal of marker genes that are common within a given level, to promote marker uniqueness. Default is TRUE.

`p_value_attribute`
: String indicating the attribute name of the p-value in the single cell data. Default is 'p_val_adj'.

`param.logfc.threshold`
: Threshold for the log-fold change, determining the minimum acceptable change for a gene to be considered a marker. Default is 2.0.

`param.p_val_adj`
: Adjusted p-value threshold. This threshold is used to control the false discovery rate. Default is 0.05.

`test.use.value`
: String indicating the statistical test used for identifying markers. Default is 'wilcox'.

`filter_markers`
: A vector containing markers to be filtered out before the analysis. NULL by default.

`marker_strategy`
: Strategy used to select marker genes. This could be a specific algorithm or statistical method. NULL by default.

`verbose`
: Logical value indicating whether to display additional information during the function's execution. Default is TRUE.

Details

The function calculates marker genes for each cell type in the provided data, with options to apply different strategies and filters during the process. It considers factors such as p-value, log-fold change, and uniqueness at different levels to ensure the identified markers are most representative of their respective cell types.

Value

A list of marker genes for each cell type.

See Also

`Matrix`

`calculate_markers_level_intersection` *Calculation of Marker Genes Intersection at Different Levels*

Description

`calculate_markers_level_intersection` computes the intersection of marker genes at various levels of clustering. This function is currently under testing and development, intending to streamline the code and improve the algorithm performance.

Usage

```
calculate_markers_level_intersection(
  single_cell_data_exp,
  top_clusters_var,
  sub_clusters_var,
  top_clusters_list,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  marker_strategy = NULL,
  verbose = FALSE
)
```

Arguments

`single_cell_data_exp`	A list or data.frame representing the single cell data.
`top_clusters_var`	A string representing the variable in `single_cell_data_exp` used to indicate the top clusters.
`sub_clusters_var`	A string representing the variable in `single_cell_data_exp` used to indicate the subclusters.
`top_clusters_list`	A list or vector indicating the top clusters to be considered.
`param.logfc.threshold`	A numeric value representing the log fold change threshold for marker genes. Default is 2.0.
`param.p_val_adj`	A numeric value representing the adjusted p-value threshold for marker genes. Default is 0.05.
`marker_strategy`	A character string specifying the overall strategy for marker gene handling. Default is NULL.
`verbose`	A logical value indicating whether to print detailed output during execution. Default is FALSE.

Value

A list containing the intersection of marker genes at different levels.

`calculate_min_correlation_incremental_markers` *Calculate Minimum Correlation with Incremental Markers*

Description

This function processes a list of marker genes for each group, sorted in descending order by log2 fold change. The objective is to determine the number of genes that must be selected to achieve the minimum correlation between samples in the reference dataset. The function is designed to incrementally add markers, starting with those that display the greatest change (as indicated by the log2 fold change). For each new set of markers, the function calculates the correlation between samples in the reference dataset.

Usage

```
calculate_min_correlation_incremental_markers(
  list_markers,
  reference,
  minimum_markers = 4,
  verbose = TRUE,
  verbose_detailed = TRUE
)
```

Arguments

list_markers	A list of marker genes for each group, ordered by log2 fold change in descending order.
reference	The reference dataset to which the correlations will be calculated.
minimum_markers	The minimum number of marker genes to be selected, default is set to 4. This can be adjusted based on the analysis requirements.
verbose	If TRUE, prints detailed information during the function execution. Default is TRUE.
verbose_detailed	If TRUE, prints even more detailed information during the function execution. Default is TRUE.

Details

The function begins by including the most significant markers (those with the highest log2 fold change) in the analysis and calculates the correlation between samples in the reference dataset. This process is repeated, each time adding more markers, until a satisfactory level of correlation is reached. This way, the function attempts to balance the inclusion of informative markers against the potential noise introduced by less significant markers.

Value

A data frame with the minimum number of markers required to achieve a satisfactory level of correlation.

---

calculate_performance_metrics

Compute Performance Metrics

Description

This function calculates several performance metrics to assess the quality of deconvolution results. These metrics quantify the accuracy of the estimated cell type proportions compared to the true proportions.

Usage

```
calculate_performance_metrics(true_prop, calc_prop, digits = 20)
```

Arguments

true_prop	A matrix or dataframe of the true proportions. Rows correspond to cell types and columns to samples.
calc_prop	A matrix or dataframe of the estimated proportions. It should have the same structure as true_prop.
digits	The number of decimal places to which the computed metrics should be rounded. Default is 20.

Details

This function computes performance metrics including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Pearson Correlation Coefficient (r) to assess the accuracy of deconvolution results. The metrics are calculated separately for each cell type and for each sample. The precision of the returned metrics can be adjusted with the 'digits' parameter.

Value

A list of calculated performance metrics. Each element of the list represents a different metric.

---

calculate_prop_back_propagation *Rescale Clustering Proportions Based on Higher Level Data*

Description

`calculate_prop_back_propagation` function computes rescaled proportions of higher level clustering based on more detailed proportions. It assumes the input `top_proportions` represent proportions for single-leaf samples in the deconvolution.

Usage

```
calculate_prop_back_propagation(
  top_proportions,
  level_proportions,
  verbose = FALSE
)
```

Arguments

top_proportions	A numeric or matrix of proportions representing higher level clustering.
level_proportions	A numeric or matrix of proportions representing more detailed clustering.
verbose	A logical value indicating whether to print detailed output during execution. Default is FALSE.

Details

The function applies a specific calculation depending on whether `top_proportions` is numeric or a matrix. If it's numeric, it simply multiplies `level_proportions` by `top_proportions`. If it's a matrix, it applies a sweep operation to multiply each column of `level_proportions` by the corresponding column in `top_proportions`. The resulting proportions should sum up to the same values as `top_proportions`.

Value

A numeric or matrix of rescaled proportions.

`calculate_p_values_wt_vs_ko` *Calculate p-values for WT vs KO*

Description

This function calculates p-values for wild type (WT) vs knockout (KO) based on given results. For more information, check: https://www.cyclismo.org/tutorial/R/pValues.html

Usage

```
calculate_p_values_wt_vs_ko(results.prop)
```

Arguments

`results.prop`	DataFrame, the results that will be used to perform the t-tests.

Details

The function takes a DataFrame of results and transposes it. Then for each cell type present in the data, it extracts the WT and KO results, performs a t-test, and prints the result.

Value

No return value. The function prints the result of the t-test for each cell type.

`calculate_sparseness_matrix` *Calculate Sparseness of a Matrix*

Description

This function calculates the sparseness of a given matrix, i.e., the percentage of zero and NA values in the matrix.

Usage

```
calculate_sparseness_matrix(matrix_object)
```

Arguments

`matrix_object` Matrix, the matrix for which to calculate sparseness.

Details

The sparseness of a matrix is defined as the percentage of zero and NA values out of the total number of entries in the matrix. This function counts the number of non-zero and non-NA values in the matrix, and then calculates sparseness as (1 - (non-zero/total)) * 100.

Value

Numeric, the sparseness of the matrix in percentage.

---

checkHT *Check Acceptability of 'n' for 'head()' and 'tail()' Methods*

Description

This function checks the acceptability of 'n' for 'head()' and 'tail()' methods. If you encounter an error related to this function, you can load it from: https://svn.r-project.org/R/trunk/src/library/utils/R/head.R

Usage

```
checkHT(n, d)
```

Arguments

`n`	The number to be checked for its acceptability.
`d`	Data input for the function.

Value

Returns TRUE if 'n' is acceptable, and FALSE otherwise.

---

choose_model_from_grid_search_object *Choose Model from Grid Search Result*

Description

This function retrieves a specific model from a grid search result based on provided alpha and beta parameters.

Usage

```
choose_model_from_grid_search_object(
  grid_search_result,
  alpha = NULL,
  beta = NULL
)
```

Arguments

grid_search_result	A list containing the results of a grid search. Each element of the list should represent one model and include the alpha and beta parameters for that model.
alpha	Numeric, the alpha parameter of the model to be returned.
beta	Numeric, the beta parameter of the model to be returned.

Details

The function iterates through the 'grid_search_result' list until it finds a model that matches the provided alpha and beta parameters. If a matching model is found, the function returns the model; otherwise, it returns NULL.

Value

The model from 'grid_search_result' that matches the provided alpha and beta parameters, or NULL if no such model exists.

---

create_basis_multiple *create_basis_multiple*

Description

This function constructs a basis matrix based on an ExpressionSet object for single cells. It is based on the function created in the SCDC R implementation. Please refer to the original implementation for more information: https://meichendong.github.io/SCDC/. Version: 0.0.0.9000

Usage

```
create_basis_multiple(
  x,
  ct.sub = NULL,
  ct.varname,
  sample,
  ct.cell.size = NULL,
  verbose = FALSE
)
```

Arguments

x	An ExpressionSet object for single cells. This object will serve as the basis for the construction of the basis matrix.
ct.sub	A vector of cell types that are selected to construct the basis matrix. This parameter is optional, with the default being NULL.
ct.varname	A string that indicates the variable name for 'cell types' in the 'x' dataset.
sample	A string that indicates the variable name for subject/samples in the 'x' dataset.
ct.cell.size	A vector of cell size factors corresponding to the ct.sub according to prior knowledge. The default is NULL, which means the "library size" is calculated based on the data. If a vector is provided, the names of the vector (names(ct.cell.size)) should not be NULL.

verbose	A boolean flag for controlling the amount of information printed during the execution of the function. Default is FALSE.

Value

A list with the following components:

- Basis matrix
- Sum of cell-type-specific library size
- Sample variance matrix
- Basis matrix by mvw
- mvw matrix.

References

SCDC: a proportion estimation method for single cell RNA-Seq data. https://meichendong.github.io/SCDC/.

---

create_basis_one *create_basis_one*

Description

This function constructs a basis matrix for single cells from one subject, based on an ExpressionSet object. It is based on the function created in the SCDC R implementation. Please refer to the original implementation for more information: https://meichendong.github.io/SCDC/. Version: 0.0.0.9000

Usage

```
create_basis_one(
  x,
  ct.sub = NULL,
  ct.varname,
  sample,
  ct.cell.size = NULL,
  verbose = FALSE
)
```

Arguments

x	An ExpressionSet object for single cells. This object will serve as the basis for the construction of the basis matrix.
ct.sub	A vector of cell types that are selected to construct the basis matrix. This parameter is optional, with the default being NULL.
ct.varname	A string that indicates the variable name for 'cell types' in the 'x' dataset.
sample	A string that indicates the variable name for subject/samples in the 'x' dataset.
ct.cell.size	A vector of cell size factors corresponding to the ct.sub according to prior knowledge. The default is NULL, which means the "library size" is calculated based on the data. If a vector is provided, the names of the vector (names(ct.cell.size)) should not be NULL.
verbose	A boolean flag for controlling the amount of information printed during the execution of the function. Default is FALSE.

Value

A list with the following components:

- Basis matrix
- Sum of cell-type-specific library size
- Sample variance matrix
- Basis matrix by mvw
- mvw matrix.

References

SCDC: a proportion estimation method for single cell RNA-Seq data. https://meichendong.github.io/SCDC/.

`create_dataframe_results_multiple_deconvolutions`
Create Data Frame of Multiple Deconvolution Results

Description

This function creates a summary data frame from the results of multiple deconvolutions, typically obtained from a grid search procedure.

Usage

```
create_dataframe_results_multiple_deconvolutions(grid_info, verbose = TRUE)
```

Arguments

`grid_info`	A list of deconvolution results obtained from a grid search.
`verbose`	Shows details about the added deconvolutions.

Details

The function iterates through all the deconvolution results in the input list. For each result, it extracts relevant performance metrics such as alpha, beta, divergence value, and delta divergence value. These metrics are then combined into a single data frame. Additionally, it computes the logarithm of the divergence and delta divergence values, and includes them in the data frame as well.

Value

A data frame that contains performance metrics for each deconvolution result, including alpha, beta, divergence value, delta divergence value, and their logarithmic forms.

decoflex_build_cell_reference
decoflex_build_cell_reference

Description

This function constructs a reference matrix for single cells, based on an ExpressionSet object. It is based on the function created in the SCDC R implementation. Please refer to the original implementation for more information: https://meichendong.github.io/SCDC/. Version: 0.0.0.9000

Usage

```
decoflex_build_cell_reference(
  x,
  ct.sub = NULL,
  ct.varname,
  sample,
  ct.cell.size = NULL,
  verbose = FALSE
)
```

Arguments

x	An ExpressionSet object for single cells. This object will serve as the basis for the construction of the reference matrix.
ct.sub	A vector of cell types that are selected to construct the reference matrix. This parameter is optional, with the default being NULL.
ct.varname	A string that indicates the variable name for 'cell types' in the 'x' dataset.
sample	A string that indicates the variable name for subject/samples in the 'x' dataset.
ct.cell.size	A vector of cell size factors corresponding to the ct.sub according to prior knowledge. The default is NULL, which means the "library size" is calculated based on the data. If a vector is provided, the names of the vector (names(ct.cell.size)) should not be NULL.
verbose	A boolean flag for controlling the amount of information printed during the execution of the function. Default is FALSE.

Value

A list with the following components:

- basis: The constructed reference matrix.
- detailed: The detailed output from create_basis_one or create_basis_multiple, which includes:
  - Basis matrix
  - Sum of cell-type-specific library size
  - Sample variance matrix
  - Basis matrix by mvw
  - mvw matrix.

References

SCDC: a proportion estimation method for single cell RNA-Seq data. https://meichendong.github.io/SCDC/.

`environment_creation` *Set up environment for NMMFlex*

Description

The environment_creation function installs and verifies the NMMFlex Python package to prepare the required environment for using NMMFlex in R.

Usage

```
environment_creation()
```

Value

This function does not return any value.

`filter_main_celltypes` *Filter Main Cell Types with Markers*

Description

Filters the main cell types using marker genes and performs clustering optimization on the filtered data.

Usage

```
filter_main_celltypes(
  single_cell_data_exp,
  reference_celltypes_top,
  group_clusters_var,
  use_min_cor_strategy = TRUE,
  delete_shared_level_markers = FALSE,
  delete_shared_internal_markers = FALSE,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame or matrix containing single-cell expression data.
reference_celltypes_top	A list of reference cell types to be used for marker identification.
group_clusters_var	A string or variable indicating the clusters for each cell.
use_min_cor_strategy	A logical value indicating whether to use the minimum correlation strategy. Default is TRUE.
delete_shared_level_markers	A logical value indicating whether to delete markers shared at level of the hierarchy. Default is FALSE.
delete_shared_internal_markers	A logical value indicating whether to delete markers shared internally. Default is FALSE.
param.logfc.threshold	A numeric value indicating the log-fold-change threshold for marker selection. Default is 2.0.
param.p_val_adj	A numeric value indicating the adjusted p-value threshold for marker selection. Default is 0.05.
verbose	A logical value indicating whether to print additional output during the computation. Default is FALSE.

Value

A data frame or matrix containing the filtered single-cell data.

FindMarkers.proxy *Calculate Marker Genes Proxy*

Description

The FindMarkers.proxy function is a flexible interface for identifying marker genes using various methods.It's designed to be readily extensible, facilitating the integration of novel marker calculation techniques in the future. The function calculates the markers for a specific group relative to all other cells (rest_groups) in a given Seurat object (seurat.reference).

Usage

```
## S3 method for class 'proxy'
FindMarkers(
  seurat.reference,
  group,
  rest_groups,
  test.use.value = "wilcox",
  marker_strategy = "keep_default_values",
  max.size.percentage = 0.7,
  param.logfc.threshold = 2,
  verbose = FALSE
)
```

Arguments

`seurat.reference`	A Seurat object used as the reference for marker gene calculation. This object contains the single-cell RNA-seq data.
`group`	A string specifying the cell group (based on clustering) for which to calculate marker genes.
`rest_groups`	A list or vector specifying the remaining cell groups to be considered as background in the marker genes calculation.
`test.use.value`	String specifying the statistical test to use for marker identification. The methods currently supported include: 'wilcox', 'bimod', 't', 'negbinom', 'poisson', 'LR', and 'MAST'. Please note that 'roc' and 'DESeq2' methods are currently not supported due to discrepancies in output values and support issues, respectively. Default is 'wilcox'.
`marker_strategy`	A string specifying the strategy to use in marker gene calculation. 'keep_default_values' (default) retains the default values. Other strategies may be added in the future.
`max.size.percentage`	Maximum acceptable size (as a proportion) for a group in the marker gene calculation. Default is 0.70.
`param.logfc.threshold`	Threshold for the log-fold change, which determines the minimum change required for a gene to be considered as a marker. Default is 2.0.
`verbose`	Logical value indicating whether to display detailed progress messages during function execution. Default is FALSE.

Details

This function serves as a proxy to different methods for marker gene calculation. It currently supports several statistical tests and provides a convenient point of extension for the inclusion of additional marker calculation methods. Note that the support for different tests can be influenced by the characteristics of the single-cell experiments, and it's important to choose a method that's appropriate for your data.

Value

A list of marker genes for the specified cell group.

`generateBulk_allcells` *generateBulk_allcells*

Description

This function constructs Pseudo bulk samples for single cells from one subject based on an ExpressionSet object. It is based on the function created in the SCDC R implementation. Please refer to the original implementation for more information: https://meichendong.github.io/SCDC/. Version: 0.0.0.9000

Usage

```
generateBulk_allcells(
  eset,
  ct.varname,
  sample,
  disease = NULL,
  ct.sub = NULL,
  verbose = FALSE
)
```

Arguments

eset	An ExpressionSet object for single cells. This object will serve as the basis for the construction of the Pseudo bulk samples.
ct.varname	A string that indicates the variable name for 'cell types' in the 'eset' dataset.
sample	A string that indicates the variable name for subject/samples in the 'eset' dataset.
disease	A string indicating the health condition of subjects.
ct.sub	A vector of cell types that are selected to construct Pseudo bulk samples. This parameter is optional, with the default being NULL. If NULL, then all cell types are used.
verbose	A boolean flag for controlling the amount of information printed during the execution of the function. Default is FALSE.

Value

A list with the following components:

- Pseudo bulk samples ExpressionSet
- Actual cell-type proportions

References

SCDC: a proportion estimation method for single cell RNA-Seq data. https://meichendong.github.io/SCDC/.

---

getESET *Create an ExpressionSet object*

Description

This function creates an ExpressionSet object which is a container for storing gene expression data along with related experimental data. An ExpressionSet object includes an expression matrix, feature data (fdata) and phenotype data (pdata).

Usage

```
getESET(exprs, fdata, pdata)
```

Arguments

exprs	A matrix or data frame of expression values. Rows correspond to features (e.g., genes) and columns correspond to samples.
fdata	A data frame or matrix of feature data. Each row corresponds to feature in the expression set and columns correspond to feature variables or a annotations. The row names should match the row names of the exprs parameter.
pdata	A data frame or matrix of phenotype data. Each row corresponds to a sample in the expression set and columns correspond to phenotype variables. The row names should match the column names of the exprs parameter.

Value

An ExpressionSet object which includes the expression data (exprs), phenotype data (pdata), and feature data (fdata).

---

hierachy_selection_plus_deconvolution
Hierarchy Selection Plus Deconvolution

Description

Runs the clustering method and the deconvolution process on given single-cell expression data.

Usage

```
hierachy_selection_plus_deconvolution(
  single_cell_data_exp,
  var_cell_type,
  subset_celltypes = NULL,
  var_sample,
  number_clusters_one_celltype = 1,
  distance_method = "euclidean",
  hclust_method = "average",
  min_size_leaf = 3,
  max_k_tried_hier_clustering = 3,
  random_seed = NULL,
  use_min_cor_strategy = TRUE,
  delete_shared_level_markers = FALSE,
  delete_shared_internal_markers = FALSE,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  filter_main_markers = TRUE,
  verbose = FALSE
)
```

Arguments

`single_cell_data_exp`
: A data frame or matrix containing single-cell expression data.

`var_cell_type`
: A vector or factor indicating the cell type for each cell.

`subset_celltypes`
: A vector of cell types to subset for the analysis. Default is NULL, implying all cell types are used.

`var_sample`
: A vector or factor indicating the sample source for each cell.

`number_clusters_one_celltype`
: An integer specifying the number of clusters for each cell type. Default is 1.

`distance_method`
: A string specifying the method to compute the distance. Default is 'euclidean'.

`hclust_method`
: A string specifying the method to perform hierarchical clustering. Default is 'average'.

`min_size_leaf`
: An integer specifying the minimum size of the leaf in the hierarchical clustering. Default is 3.

`max_k_tried_hier_clustering`
: An integer specifying the maximum number of clusters tried in hierarchical clustering. Default is 3.

`random_seed`
: An integer specifying the random seed for reproducibility. Default is NULL.

`use_min_cor_strategy`
: A logical value indicating whether to use the minimum correlation strategy. Default is TRUE.

`delete_shared_level_markers`
: A logical value indicating whether to delete markers shared at level of the hierarchy. Default is FALSE.

`delete_shared_internal_markers`
: A logical value indicating whether to delete markers shared internally. Default is FALSE.

`filter_markers`
: A vector specifying which markers to filter. Default is NULL.

`param.logfc.threshold`
: A numeric value indicating the log-fold-change threshold for marker selection. Default is 2.0.

`param.p_val_adj`
: A numeric value indicating the adjusted p-value threshold for marker selection. Default is 0.05.

`filter_main_markers`
: A logical value indicating whether to filter main markers. Default is TRUE.

`verbose`
: A logical value indicating whether to print additional output during the computation. Default is FALSE.

Value

A list containing results of the clustering and deconvolution process.

join_back_propagation_proportions_leaves *Join Back-Propagation Proportions from Multiple Deconvolution Results*

Description

`join_back_propagation_proportions_leaves` function iterates over a list of deconvolution results and merges the back-propagated proportions from each result.

Usage

```
join_back_propagation_proportions_leaves(deco_results_list, verbose = FALSE)
```

Arguments

`deco_results_list`	A list containing deconvolution results. Each element in the list shouldbe a list with an element named "back_propagation_proportions_top_detailed" that contains the back-propagated proportions.
`verbose`	A logical value indicating whether to print detailed output during execution. Default is FALSE.

Details

The function iterates over the deconvolution results in `deco_results_list`. From each result, it extracts the back-propagation proportions, then merges them row-wise using the `rbind` function. This process continues until all the back-propagation proportions from all deconvolution results have been merged.

Value

A matrix that contains the merged back-propagation proportions from all the input deconvolution results.

load_libraries *Load Required Libraries*

Description

This function is designed to load the NMMFlex library, a Python package. The function uses the `reticulate` package to facilitate interoperability between R and Python, allowing the use of Python scripts within the R environment. NMMFlex needs to be located at the specific path on your local machine for this function to work.

Usage

```
load_libraries()
```

Details

This function first attempts to install the Python package from a local directory using `reticulate::py_install`. It then tries to import the package. If the import is successful, it confirms that the package is installed and loaded. If not, it throws an error message. Make sure to adjust the path to the location of the NMMFlex package on your machine.

Value

A message indicating whether or not the NMMFlex library was successfully loaded.

Note

Ensure that the Python package is located at the provided path on your machine before using this function.

See Also

`reticulate`

`order_result_generic` *Order Columns in a Matrix*

Description

This function orders columns in a matrix based on a given starting position.

Usage

```
order_result_generic(matrix, start = 7, verbose = FALSE)
```

Arguments

`matrix`	Matrix, the matrix to be ordered.
`start`	Integer, the column from which to start ordering (default = 7).
`verbose`	Logical, if TRUE, displays detailed messages during execution (default = FALSE).

Details

The function orders the columns in the matrix starting from the position indicated by the 'start' parameter. It creates a new matrix by concatenating the ordered columns and returns it.

Value

Matrix, the ordered matrix.

order_simple_vector *Order Elements in a Vector*

Description

This function orders elements in a vector based on a given starting position.

Usage

```
order_simple_vector(vector, start = 5, verbose = FALSE)
```

Arguments

vector	Vector, the vector to be ordered.
start	Integer, the position from which to start ordering (default = 5).
verbose	Logical, if TRUE, displays detailed messages during execution (default = FALSE).

Details

The function orders the elements in the vector starting from the position indicated by the 'start' parameter. It creates a new vector by concatenating the ordered elements and returns it.

Value

Vector, the ordered vector.

performance_metrics_general *Calculate and Print Performance Metrics*

Description

This function computes various performance metrics to assess the accuracy of calculated cell type proportions relative to the true proportions. The performance metrics are computed both by clusters and by samples. The results can be optionally printed.

Usage

```
performance_metrics_general(
  true_prop,
  calc_prop,
  title = NULL,
  verbose = FALSE
)
```

Arguments

true_prop	A matrix or dataframe of the true proportions. Rows correspond to cell types and columns to samples.
calc_prop	A matrix or dataframe of the calculated proportions. It should have the same structure as true_prop.
title	Optional title for the print output.
verbose	If TRUE, the performance metrics will be printed. Default is FALSE.

Details

The function computes performance metrics, such as mean absolute error, root mean squared error, and correlation coefficient, using the 'calculate_performance_metrics' function. These metrics are calculated for each cell type (i.e., by clusters) and for each sample. If verbose is set to TRUE, these metrics will be printed with the 'print_performance_metrics' function.

Value

A list with three elements: 'pm_clusters' contains the performance metrics by clusters, 'pm_samples' contains the performance metrics by samples, true_prop' contains the true proportions, and 'calc_prop' contains the calculated proportions.

---

performance_plot_multiple_deconvolutions

Plot Performance of Multiple Deconvolutions

Description

This function generates a plot showing the performance of multiple deconvolutions. The plot visualizes the divergence and delta divergence metrics against a main parameter of interest. Besides it is used for plotting the results for the gridSearch in a general level.

Usage

```
performance_plot_multiple_deconvolutions(
  performance_df,
  main_parameter = "alpha"
)
```

Arguments

performance_df	A data frame that contains performance metrics for each deconvolution result. The data frame should include the main parameter, divergence, and delta divergence values.
main_parameter	A string that specifies the name of the main parameter to be plotted on the x-axis.This parameter should be a column in the 'performance_df' data frame.

Details

The function uses ggplot2 to generate a line plot where the y-axis represents the logarithmic values of the divergence and delta divergence metrics, and the x-axis represents the main parameter. The divergence and delta divergence metrics are differentiated by color.

Value

A ggplot object showing the performance of multiple deconvolutions against the specified main parameter.

`plot_deconvolution_performance`

Plot Deconvolution Performance Metrics

Description

This function plots various performance metrics for a deconvolution process across multiple iterations. Function for gridSearch analysis: General plots.

Usage

```
plot_deconvolution_performance(running_info, title_addition = "")
```

Arguments

`running_info`	A data frame with iteration number and the corresponding values for divergence and delta divergence.
`title_addition`	An optional string that is added to the plot title.

Details

This function creates three plots to visualize the performance of a deconvolution process:

1. The natural logarithm of the divergence value over iterations.
2. The natural logarithm of the delta divergence value over iterations.
3. Both the above metrics in the same plot. The divergence value provides a measure of how much the deconvolution results deviate from the ground truth, while the delta divergence gives a measure of how much the divergence changes between successive iterations.

Value

The function returns a list of ggplot objects, each representing a different visualization of the deconvolution performance.

`plot_deconvolution_performance_x_y_z_total` *Plot Detailed Deconvolution Performance*

Description

This function plots detailed performance of the deconvolution with respect to x, y, z, and total divergence. If specified, the y and z divergence values can be scaled by the given alpha and beta parameters.

Usage

```
plot_deconvolution_performance_x_y_z_total(
  running_info,
  title_addition = "",
  alpha = NULL,
  beta = NULL,
  scale_parameters = FALSE
)
```

Arguments

`running_info`	Data frame, the running information of the deconvolution.
`title_addition`	Character, an optional string to be added to the plot title. Optional, default is ''.
`alpha`	Numeric, the alpha value to scale the y divergence. Optional, default is NULL.
`beta`	Numeric, the beta value to scale the z divergence. Optional, default is NULL.
`scale_parameters`	Logical, whether to scale the y and z divergence values by the given alpha and beta. Optional, default is FALSE.

Details

The function generates a line plot showing the log of the divergence values (for x, y, z, total, and delta) against iterations. The line color is specified according to the type of divergence. The divergence values for y and z can be scaled by the specified alpha and beta parameters if 'scale_parameters' is TRUE.

Value

A ggplot2 object representing the plot.

`plot_heatmap_alpha_beta` *Plot Heatmap of Alpha and Beta Values*

Description

This function generates a heatmap showing the divergence value for each combination of alpha and beta values. This visualization aids in identifying the optimal alpha and beta parameters.

Usage

```
plot_heatmap_alpha_beta(
  performance_df,
  fill_value = "divergence_value",
  title = "Divergence performance: alpha vs beta"
)
```

Arguments

`performance_df`	A data frame that contains performance metrics for each alpha, beta combination. The data frame should include columns for alpha, beta, and divergence value.
`fill_value`	A string indicating the column to be used for filling the heatmap. Default is "divergence_value".
`title`	A string specifying the title of the plot. Default is 'Divergence performance: alpha vs beta'.

Details

The function uses ggplot2 to generate a heatmap where the x-axis represents alpha values and the y-axis represents beta values. The color fill of each tile is determined by the 'fill_value' parameter, which is typically set to 'divergence_value'.

Value

A ggplot object showing a heatmap of divergence values for different combinations of alpha and beta values.

`plot_heatmap_correlation_reference` *Plot Heatmap of Correlation Among Cell-Types or Groups*

Description

This function creates a heatmap to visually display the correlation between different cell-types or groups of cell-types. The color gradient of the heatmap represents the correlation coefficient values, providing a clear visual summary of the correlations in the data.

Usage

```
plot_heatmap_correlation_reference(reference_w.k.celltypes, title = "Heatmap")
```

Arguments

reference_w.k.celltypes	A data frame containing the cell-types or groups of cell-types and their corresponding basis values.
title	Optional; a character string representing the title of the heatmap. If not provided, the title defaults to 'Heatmap'.

Value

This function generates a heatmap as a side-effect. It does not return any value.

@import grDevices @import ggplot2 @import gplots @import graphics @import stats

---

plot_hierarchical_clustering

Plot hierarchical clustering

Description

This function generates a plot of the hierarchical clustering and includes the option for a custom title. The plot visualizes the dendrogram resulting from the clustering, with the height of the cut denoted by a green line. Clusters resulting from the cut are highlighted with green borders. The function also outputs a table summarizing the resulting cluster assignments.

Usage

```
plot_hierarchical_clustering(
  hierar_clust,
  k_value,
  fit_celltypes,
  title = NULL
)
```

Arguments

hierar_clust	A hierarchical clustering object, which is the result of hclust function or similar.
k_value	An integer value indicating the number of clusters to highlight in the plot.
fit_celltypes	A vector containing the assignment of each data point to a cluster.
title	Optional; a character string representing the title of the plot. If NULL, no title is added.

Value

This function generates a plot as a side-effect. The returned value is a table summarizing the cluster assignments.

`plot_performance_for_each_model` *Plot Performance for Each Model*

Description

This function plots the performance of each model from a grid search result based on specified alpha and beta values.

Usage

```
plot_performance_for_each_model(
  grid_search_result,
  alpha = NULL,
  beta = NULL,
  plot_type = "summary",
  scale_parameters = FALSE,
  save.pdf = FALSE,
  path.pdf = NULL
)
```

Arguments

`grid_search_result`	A list object that contains the results of a grid search.
`alpha`	Numeric, the alpha value to filter the models to plot. Optional, default is NULL.
`beta`	Numeric, the beta value to filter the models to plot. Optional, default is NULL.
`plot_type`	Character, specifies the type of plot to produce. Options are 'summary' (default) or other types supported by the underlying plotting function.
`scale_parameters`	Logical, whether to scale the parameters in the plot. Optional, default is FALSE.
`save.pdf`	Logical, whether to save the plot as a PDF. Optional, default is FALSE.
`path.pdf`	String, the path where to save the PDF file if 'save.pdf' is TRUE. Optional, default is NULL.

Details

The function iterates over each model in the grid search result. For each model, it produces a plot based on the 'plot_type' argument. It filters the models to plot based on the 'alpha' and 'beta' arguments.

If 'save.pdf' is TRUE, it saves the plot as a PDF file to the specified path or the current working directory if no path is specified.

Value

The function is used for its side effects of creating and optionally saving plots. It does not return anything.

print_performance_metrics

Display Performance Metrics

Description

This function neatly prints the calculated performance metrics, facilitating the interpretation of deconvolution results.

Usage

```
print_performance_metrics(list_metrics, title = NULL)
```

Arguments

list_metrics	A list of calculated performance metrics, as produced by the calculate_performance_metrics() function.
title	An optional title for the printed metrics. Default is NULL.

Details

The function takes as input a list of performance metrics, which should include metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Pearson Correlation Coefficient (r). These metrics are then printed in a structured format for easy interpretation. The optional 'title' parameter can be used to give the printed metrics a descriptive title.

Value

NULL. The function is used for its side effect of printing the metrics.

---

recalculate_proportions

Recalculate Cell Type Proportions

Description

This function recalculates the proportions of cell types based on the top-level clustering proportions and the more granular sub-cluster proportions.

Usage

```
recalculate_proportions(
  top_cluster_deco_list,
  top_clusters_list,
  subcluster_deco_list,
  verbose = FALSE
)
```

Arguments

top_cluster_deco_list	A list containing the deconvolution results for the top-level clusters.
top_clusters_list	A vector containing the names or identifiers of the top-level clusters.
subcluster_deco_list	A list of deconvolution results for the sub-clusters.
verbose	A logical indicating whether or not to print detailed messages while the function is running. Default is FALSE.

Details

This function takes as input the top-level cluster proportions and a list of sub-cluster deconvolution results. It scales the sub-cluster proportions by the corresponding top-level cluster proportions and then combines all the proportions. This results in an updated set of cell type proportions that reflects the hierarchical structure of the clustering.

Value

A matrix where each row represents a cell type and each column represents a sample. Each entry in the matrix represents the recalculated proportion of a cell type in a sample.

---

run_complete_deconvolution

title Run Complete Deconvolution

Description

Function that runs the src with three matrices as input. You can define if the model is complete or partial in terms of the input and therefore the output. For example, if some matrices are fixed (e.g., WH), and Y is provided, then the variable Gamma = 0, resulting in Divergence(X|WG) being zero, while the rest can still be calculated.

Deconvolution using three input matrices.

Usage

```
run_complete_deconvolution(
  x_matrix,
  y_matrix = NULL,
  z_matrix = NULL,
  k,
  gamma = 1,
  alpha = 0,
  beta = 0,
  delta_threshold = 1e-10,
  max_iterations = 200,
  print_limit = 100,
  proportion_constraint_h = TRUE,
  regularize_w = NULL,
  alpha_regularizer_w = 0,
```

```
  fixed_w = NULL,
  fixed_h = NULL,
  fixed_a = NULL,
  fixed_b = NULL,
  initialized_w = NULL,
  initialized_h = NULL,
  initialized_a = NULL,
  initialized_b = NULL,
  init_method_w = "random_based.uniform",
  init_method_h = "random_based.uniform",
  init_method_a = "random_based.uniform",
  init_method_b = "random_based.uniform",
  verbose = TRUE
)
```

Arguments

x_matrix	A matrix, Input matrix corresponding to a dataframe.
y_matrix	A matrix or NULL, Input matrix.
z_matrix	A matrix or NULL, Input matrix.
k	An integer, The rank used for deconvolution.
gamma	A numeric, The gamma parameter value. Default is 1.
alpha	A numeric, The alpha parameter value. Default is 0.0.
beta	A numeric, The beta parameter value. Default is 0.0.
delta_threshold	A numeric, The convergence threshold for stopping the deconvolution iterations. Default is 1e-10.
max_iterations	An integer, The maximum number of iterations to perform during deconvolution. Default is 200.
print_limit	An integer, The iteration interval at which to print progress messages during deconvolution. Default is 100.
proportion_constraint_h	A logical, Whether to apply a proportion constraint to matrix H. Default is TRUE.
regularize_w	A matrix or NULL, The regularization matrix for W. Default is NULL.
alpha_regularizer_w	A numeric, The alpha regularization parameter for W. Default is 0.
fixed_w	A matrix or NULL, The fixed matrix for W. Default is NULL.
fixed_h	A matrix or NULL, The fixed matrix for H. Default is NULL.
fixed_a	A matrix or NULL, The fixed matrix for A. Default is NULL.
fixed_b	A matrix or NULL, The fixed matrix for B. Default is NULL.
initialized_w	A matrix or NULL, The initial value for W. Default is NULL.
initialized_h	A matrix or NULL, The initial value for H. Default is NULL.
initialized_a	A matrix or NULL, The initial value for A. Default is NULL.
initialized_b	A matrix or NULL, The initial value for B. Default is NULL.
init_method_w	A string, The initialization method for W. Default is 'random_based.uniform'.

init_method_h	A string, The initialization method for H. Default is 'random_based.uniform'.
init_method_a	A string, The initialization method for A. Default is 'random_based.uniform'.
init_method_b	A string, The initialization method for B. Default is 'random_based.uniform'.
verbose	A logical, Whether to display progress messages during deconvolution. Default is TRUE.

Details

Raises an error if any of the input matrices. Raises an error if the shape of the input matrices and the rank k is not compatible. Raises an error if any of the parameter values are invalid or not in the expected range.

Value

The deconvoluted matrix.

`run_deconvolution_simulation`
Deconvolution Simulation

Description

This function performs a detailed simulation of the deconvolution process on single cell expression data. It allows for the inclusion of multiple cluster levels (both top level and sub level clusters), and customization of various parameters such as marker strategy, log fold change threshold, and adjusted p-value threshold. The function also supports a minimum correlation strategy for deconvolution, providing flexibility in the simulation approach. Verbose mode can be enabled for additional execution details. Note: If you encounter an error related to the function 'checkHT', you can load it from: https://svn.r-project.org/R/trunk/src/library/utils/R/head.R

Usage

```
run_deconvolution_simulation(
  single_cell_data_exp,
  top_clusters_var,
  top_clusters_list,
  sub_clusters_var,
  sub_clusters_list,
  sample,
  use_min_cor_strategy = FALSE,
  delete_shared_level_markers = TRUE,
  delete_shared_internal_markers = TRUE,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  marker_strategy = "keep_default_values",
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame representing the single cell data for the deconvolution simulation.
top_clusters_var	A vector indicating the top clusters variables.
top_clusters_list	A list containing the top clusters.
sub_clusters_var	A vector indicating the sub clusters variables.
sub_clusters_list	A list containing the sub clusters.
sample	A numeric or integer value indicating the sample size for the deconvolution simulation.
use_min_cor_strategy	A boolean. If TRUE, the function uses minimum correlation strategy. Default is FALSE.
delete_shared_level_markers	A boolean. If TRUE, shared level markers will be deleted. Default is TRUE.
delete_shared_internal_markers	A boolean. If TRUE, shared internal markers will be deleted. Default is TRUE.
filter_markers	A vector of markers to filter, or NULL. Default is NULL
param.logfc.threshold	A numeric value indicating the log fold change threshold. Default is 2.0.
param.p_val_adj	A numeric value indicating the adjusted p-value threshold. Default is 0.05.
marker_strategy	A character string specifying the marker strategy. Default is 'keep_default_values'.
verbose	A boolean. If TRUE, the function will print additional details during the execution. Default is FALSE.

Details

Tree Guided Deconvolution Functions

This script, tree_guided_deconvolution.R, is part of the Decoflex package. It includes the following functions:

add_grouping_single_cell() A function for adding single cell groupings.

aggregate_true_top_clustering_prop() A function to aggregate true top clustering proportions.

calculate_joint_cor() A function to calculate joint correlation.

calculate_markers() A function to calculate markers.

calculate_markers_level_intersection() A function to calculate markers level intersection.

calculate_min_correlation_incremental_markers() A function to calculate minimum correlation incremental markers.

calculate_performance_metrics() A function to calculate performance metrics.

calculate_prop_back_propagation() A function to calculate proportion back propagation.

FindMarkers.proxy() A proxy function to find markers.

join_back_propagation_proportions_leaves() A function to join back propagation proportions leaves.

performance_metrics_general() A function to calculate general performance metrics.

print_performance_metrics() A function to print performance metrics.

recalculate_proportions() A function to recalculate proportions.

run_deconvolution_simulation_generic() A function to run generic deconvolution simulations.

run_deconvolution_tree_guided() A function to run tree-guided deconvolution.

run_deconvolution_tree_guided_recursive() A recursive function to run tree-guided deconvolution.

Value

A list containing the results of the deconvolution simulation. This might include the estimated sources, the final weights, and any other relevant output depending on the actual implementation of the function.

`run_deconvolution_simulation_generic`
Generic Deconvolution Simulation

Description

This function is a generalized form of the deconvolution simulation process applicable to single cell data. It allows users to modify various parameters like the marker strategy, log fold change threshold, and adjusted p-value threshold. By setting use_min_cor_strategy to TRUE, the function will apply a minimum correlation strategy during the deconvolution.

Usage

```
run_deconvolution_simulation_generic(
  single_cell_data_exp,
  top_clusters_var,
  top_clusters_list,
  sub_clusters_var,
  sub_clusters_list,
  sample,
  use_min_cor_strategy = FALSE,
  delete_shared_level_markers = TRUE,
  delete_shared_internal_markers = TRUE,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  marker_strategy = NULL,
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame representing the single cell data for the deconvolution simulation.
top_clusters_var	A vector indicating the top clusters variables.
top_clusters_list	A list containing the top clusters.
sub_clusters_var	A vector indicating the sub clusters variables.
sub_clusters_list	A list containing the sub clusters.
sample	A numeric or integer value indicating the sample size for the deconvolution simulation.
use_min_cor_strategy	A boolean. If TRUE, the function uses minimum correlation strategy. Default is FALSE.
delete_shared_level_markers	A boolean. If TRUE, shared level markers will be deleted. Default is TRUE.
delete_shared_internal_markers	A boolean. If TRUE, shared internal markers will be deleted. Default is TRUE.
filter_markers	A vector of markers to filter, or NULL. Default is NULL
param.logfc.threshold	A numeric value indicating the log fold change threshold. Default is 2.0.
param.p_val_adj	A numeric value indicating the adjusted p-value threshold. Default is 0.05.
marker_strategy	A character string specifying the marker strategy, or NULL. Default is NULL.
verbose	A boolean. If TRUE, the function will print additional details during the execution. Default is FALSE.

Value

A list containing the results of the deconvolution simulation. This might include the estimated sources, the final weights, and any other relevant output depending on the actual implementation of the function.

---

run_deconvolution_simulation_generic_recursive

run_deconvolution_simulation_generic_recursive

Description

Perform a recursive deconvolution simulation on given single cell data. This function handles the first top node of the hierarchy and then send the rest to the recursive algorithm

Usage

```
run_deconvolution_simulation_generic_recursive(
  single_cell_data_exp,
  hierarchy,
  sub_clusters_var,
  sample,
  use_min_cor_strategy = TRUE,
  delete_shared_level_markers = FALSE,
  delete_shared_internal_markers = FALSE,
  deconvolute_just_top = FALSE,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  marker_strategy = NULL,
  verbose = FALSE
)
```

Arguments

single_cell_data_exp	A data frame that contains single cell expression data.
hierarchy	A list that outlines the hierarchy of cell types.
sub_clusters_var	A variable that represents the subclusters in the data.
sample	A variable that represents the samples in the data.
use_min_cor_strategy	A logical parameter indicating whether to use minimum correlation strategy. Defaults to TRUE.
delete_shared_level_markers	A logical parameter indicating whether to delete shared level markers. Defaults to FALSE.
delete_shared_internal_markers	A logical parameter indicating whether to delete shared internal markers. Defaults to FALSE.
deconvolute_just_top	A logical parameter indicating whether to only deconvolute top. Defaults to FALSE.
filter_markers	A parameter for specifying markers to filter, defaults to NULL.
param.logfc.threshold	A numeric value specifying the threshold for log fold change, defaults to 2.0.
param.p_val_adj	A numeric value specifying the threshold for adjusted p-values, defaults to 0.05.
marker_strategy	Strategy for selecting markers, defaults to NULL.
verbose	A logical parameter indicating whether to print detailed messages during the execution of the function. Defaults to FALSE.

Value

A list containing the deconvolution result and the pseudo bulk data.

`run_deconvolution_tree_guided` *Tree-guided Deconvolution of Bulk Data*

Description

`run_deconvolution_tree_guided` is a generic function for deconvoluting bulk data based on a reference single-cell dataset using a tree-guided method.

Usage

```
run_deconvolution_tree_guided(
  bulk_data,
  single_cell_data_exp,
  top_clusters_var,
  top_clusters_list,
  sub_clusters_var,
  sub_clusters_list,
  sample,
  use_min_cor_strategy = FALSE,
  true_proportions = NULL,
  delete_shared_level_markers = TRUE,
  delete_shared_internal_markers = TRUE,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  test.use.value = "wilcox",
  calculate_markers = "",
  marker_strategy = NULL,
  verbose = FALSE
)
```

Arguments

`bulk_data`	A numeric matrix representing the bulk data to be deconvoluted.
`single_cell_data_exp`	A list or data.frame representing the single cell data.
`top_clusters_var`	A string representing the variable in `single_cell_data_exp` used to indicate the top clusters.
`top_clusters_list`	A list or vector indicating the top clusters to be considered.
`sub_clusters_var`	A string representing the variable in `single_cell_data_exp` used to indicate the subclusters.
`sub_clusters_list`	A list or vector indicating the subclusters to be considered.
`sample`	A string or numeric value indicating the specific sample to use.

use_min_cor_strategy
: A logical value indicating whether to use the minimal correlation strategy. Default is FALSE.

true_proportions
: A numeric vector representing the true proportions of the clusters, if known. Default is NULL.

delete_shared_level_markers
: A logical value indicating whether to delete shared markers at the same level. Default is TRUE.

delete_shared_internal_markers
: A logical value indicating whether to delete shared markers within a group. Default is TRUE.

filter_markers
: A character string specifying the filtering strategy for markers. Default is NULL.

param.logfc.threshold
: A numeric value representing the log fold change threshold for marker genes. Default is 2.0.

param.p_val_adj
: A numeric value representing the adjusted p-value threshold for marker genes. Default is 0.05.

test.use.value
: A string specifying the test to use for marker gene identification. Default is 'wilcox'.

calculate_markers
: A character string specifying the strategy for marker gene calculation. Default is ".

marker_strategy
: A character string specifying the overall strategy for marker gene handling. Default is NULL.

verbose
: A logical value indicating whether to print detailed output during execution. Default is FALSE.

Value

A list containing the deconvoluted bulk data.

---

run_deconvolution_tree_guided_recursive

Run Deconvolution Tree Guided Recursively

Description

This function uses a tree-guided method to perform a recursive deconvolution operation on bulk and single-cell data. The recursive method helps in analyzing hierarchically structured data, usually encountered in biological systems where a set of cell types are hierarchically organized based on their molecular similarity. The method starts at the top level, identifies clusters, calculates markers, performs deconvolution, and propagates the information back to guide the deconvolution at the next level.

Usage

```
run_deconvolution_tree_guided_recursive(
  result_deco_top = NULL,
  bulk_data,
  true_proportions = NULL,
  single_cell_data_exp,
  sub_clusters_var,
  hierarchy,
  sample,
  use_min_cor_strategy = TRUE,
  delete_shared_level_markers = FALSE,
  delete_shared_internal_markers = FALSE,
  deconvolute_just_top = FALSE,
  deconvolute_top_hierarchy_limit = 1,
  filter_markers = NULL,
  param.logfc.threshold = 2,
  param.p_val_adj = 0.05,
  test.use.value = "wilcox",
  marker_strategy = NULL,
  verbose = FALSE
)
```

Arguments

`result_deco_top`
: A list representing the result of deconvolution at the top level of the hierarchy. If NULL (default), the function starts at the root of the tree.

`bulk_data`
: The bulk data on which the deconvolution needs to be performed.

`true_proportions`
: An optional matrix representing the true proportions of cell types. This can be used to calculate the accuracy of the deconvolution. If NULL (default), the accuracy calculation is skipped.

`single_cell_data_exp`
: A data frame representing single cell expression data, used to guide the deconvolution process.

`sub_clusters_var`
: A string representing the variable in single_cell_data_exp which indicates subclusters.

`hierarchy`
: A list structure describing the hierarchy of cell type clustering.

`sample`
: A string representing the sample name.

`use_min_cor_strategy`
: A boolean indicating whether to use the minimum correlation strategy during the marker calculation phase. Default is TRUE.

`delete_shared_level_markers`
: A boolean indicating whether to delete shared level markers during the marker calculation phase. Default is FALSE.

`delete_shared_internal_markers`
: A boolean indicating whether to delete shared internal markers during the marker calculation phase. Default is FALSE.

deconvolute_just_top
: A boolean indicating whether to perform deconvolution only at the top level. If TRUE (default is FALSE), the function stops after deconvoluting the top level and returns the result.

deconvolute_top_hierarchy_limit
: An integer indicating the top hierarchy limit for deconvolution. Default is 1.

filter_markers
: An optional list of markers to be used for filtering during the marker calculation phase. If NULL (default), no filtering is performed.

param.logfc.threshold
: A numeric value representing the log-fold-change threshold for marker calculation. Default is 2.0.

param.p_val_adj
: A numeric value representing the adjusted p-value threshold for marker calculation. Default is 0.05.

test.use.value
: A string indicating the statistical test to use for marker calculation. Default is 'wilcox'.

marker_strategy
: An optional string representing the marker selection strategy. If NULL (default), a default strategy is used.

verbose
: A boolean indicating whether to print detailed messages during the execution of the function. Default is FALSE.

Value

A list containing the result of the top level deconvolution, a list of deconvolution results at each level, and the back-propagated proportions at the top level.

---

run_grid_search *Run Grid Search*

Description

Performs a grid search in a parallelized manner over different values of alpha and beta in the Non-negative matrix factorization. The search is conducted over the combinations of given alpha and beta values.

Usage

```
run_grid_search(
  bulk_data_methylation,
  bulk_data_expression = NULL,
  data_expression_auxiliary = NULL,
  k,
  alpha_list = NULL,
  beta_list = NULL,
  delta_threshold = 1e-20,
  max_iterations = 200,
  print_limit = 100,
  threads = 0,
  proportion_constraint_h = TRUE,
```

```
  regularize_w = NULL,
  alpha_regularizer_w_list = NULL,
  fixed_w = NULL,
  fixed_h = NULL,
  fixed_a = NULL,
  fixed_b = NULL
)
```

Arguments

bulk_data_methylation	A data.frame, Bulk data methylation matrix.
bulk_data_expression	A data.frame, Bulk data expression matrix.
data_expression_auxiliary	A data.frame, Auxiliary expression data matrix.
k	An integer, Number of clusters.
alpha_list	A list of numerics, List of alpha values to be considered in the grid search. If NULL, no grid search is performed over alpha values.
beta_list	A list of numerics, List of beta values to be considered in the grid search. If NULL, no grid search is performed over beta values.
delta_threshold	A numeric, The threshold value for convergence. Default is 1e-20.
max_iterations	An integer, Maximum number of iterations for convergence. Default is 200.
print_limit	An integer, Limit for print statements. Default is 100.
threads	An integer, Number of CPU threads to be used. If 0, then it uses the total number of CPUs minus one. Default is 0.
proportion_constraint_h	A logical, Whether to apply proportion constraint on H matrix. Default is TRUE.
regularize_w	A logical, Whether to apply regularization on W matrix. If NULL, no regularization is applied.
alpha_regularizer_w_list	A list of numerics, List of alpha regularizer values to be considered in the grid search. If NULL, no grid search is performed over alpha regularizer values.
fixed_w	A matrix or NULL, Fixed matrix W. If NULL, it implies that W is not fixed.
fixed_h	A matrix or NULL, Fixed matrix H. If NULL, it implies that H is not fixed.
fixed_a	A matrix or NULL, Fixed matrix A. If NULL, it implies that A is not fixed.
fixed_b	A matrix or NULL, Fixed matrix B. If NULL, it implies that B is not fixed.

Details

Raises an error if gamma, alpha, and beta are all 0, indicating that the model cannot be executed. Suggests the user to switch to the more direct function run_deconvolution_multiple() if any two of gamma, alpha, beta are zero.

Value

A list of results from the grid search.

run_standard_deconvolution *Run standard deconvolution*

Description

This function is a wrapper for the MMNF library in python that is used for running a standard deconvolution process. This involves estimating the source distribution in a mixture model given some observed data.

Usage

```
run_standard_deconvolution(
  bulk_data_x,
  references_w,
  markers = NULL,
  max_iterations = 10000,
  verbose = FALSE
)
```

Arguments

bulk_data_x	A numeric vector, matrix, or data frame representing the bulk data for the deconvolution.
references_w	A numeric vector, matrix, or data frame representing the references for the deconvolution.
markers	A numeric vector or NULL. Represents the markers to be used in the deconvolution. By default, this is set to NULL.
max_iterations	A positive integer. Represents the maximum number of iterations for the deconvolution process. By default, this is set to 10000.
verbose	A boolean. If TRUE, the function will print additional details during the execution. By default, this is set to FALSE.

Value

A list containing the results of the deconvolution. This might include the estimated sources and the final weights depending on the implementation of MMNF library in python.

save_plot_as_pdf *Save Plot as PDF*

Description

This function saves a given plot object as a PDF file to a specific path.

Usage

```
save_plot_as_pdf(
  plot_object,
  file_name,
  width.parameter = NULL,
  height.parameter = NULL,
  path = NULL
)
```

Arguments

plot_object	The plot object to be saved.
file_name	String, the name of the file to save the plot as. It should not include the path or file extension.
width.parameter	Numeric, the width of the output PDF in inches. Optional, default is NULL.
height.parameter	Numeric, the height of the output PDF in inches. Optional, default is NULL.
path	String, the directory where the PDF should be saved. Optional, default is '/mnt_volumen/GIT_REPOS

Details

The function first checks whether a path was provided. If not, it uses a default path. Then, it opens a new PDF device, prints the 'plot_object' to this device, and then closes the device, which saves the plot to the specified file. The 'width.parameter' and 'height.parameter' arguments allow for control over the dimensions of the output PDF. If these are not provided, the PDF is created with the default dimensions.

Value

The function is used for its side effects of creating a PDF file and does not return anything.

Index