Package 'scMINER'

July 17, 2024

Type Package

Title Single-cell Mutual Information-based Network Enginerring Ranger

Version 1.1.0

Description
scMINER is a mutual information-based framework for single-cell/nucleus RNA-Seq data analysis. It enables the mutual information-based clustering, cell type-specific gene network reverse engineering and protein activity inference for not only transcriptional factors (TFs) but also signaling genes (SIGs) on a cell type-specific basis. scMINER can identify the hidden drivers underlying cellular lineage differentiation, tissue specification and beyond.

License Apache License (>= 2)

Encoding UTF-8

LazyData TRUE

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 4.3),
base (>= 4.3.3),
Biobase (>= 2.62.0),
Matrix (>= 1.6.5),
methods (>= 4.3.3),
stats (>= 4.3.3)

Imports anndata (>= 0.7.5.6),
dplyr (>= 1.1.4),
gridExtra (>= 2.3),
hdf5r (>= 1.3.10),
igraph (>= 2.0.3),
limma (>= 3.58.1),
pheatmap (>= 1.0.12),
reshape2 (>= 1.4.4),
rmarkdown (>= 2.27),
SuperCell (>= 1.0)

Suggests openxlsx (>= 4.2.5.2),
Seurat (>= 5.1.0),
testthat (>= 3.2.1.1)

R topics documented:

`addMICAoutput` *Add the MICA output (cluster labels and UMAP/tSNE coordinates) to sparse eset object*

Description

This function is used to add the clustering results by MICA into the sparse eset object. Two types of resluts would be added to the phenoData slot of sparse eset object:

- cluster ID of each cell
- dimension reduction coordinates of each cell, either `UMAP` ot `tSNE`.

Usage

```
addMICAoutput(input_eset, mica_output_file, visual_method = "umap")
```

Arguments

input_eset	The sparse eset object to add the MICA output into
mica_output_file	The .txt file generated by MICA. It includes 4 columns: "ID" (Cell ID), "X" (UMAP_1 or tSNE_1), "Y" (UMAP_2 or tSNE_2), "label" (ClusterID)
visual_method	Character, method used for visualizing the clustering results: umap (the default) or tsne.

Value

A sparse eset object with clustering results added.

Examples

```
clustered.eset <- addMICAoutput(input_eset = input_eset, mica_output_file = "/path-to-mica-input/clustering_U
```

---

combinePvalVector *Combine P values using Fisher's method or Stouffer's method*

Description

This function is used to calculate the combined P value and Z score from multiple P values. it can also be used to convert P value to Z score.

Usage

```
combinePvalVector(pvals, method = "Stouffer", signed = TRUE, twosided = TRUE)
```

Arguments

pvals	A vector of numeric, P values to be combined or converted to Z scores
method	Character, method used to combine P values: "Stouffer" (the default) or "Fisher".
signed	Logical, whether the input P values are signed or not. Usually they are signed by folder change. Default: TRUE.
twosided	Logical, whether the input P values are two-sided. If FALSE, the input P values will be treated as one-tailed. Default: TRUE.

Value

A vector containing "Z-statistics" and "P.Value".

Examples

```
## 1. combine P values from a vector
combinePvalVector(c(0.1,1e-3,1e-5), method = 'Stouffer')

## 2. combine P values from a table
df_Pcombined <- sapply(df$Pval*sign(df$log2FC), function(x) {combinePvalVector(x, twosided = TRUE)[2]})
df_Zscore <- sapply(df$Pval*sign(df$log2FC), function(x) {combinePvalVector(x, twosided = TRUE)[1]})
```

`combineSparseEset` *Combine multiple sparse expression set objects*

Description

This function is used to combine the sparse expression set objects. The combined eset object contains all cells and features of all input eset objects. If the eset objects are of different features, `NA` values will be generated and automatically imputed by the minimum value of the combined gene expression matrix.

Usage

```
combineSparseEset(
  eset_list,
  projectID = NULL,
  addPrefix = NULL,
  addSurfix = NULL,
  addMetaData = TRUE
)
```

Arguments

`eset_list`	A vector of sparse expression set objects to combine
`projectID`	A character vector or `NULL`, set the project names of the eset objects to combine. Default: `NULL`.
`addPrefix`	A character vector or `NULL`, add a **prefix** to the cell barcodes of each eset object to combine. It is highly recommended to use a prefix containing letters and/or numbers only, and not starting with numbers. Default: `NULL`.
`addSurfix`	A character vector or `NULL`, add a **surfix** to the cell barcodes of each eset object to combine. It is highly recommended to use a surfix containing letters and/or numbers only, and not starting with numbers. Default: `NULL`.
`addMetaData`	Logical, whether to update the meta data of cells and features after combination. Default: `TRUE`.

Value

A sparse eset object with combined features and cells of multiple eset objects.

Examples

```
combined.eset <- combineSparseEset(c(sample_1.eset, sample_2.eset, sample_3.eset), projectID = c("sample1", "
```

createSparseEset *Create a sparse expression set object from a data matrix*

Description

This function is used to create a pre-defined sparse expression set object from a data matrix of differnt classes: "dgCMatrix", "dgTMatrix", "dgeMatrix", "matrix", "data.frame". It allows the users to provide self-customized meta data for both cells (parameter cellData) and genes (parameter featureData). It can also generate the meta data for both automatically, if addMetaData = TRUE. The automatically generated meta data includes:

- **"nUMI"**: number of total UMIs in each cell, only valid when the values in data matrix are raw UMI counts;
- **"nFeature"**: number of expressed features/genes in each cell;
- **"pctMito"**: percentage of UMIs of mitochondrial genes (defined by "^mt-|^MT-") in each cell;
- **"pctSpikeIn"**: percentage of UMI of spike-in RNAs (defined by "^ERCC-|^Ercc-")) in each cell;
- **"nCell"**: number of cells that each feature/gene was identified in).

Usage

```
createSparseEset(
  input_matrix,
  do.sparseConversion = TRUE,
  cellData = NULL,
  featureData = NULL,
  projectID = NULL,
  addMetaData = TRUE
)
```

Arguments

input_matrix	A data matrix with Features/Genes as the rows and Cells as the columns. It should be one of: 'dgCMatrix', 'dgTMatrix', 'dgeMatrix', 'matrix', 'data.frame'.
do.sparseConversion	Logical, whether to convert the input_matrix to a sparse matrix if it's not. Default: TRUE.
cellData	A data frame containing meta data of cells or NULL. It's row.names should be consistent with the colnames of input_matrix. Default: NULL.
featureData	A data frame containing meata data of features or NULL. It's row.names should be consistent with the row.names of input_matrix. Default: NULL.
projectID	Character or NULL, the project name of the sparse eset object. Default: NULL.
addMetaData	Logical, whether to calculate and add extra statistics (a.k.a. meta data) to cells and features. Default: TRUE.

Value

A sparse eset object with three slot: 1) gene by cell matrix; 2) data frame of cell information; 3) data frame of feature/gene information.

Examples

```
expression_raw.eset <- createSparseEset(input_matrix = sparseMatrix, projectID = "demoSample", addMetaData = T
```

`drawNetworkQC` *Assess the quality of each network generated by SJARACNe*

Description

This function is used to assess the quality of networks generated by SJARACNe. It returns a summary table of key statistics of networks. The users can also generate the html quality control report by turning `generate_html = TRUE`.

Usage

```
drawNetworkQC(
  network_file = NULL,
  sjaracne_dir = NULL,
  directed = TRUE,
  weighted = TRUE,
  generate_html = TRUE,
  outdir = NULL,
  prefix = NULL
)
```

Arguments

`network_file`	The path to the network file (`consensus_network_ncol_.txt`) for quality control
`sjaracne_dir`	The path to the folder of SJARACNe runs. When this argument is given, scMINER will automatically retrieve all network files (`consensus_network_ncol_.txt`) in this folder and generate quality control report for all of them.
`directed`	Logical, whether the network is directed or not. All networks generated by SJARACNe are directed. Default: `TRUE`.
`weighted`	Logical, whether the edges of the network is weighted or not. All networks generated by SJARACNe are weighted by mutual information (`"MI"` column). Default: `TRUE`.
`generate_html`	Logical, whether to generate a html report. Default: `TRUE`.
`outdir`	Character or `NULL`, the path to save the html report. If `NULL`, the html report will be saved in the same folder of the network file. Default: NULL. Ignored if `generate_html = FALSE`.
`prefix`	Character or `NULL`, the character string to add in front of the html report file name. Default: `NULL`. Ignored if `generate_html = FALSE`.

Value

This function will print the statistics of several key quality metrics of network(s). If `generate_html` is set `TRUE`, it also generates a html file of quality control report and save it to the same folder of the network file (by default) or the folder specified by "`outdir`".

Examples

```
## 1. assess the quality of network from a network file
drawNetworkQC(network_file = ./SJARACNE/B/SIG/b100/consensus_network_ncol_.txt, generate_htmal = TRUE) # the
drawNetworkQC(network_file = ./SJARACNE/B/SIG/b100/consensus_network_ncol_.txt, generate_htmal = TRUE, outdi
drawNetworkQC(network_file = ./SJARACNE/B/SIG/b100/consensus_network_ncol_.txt, generate_htmal = TRUE, prefi

## 2. assess the qulity of network from a directory
drawNetworkQC(network_dir = ./SJARACNE, generate_htmal = TRUE) # the html file will be saved to the same folder
```

`drawSparseEsetQC` *Generate a quality control report from sparse eset object*

Description

This function is used to generate a html quality control report from a sparse eset object. Compared with the summary table return by filterSparseEset(), the output report contains more comprehensive and detailed QC results and can be used to estimate the cutoffs to filter the eset object. It also contains some plots for presentation purpose.

Usage

```
drawSparseEsetQC(input_eset, output_html_file, overwrite = FALSE, group = NULL)
```

Arguments

`input_eset`	The sparse eset object for quality control analysis
`output_html_file`	The path of the output .html file
`overwrite`	Logical, whether to overwrite the output .html file if it already exists. Default: `FALSE`.
`group`	Character or `NULL`, Name of column in pData(eset) used for grouping. Default: `NULL`.

Value

A html-formatted quality control report of sparse eset object

Examples

```
drawSparseEsetQC(input_eset, output_html_file = "./QC/esetQCreport.html", overwrite = FALSE, group = "projectI
```

`draw_barplot` *Bar plot showing the cell composition of self-defined groups*

Description

This function is used to draw a bar plot showing the cell composition of self-defined groups.

Usage

```
draw_barplot(
  input_eset,
  group_by = "clusterID",
  color_by = "cell_type",
  colors = NULL,
  legend.position = "right",
  xlabel.angel = 0,
  fontsize.legend_title = 12,
  fontsize.legend_text = 10,
  fontsize.axis_title = 12,
  fontsize.axis_text = 10
)
```

Arguments

`input_eset`	The expression set object that filtered, normalized and log-transformed
`group_by`	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "`clusterID`".
`color_by`	Character, name of the column for color-coding, usually the column of cell types or clusters. Default: "`cell_type`".
`colors`	A vector of colors for filling the violins. The length should be same as the number of groups. Default: `NULL` (ggplot default colors).
`legend.position`	Character, position of legend: "`right`" (the default), "`left`", "`top`", "`bottom`" or "`none`".
`xlabel.angel`	Numeric, the angel of the a-axis title. When it's set not 0, the x-axis text will automatically right-justified. Default: 0.
`fontsize.legend_title`	Integer, font size of the legend title. Default: 10.
`fontsize.legend_text`	Integer, font size of the legend text. Default: 8.
`fontsize.axis_title`	Integer, font size of the axis label and text. Default: 10.
`fontsize.axis_text`	Integer, font size of the axis label and text. Default: 8.

Value

A ggplot object that can be visualized by "`p`" or `ggsave(file = "output.pdf", p)`

Examples

```
## 1. bar plot grouped by clusters ("clusterID") and colored by true labels ("true_label)
p_bar <- draw_barplot(input_eset = clustered.eset, group_by = "clusterID", color_by = "true_label")

## 2. customize the colors
p_bar <- draw_barplot(input_eset = clustered.eset, group_by = "clusterID", color_by = "true_label", colors = c(
```

`draw_bubbleplot` *Bubble plot showing the signature scores by self-defined groups*

Description

This function is used to draw a bubble plot of signature scores among self-defined groups.

Usage

```
draw_bubbleplot(
  input_eset,
  signature_table = NULL,
  group_by = "clusterID",
  colors = NULL,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8,
  xlabel.angel = 0
)
```

Arguments

`input_eset`	The expression set object that filtered, normalized and log-transformed
`signature_table`	A matrix or data frame containing three columns: signature_name, signature_feature, weight. Default: `NULL`.
`group_by`	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "`clusterID`".
`colors`	A vector of two colors indicating the low and high values respectively. Default: c("`lightgrey`", "`red`").
`legend.position`	Character, position of legend: "`right`" (the default), "`left`", "`top`", "`bottom`" or "`none`".
`fontsize.legend_title`	Integer, font size of the legend title. Default: 10.
`fontsize.legend_text`	Integer, font size of the legend text. Default: 8.
`fontsize.axis_title`	Integer, font size of the axis label and text. Default: 10.
`fontsize.axis_text`	Integer, font size of the axis label and text. Default: 8.
`xlabel.angel`	Numeric, the angel of the a-axis title. When it's set not 0, the x-axis text will automatically right-justified. Default: 0.

Value

A ggplot object of bubble plot

Examples

```
marker_file <- system.file('PBMC14KDS_DemoDataSet/DATA/', 'Immune_signatures.xlsx', package = "scMINER")
signature_table <- openxlsx::read.xlsx(marker_file)
head(signature_table)
## 1. the most commonly used command
p_bubbleplot <- draw_bubbleplot(input_eset = clustered.eset, signature_table = signature_table, group_by = "cl

## 2. customize the colors
p_bubbleplot <- draw_bubbleplot(input_eset = clustered.eset, signature_table = signature_table, group_by = "cl
```

---

feature_boxplot *Box plot showing the expression or activity of selected features by self-defined groups*

Description

This function is used to draw a box plot of selected features among self-defined groups from a sparse eset object.

Usage

```
feature_boxplot(
  input_eset,
  features = NULL,
  group_by = "clusterID",
  ncol = 3,
  colors = NULL,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.strip = 10,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8,
  xlabel.angel = 0,
  ylabel.text = "Expression (log2CPM)",
  stat_method = "median",
  add_jitter = FALSE,
  jitter.height = 0,
  jitter.width = 0.3,
  jitter.size = 0.1
)
```

Arguments

input_eset	The expression set object that filtered, normalized and log-transformed
features	A vector of genes or drivers (row.names of the input eset) to plot

group_by Character, name of the column for grouping, usually the column of cell types or clusters. Default: "clusterID".

ncol Integer, number of columns when multiple plots are displayed. Default: 3.

colors A vector of colors for filling the violins. The length should be same as the number of groups. Default: NULL (ggplot default colors).

legend.position
Character, position of legend: "right" (the default), "left", "top", "bottom" or "none".

fontsize.legend_title
Integer, font size of the legend title. Default: 10.

fontsize.legend_text
Integer, font size of the legend text. Default: 8.

fontsize.strip Integer, font size of the plot strip. Default: 10.

fontsize.axis_title
Integer, font size of the axis label and text. Default: 10.

fontsize.axis_text
Integer, font size of the axis label and text. Default: 8.

xlabel.angel Numeric, the angel of the a-axis title. When it's set not 0, the x-axis text will automatically right-justified. Default: 0.

ylabel.text Character, the title of y-axis. Default: "Expression (log2CPM)"

stat_method Character or NULL. method of the stat point to show: "median" (the default), "mean". If NULL, the stat point won't show up.

add_jitter Logical, whether to add jittered points. Default: FALSE.

jitter.height Numeric, amount of vertical jitter. Default: 0.

jitter.width Numeric, amount of horizontal jitter. Default: 0.3.

jitter.size Numeric, size of the jittered points. Default: 0.1.

Value

A ggplot object with one or multiple box plots

Examples

```
## 1. violin plots grouped by clusters (say the column name is 'clusterID')
p_box <- feature_boxplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 2. violin plots grouped by cell types (say the column name is 'cellType')
p_box <- feature_boxplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cellTyp

## 3. customize the colors to fill the violin plots
p_box <- feature_boxplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 4. add jittered points
p_box <- feature_boxplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 5. using activity data
p_box <- feature_boxplot(input_eset = activity_clustered.eset, features = c("CD14_SIG", "CD19_SIG", "CD8A_SIG"
```

`feature_bubbleplot` *Bubble blot showing the expression or activity of selected features by self-defined groups*

Description

This function is used to draw a bubble plot of selected features among self-defined groups from a sparse eset object.

Usage

```
feature_bubbleplot(
  input_eset,
  features = NULL,
  group_by = "clusterID",
  colors = NULL,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8,
  xlabel.angel = 0
)
```

Arguments

`input_eset`	The expression set object that filtered, normalized and log-transformed
`features`	A vector of genes or drivers (row.names of the input eset) to plot
`group_by`	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "`clusterID`".
`colors`	A vector of two colors indicating the low and high values respectively. Default: c("`lightgrey`", "`red`").
`legend.position`	Character, position of legend: "`right`" (the default), "`left`", "`top`", "`bottom`" or "`none`".
`fontsize.legend_title`	Integer, font size of the legend title. Default: 10.
`fontsize.legend_text`	Integer, font size of the legend text. Default: 8.
`fontsize.axis_title`	Integer, font size of the axis label and text. Default: 10.
`fontsize.axis_text`	Integer, font size of the axis label and text. Default: 8.
`xlabel.angel`	Numeric, the angel of the a-axis title. When it's set not 0, the x-axis text will automatically right-justified. Default: 0.

Value

A ggplot object of bubble plot

Examples

```
features_of_interest <- c("CD3D","CD27","IL7R","SELL","CCR7","IL32","GZMA","GZMK","DUSP2","CD8A","GZMH","GZ
## 1. the most commonly used command
p_bubble <- feature_bubbleplot(input_eset = clustered.eset, features = features_of_interest, group_by = "clust

## 2. customize the colors
p_bubble <- feature_bubbleplot(input_eset = clustered.eset, features = features_of_interest, group_by = "clust
```

`feature_heatmap` *Heatmap showing the expression or activity of selected features by self-defined groups*

Description

This function is used to draw a heatmap of selected features among self-defined groups from a sparse eset object. By default, the groups are sorted by size, from largest to smallest. Within each group, the cells are sorted alphabetically.

Usage

```
feature_heatmap(
  input_eset,
  features = NULL,
  group_by = "clusterID",
  scale_method = "none",
  annotation_columns = NULL,
  use_gaps.column = FALSE,
  cluster_rows = FALSE,
  show_rownames = TRUE,
  fontsize.row = 10,
  use_gaps.row = FALSE
)
```

Arguments

`input_eset`	The expression set object that filtered, normalized and log-transformed
`features`	A vector of genes or drivers (row.names of the input eset) to plot
`group_by`	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "`clusterID`".
`scale_method`	Character, method for data scaling: "`none`" (the default), "`column`", "`row`".
`annotation_columns`	Character, name(s) of the column(s) to add for cell annotation. Default: `NULL`.
`use_gaps.column`	Logical, whether to put a gap between cell groups. Default: `FALSE`.
`cluster_rows`	Logical, whether to cluster the rows. If `TRUE`, the rows will be clustered. If `FALSE`, the rows are displays following the order in '`features`'. Default: `FALSE`.
`show_rownames`	Logical, whether to show the rownames. Default: `TRUE`.
`fontsize.row`	Numeric, font size of the rownames. Defualt: 10.
`use_gaps.row`	Logical, whether to put a gap between features. Default: `FALSE`.

Value

Print the heatmap to screen

Examples

```
features_of_interest <- c("CD3D","CD27","IL7R","SELL","CCR7","IL32","GZMA","GZMK","DUSP2","CD8A","GZMH","GZ
## 1. the most commonly used command
feature_heatmap(input_eset = clustered.eset, features = features_of_interest, group_by = "clusterID")

## 2. add one more column ('true_label') for cell annotation
feature_heatmap(input_eset = clustered.eset, features = features_of_interest, group_by = "clusterID", annotati

## 3. scale the data by row
feature_heatmap(input_eset = clustered.eset, features = features_of_interest, group_by = "clusterID", scale_me

## 4. cluster the rows
feature_heatmap(input_eset = clustered.eset, features = features_of_interest, group_by = "clusterID", cluster_

## 5. add gaps
feature_heatmap(input_eset = clustered.eset, features = features_of_interest, group_by = "clusterID", use_gaps
```

---

feature_scatterplot *Scatter plot showing the expression or activity of selected features on UMAP or t-SNE coordinates*

Description

This function is used to draw a scatter plot of selected features on UMAP or t-SNE coordinates from a sparse eset object.

Usage

```
feature_scatterplot(
  input_eset,
  features = NULL,
  location_x = "UMAP_1",
  location_y = "UMAP_2",
  colors = NULL,
  ncol = 3,
  point.size = 0.5,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.strip = 10,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8
)
```

Arguments

input_eset	The expression set object that filtered, normalized and log-transformed
features	A vector of genes or drivers (row.names of the input eset) to plot

location_x Character, name of the column of x-axis coordinates. Default: "UMAP_1".

location_y Character, name of the column of y-axis coordinates. Default: "UMAP_2".

colors A vector of two colors indicating the low and high values respectively. Default: c("lightgrey", "red").

ncol Integer, number of columns when multiple plots are displayed. Default: 3.

point.size Numeric, size of the scatter points. Default: 0.5.

legend.position
Character, position of legend: "right" (the default), "left", "top", "bottom" or "none".

fontsize.legend_title
Integer, font size of the legend title. Default: 10.

fontsize.legend_text
Integer, font size of the legend text. Default: 8.

fontsize.strip Integer, font size of the plot strip. Default: 10.

fontsize.axis_title
Integer, font size of the axis label and text. Default: 10.

fontsize.axis_text
Integer, font size of the axis label and text. Default: 8.

Value

Print a plot to screen and return a gtable containing a list of plots, can be visualized by plot(g), and saved by ggsave(file = "output.pdf", g)

Examples

```
## 1. scatter plots with UMAP projections
p_scatter <- feature_scatterplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), location_x

## 2. scatter plots with t-SNE projections
p_scatter <- feature_scatterplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), location_x

## 3. change the point size and font size
p_scatter <- feature_scatterplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), location_x
```

---

feature_vlnplot *Violin plot showing the expression or activity of selected features by self-defined groups*

Description

This function is used to draw a violin plot of selected features among self-defined groups from a sparse eset object.

Usage

```
feature_vlnplot(
  input_eset,
  features = NULL,
  group_by = "clusterID",
  ncol = 3,
  colors = NULL,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.strip = 10,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8,
  xlabel.angel = 0,
  ylabel.text = "Expression (log2CPM)",
  stat_method = "median",
  add_boxplot = FALSE,
  boxplot.width = 0.3,
  boxplot.fill = "white",
  boxplot.alpha = 0.8,
  add_jitter = FALSE,
  jitter.height = 0,
  jitter.width = 0.3,
  jitter.size = 0.1
)
```

Arguments

input_eset	The expression set object that filtered, normalized and log-transformed
features	A vector of genes or drivers (row.names of the input eset) to plot
group_by	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "clusterID".
ncol	Integer, number of columns when multiple plots are displayed. Default: 3.
colors	A vector of colors for filling the violins. The length should be same as the number of groups. Default: NULL (ggplot default colors).
legend.position	Character, position of legend: "right" (the default), "left", "top", "bottom" or "none".
fontsize.legend_title	Integer, font size of the legend title. Default: 10.
fontsize.legend_text	Integer, font size of the legend text. Default: 8.
fontsize.strip	Integer, font size of the plot strip. Default: 10.
fontsize.axis_title	Integer, font size of the axis title. Default: 10.
fontsize.axis_text	Integer, font size of the axis text. Default: 8.
xlabel.angel	Numeric, the angel of the a-axis title. When it's set not 0, the x-axis text will automatically right-justified. Default: 0.

ylabel.text	Character, the title of y-axis. Default: "Expression (log2CPM)"
stat_method	Character or NULL. method of the stat point to show: "median" (the default), "mean". If NULL, the stat point won't show up.
add_boxplot	Logical, whether to add box plot. Default: FALSE.
boxplot.width	Numeric, width of the box plot relative to the body of violin plot, ranging from 0 to 1. Default: 0.3. Ignored if add_boxplot = FALSE.
boxplot.fill	Character, color used to fill the box plots. Default: "white". Ignored if add_boxplot = FALSE.
boxplot.alpha	Numerical, transparency of box plots, ranging from 0 (more transparent) to 1 (less transparent). Default: 0.8. Ignored if add_boxplot = FALSE.
add_jitter	Logical, whether to add jittered points. Default: FALSE.
jitter.height	Numeric, amount of vertical jitter. Default: 0.
jitter.width	Numeric, amount of horizontal jitter. Default: 0.3.
jitter.size	Numeric, size of the jittered points. Default: 0.1.

Value

A ggplot object with one or multiple violin plots

Examples

```
## 1. violin plots grouped by clusters (say the column name is 'clusterID')
p_vln <- feature_vlnplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 2. violin plots grouped by cell types (say the column name is 'cellType')
p_vln <- feature_vlnplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cellTyp

## 3. customize the colors to fill the violin plots
p_vln <- feature_vlnplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 4. add jittered points
p_vln <- feature_vlnplot(input_eset = clustered.eset, features = c("CD14", "CD19", "CD8A"), group_by = "cluster

## 5. using activity data
p_vln <- feature_vlnplot(input_eset = activity_clustered.eset, features = c("CD14_SIG", "CD19_SIG", "CD8A_SIG"
```

---

filterSparseEset *Filter the cells and/or features of sparse eset object using automatic or self-customized cutoffs*

Description

This function is used to remove the cells and features of low quality. It provides two modes to define the cutoffs:

- **"auto"**: in this mode, scMINER will estimate the cutoffs based on Median ± 3*MAD (maximum absolute deviation). This mode works well for the matrix of raw UMI counts or TPM (Transcripts Per Million) values.
- **"manual"**: in this mode, the users can manually specify the cutoffs, both low and high, of all 5 metrics: **nUMI**, **nFeature**, **pctMito**, **pctSpikeIn** for cells, and **nCell** for genes. No cells or features would be removed under the default cutoffs of each metrics.

Usage

```
filterSparseEset(
  eset,
  mode = "auto",
  gene.nCell_min = 1,
  gene.nCell_max = Inf,
  cell.nUMI_min = 1,
  cell.nUMI_max = Inf,
  cell.nFeature_min = 1,
  cell.nFeature_max = Inf,
  cell.pctMito_min = 0,
  cell.pctMito_max = 1,
  cell.pctSpikeIn_min = 0,
  cell.pctSpikeIn_max = 1
)
```

Arguments

`eset`	The sparse eset object to be filtered
`mode`	Character, mode to apply the filtration cutoffs: **"auto"** (the default) or **"manual"**.
`gene.nCell_min`	Numeric, the minimum number of cells that the qualified genes are identified in. Default: 1.
`gene.nCell_max`	Numeric, the maximum number of cells that the qualified genes are identified in. Default: Inf.
`cell.nUMI_min`	Numeric, the minimum number of total UMI counts per cell that the qualified cells carry. Default: 1.
`cell.nUMI_max`	Numeric, the maximum number of total UMI counts per cell that the qualified cells carry. Default: Inf.
`cell.nFeature_min`	Numeric, the minimum number of non-zero Features per cell that the qualified cells carry. Default: 1.
`cell.nFeature_max`	Numeric, the maximum number of non-zero Features per cell that the qualified cells carry. Default: Inf.
`cell.pctMito_min`	Numeric, the minimum percentage of UMI counts of mitochondrial genes that the qualified cells carry. Default: 0.
`cell.pctMito_max`	Numeric, the maximum percentage of UMI counts of mitochondrial genes that the qualified cells carry. Default: 1.
`cell.pctSpikeIn_min`	Numeric, the minimum percentage of UMI counts of spike-in that the qualified cells carry. Default: 0.
`cell.pctSpikeIn_max`	Numeric, the maximum percentage of UMI counts of spike-in that the qualified cells carry. Default: 1.

Value

A filtered sparse eset object. It also prints the summary of filtration to the screen.

Examples

```
filtered.eset <- filterSparseEset(raw.eset) ## filter the input eset using the cutoffs calculated by scMINER.
filtered.eset <- filterSparseEset(raw.eset, gene.nCell_min = 10, cell.nUMI_min = 500, cell.nFeature_min = 100,
```

`generateMICAinput` *Generate the standard input files for MICA from sparse eset object*

Description

This function is used to generate the standard input files for MICA (Mutual Information-based Clustering Analysis) from a sparse eset object. It supports two file formats, "**.txt**" or "**.h5ad**". To generate a "**.h5ad**" file, the "**anndata**" package is required.

Usage

```
generateMICAinput(
  input_eset,
  output_file,
  overwrite = F,
  downSample_N = NULL,
  seed = 1
)
```

Arguments

`input_eset`	The sparse eset object to generate MICA input from. It must be normalized and log-transformed.
`output_file`	The output file, or MICA input file. Should be in either "**.txt**" or "**.h5ad**" format.
`overwrite`	Logical, whether to overwrite the output_file if it already exists. Default: `FALSE`.
`downSample_N`	A non-negative integer or `NULL`, number of cells to downsample to. Default: `NULL`.
`seed`	Integer or `NULL`, the seed for sampling. Default: 1. Ignored if `downSample_N = NULL`.

Value

A .txt or .h5ad file that can be used as the MICA input

Examples

```
generateMICAinput(input_eset = log2cpm.ese, output_file = "./MICA/micaInput.txt")
```

`generatePortalInputs` *Prepare the standard input files for scMINER Portal*

Description

This function is used to generated the standard input files that can be directly uploaded to scMINER Portal for visualization and beyond.

Usage

```
generatePortalInputs(
  input_expression.eset = NULL,
  input_expression.seuratObj = NULL,
  group_by = NULL,
  input_activity.eset = NULL,
  input_network.dir = NULL,
  input_network.table = NULL,
  output_dir = NULL
)
```

Arguments

`input_expression.eset`	A eset object of expression data which has been filerted, normalized and log-transformed. Default: `NULL`.
`input_expression.seuratObj`	A Seurat object with reduction results (umap and/or tsne). Default: `NULL`.
`group_by`	Character, name of the column in phenoData of eset object or meta.data of Seurat Obj that defines the groups for colorcoding in scMINER Portal. Default: `NULL`.
`input_activity.eset`	A eset object of activity data calculated by scMINER. Default: `NULL`.
`input_network.dir`	Character, the path to the SJARACNe directory. Default: `NULL`.
`input_network.table`	A table of network information, it contains at least three column: `"CellGroup"` (name of groups), `"NetworkType"` (type of network, TF or SIG) and `"NetworkFile"` (path to network files). Default: `NULL`.
`output_dir`	Character, the path to the output directory. Default: `NULL`.

Value

This function generated 1-3 standard files which can be uploaded to scMINER Portal directly.

Examples

```
## 1. the most commonly used command
generatePortalInputs(input_expression.eset = expression_clustered.eset, group_by = "cellType", input_activity

## 2. prepare expression data from Seurat object ("pbmc14.obj")
generatePortalInputs(input_expression.seuratObj = pbmc14.obj, output_dir = "./path-to-output_dir")
```

```
## 3. prepare network data from a table
network.table <- data.frame(CellGroup = c("CD4__CD25_T_Reg", "CD4__CD25_T_Reg", "CD19__B", "CD19__B"),Network
                  NetworkFile = c("./sjaracne/CD4__CD25_T_Reg/SIG/b100_pce-3/sjaracne_workflow-1474c41b-067
                                  "./sjaracne/CD4__CD25_T_Reg/TF/b100_pce-3/sjaracne_workflow-a93cd6db-7253-4ff
                                  "./sjaracne/CD19__B/SIG/sjaracne_workflow-da0c3c72-7afb-44fa-973b-e4d767e20b6
                                  "./sjaracne/CD19__B/TF/sjaracne_workflow-0426ea12-10bf-428c-b199-d5bd1a7aab5f
generatePortalInputs(input_expression.eset = expression_clustered.eset, group_by = "cellType", input_network.
```

`generateSJARACNeInput` *Generate the standard input files for SJARACNe from sparse eset object*

Description

This function is used to generate the standard input files for SJARACNe, a scalable software tool for gene network reverse engineering from big data.

Usage

```
generateSJARACNeInput(
  input_eset,
  group_name = "clusterID",
  group_name.refine = FALSE,
  sjaracne_dir,
  species_type = "hg",
  driver_type = "TF_SIG",
  customDriver_TF = NULL,
  customDriver_SIG = NULL,
  downSample_N = 1000,
  seed = 123,
  superCell_N = NULL,
  superCell_count = 100,
  superCell_gamma = 10,
  superCell_knn = 5,
  superCell_nHVG = 1000,
  superCell_nPC = 10,
  superCell_save = TRUE,
  print_command = FALSE,
  save_command = TRUE
)
```

Arguments

`input_eset`	The expression set object that filtered, normalized and log-transformed
`group_name`	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "`clusterID`".
`group_name.refine`	Logical, whether to replace the non-word characters in group names with underscore symbol ("_"). The improper filename characters may cause troubles, since scMINER creates a folder for each group using the group names. Set this argument to `TRUE` can help avoid this issue. Default: `FALSE`.

sjaracne_dir	The path to the folder for SJARACNe runs. Both the inputs and outputs will be saved here.
species_type	Character, species of the pre-defined driver list to use: "hg" for human or "mm" for mouse. Default: hg.
driver_type	Character, type of the pre-defined driver list to use: "TF" for transcriptional factors only, "SIG" for signaling genes only, or "TF_SIG" for both. Default: "TF_SIG".
customDriver_TF	A character vector or NULL, genes used to replace the pre-defined transcriptional factor driver list. This allows the user to customize the TF driver list. Default: NULL.
customDriver_SIG	A character vector or NULL, genes used to replace the pre-defined signaling gene driver list. This allows the user to customize the SIG driver list. Default: NULL.
downSample_N	Integer or NULL, if an integer is given, the groups with more cells than this integer will be down-sampled to this integer. A number between 500 to 3000 gives a good balance between robustness and computational efficiency. If NULL, the downsampling would be skipped. Default: 1000.
seed	Non-negative integer, seed of random sampling. Default: 123.
superCell_N	Integer or NULL, if an integer is given, the metacell method would be performed by SuperCell package to the groups with more cells than this integer. If NULL, no metacell method would be done. Default: NULL.
superCell_count	Integer, number of metacells to generate by SuperCell. Default: 100. Ignored if superCell_N = NULL.
superCell_gamma	Integer, graining level of data by SuperCell (proportion of number of single cells in the initial dataset to the number of metacells in the final dataset). Default: 10. Ignored if superCell_N = NULL.
superCell_knn	Integer, the k value to compute single-cell kNN network by SuperCell. Default: 5. Ignored if superCell_N = NULL.
superCell_nHVG	Integer, number of genes with the largest variation to use by SuperCell. Default: 1000. Ignored if superCell_N = NULL.
superCell_nPC	Integer, number of principal components to use for construction of single-cell kNN network by SuperCell.Default: 10. Ignored if superCell_N = NULL.
superCell_save	Logical, whether to save the results generated by SuperCell, including membership and other components. Default: TRUE. Ignored if superCell_N = NULL.
print_command	Logical, whether to print the command to run SJARACNe to screen. Default: FALSE.
save_command	Logical, whether to save the command to run SJARACNe. Default: TRUE.

Value

This function will generate several folders and files in the directory specified by "sjaracne_dir":

1. a folder for each group in the column specified by "group_name";
2. In each folder:
   - a ".exp.txt" file: expression matrix, features by cells.
   - a "TF" folder containing a ".tf.txt" file: this file contains the TF driver list.

- a "SIG" folder containing a ".sig.txt" file: this file contains the SIG driver list.
- a bash script (runSJARACNe.sh) to run SJARACNe. Further modification is needed to run it.
- a json file (config_cwlexec.json) containing parameters to run SJARACNe.

Examples

```
## 1. The most commonly used command: pre-defined driver lists, automatic down-sampling, no metacell method
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe

## 2. to disable the downsampling
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe

## 3. Use the customized driver list: TUBB4A is the gene of interest but currently not in the pre-defined driver
hg_driver <- getDriverList(species_type = "hg", driver_type = "TF_SIG")
"TUBB4A" %in% hg_driver # It would returen FALSE if TUBB4A is not in the pre-defined driver lists
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe

## 4. Use the metacell method
generateSJARACNeInput(input_eset = normalized.eset, group_name = "cell_type", sjaracne_dir = "./SJARACNe", spe
```

---

getActivity_inBatch *Calculate driver activities in batch from the SJARACNe directory*

Description

This function is used to calculate the driver activities of multiple groups from a scMINER directory. To calculate driver activities of one single group, please use getActivity_individual().

Usage

```
getActivity_inBatch(
  input_eset,
  sjaracne_dir,
  group_name,
  group_exclude = NULL,
  network_tag.tf = NULL,
  network_tag.sig = NULL,
  driver_type = "TF_SIG",
  activity_method = "mean",
  do.z_normalization = TRUE
)
```

Arguments

input_eset	The expression set object which has been filtered, normalized and log-transformed
sjaracne_dir	The path to the SJARACNe directory
group_name	Character, name of the column for grouping, usually the column of cell types or clusters

group_exclude A vector of group names to exclude in activity calculation

network_tag.tf Character or NULL, the tag used to distinguish different SJARACNe runs from the same input files. This is usually the name of the folder(s) in "TF" folder of each group. Default: NULL.

network_tag.sig Character or NULL, the tag used to distinguish different SJARACNe runs from the same input files. This is usually the name of the folder(s) in "SIG" folder of each group. Default: NULL.

driver_type Character, type of the pre-defined driver list to use. Should be one of: "TF" for transcriptional factors only, "SIG" for signaling genes only, or "TF_SIG" for both. Default: "TF_SIG".

activity_method Character, method used to calculate the activity: "mean" (the default), "weightedmean", "absmean" or "maxmean".

do.z_normalization Logical, whether to do the z-normalization on the gene expression values in each sample. Set if to FALSE only when the expression values has been scaled in each cell. Default: TRUE.

Value

A expression set object. The assayData is the activity matrix of all cells of all groups, drivers by cells. The phenoData and featureData are exactly save with the input eset.

Examples

```
## 1. when no tag was used in runing SJARACNE: the network file folder ("sjaracne_workflow-*") is directly under
activity.eset <- getActivity_inBatch(input_eset = normalized.eset, sjaracne_dir = "./SJARACNe", group_name = "

## 2. when tag (e.g. "bs_100" ) was used: the nework file folder ("sjaracne_workflow-*") is directly under a subf
activity.eset <- getActivity_inBatch(input_eset = normalized.eset, sjaracne_dir = "./SJARACNe", group_name = "

## 3. to calculate the activities of TF only
activity.eset <- getActivity_inBatch(input_eset = normalized.eset, sjaracne_dir = "./SJARACNe", group_name = "

## 4. to exclude some groups in the activity calculation (e.g. "NK" and "Monocyte")
activity.eset <- getActivity_inBatch(input_eset = normalized.eset, sjaracne_dir = "./SJARACNe", group_name = "

## 5. when calculate the activities from the gene expression values scaled by other methods (e.g. ScaleData() fr
activity.eset <- getActivity_inBatch(input_eset = normalized.eset, sjaracne_dir = "./SJARACNe", group_name = "
```

---

getActivity_individual

Calculate driver activities per group from network files

Description

This function is used to calculate the driver activities of one single group from the sparse eset obj and networks generated by SJARACNe. To calculate driver activities of multiple groups from a scMINER directory, please use getActivity_inBatch().

Usage

```
getActivity_individual(
  input_eset,
  network_file.tf = NULL,
  network_file.sig = NULL,
  driver_type = "TF_SIG",
  activity_method = "mean",
  do.z_normalization = TRUE
)
```

Arguments

input_eset	The group-specific expression set object which has been filtered, normalized and log-transformed
network_file.tf	The path to the TF network file generated by SJARACNe
network_file.sig	The path to the SIG network file generated by SJARACNe
driver_type	Character, type of the pre-defined driver list to use: "TF" for transcriptional factors only, "SIG" for signaling genes only, or "TF_SIG" for both. Default: "TF_SIG".
activity_method	Character, method used to calculate the activity: "mean" (the default), "weightedmean", "absmean" or "maxmean".
do.z_normalization	Logical, whether to do the z-normalization on the gene expression values in each sample. Set if to FALSE only when the expression values has been scaled in each cell. Default: TRUE.

Value

A expression set object of the group-of-interest. The assayData is the activity matrix of all cells of all groups, drivers by cells. The phenoData and featureData are exactly save with the input eset.

Examples

```
## 1. when use the eset with all groups
activity_group.eset <- getActivity_individual(input_est = normalized.eset[, pData(normalized.eset)$cell_type

## 2. when the group-specific eset is available
activity_group.eset <- getActivity_individual(input_est = group_specific.est, network_file.tf = "./TF/tag/sja
```

---

getDA *Perform differential activity analysis on expression set*

Description

Perform differential activity analysis on expression set

Usage

```
getDA(
  input_eset,
  group_by = "clusterID",
  g1 = NULL,
  g0 = NULL,
  use_method = "t.test"
)
```

Arguments

input_eset	The expression set object that filtered, normalized and log-transformed
group_by	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "clusterID".
g1	A vector of character defining the fore-ground group or NULL. Default: NULL.
g0	A vector of character defining the back-ground group or NULL. Default: NULL.
use_method	Character, method used for differential analysis: "limma", "wilcoxon", and "t.test" (the default).

Value

A data frame. Rows are genes/drivers, and columns are 11 statistics of differential analysis.

Examples

```
## 1. To perform differential activity analysis in a 1-vs-rest manner for all groups in "clusterID" column
da_res <- getDA(input_eset = activity_clustered.eset, group_by = "clusterID", use_method = "t.test")

## 2. To perform differential activity analysis in a 1-vs-rest manner for one specific group in "clusterID" colu
da_res <- getDA(input_eset = activity_clustered.eset, group_by = "clusterID", g1 = c("1"), use_method = "t.test

## 3. To perform differential activity analysis in a rest-vs-1 manner for one specific group in "clusterID" colu
da_res <- getDA(input_eset = activity_clustered.eset, group_by = "clusterID", g0 = c("1"), use_method = "t.test

## 4. To perform differential activity analysis in a 1-vs-1 manner for groups in "clusterID" column
da_res <- getDA(input_eset = activity_clustered.eset, group_by = "clusterID", g1 = c("1"), g0 = c("3"), use_met
```

---

getDE *Perform differential expression analysis on expression set object*

Description

This function is used to perform the differential expression analysis on sparse eset object. It supports there methods: "limma", "wilcoxon", and "t.test".

Usage

```
getDE(
  input_eset,
  group_by = "clusterID",
  g1 = NULL,
  g0 = NULL,
  use_method = "limmma"
)
```

Arguments

input_eset	The expression set object that filtered, normalized and log-transformed
group_by	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "clusterID".
g1	A vector of character defining the fore-ground group or NULL. Default: NULL.
g0	A vector of character defining the back-ground group or NULL. Default: NULL.
use_method	Character, method used for differential analysis: "limma" (the default), "wilcoxon", and "t.test".

Value

A data frame. Rows are genes/drivers, and columns are 11 statistics of differential analysis.

Examples

```
## 1. To perform differential expression analysis in a 1-vs-rest manner for all groups in "clusterID" column
de_res <- getDE(input_eset = clustered.eset, group_by = "clusterID", use_method = "limma")

## 2. To perform differential expression analysis in a 1-vs-rest manner for one specific group in "clusterID" co
de_res <- getDE(input_eset = clustered.eset, group_by = "clusterID", g1 = c("1"), use_method = "limma")

## 3. To perform differential expression analysis in a rest-vs-1 manner for one specific group in "clusterID" co
de_res <- getDE(input_eset = clustered.eset, group_by = "clusterID", g0 = c("1"), use_method = "limma")

## 4. To perform differential expression analysis in a 1-vs-1 manner for groups in "clusterID" column
de_res <- getDE(input_eset = clustered.eset, group_by = "clusterID", g1 = c("1"), g0 = c("3"), use_method = "lim
```

---

getDriverList *Extract the pre-defined driver lists of human or mouse*

Description

This function is used to extract the pre-defined driver lists of human or mouse.

Usage

```
getDriverList(species_type = "hg", driver_type = "TF")
```

Arguments

species_type	Character, species of the driver lists to be extracted: "hg" for human, "mm" for mouse. Default: "hg".
driver_type	Character, type of drivers to be extracted: "TF" for transcriptional factor, "SIG" for signaling genes, and "TF_SIG" for both. Default: "TF".

Value

A vector of pre-defined driver genes

Examples

```
hg_tf <- getDriverList(species_type = "hg", driver_type = "TF") # get the TF driver list of human
mm_driver <- getDriverList(species_type = "mm", driver_type = "TF_SIG") # get the total driver list, including b
```

---

getTopFeatures *Pick the top genes/drivers for each group from differential analysis results*

Description

This function is used to pick the top genes/drivers for each group from differential analysis results based on either fold change, p value or FDR.

Usage

```
getTopFeatures(
  input_table,
  number = 10,
  group_by = "g1_tag",
  sort_by = "log2FC",
  sort_decreasing = TRUE
)
```

Arguments

input_table	The table generated by getDE() or getDA(), containing 11 statistics of differential analysis.
number	An Integer, number of top genes/driver to pick from each group. Default: 10.
group_by	Character, name of the column for grouping, usually the column of cell types or clusters. Default: "g1_tag".
sort_by	Character, name of the column for sorting. Default: "log2FC".
sort_decreasing	Logical, whether to sort the column specified by sort_by in decreasing order. If FALSE, the column will be sorted in increasing order. Default: TRUE.

Value

A data frame with top genes/driver of each group

Examples

```
top_drivers <- getTopFeature(da_res, number = 10, group_by, "g1_tag")
```

MICAplot *Draw a scatter plot showing the coordinates and cluster id of each cell*

Description

This function is used to visualize the clustering results generated by MICA.

Usage

```
MICAplot(
  input_eset,
  color_by = "clusterID",
  colors = NULL,
  X = "UMAP_1",
  Y = "UMAP_2",
  point.size = 0.3,
  point.alpha = 1,
  name.plot_title = NULL,
  fontsize.plot_table = 20,
  show.cluster_label = TRUE,
  fontsize.cluster_label = 12,
  legend.position = "right",
  fontsize.legend_title = 10,
  fontsize.legend_text = 8,
  fontsize.axis_title = 10,
  fontsize.axis_text = 8
)
```

Arguments

input_eset	The sparse eset object
color_by	Character, name of the column of MICA cluster labels. Default: "clusterID".
colors	A character vector or NULL, colors of the MICA cluster labels. The length of this vector should be same as the number of groups in color_by column. If NULL, the ggplot default colors will be use. Default: NULL.
X, Y	Character, name of the columns of x-axis and y-axis coordinates. Default: "UMAP_1", and "UMAP_2".
point.size	Numeric, size of points. Default: 0.3.
point.alpha	Numeric, transparency of points, ranging from 0 (more transparent) to 1 (less transparent). Default: 1.
name.plot_title	Character or NULL, title of the plot. Default: NULL.
fontsize.plot_table	Numeric, font size of the title. Default: 20.
show.cluster_label	Logical, whether to show labels on the plot. Default: TRUE.

`fontsize.cluster_label`
: Numeric, font size of the labels. Default: 12.

`legend.position`
: Character, position of legend: `"right"`, `"left"`, `"top"`, `"bottom"` or `"none"`. Default: `"right"`.

`fontsize.legend_title`
: Integer, font size of the legend title. Default: 10.

`fontsize.legend_text`
: Integer, font size of the legend text. Default: 8.

`fontsize.axis_title`
: Integer, font size of the axis title. Default: 10.

`fontsize.axis_text`
: Integer, font size of the axis text. Default: 8.

Value

A UMAP or T-SNE plot. It also print the plot to screen.

Examples

```
p_umap <- MICAplot(input_eset = clustered.eset, color_by = "clusterID", X = "UMAP_1", Y = "UMAP_2", point.size =
```

---

normalizeSparseEset *Normalize and log-transform the sparse eset object*

Description

This function is used to normalize and log-transform the sparse eset object. The default method is "log21p".

Usage

```
normalizeSparseEset(
  input_eset,
  scale_factor = 1e+06,
  do.logTransform = TRUE,
  log_base = 2,
  log_pseudoCount = 1
)
```

Arguments

`input_eset` The sparse eset object for normalization

`scale_factor` Numeric, the library size to normalize to. Default: 1000000.

`do.logTransform`
: Logical, whether to do log-transformation. Default: `TRUE`.

`log_base` Numeric, the base of log-transformation. Usually **2**, **exp(1)** or **10**. Default: 2.

`log_pseudoCount`
: Numeric, the pseudo count to add to avoid `"-Inf"` in log-transformation. Default: 1.

Value

A sparse eset object that has been normalized and log-transformed

Examples

```
normalized.eset <- normalizeSparseEset(input_eset = filtered.eset, scale_factor = 1000000, do.logTransform = T
```

`readInput_10x.dir` *Read the input data generated by 10x Genomics from a directory*

Description

This function is used to read the gene expression data from a directory containing three files generated by 10x Genomics: **matrix.mtx**, **barcodes.tsv** and **features.tsv** (or **genes.tsv**). This function can handle these conditions well:

- Alternative file names for feature data: **features.tsv** by CellRanger > 3.0, and **genes.tsv** by CellRanger < 3.0;
- One or more input files are compressed, usually in "**.gz**" format;
- Data with multiple modalities: like the single cell multiome data. In this case, it only retains the data of "Gene Expression".

Usage

```
readInput_10x.dir(
  input_dir,
  featureType = "gene_symbol",
  removeSuffix = TRUE,
  addPrefix = NULL
)
```

Arguments

`input_dir`	Path to the directory containing the 3 files generated by 10x Genomics: **matrix.mtx**, **barcodes.tsv** and **features.tsv** (or **genes.tsv**)
`featureType`	Character, feature type to use as the gene name of expression matrix: `"gene_symbol"` (the default) or `"gene_id"`.
`removeSuffix`	Logical, whether to remove the suffix "**-1**" when present in all cell barcodes. Default: `TRUE`.
`addPrefix`	Character or `NULL`, add a prefix to the cell barcodes, like Sample ID. It is highly recommended to use a prefix containing letters and/or numbers only, and not starting with numbers. Default: `NULL`.

Value

A sparse gene expression matrix of raw UMI counts, genes by cells

Examples

```
input_dir <- 'path-to-directory'
list.files(input_dir, full.names = FALSE) # you should see three files: matrix.mtx, barcodes.tsv and features.t
sparseMatrix <- readInput_10x.dir(input_dir, featureType = "gene_symbol", removeSuffix = TRUE, addPrefix = "de
```

readInput_10x.h5 *Read the input data generated by 10x Genomics from the HDF5 file*

Description

This function is used to read the gene expression data from the HDF5 file generated by CellRanger pipeline of 10x Genomics. This function can automatically distinguish the data of different modalities (e.g. expression data, ATAC data) and retains the gene expression data only. The **hdf5r** package is needed to use this function.

Usage

```
readInput_10x.h5(
  h5_file,
  featureType = "gene_symbol",
  removeSuffix = TRUE,
  addPrefix = NULL
)
```

Arguments

h5_file	H5 file generated by CellRanger pipeline of 10x Genomics
featureType	Character, feature type to use as the gene name of expression matrix: "gene_symbol" (the default) or "gene_id".
removeSuffix	Logical, whether to remove the suffix "**-1**" when present in all cell barcodes. Default: TRUE.
addPrefix	Character or NULL, add a prefix to the cell barcodes, like Sample ID. It is highly recommended to use a prefix containing letters and/or numbers only, and not starting with numbers. Default: NULL.

Value

A sparse gene expression matrix of raw UMI counts, genes by cells

Examples

```
h5_file <- 'path-to-h5_file'
sparseMatrix <- readInput_10x.h5(h5_file, featureType = "gene_symbol", removeSuffix = TRUE, addPrefix = "demoS
```

readInput_h5ad *Read the h5ad file*

Description

This function is used to read the h5ad file, a popular file format for storing and sharing single-cell RNA sequencing data. The **anndata** package is needed to use this function.

Usage

```
readInput_h5ad(h5ad_file, removeSuffix = FALSE, addPrefix = NULL)
```

Arguments

h5ad_file	H5ad file of sc/snRNA-seq data
removeSuffix	Logical, whether to remove the suffix "**-1**" when present in all cell barcodes. Default: TRUE.
addPrefix	Character or NULL, add a prefix to the cell barcodes, like Sample ID. It is highly recommended to use a prefix containing letters and/or numbers only, and not starting with numbers. Default: NULL.

Value

A AnnData object containing "X" (a observations x variables data matrix), "obs" (data frame of observations), "var" (data frame of variables) and more. For more details, please check out https://anndata.readthedocs.io/en/latest/generated/anndata.AnnData.html.

Examples

```
h5ad_file <- 'path-to-h5ad_file'
sparseMatrix <- readInput_h5ad(h5ad_file, removeSuffix = FALSE, addPrefix = "demoSample")
```

---

readInput_table *Read the table format file*

Description

This function is used to read data from a table-format file. The user needs to specify the format of the table using the parameter is.geneBYcell:

- TRUE (the default): the rows will be treated as genes, while the columns will be treated as cells;
- FALSE: the rows will be treated as cells, while the columns will be treated as genes.

Usage

```
readInput_table(
  table_file,
  sep = "\t",
  is.geneBYcell = TRUE,
  removeSuffix = FALSE,
  addPrefix = NULL
)
```

Arguments

table_file	The table format file (e.g. **txt**, **tsv**, **csv**, and others) which the data are to be read from.
sep	String, The field separator character. Default: "\t".
is.geneBYcell	Logical, whether the table is organized in **gene (row) by cell (column)** format. If FALSE, the rows will be treated as cells. Default: TRUE.
removeSuffix	Logical, whether to remove the suffix "**-1**" when present in all cell barcodes. Default: FALSE.
addPrefix	Character or NULL, add a prefix to the cell barcodes, like Sample ID. It is highly recommended to use a prefix containing letters and/or numbers only, and not starting with numbers. Default: NULL.

Value

A sparse gene expression matrix, genes by cells

Examples

```
table_file <- 'path-to-table_file'
sparseMatrix <- readInput_table(table_file, sep = "\t", is.geneBYcell = TRUE, removeSuffix = FALSE, addPrefix =
```

SparseExpressionSet-class *SparseExpressionSet*

Description

Define the class: SparseExpressionSet

---

updateSparseEset *Update the slots and/or meta data of the sparse eset object*

Description

This function is used to update the three slots ('**assayData**', '**phenoData**', '**featureData**') and/or '**meta data**' of sparse eset object.

Usage

```
updateSparseEset(
  input_eset,
  dataMatrix = NULL,
  cellData = NULL,
  featureData = NULL,
  addMetaData = FALSE
)
```

Arguments

input_eset	The sparse eset object to update
dataMatrix	A data matrix with Features/Genes as the rows and Cells as the columns. It's row.names and colnames must be consistent with the input_eset. Default: NULL.
cellData	A data frame containing meta data of cells or NULL. It's row.names should be consistent with the colnames of input_eset. Default: NULL.
featureData	A data frame containing meata data of features or NULL. It's row.names should be consistent with the row.names of input_eset. Default: NULL.
addMetaData	Logical, whether to update the meta data of features and cells based on the expression matrix. Default: FALSE.

Value

A sparse eset object with updated information

Examples

```
updated.eset <- updateSparseEset(input_eset = input.eset, cellData = data.frame(pData(input.eset), cellType =
```

Index