Package 'inferCSN'

May 8, 2026

Type Package

Title Inferring Cell-Specific Gene Regulatory Network

Version 1.2.0

Date 2025-10-15

Maintainer Meng Xu <mengxu98@qq.com>

Description An R package for inferring cell-type specific gene regulatory network from single-cell RNA-seq data.

License MIT + file LICENSE

URL https://mengxu98.github.io/inferCSN/

BugReports https://github.com/mengxu98/inferCSN/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, ggnetwork, ggplot2, ggraph, L0Learn, Matrix, methods, purrr, Rcpp, stats, thisutils

Suggests ComplexHeatmap, circlize, gtools, gganimate, ggExtra, ggpointdensity, ggpubr, igraph, irlba, network, patchwork, plotly, precrec, pROC, proxy, tidygraph, RANN, RColorBrewer, Rtsne, RTransferEntropy, uwot, viridis

LinkingTo Rcpp

Config/Needs/website mengxu98/thistemplate

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Language en-US

NeedsCompilation yes

Author Meng Xu [aut, cre] (ORCID: <https://orcid.org/0000-0002-8300-1054>)

Repository CRAN

Date/Publication 2025-10-16 02:30:02 UTC

Contents

inferCSN-package *Inferring Cell-Specific Gene Regulatory Network*

Description

An R package for inferring cell-type specific gene regulatory network from single-cell RNA-seq data.

Author(s)

Meng Xu (Maintainer), <mengxu98@qq.com>

Source

https://mengxu98.github.io/inferCSN/

See Also

Useful links:

- https://mengxu98.github.io/inferCSN/
- Report bugs at https://github.com/mengxu98/inferCSN/issues

calculate_accuracy *Calculate Accuracy*

Description

Calculates accuracy metric

Usage

```
calculate_accuracy(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_accuracy(
  network_table,
  example_ground_truth
)
```

`calculate_auc` *Calculate AUC Metrics*

Description

Calculates AUROC and AUPRC metrics with optional visualization

Usage

```
calculate_auc(
  network_table,
  ground_truth,
  return_plot = FALSE,
  line_color = "#1563cc",
  line_width = 1,
  tag_levels = "A"
)
```

Arguments

`network_table`	A data frame of predicted network structure
`ground_truth`	A data frame of ground truth network
`return_plot`	Logical value indicating whether to generate plots
`line_color`	Color for plot lines
`line_width`	Width for plot lines
`tag_levels`	Tag levels for plot annotations

Value

A list containing metrics and optional plots

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_auc(
  network_table,
  example_ground_truth,
  return_plot = TRUE
)
```

`calculate_auprc` *Calculate AUPRC Metric*

Description

Calculates AUPRC metric with optional visualization

Usage

```
calculate_auprc(
  network_table,
  ground_truth,
  return_plot = FALSE,
  line_color = "#1563cc",
  line_width = 1
)
```

Arguments

`network_table`	A data frame of predicted network structure
`ground_truth`	A data frame of ground truth network
`return_plot`	Logical value indicating whether to generate plot
`line_color`	Color for plot lines
`line_width`	Width for plot lines

Value

A list containing metric and optional plot

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_auprc(
  network_table,
  example_ground_truth,
  return_plot = TRUE
)
```

`calculate_auroc` *Calculate AUROC Metric*

Description

Calculates AUROC metric with optional visualization

Usage

```
calculate_auroc(
  network_table,
  ground_truth,
  return_plot = FALSE,
  line_color = "#1563cc",
  line_width = 1
)
```

Arguments

`network_table`	A data frame of predicted network structure
`ground_truth`	A data frame of ground truth network
`return_plot`	Logical value indicating whether to generate plot
`line_color`	Color for plot lines
`line_width`	Width for plot lines

Value

A list containing metric and optional plot

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_auroc(
  network_table,
  example_ground_truth,
  return_plot = TRUE
)
```

calculate_f1 *Calculate F1 Score*

Description

Calculates F1 score

Usage

```
calculate_f1(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_f1(
  network_table,
  example_ground_truth
)
```

calculate_gene_rank *Rank TFs and genes in network*

Description

Rank TFs and genes in network

Usage

```
calculate_gene_rank(
  network_table,
  regulators = NULL,
  targets = NULL,
  directed = FALSE
)
```

Arguments

network_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
directed	Whether the network is directed.

Value

A table of gene rank.

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
head(calculate_gene_rank(network_table))
head(calculate_gene_rank(network_table, regulators = "g1"))
head(calculate_gene_rank(network_table, targets = "g1"))
```

---

calculate_ji *Calculate Jaccard Index*

Description

Calculates Jaccard Index (JI) metric

Usage

```
calculate_ji(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_ji(
  network_table,
  example_ground_truth
)
```

`calculate_metrics` *Calculate Network Prediction Performance Metrics*

Description

Calculates comprehensive performance metrics for evaluating predicted network structures, including classification performance, precision-recall metrics, and network topology metrics.

Usage

```
calculate_metrics(
  network_table,
  ground_truth,
  metric_type = c("all", "auc", "auroc", "auprc", "precision", "recall", "f1",
    "accuracy", "si", "ji"),
  return_plot = FALSE,
  line_color = "#1563cc",
  line_width = 1
)
```

Arguments

`network_table`	A data frame.
`ground_truth`	A data frame of ground truth network with the same format as network_table.
`metric_type`	The type of metric to return. Default is `"all"`. This can take any of the following choices: • all Returns all available metrics with *Performance Metrics* plot. • auc Returns both AUROC and AUPRC with their plots. • auroc Area Under ROC Curve with plot. • auprc Area Under Precision-Recall Curve with plot. • precision Proportion of correct predictions among positive predictions. • recall Proportion of actual positives correctly identified. • f1 Harmonic mean of precision and recall. • accuracy Overall classification accuracy. • si Set Intersection, counting correctly predicted edges. • ji Jaccard Index, measuring overlap between predicted and true networks.
`return_plot`	Whether to generate visualization plots. Default is `FALSE`.
`line_color`	Color for plot lines. Default is `#1563cc`.
`line_width`	Width for plot lines. Default is `1`.

Value

A list containing:

- metrics A data frame with requested metrics.
- plot A plot object if return_plot = TRUE (optional).

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_metrics(
  network_table,
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
  network_table,
  example_ground_truth,
  metric_type = "auroc"
)
```

---

calculate_precision *Calculate Precision Metric*

Description

Calculates precision metric

Usage

```
calculate_precision(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_precision(
  network_table,
  example_ground_truth
)
```

calculate_recall *Calculate Recall Metric*

Description

Calculates recall metric

Usage

```
calculate_recall(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_recall(
  network_table,
  example_ground_truth
)
```

calculate_si *Calculate Set Intersection*

Description

Calculates Set Intersection (SI) metric

Usage

```
calculate_si(network_table, ground_truth)
```

Arguments

network_table	A data frame of predicted network structure
ground_truth	A data frame of ground truth network

Value

A list containing the metric

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
calculate_si(
  network_table,
  example_ground_truth
)
```

`example_ground_truth` *Example ground truth data*

Description

The data used for calculate the evaluating indicator.

`example_matrix` *Example matrix data*

Description

The matrix used for reconstruct gene regulatory network.

`example_meta_data` *Example meta data*

Description

The data contains cells and pseudotime information.

filter_sort_matrix *Filter and sort matrix*

Description

Filter and sort matrix

Usage

```
filter_sort_matrix(network_matrix, regulators = NULL, targets = NULL)
```

Arguments

network_matrix	The matrix of network weight.
regulators	Regulators list.
targets	Targets list.

Value

Filtered and sorted matrix

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
colnames(network_table) <- c("row", "col", "value")
network_matrix <- thisutils::table_to_matrix(network_table)
filter_sort_matrix(network_matrix)[1:6, 1:6]

filter_sort_matrix(
  network_matrix,
  regulators = c("g1", "g2"),
  targets = c("g3", "g4")
)
```

fit_srm *Sparse regression model*

Description

Sparse regression model

Usage

```
fit_srm(
  x,
  y,
  cross_validation = FALSE,
  seed = 1,
  penalty = "L0",
  regulators_num = ncol(x),
  n_folds = 5,
  verbose = TRUE,
  ...
)
```

Arguments

x	The matrix of regulators.
y	The vector of target.
cross_validation	Whether to use cross-validation. Default is FALSE.
seed	The random seed for cross-validation. Default is 1.
penalty	The type of regularization, default is "L0". This can take either one of the following choices: "L0", "L0L1", and "L0L2". For high-dimensional and sparse data, "L0L2" is more effective.
regulators_num	The number of regulators for target.
n_folds	The number of folds for cross-validation. Default is 5.
verbose	Whether to print progress messages. Default is TRUE.
...	Parameters for other methods.

Value

A list of the sparse regression model. The list has the three components: model, metrics, and coefficients.

Examples

```
data(example_matrix)
fit_srm(
  x = example_matrix[, -1],
  y = example_matrix[, 1]
)
```

inferCSN *inferring cell-type specific gene regulatory network*

Description

inferring cell-type specific gene regulatory network

Usage

```
inferCSN(
  object,
  penalty = "L0",
  cross_validation = FALSE,
  seed = 1,
  n_folds = 5,
  subsampling_method = c("sample", "meta_cells", "pseudobulk"),
  subsampling_ratio = 1,
  r_squared_threshold = 0,
  regulators = NULL,
  targets = NULL,
  cores = 1,
  verbose = TRUE,
  ...
)

## S4 method for signature 'matrix'
inferCSN(
  object,
  penalty = "L0",
  cross_validation = FALSE,
  seed = 1,
  n_folds = 5,
  subsampling_method = c("sample", "meta_cells", "pseudobulk"),
  subsampling_ratio = 1,
  r_squared_threshold = 0,
  regulators = NULL,
  targets = NULL,
  cores = 1,
  verbose = TRUE,
  ...
)

## S4 method for signature 'sparseMatrix'
inferCSN(
  object,
  penalty = "L0",
  cross_validation = FALSE,
```

```
  seed = 1,
  n_folds = 5,
  subsampling_method = c("sample", "meta_cells", "pseudobulk"),
  subsampling_ratio = 1,
  r_squared_threshold = 0,
  regulators = NULL,
  targets = NULL,
  cores = 1,
  verbose = TRUE,
  ...
)

## S4 method for signature 'data.frame'
inferCSN(
  object,
  penalty = "L0",
  cross_validation = FALSE,
  seed = 1,
  n_folds = 5,
  subsampling_method = c("sample", "meta_cells", "pseudobulk"),
  subsampling_ratio = 1,
  r_squared_threshold = 0,
  regulators = NULL,
  targets = NULL,
  cores = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	The input data for `inferCSN`.
`penalty`	The type of regularization, default is `"L0"`. This can take either one of the following choices: `"L0"`, `"L0L1"`, and `"L0L2"`. For high-dimensional and sparse data, `"L0L2"` is more effective.
`cross_validation`	Whether to use cross-validation. Default is `FALSE`.
`seed`	The random seed for cross-validation. Default is `1`.
`n_folds`	The number of folds for cross-validation. Default is `5`.
`subsampling_method`	The method to use for subsampling. Options are `"sample"`, `"pseudobulk"` or `"meta_cells"`.
`subsampling_ratio`	The percent of all samples used for fit_srm. Default is `1`.
`r_squared_threshold`	Threshold of R^2 coefficient. Default is `0`.
`regulators`	The regulator genes for which to infer the regulatory network.

targets	The target genes for which to infer the regulatory network.
cores	The number of cores to use for parallelization with foreach::foreach. Default is 1.
verbose	Whether to print progress messages. Default is TRUE.
...	Parameters for other methods.

Value

A data table of regulator-target regulatory relationships. The data table has the three columns: regulator, target, and weight.

Examples

```
data(example_matrix)
network_table_1 <- inferCSN(
  example_matrix
)

network_table_2 <- inferCSN(
  example_matrix,
  cores = 2
)

head(network_table_1)

identical(
  network_table_1,
  network_table_2
)

inferCSN(
  example_matrix,
  regulators = c("g1", "g2"),
  targets = c("g3", "g4")
)
inferCSN(
  example_matrix,
  regulators = c("g1", "g2"),
  targets = c("g3", "g0")
)

## Not run:
data("example_ground_truth")
network_table_07 <- inferCSN(
  example_matrix,
  r_squared_threshold = 0.7
)
calculate_metrics(
  network_table_1,
  example_ground_truth,
  return_plot = TRUE
```

```
)
calculate_metrics(
  network_table_07,
  example_ground_truth,
  return_plot = TRUE
)

## End(Not run)
## Not run:
data(example_matrix)
network_table <- inferCSN(example_matrix)
head(network_table)

network_table_sparse_1 <- inferCSN(
  as(example_matrix, "sparseMatrix")
)
head(network_table_sparse_1)

network_table_sparse_2 <- inferCSN(
  as(example_matrix, "sparseMatrix"),
  cores = 2
)
identical(
  network_table,
  network_table_sparse_1
)

identical(
  network_table_sparse_1,
  network_table_sparse_2
)

plot_scatter(
  data.frame(
    network_table$weight,
    network_table_sparse_1$weight
  ),
  legend_position = "none"
)

plot_weight_distribution(
  network_table
) + plot_weight_distribution(
  network_table_sparse_1
)

## End(Not run)
```

`infercsn_logo` *The logo of inferCSN*

Description

The inferCSN logo, using ASCII or Unicode characters Use cli::ansi_strip to get rid of the colors.

Usage

```
infercsn_logo(unicode = cli::is_utf8_output())
```

Arguments

unicode	Unicode symbols on UTF-8 platforms. Default is cli::is_utf8_output.

References

https://github.com/tidyverse/tidyverse/blob/main/R/logo.R

Examples

```
infercsn_logo()
```

---

meta_cells *Detection of metacells from single-cell gene expression matrix*

Description

This function detects metacells from a single-cell gene expression matrix using dimensionality reduction and clustering techniques.

Usage

```
meta_cells(
  matrix,
  genes_use = NULL,
  genes_exclude = NULL,
  var_genes_num = min(1000, nrow(matrix)),
  gamma = 10,
  knn_k = 5,
  do_scale = TRUE,
  pc_num = 25,
  fast_pca = FALSE,
  do_approx = FALSE,
  approx_num = 20000,
  directed = FALSE,
  use_nn2 = TRUE,
  seed = 1,
  cluster_method = "walktrap",
  block_size = 10000,
  weights = NULL,
```

```
  do_median_norm = FALSE,
  ...
)
```

Arguments

matrix	A gene expression matrix where rows represent genes and columns represent cells.
genes_use	A character vector specifying genes to be used for PCA dimensionality reduction. Default is NULL.
genes_exclude	A character vector specifying genes to be excluded from PCA computation. Default is NULL.
var_genes_num	Number of most variable genes to select when genes_use is not provided. Default is min(1000, nrow(matrix)).
gamma	Default is 10. Coarse-graining parameter defining the target ratio of input cells to output metacells (e.g., gamma=10 yields approximately n/10 metacells for n input cells).
knn_k	Default is 5. Number of nearest neighbors for constructing the cell-cell similarity network.
do_scale	Whether to standardize (center and scale) gene expression values before PCA. Default is TRUE.
pc_num	Default is 25. Number of principal components to retain for downstream analysis.
fast_pca	Default is TRUE. Whether to use the faster irlba algorithm instead of standard PCA. Recommended for large datasets.
do_approx	Default is FALSE. Whether to use approximate nearest neighbor search for datasets with >50000 cells to improve computational efficiency.
approx_num	Default is 20000. Number of cells to randomly sample for approximate nearest neighbor computation when do_approx = TRUE.
directed	Default is FALSE. Whether to construct a directed or undirected nearest neighbor graph.
use_nn2	Default is TRUE. Whether to use the faster RANN::nn2 algorithm for nearest neighbor search (only applicable with Euclidean distance).
seed	Default is 1. Random seed for reproducibility when subsampling cells in approximate mode.
cluster_method	Default is walktrap. Algorithm for community detection in the cell similarity network. Options: walktrap (recommended) or louvain (gamma parameter ignored).
block_size	Default is 10000. Number of cells to process in each batch when mapping cells to metacells in approximate mode. Adjust based on available memory.
weights	Default is NULL. Numeric vector of cell-specific weights for weighted averaging when computing metacell expression profiles. Length must match number of cells.
do_median_norm	Default is FALSE. Whether to perform median-based normalization of the final metacell expression matrix.
...	Additional arguments passed to internal functions.

Value

A matrix where rows represent metacells and columns represent genes.

References

https://github.com/GfellerLab/SuperCell https://github.com/kuijjerlab/SCORPION

Examples

```
data(example_matrix)
meta_cells_matrix <- meta_cells(
  example_matrix
)
dim(meta_cells_matrix)
meta_cells_matrix[1:6, 1:6]
```

---

network_format *Format network table*

Description

Format network table

Usage

```
network_format(
  network_table,
  regulators = NULL,
  targets = NULL,
  abs_weight = TRUE
)
```

Arguments

network_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
abs_weight	Logical value, default is TRUE, whether to perform absolute value on weights, and when set abs_weight to TRUE, the output of weight table will create a new column named Interaction.

Value

Formated network table

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)

network_format(
  network_table,
  regulators = "g1"
)

network_format(
  network_table,
  regulators = "g1",
  abs_weight = FALSE
)

network_format(
  network_table,
  targets = "g3"
)

network_format(
  network_table,
  regulators = c("g1", "g3"),
  targets = c("g3", "g5")
)
```

---

network_sift *Sifting network*

Description

Sifting network

Usage

```
network_sift(
  network_table,
  matrix = NULL,
  meta_data = NULL,
  pseudotime_column = NULL,
  method = c("entropy", "max"),
  entropy_method = c("Shannon", "Renyi"),
  effective_entropy = FALSE,
  shuffles = 100,
  entropy_nboot = 300,
  lag_value = 1,
  entropy_p_value = 0.05,
  cores = 1,
  verbose = TRUE
)
```

Arguments

network_table	The weight data table of network.
matrix	The expression matrix.
meta_data	The meta data for cells or samples.
pseudotime_column	The column of pseudotime.
method	The method used for filter edges. Could be choose "entropy" or "max".
entropy_method	If setting method to "entropy", could be choose "Shannon" or "Renyi" to compute entropy.
effective_entropy	Default is FALSE. Whether to use effective entropy to filter weights.
shuffles	Default is 100. The number of shuffles used to calculate the effective transfer entropy.
entropy_nboot	Default is 300. The number of bootstrap replications for each direction of the estimated transfer entropy.
lag_value	Default is 1. Markov order of gene expression values, i.e. the number of lagged values affecting the current value of gene expression values.
entropy_p_value	P value used to filter edges by entropy. Default is 0.05.
cores	The number of cores to use for parallelization with foreach::foreach. Default is 1.
verbose	Whether to print progress messages. Default is TRUE.

Value

A data table of regulator-target regulatory relationships. The data table has the three columns: regulator, target, and weight.

Examples

```
## Not run:
data(example_matrix)
data(example_meta_data)
data(example_ground_truth)

network_table <- inferCSN(example_matrix)
network_table_sifted <- network_sift(network_table)
network_table_sifted_entropy <- network_sift(
  network_table,
  matrix = example_matrix,
  meta_data = example_meta_data,
  pseudotime_column = "pseudotime",
  lag_value = 2,
  shuffles = 0,
  entropy_nboot = 0
)
```

```
plot_network_heatmap(
  example_ground_truth[, 1:3],
  heatmap_title = "Ground truth",
  show_names = TRUE,
  rect_color = "gray70"
)
plot_network_heatmap(
  network_table,
  heatmap_title = "Raw",
  show_names = TRUE,
  rect_color = "gray70"
)
plot_network_heatmap(
  network_table_sifted,
  heatmap_title = "Filtered",
  show_names = TRUE,
  rect_color = "gray70"
)
plot_network_heatmap(
  network_table_sifted_entropy,
  heatmap_title = "Filtered by entropy",
  show_names = TRUE,
  rect_color = "gray70"
)

calculate_metrics(
  network_table,
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
  network_table_sifted,
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
  network_table_sifted_entropy,
  example_ground_truth,
  return_plot = TRUE
)

## End(Not run)
```

`plot_coefficient` *Plot coefficients*

Description

Plot coefficients

Usage

```
plot_coefficient(
  data,
  style = "continuous",
  positive_color = "#3d67a2",
  negative_color = "#c82926",
  neutral_color = "#cccccc",
  bar_width = 0.7,
  text_size = 3,
  show_values = TRUE
)
```

Arguments

data	Input data.
style	Plotting style: "binary", "gradient", or "continuous".
positive_color	Color for positive weights. Default is "#3d67a2".
negative_color	Color for negative weights. Default is "#c82926".
neutral_color	Color for weights near zero (used in "continuous" style). Default is "#cccccc".
bar_width	Width of the bars. Default is 0.7.
text_size	Size of the text for weight values. Default is 3.
show_values	Whether to show weight values on bars. Default is TRUE.

Value

A ggplot object

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix, targets = "g1")
plot_coefficient(network_table)
plot_coefficient(network_table, style = "binary")
```

---

plot_coefficients *Plot coefficients for multiple targets*

Description

Plot coefficients for multiple targets

Usage

```
plot_coefficients(data, targets = NULL, nrow = NULL, ...)
```

Arguments

data	Input data.
targets	Targets to plot. Default is 'NULL'.
nrow	Number of rows for the plot. Default is 'NULL'.
...	Other arguments passed to [plot_coefficient].

Value

A list of ggplot objects

Examples

```
data(example_matrix)
network_table <- inferCSN(
  example_matrix,
  targets = c("g1", "g2", "g3")
)
plot_coefficients(network_table, show_values = FALSE)
plot_coefficients(network_table, targets = "g1")
```

---

plot_contrast_networks

Plot contrast networks

Description

Plot contrast networks

Usage

```
plot_contrast_networks(
  network_table,
  degree_value = 0,
  weight_value = 0,
  legend_position = "bottom"
)
```

Arguments

network_table	The weight data table of network.
degree_value	Degree value to filter nodes. Default is 0.
weight_value	Weight value to filter edges. Default is 0.
legend_position	The position of legend. Default is "bottom".

Value

A ggplot2 object.

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
plot_contrast_networks(network_table[1:50, ])
```

`plot_dynamic_networks` *Plot dynamic networks*

Description

Plot dynamic networks

Usage

```
plot_dynamic_networks(
  network_table,
  celltypes_order,
  ntop = 10,
  title = NULL,
  theme_type = "theme_void",
  plot_type = "ggplot",
  layout = "fruchtermanreingold",
  nrow = 2,
  figure_save = FALSE,
  figure_name = NULL,
  figure_width = 6,
  figure_height = 6,
  seed = 1
)
```

Arguments

`network_table`	The weight data table of network.
`celltypes_order`	The order of cell types.
`ntop`	The number of top genes to plot. Default is `10`.
`title`	The title of figure. Default is `NULL`.
`theme_type`	The theme of figure. Could be `"theme_void"`, `"theme_blank"`, or `"theme_facet"`. Default is `"theme_void"`.
`plot_type`	The type of figure. Could be `"ggplot"`, `"animate"`, or `"ggplotly"`. Default is `"ggplot"`.

layout	The layout of figure. Could be "fruchtermanreingold" or "kamadakawai". Default is "fruchtermanreingold".
nrow	The number of rows of figure. Default is 2.
figure_save	Whether to save the figure file. Default is FALSE.
figure_name	The name of figure file. Default is NULL.
figure_width	The width of figure. Default is 6.
figure_height	The height of figure. Default is 6.
seed	The seed random use to plot network. Default is 1.

Value

A dynamic figure object.

Examples

```
data(example_matrix)
network <- inferCSN(example_matrix)[1:100, ]
network$celltype <- c(
  rep("cluster1", 20),
  rep("cluster2", 20),
  rep("cluster3", 20),
  rep("cluster5", 20),
  rep("cluster6", 20)
)

celltypes_order <- c(
  "cluster5", "cluster3",
  "cluster2", "cluster1",
  "cluster6"
)

plot_dynamic_networks(
  network,
  celltypes_order = celltypes_order
)

plot_dynamic_networks(
  network,
  celltypes_order = celltypes_order[1:3]
)

plot_dynamic_networks(
  network,
  celltypes_order = celltypes_order,
  plot_type = "ggplotly"
)

## Not run:
# If setting `plot_type = "animate"` to plot and save `gif` figure,
# please install `gifski` package first.
```

```
plot_dynamic_networks(
  network,
  celltypes_order = celltypes_order,
  plot_type = "animate"
)

## End(Not run)
```

`plot_edges_comparison` *Network Edge Comparison Visualization*

Description

Generates visualizations comparing edges of two networks.

Usage

```
plot_edges_comparison(
  network_table,
  ground_truth,
 color_pattern = list(predicted = "gray", ground_truth = "#bb141a", overlap = "#1966ad",
    total = "#6C757D")
)
```

Arguments

`network_table`	A data frame of predicted network structure.
`ground_truth`	A data frame of ground truth network.
`color_pattern`	A list of colors for different categories.

Value

A patchwork plot object containing network edge comparison and distribution plots

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)
plot_edges_comparison(
  network_table,
  example_ground_truth
)
```

`plot_embedding` *Plot embedding*

Description

Plot embedding

Usage

```
plot_embedding(
  matrix,
  labels = NULL,
  method = "pca",
  colors = RColorBrewer::brewer.pal(length(unique(labels)), "Set1"),
  seed = 1,
  point_size = 1,
  cores = 1
)
```

Arguments

`matrix`	Input matrix.
`labels`	Input labels.
`method`	Method to use for dimensionality reduction.
`colors`	Colors to use for the plot.
`seed`	Seed for the random number generator.
`point_size`	Size of the points.
`cores`	Set the number of threads when setting for uwot::umap and Rtsne::Rtsne.

Value

An embedding plot

Examples

```
data(example_matrix)
samples_use <- 1:200
plot_data <- example_matrix[samples_use, ]
labels <- sample(
  c("A", "B", "C", "D", "E"),
  nrow(plot_data),
  replace = TRUE
)

plot_embedding(
  plot_data,
```

```
  labels,
  method = "pca",
  point_size = 2
)

plot_embedding(
  plot_data,
  labels,
  method = "tsne",
  point_size = 2
)
```

`plot_histogram` *Plot histogram*

Description

Plot histogram

Usage

```
plot_histogram(
  data,
  binwidth = 0.01,
  show_border = FALSE,
  border_color = "black",
  alpha = 1,
  theme = "viridis",
  theme_begin = 0,
  theme_end = 0.5,
  theme_direction = -1,
  legend_position = "right"
)
```

Arguments

`data`	A numeric vector.
`binwidth`	Width of the bins.
`show_border`	Logical value, whether to show border of the bins.
`border_color`	Color of the border.
`alpha`	Alpha value of the bins.
`theme`	Theme of the bins.
`theme_begin`	Begin value of the theme.
`theme_end`	End value of the theme.
`theme_direction`	Direction of the theme.
`legend_position`	The position of legend.

Value

A ggplot object

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
plot_histogram(network_table[, 3])
```

`plot_network_heatmap` *Plot network heatmap*

Description

Plot network heatmap

Usage

```
plot_network_heatmap(
  network_table,
  regulators = NULL,
  targets = NULL,
  switch_matrix = TRUE,
  show_names = FALSE,
  heatmap_size_lock = TRUE,
  heatmap_size = 5,
  heatmap_height = NULL,
  heatmap_width = NULL,
  heatmap_title = NULL,
  heatmap_color = c("#1966ad", "white", "#bb141a"),
  border_color = "gray",
  rect_color = NA,
  anno_width = 1,
  anno_height = 1,
 row_anno_type = c("boxplot", "barplot", "histogram", "density", "lines", "points",
    "horizon"),
 column_anno_type = c("boxplot", "barplot", "histogram", "density", "lines", "points"),
  legend_name = "Weight",
  row_title = "Regulators"
)
```

Arguments

`network_table`	The weight data table of network.
`regulators`	Regulators list.
`targets`	Targets list.

switch_matrix Whether to weight data table to matrix. Default is TRUE.

show_names Whether to show names of row and column. Default is FALSE.

heatmap_size_lock Lock the size of heatmap.

heatmap_size The size of heatmap. Default is 5.

heatmap_height The height of heatmap.

heatmap_width The width of heatmap.

heatmap_title The title of heatmap.

heatmap_color Colors of heatmap.

border_color Default is "gray". Color of heatmap border.

rect_color Default is NA. Color of heatmap rect.

anno_width Width of annotation.

anno_height Height of annotation.

row_anno_type Default is "boxplot", could add a annotation plot to row. choose one of "boxplot", "barplot", "histogram", "density", "lines", "points", and "horizon".

column_anno_type Default is "boxplot", could add a annotation plot to column. choose one of "boxplot", "barplot", "histogram", "density", "lines", and "points".

legend_name The name of legend.

row_title The title of row.

Value

A heatmap

Examples

```
data(example_matrix)
data("example_ground_truth")
network_table <- inferCSN(example_matrix)

p1 <- plot_network_heatmap(
  example_ground_truth[, 1:3],
  heatmap_title = "Ground truth",
  legend_name = "Ground truth"
)
p2 <- plot_network_heatmap(
  network_table,
  heatmap_title = "inferCSN",
  legend_name = "inferCSN"
)
ComplexHeatmap::draw(p1 + p2)
## Not run:
p3 <- plot_network_heatmap(
  network_table,
  legend_name = "Weight1",
```

```
  heatmap_color = c("#20a485", "#410054", "#fee81f")
)
p4 <- plot_network_heatmap(
  network_table,
  legend_name = "Weight2",
  heatmap_color = c("#20a485", "white", "#fee81f")
)
ComplexHeatmap::draw(p3 + p4)

## End(Not run)

plot_network_heatmap(
  network_table,
  show_names = TRUE,
  rect_color = "gray90",
  row_anno_type = "density",
  column_anno_type = "barplot"
)

plot_network_heatmap(
  network_table,
  regulators = c("g1", "g3", "g5"),
  targets = c("g3", "g6", "g9"),
  show_names = TRUE
)
## Not run:
plot_network_heatmap(
  network_table,
  regulators = c("g1", "g2"),
  show_names = TRUE
)

plot_network_heatmap(
  network_table,
  targets = c("g1", "g2"),
  row_anno_type = "boxplot",
  column_anno_type = "histogram",
  show_names = TRUE
)

## End(Not run)
```

`plot_scatter` *Plot expression data in a scatter plot*

Description

Plot expression data in a scatter plot

Usage

```
plot_scatter(
  data,
  smoothing_method = "lm",
  group_colors = RColorBrewer::brewer.pal(9, "Set1"),
  title_color = "black",
  title = NULL,
  col_title = NULL,
  row_title = NULL,
  legend_title = NULL,
  legend_position = "bottom",
  margins = "both",
  marginal_type = NULL,
  margins_size = 10,
  compute_correlation = TRUE,
  compute_correlation_method = "pearson",
  keep_aspect_ratio = TRUE,
  facet = FALSE,
  se = FALSE,
  pointdensity = TRUE
)
```

Arguments

`data`	Input data.
`smoothing_method`	Method for smoothing curve, `lm` or `loess`.
`group_colors`	Colors for different groups.
`title_color`	Color for the title.
`title`	Main title for the plot.
`col_title`	Title for the x-axis.
`row_title`	Title for the y-axis.
`legend_title`	Title for the legend.
`legend_position`	The position of legend.
`margins`	The position of marginal figure ("both", "x", "y").
`marginal_type`	The type of marginal figure (`density`, `histogram`, `boxplot`, `violin`, `densigram`).
`margins_size`	The size of marginal figure, note the bigger size the smaller figure.
`compute_correlation`	Whether to compute and print correlation on the figure.
`compute_correlation_method`	Method to compute correlation (`pearson` or `spearman`).
`keep_aspect_ratio`	Logical value, whether to set aspect ratio to 1:1.

facet	Faceting variable. If setting TRUE, all settings about margins will be inalidation.
se	Display confidence interval around smooth.
pointdensity	Plot point density when only provide 1 cluster.

Value

ggplot object

Examples

```
data(example_matrix)
test_data <- data.frame(
  example_matrix[1:200, c(1, 7)],
  c = c(
    rep("c1", 40),
    rep("c2", 40),
    rep("c3", 40),
    rep("c4", 40),
    rep("c5", 40)
  )
)

p1 <- plot_scatter(
  test_data
)
p2 <- plot_scatter(
  test_data,
  marginal_type = "boxplot"
)
p1 + p2

p3 <- plot_scatter(
  test_data,
  facet = TRUE
)
p3

p4 <- plot_scatter(
  test_data[, 1:2],
  marginal_type = "histogram"
)
p4
```

---

plot_static_networks *Plot dynamic networks*

Description

Plot dynamic networks

Usage

```
plot_static_networks(
  network_table,
  regulators = NULL,
  targets = NULL,
  legend_position = "right"
)
```

Arguments

network_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
legend_position	The position of legend.

Value

A ggplot2 object

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
plot_static_networks(
  network_table,
  regulators = "g1"
)
plot_static_networks(
  network_table,
  targets = "g1"
)
plot_static_networks(
  network_table,
  regulators = "g1",
  targets = "g2"
)
```

---

print.infercsn_logo *Print logo*

Description

Print logo

Usage

```
## S3 method for class 'infercsn_logo'
print(x, ...)
```

Arguments

x	Input information.
...	Other parameters.

Value

Print the ASCII logo

`single_network` *Construct network for single target gene*

Description

Construct network for single target gene

Usage

```
single_network(
  matrix,
  regulators,
  target,
  cross_validation = FALSE,
  seed = 1,
  penalty = "L0",
  r_squared_threshold = 0,
  n_folds = 5,
  verbose = TRUE,
  ...
)
```

Arguments

matrix	An expression matrix.
regulators	The regulator genes for which to infer the regulatory network.
target	The target gene.
cross_validation	Whether to use cross-validation. Default is FALSE.
seed	The random seed for cross-validation. Default is 1.
penalty	The type of regularization, default is "L0". This can take either one of the following choices: "L0", "L0L1", and "L0L2". For high-dimensional and sparse data, "L0L2" is more effective.
r_squared_threshold	Threshold of R^2 coefficient. Default is 0.
n_folds	The number of folds for cross-validation. Default is 5.
verbose	Whether to print progress messages. Default is TRUE.
...	Parameters for other methods.

Value

A data frame of the single target gene network. The data frame has three columns: regulator, target, and weight.

Examples

```
data(example_matrix)
head(
  single_network(
    example_matrix,
    regulators = colnames(example_matrix),
    target = "g1"
  )
)
head(
  single_network(
    example_matrix,
    regulators = colnames(example_matrix),
    target = "g1",
    cross_validation = TRUE
  )
)

single_network(
  example_matrix,
  regulators = c("g1", "g2", "g3"),
  target = "g1"
)
single_network(
  example_matrix,
  regulators = c("g1", "g2"),
  target = "g1"
)
```

---

subsampling *Subsampling function*

Description

This function subsamples a matrix using either random sampling or meta cells method.

Usage

```
subsampling(
  matrix,
  subsampling_method = c("sample", "meta_cells", "pseudobulk"),
  subsampling_ratio = 1,
  seed = 1,
  verbose = TRUE,
```

```
  ...
)
```

Arguments

matrix	The input matrix to be subsampled.
subsampling_method	The method to use for subsampling. Options are "sample", "pseudobulk" or "meta_cells".
subsampling_ratio	The percent of all samples used for fit_srm. Default is 1.
seed	The random seed for cross-validation. Default is 1.
verbose	Whether to print progress messages. Default is TRUE.
...	Parameters for other methods.

Value

The subsampled matrix.

Examples

```
data(example_matrix)
data("example_ground_truth")
subsample_matrix <- subsampling(
  example_matrix,
  subsampling_ratio = 0.5
)
subsample_matrix_2 <- subsampling(
  example_matrix,
  subsampling_method = "meta_cells",
  subsampling_ratio = 0.5,
  fast_pca = FALSE
)
subsample_matrix_3 <- subsampling(
  example_matrix,
  subsampling_method = "pseudobulk",
  subsampling_ratio = 0.5
)

calculate_metrics(
  inferCSN(example_matrix),
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
  inferCSN(subsample_matrix),
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
```

```
  inferCSN(subsample_matrix_2),
  example_ground_truth,
  return_plot = TRUE
)
calculate_metrics(
  inferCSN(subsample_matrix_3),
  example_ground_truth,
  return_plot = TRUE
)
```

`weight_sift` *Weight sift*

Description

Remove edges with smaller weights in the reverse direction.

Usage

```
weight_sift(table)
```

Arguments

`table`	A data frame with three columns: "regulator", "target", and "weight".

Examples

```
data(example_matrix)
network_table <- inferCSN(example_matrix)
weight_sift(network_table) |> head()
```

Index