

Package ‘scGraphVerse’

May 15, 2025

Title scGraphVerse: A Gene Regulatory Network Analysis Package

Version 0.99.0

Description A package for inferring, comparing, and visualizing gene regulatory networks from single-cell RNA sequencing data. It integrates multiple methods (GENIE3, GRNBoost2, ZILGM, PCzinb, and JRF) for robust network inference, supports consensus building across methods or datasets, and provides tools for evaluating regulatory structure and community similarity.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

biocViews GeneRegulation, NetworkInference, SingleCell, RNASeq, Visualization, Software, GraphAndNetwork, GeneSetEnrichment, NetworkEnrichment, Pathways, Sequencing, Reactome, Network, KEGG

URL <https://ngsFC.github.io/scGraphVerse>

BugReports <https://github.com/ngsFC/scGraphVerse/issues>

Depends R (>= 4.4.0)

Imports AnnotationDbi,
BiocParallel (>= 1.30.0),
GENIE3,
Matrix,
RColorBrewer,
ReactomePA,
STRINGdb,
Seurat,
SingleCellExperiment,
SummarizedExperiment,
clusterProfiler,
distributions3,
doParallel,
dplyr,
fmsb,

ggraph,	
ggplot2,	
gridExtra,	
grDevices,	
graphics,	
httr,	
igraph,	
jsonlite,	
methods,	
org.Hs.eg.db,	
parallel,	
patchwork,	
pROC,	
rentrez,	
reticulate,	
robin,	
scales,	
tidyr,	
cli	
Suggests testthat (>= 3.0.0),	
knitr,	
rmarkdown,	
tidyverse,	
magick,	
JRF,	
ZILGM,	
learn2count,	
INetTool	
VignetteBuilder knitr	
LazyData true	

Contents

adj_truth	3
community_path	3
community_similarity	5
compare_consensus	6
count_matrices	8
create_consensus	9
cutoff_adjacency	10
download_Atlas	12
earlyj	13
edge_mining	14
generate_adjacency	15
infer_networks	16
init_py	18
plotg	19

adj_truth

3

plotROC

20

pscores

21

selgene

22

stringdb_adjacency

24

symmetrize

26

zinb_simdata

27

Index

29

adj_truth	<i>Toy adjacency matrix for examples</i>
-----------	--

Description

An adjacency matrix generated using real dataset and STRINGdb for demonstrating functions in the scGraphVerse package.

Usage

```
data(adj_truth)
```

Format

A matrix of dimension 38x38.

Examples

```
data(adj_truth)
str(adj_truth)
```

community_path	<i>Community Detection and Pathway Enrichment Analysis</i>
----------------	--

Description

Detects gene communities within an adjacency network using one or two community detection methods, and performs pathway enrichment for each detected community.

Usage

```
community_path(
  adj_matrix,
  methods = "louvain",
  pathway_db = "KEGG",
  genes_path = 5,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

adj_matrix	A square adjacency matrix. Row and column names must correspond to gene symbols.
methods	A character vector of one or two community detection methods supported by robin . If two are given, performance is compared and the best is selected. Default: "louvain".
pathway_db	Character string specifying the pathway database to use: "KEGG" or "Reactome". Default: "KEGG".
genes_path	Integer. Minimum number of genes per community to run enrichment analysis. Default: 5.
plot	Logical. If TRUE, generates a plot of detected communities. Default: TRUE.
verbose	Logical. If TRUE, shows progress messages. Default: TRUE.

Details

If two methods are provided, the function uses `robinCompare` and selects the method with higher AUC. Pathway enrichment is done via **clusterProfiler** (KEGG) or via **ReactomePA** (Reactome). Communities smaller than `genes_path` are excluded.

Value

A list with elements:

- `communities`: List with `best_method` and a named vector of community membership per gene.
- `pathways`: List of enrichment results per community (only for communities meeting size threshold).
- `graph`: The **igraph** object with community annotations.

Examples

```
data(count_matrices)
data(adj_truth)
networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
```

```
    quantile_threshold = 0.99,  
    nCores = 1,  
    debug = TRUE  
  )  
  head(binary_listj[[1]])  
  
consensus <- create_consensus(binary_listj, method = "vote")  
comm_cons <- community_path(consensus)
```

community_similarity *Compare Community Assignments and Topological Properties*

Description

Evaluates similarity between a ground truth community structure and one or more predicted community structures. Computes community assignment metrics (VI, NMI, ARI) and raw topological properties (Modularity, Number of Communities, Density, Transitivity). Visualizes results via a radar plot for community assignment and bar plots for topology.

Usage

```
community_similarity(control_output, predicted_list)
```

Arguments

control_output A list output from `community_path()` representing the ground truth network. Must contain a graph (igraph object) and `communities$membership`.

predicted_list A list of lists, each output from `community_path()` representing predicted networks to compare.

Details

This function requires the **igraph** and **fmsb** packages. Community similarity is measured using variation of information (VI), normalized mutual information (NMI), and adjusted Rand index (ARI). Topological properties are compared by directly plotting raw values without normalization.

Value

A list containing:

- **community_metrics**: A data frame with VI, NMI, and ARI scores for each prediction.
- **topology_measures**: A data frame with raw topological metrics for each prediction.
- **control_topology**: A list of raw topological metrics for the ground truth network.

Examples

```
data(count_matrices)
data(adj_truth)
networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
  quantile_threshold = 0.95,
  nCores = 1,
  debug = TRUE
)
head(binary_listj[[1]])

consensus <- create_consensus(binary_listj, method = "vote")
comm_cons <- community_path(consensus)
comm_truth <- community_path(adj_truth)

sim_score <- community_similarity(comm_truth, list(comm_cons))
```

compare_consensus

Compare Consensus and Reference Graphs or STRINGdb Networks

Description

Compares a consensus adjacency matrix to a reference network, either provided manually or generated from STRINGdb. Visualizes True Positives (TP), False Negatives (FN), and optionally False Positives (FP) edges.

Usage

```
compare_consensus(
  consensus_matrix,
  reference_matrix = NULL,
  false_plot = FALSE
)
```

Arguments

<code>consensus_matrix</code>	A binary square adjacency matrix representing the consensus network. Row and column names should correspond to gene symbols.
<code>reference_matrix</code>	Optional. A binary square adjacency matrix representing the reference (ground truth) network. If NULL, a STRINGdb high-confidence physical interaction network (human, score > 900) is used.
<code>false_plot</code>	Logical. If TRUE, an additional plot of False Positives (FP) is generated. Default is FALSE.

Details

If no `reference_matrix` is provided, the function automatically queries STRINGdb to generate a high-confidence physical interaction network.

The plots differentiate:

- Confirmed Edges (TP or CE): Present in both consensus and reference.
- Missing Edges (FN or ME): Present in reference but absent in consensus.
- Extra Edges (FP or EE): Present in consensus but absent in reference (only if `false_plot = TRUE`).

Value

A ggplot object visualizing the comparison. If `false_plot = TRUE`, a combined plot of True Positives / False Negatives and False Positives is returned.

Note

Requires the **igraph**, **ggraph**, **patchwork**, **Matrix**, and **STRINGdb** packages.

Examples

```
data(count_matrices)
data(adj_truth)
networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
```

```
    method = "GENIE3",
    quantile_threshold = 0.99,
    nCores = 1,
    debug = TRUE
  )
  head(binary_listj[[1]])

consensus <- create_consensus(binary_listj, method = "vote")

compare_consensus(
  consensus,
  false_plot = TRUE
)

compare_consensus(
  consensus,
  reference_matrix = adj_truth,
  false_plot = TRUE
)
```

count_matrices

Toy gene expression dataset for examples

Description

A simulated dataset generated using `zinb_simdata()` for demonstrating functions in the `scGraphVerse` package.

Usage

```
data(count_matrices)
```

Format

A list with 3 matrices, each of dimension 38 genes x 40 cells.

Examples

```
data(count_matrices)
str(count_matrices)
```


create_consensus

*Create a Consensus Adjacency Matrix from Multiple Networks***Description**

Builds a consensus adjacency matrix from a list of networks using one of three methods: "vote", "union", or "INet".

Usage

```
create_consensus(
  adj_matrix_list,
  method = "vote",
  weighted_list = NULL,
  theta = 0.04,
  threshold = 0.5,
  ncores = 1
)
```

Arguments

adj_matrix_list	A list of binary adjacency matrices (square, 0/1) with identical dimensions and matching row/column names.
method	Character string specifying the consensus strategy. One of: <ul style="list-style-type: none"> "vote" (default): An edge is included if supported by at least threshold fraction of matrices. "union": An edge is included if present in any matrix. "INet": Combines normalized weighted matrices using consensusNet.
weighted_list	A list of weighted adjacency matrices (required if method = "INet").
theta	Numeric. Tuning parameter passed to consensusNet (default: 0.04).
threshold	Numeric between 0 and 1. Threshold for "vote" and "INet" methods. Default is 0.5.
ncores	Integer. Number of CPU cores to use when method = "INet". Default is 1.

Details

Consensus construction depends on the selected method:

vote Counts the presence of each edge across all matrices and includes edges supported by at least threshold × N matrices.

union Includes any edge that appears in any matrix.

INet Multiplies binary matrices by corresponding weighted matrices, normalizes the results, and applies consensusNet to generate a consensus network.

For "INet", both binary and weighted adjacency matrices must be provided with matching dimensions.

Value

A square consensus adjacency matrix (binary or weighted, depending on the method).

Examples

```
data(count_matrices)

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
  quantile_threshold = 0.99,
  nCores = 1,
  debug = TRUE
)
head(binary_listj[[1]])

consensus <- create_consensus(binary_listj, method = "vote")
head(consensus)
```

cutoff_adjacency

Threshold Adjacency Matrices Based on Shuffled Network Quantiles

Description

Applies a cutoff to weighted adjacency matrices using a percentile estimated from shuffled versions of the original expression matrices. Supports inference methods "GENIE3", "GRNBoost2", and "JRF".

Usage

```
cutoff_adjacency(
  count_matrices,
  weighted_adjm_list,
  n,
  method = "GENIE3",
  quantile_threshold = 0.99,
  weight_function = "mean",
```

```

nCores = 1,
grnboost_modules = NULL,
debug = FALSE
)

```

Arguments

count_matrices	A list of expression matrices (genes \times cells) or Seurat or SingleCellExperiment objects.
weighted_adjm_list	A list of weighted adjacency matrices (one per expression matrix) to threshold.
n	Integer. Number of shuffled replicates generated per original expression matrix.
method	Character string. One of "GENIE3", "GRNBoost2", or "JRF".
quantile_threshold	Numeric. The quantile used to define the cutoff. Default is 0.99.
weight_function	Character string or function used to symmetrize adjacency matrices ("mean", "max", etc.).
nCores	Integer. Number of CPU cores to use for parallelization. Default is the number of workers in the current BiocParallel backend.
grnboost_modules	Python modules needed for GRNBoost2 if using reticulate.
debug	Logical. If TRUE, prints detailed progress messages. Default is FALSE.

Details

For each input expression matrix, n shuffled versions are generated by randomly permuting each gene's expression across cells. Network inference is performed on the shuffled matrices, and a cutoff is determined as the specified quantile (`quantile_threshold`) of the resulting edge weights. The original weighted adjacency matrices are then thresholded using these estimated cutoffs.

Parallelization is handled via **BiocParallel**.

The methods are based on:

- **GENIE3**: Random Forest-based inference (Huynh-Thu et al., 2010).
- **GRNBoost2**: Gradient boosting trees using arboreto (Moerman et al., 2019).
- **JRF**: Joint Random Forests across multiple conditions (Petrulia et al., 2015).

Value

A list of binary (thresholded) adjacency matrices, each corresponding to an input weighted matrix.

Examples

```

data(count_matrices)

networks <- infer_networks(
  count_matrices_list = count_matrices,

```

```

      method = "GENIE3",
      nCores = 1
    )
  head(networks[[1]])

  wadj_list <- generate_adjacency(networks)
  swadj_list <- symmetrize(wadj_list, weight_function = "mean")

  binary_listj <- cutoff_adjacency(
    count_matrices = count_matrices,
    weighted_adjm_list = swadj_list,
    n = 2,
    method = "GENIE3",
    quantile_threshold = 0.99,
    nCores = 1,
    debug = TRUE
  )
  head(binary_listj[[1]])

```

download_Atlas

Download and Load an RDS File from a URL

Description

Downloads an RDS file from a specified URL and reads its contents into R. We used it for <https://www.singlecellatlas.org>

Usage

```
download_Atlas(file_url)
```

Arguments

`file_url` Character; URL of the RDS file to download.

Details

This function uses **httr** to perform the download. The RDS file is read directly from a raw connection without saving to disk. An internet connection is required.

If the download fails (e.g., invalid URL, server error), an informative error message is returned.

Value

An R object loaded from the downloaded RDS file.

Examples

```
url <- paste0(
  "https://www.dropbox.com/s/r8qwsng79rhp9gf/",
  "SCA_scrNASEQ_TISSUE_WHOLE_BLOOD.RDS?dl=1"
)
atlas_data <- download_Atlas(url)
```

earlyj

*Modify Cell Names and Combine Datasets***Description**

Modifies cell identifiers of each element in a list of matrices, [Seurat](#) objects, or [SingleCellExperiment](#) objects by appending a unique matrix index (e.g., "-m1", "-m2", etc.). After renaming, the datasets are merged into a single object.

Usage

```
earlyj(input_list, rowg = TRUE)
```

Arguments

input_list	A list of matrices, Seurat objects, or SingleCellExperiment objects. All elements must be of the same class.
rowg	Logical. If TRUE (default), genes are assumed to be rows and cells columns. If FALSE, matrices are transposed before renaming and combining.

Details

For matrices, this function optionally transposes the input before combining. For [Seurat](#) and [SingleCellExperiment](#) objects, only features (genes) common across all input datasets are retained before merging. The cell names are suffixed with "-m1", "-m2", etc., according to their original list position.

Value

A combined matrix, [Seurat](#) object, or [SingleCellExperiment](#) object with modified (unique) cell names.

Examples

```
data(count_matrices)

earlym <- earlyj(count_matrices)
head(earlym)
```

Description

Query PubMed for literature evidence supporting predicted gene–gene interactions.

Usage

```
edge_mining(
  predicted_list,
  ground_truth,
  delay = 1,
  query_field = "Title/Abstract",
  query_edge_types = c("TP", "FP", "FN"),
  max_retries = 10,
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

<code>predicted_list</code>	A list of predicted adjacency matrices (row and column names are gene symbols).
<code>ground_truth</code>	A 0/1 adjacency matrix with row and column names.
<code>delay</code>	Numeric. Seconds to wait between consecutive queries (default = 1).
<code>query_field</code>	Character. PubMed search field. Options: "Title/Abstract" (default), "Title", "Abstract".
<code>query_edge_types</code>	Character vector. Edge types to query: c("TP", "FP", "FN") (default all).
<code>max_retries</code>	Integer. Max retries for PubMed queries (default = 10).
<code>BPPARAM</code>	A BiocParallel parameter object. Default = <code>bpparam()</code> .

Details

This function compares predicted adjacency matrices against a ground truth matrix, identifies edge types (TP, FP, FN), and queries PubMed for each gene pair. Returns counts of hits, PMIDs, and query status.

Value

A named list of data.frames. Each data.frame has columns:

gene1 First gene in interaction
gene2 Second gene
edge_type One of "TP", "FP", or "FN"

pubmed_hits Number of PubMed hits
PMIDs Comma-separated PubMed IDs or NA
query_status One of "hits_found", "no_hits", or "error"

Examples

```
data(count_matrices)
data(adj_truth)

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
  quantile_threshold = 0.99,
  nCores = 1,
  debug = TRUE
)
head(binary_listj[[1]])

consensus <- create_consensus(binary_listj, method = "vote")
head(consensus)
em <- edge_mining(list(consensus), adj_truth, query_edge_types = "TP")
```

generate_adjacency	<i>Generate Adjacency Matrices from Gene Interaction Tables</i>
--------------------	---

Description

Constructs adjacency matrices from a list of data frames, each representing weighted gene–gene interactions.

Usage

```
generate_adjacency(df_list, nCores = 1)
```

Arguments

<code>df_list</code>	A list of data frames. Each data frame must have three columns: Gene1 Character. First gene in the interaction. Gene2 Character. Second gene in the interaction. Weight Numeric. Weight or strength of the interaction from Gene1 to Gene2.
<code>nCores</code>	Integer. Number of CPU cores to use for parallel processing. Defaults to the number of available workers from the current BiocParallel backend.

Details

The function first identifies all unique genes across all data frames to define the matrix dimensions. For each interaction table, it places the corresponding weights at the appropriate gene-pair positions. Parallelization is handled by **BiocParallel** for improved performance on multiple datasets.

Missing weights (NA) are ignored during construction. Only gene pairs matching the global gene list are inserted.

Value

A list of square numeric adjacency matrices. Each matrix has genes as row and column names, and weights as values. Diagonal entries are set to zero (no self-interactions).

Examples

```
data("count_matrices")

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
head(wadj_list[[1]])
```

infer_networks

Infer Gene Regulatory Networks from Expression Matrices

Description

Infers weighted gene regulatory networks (GRNs) from one or more expression matrices using different inference methods: "GENIE3", "GRNBoost2", "ZILGM", "JRF", or "PCzinb".

Usage

```
infer_networks(
  count_matrices_list,
  method = c("GENIE3", "GRNBoost2", "ZILGM", "JRF", "PCzinb"),
  adjm = NULL,
  nCores = 1,
  grnboost_modules = NULL
)
```

Arguments

count_matrices_list	A list of expression matrices (genes \times cells) or Seurat or SingleCellExperiment objects.
method	Character string. Inference method to use. One of: "GENIE3", "GRNBoost2", "ZILGM", "JRF", or "PCzinb".
adjm	Optional. Reference adjacency matrix for matching dimensions when using "ZILGM" or "PCzinb".
nCores	Integer. Number of CPU cores to use for parallelization. Defaults to the number of workers in the current BiocParallel backend.
grnboost_modules	Python modules required for GRNBoost2 (created via reticulate).

Details

Each expression matrix is preprocessed automatically depending on its object type (Seurat, SingleCellExperiment, or plain matrix).

Parallelization behavior:

- **GENIE3** and **ZILGM**: No external parallelization; internal nCores parameter controls computation.
- **GRNBoost2** and **PCzinb**: Parallelized across matrices using **BiocParallel**.
- **JRF**: Joint modeling of all matrices together; internal parallelization across random forest trees using **doParallel**.

Methods are based on:

- **GENIE3**: Random Forest-based inference (Huynh-Thu et al., 2010).
- **GRNBoost2**: Gradient boosting trees using arboreto (Moerman et al., 2019).
- **ZILGM**: Zero-Inflated Graphical Models for scRNA-seq (Zhang et al., 2021).
- **JRF**: Joint Random Forests across multiple conditions (Petràlia et al., 2015).
- **PCzinb**: Pairwise correlation under ZINB models (Li et al., 2020).

Value

A list of inferred networks:

- For "GENIE3", "GRNBoost2", "ZILGM", and "PCzinb", a list of inferred network objects (edge lists or adjacency matrices).
- For "JRF", a list of data frames with inferred edge lists for each condition or dataset.

Examples

```
data("count_matrices")

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])
```

init_py

Initialize Python Environment for GRNBoost2

Description

Sets up the Python environment and lazily loads modules required for running GRNBoost2: `arboreto`, `pandas`, and `numpy`.

Usage

```
init_py(python_path = "/usr/bin/python3", required = TRUE)
```

Arguments

<code>python_path</code>	Character string. Path to the Python executable, e.g., <code>"/usr/bin/python3"</code> .
<code>required</code>	Logical. If TRUE, errors if Python is not available or path is invalid. Default: TRUE.

Details

Uses **reticulate** to bind R to the specified Python interpreter and lazily import modules needed for GRNBoost2. If a module is missing or incompatible, an informative error is raised (when `required = TRUE`).

Value

A list with three Python module objects:

- `arboreto`: GRNBoost2 algorithm module.
- `pandas`: Data handling module.
- `numpy`: Numerical operations module.

Examples

```
# Initialize Python environment (adjust python_path as needed)
modules <- init_py()
```

plotg

Visualize Graphs from Adjacency Matrices

Description

Generates and arranges multiple graph visualizations from a list of adjacency matrices. Each matrix is converted into an undirected **igraph** object and visualized using a force-directed layout via **ggraph**.

Usage

```
plotg(adj_matrix_list)
```

Arguments

adj_matrix_list

A list of square, symmetric adjacency matrices with zeros on the diagonal (no self-loops). Each matrix represents an undirected graph.

Details

Each adjacency matrix is validated to ensure it is square and symmetric. Disconnected nodes (degree zero) are removed prior to visualization. Graphs are visualized with a force-directed layout using **ggraph**, and multiple plots are arranged into a grid with **gridExtra**.

Each subplot title includes the graph index, number of nodes, and number of edges.

Value

A grid of plots displaying all valid graphs in the input list.

Note

This function requires the following packages: **igraph**, **ggraph**, and **gridExtra**. If any are missing, an informative error will be thrown.

Examples

```
data(count_matrices)

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
```

```

head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
  quantile_threshold = 0.99,
  nCores = 1,
  debug = TRUE
)
head(binary_listj[[1]])
plotg(binary_listj)

```

plotROC

*Plot ROC Curves for Inferred Networks***Description**

Computes and visualizes Receiver Operating Characteristic (ROC) curves for a list of predicted adjacency matrices compared against a binary ground truth network.

Usage

```
plotROC(matrices_list, ground_truth, plot_title, is_binary = FALSE)
```

Arguments

<code>matrices_list</code>	A list of square matrices representing predicted interactions. Each must share dimnames with <code>ground_truth</code> ; entries may be binary (0/1) or continuous weights.
<code>ground_truth</code>	A square binary matrix indicating true interactions (1) in the upper triangle. Must match dims and names of each element of <code>matrices_list</code> .
<code>plot_title</code>	Character string. Title for the ROC plot.
<code>is_binary</code>	Logical. If TRUE, treat matrices as binary predictions. Default FALSE for weighted predictions.

Details

For binary matrices, a single TPR/FPR point is computed per matrix. For weighted ones, a full ROC curve is computed from continuous scores. Diagonals are ignored; symmetry is not enforced.

Value

A list with:

- `auc`: data frame of AUC per matrix.
- `plot`: the ROC plot (via `ggplot2`).

Examples

```

data(count_matrices)
data(adj_truth)

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

roc_res <- plotROC(
  swadj_list,
  adj_truth,
  plot_title = "ROC Curve: GENIE3",
  is_binary = FALSE
)
roc_res$plot
auc_joint <- roc_res$auc

```

pscores

*Compute Performance Scores for Predicted Adjacency Matrices***Description**

Computes classification metrics by comparing predicted adjacency matrices to a ground truth binary network and visualizes the performance via a radar (spider) plot.

Usage

```
pscores(ground_truth, predicted_list, zero_diag = TRUE)
```

Arguments

ground_truth	A square binary adjacency matrix representing the ground truth network. Values must be 0 or 1. Only the upper triangle is used for evaluation.
predicted_list	A list of predicted adjacency matrices to evaluate. Each matrix must have the same dimensions and row/column names as ground_truth.
zero_diag	Logical. If TRUE (default), sets the diagonal of ground_truth to zero before evaluation, removing self-loops.

Details

For each predicted matrix, the confusion matrix is computed using the upper triangle (non-self edges). Metrics including True Positive Rate (TPR), False Positive Rate (FPR), Precision, F1-score, and Matthews Correlation Coefficient (MCC) are calculated.

A radar plot is automatically generated summarizing the key scores across matrices.

Value

A list with one element:

Statistics: Data frame of evaluation metrics (TP, TN, FP, FN, TPR, FPR, Precision, F1, MCC) for each predicted matrix.

Note

Requires the **fmsb**, **dplyr**, and **tidyr** packages.

Examples

```
data(count_matrices)
data(adj_truth)

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")

binary_listj <- cutoff_adjacency(
  count_matrices = count_matrices,
  weighted_adjm_list = swadj_list,
  n = 2,
  method = "GENIE3",
  quantile_threshold = 0.99,
  nCores = 1,
  debug = TRUE
)

pscores_data <- pscores(adj_truth, binary_listj)
```

Description

Identifies and returns the top *n* most highly expressed genes across all cells or within a specific cell type. Supports objects of class [Seurat](#), [SingleCellExperiment](#), or a numeric expression matrix (genes \times cells).

Usage

```
selgene(
  object,
  top_n,
  cell_type = NULL,
  cell_type_col = "cell_type",
  assay = NULL,
  remove_mt = FALSE,
  remove_rib = FALSE
)
```

Arguments

<code>object</code>	A Seurat object, SingleCellExperiment object, or numeric matrix (genes \times cells).
<code>top_n</code>	Integer. Number of top expressed genes to return.
<code>cell_type</code>	Optional string. If provided, filters the expression matrix to only include cells of this type.
<code>cell_type_col</code>	Character. Name of the column in metadata (Seurat <code>meta.data</code> or SCE <code>colData</code>) containing cell type annotations. Default is "cell_type".
<code>assay</code>	Character. For SingleCellExperiment objects only. Name of the assay to use. If NULL, defaults to "logcounts".
<code>remove_mt</code>	Logical. If TRUE, remove mitochondrial genes matching "^MT-" (case-insensitive).
<code>remove_rib</code>	Logical. If TRUE, remove ribosomal genes matching "^RP[SL]" (case-insensitive).

Details

The function assumes that log-normalized values are available in the "data" slot (for Seurat objects) or the "logcounts" assay (for SingleCellExperiment). If raw counts are provided as a matrix, no transformation is applied.

Optional filtering is available to exclude mitochondrial genes ("^MT-") and ribosomal genes ("^RP[SL]"), which may otherwise dominate the top expressed genes.

Value

A character vector of the top *n* most highly expressed gene names.

Details

When using a Seurat object, the function retrieves the log-normalized data from the default assay's "data" slot. For SingleCellExperiment, it uses the specified assay (default is "logcounts"). For matrices, no checks or transformations are applied, and subsetting by cell type is not supported.

Mitochondrial and ribosomal gene removal is based on regular expressions matching gene names. These should follow standard naming conventions (e.g., MT-ND1, RPL13A, RPS6).

See Also

[Seurat](#), [SingleCellExperiment](#)

Examples

```
url <- paste0(
  "https://www.dropbox.com/s/r8qwsng79rhp9gf/",
  "SCA_scrNASEQ_TISSUE_WHOLE_BLOOD.RDS?dl=1"
)
seu <- download_Atlas(file_url = url)
genes <- selgene(
  object = seu,
  top_n = 50,
  cell_type = "T_cells",
  cell_type_col = "CELL_TYPE",
  remove_rib = TRUE,
  remove_mt = TRUE
)
```

stringdb_adjacency	<i>Build Adjacency Matrices for Physical Interactions from STRING (POST API)</i>
--------------------	--

Description

Constructs weighted and binary adj matrices for physical protein-protein interactions using a POST request to the STRING database API.

Usage

```
stringdb_adjacency(
  genes,
  species = 9606,
  required_score = 400,
  keep_all_genes = TRUE,
  verbose = TRUE
)
```

Arguments

genes	A character vector of gene symbols or identifiers, e.g., c("TP53", "BRCA1", ...).
species	Integer. NCBI taxonomy ID of the species. Default is 9606 (human).
required_score	Integer in 0,1000. Minimum confidence score for interactions. Default is 400.

`keep_all_genes` Logical. If TRUE (default), includes all input genes in the final matrix even if unmapped.

`verbose` Logical. If TRUE, displays progress messages. Default is TRUE.

Details

This function:

1. Maps input genes to STRING internal IDs.
2. Uses a POST request to retrieve physical protein-protein interactions from STRING.
3. Builds a weighted adjacency matrix using the STRING combined score.
4. Builds a binary adjacency matrix indicating presence/absence.

Genes not mapped to STRING are optionally retained as zero rows/columns if `keep_all_genes = TRUE`.

Value

A list containing:

- `weighted`: A square numeric adjacency matrix with scores as weights.
- `binary`: A corresponding binary (0/1) adjacency matrix.

Note

Requires packages: **STRINGdb**, **httr**, **jsonlite**.

Examples

```
url <- paste0(
  "https://www.dropbox.com/s/r8qwsng79rhp9gf/",
  "SCA_scrnaseq_TISSUE_WHOLE_BLOOD.RDS?dl=1"
)
seu <- download_Atlas(file_url = url)
genes <- selgene(
  object = seu,
  top_n = 50,
  cell_type = "T_cells",
  cell_type_col = "CELL_TYPE",
  remove_rib = TRUE,
  remove_mt = TRUE
)
str_res <- stringdb_adjacency(
  genes = genes,
  species = 9606,
  required_score = 900,
  keep_all_genes = FALSE
)
wadj_truth <- str_res$weighted
adj_truth <- str_res$binary
```

symmetrize

Symmetrize a List of Square Matrices

Description

Symmetrizes each square matrix in a list by ensuring entries (i, j) and (j, i) are identical, using a specified combination function.

Usage

```
symmetrize(matrix_list, weight_function = "mean", nCores = 1)
```

Arguments

matrix_list	A list of square numeric matrices to symmetrize.
weight_function	Character string or function. Method to combine entries (i, j) and (j, i). Options include "mean", "max", "min", or a user-defined function.
nCores	Integer. Number of CPU cores to use for parallel processing. Defaults to the number of available workers in the current BiocParallel backend.

Details

For each pair of off-diagonal elements (i, j) and (j, i):

- If one value is zero, the non-zero value is used.
- If both are non-zero, they are combined using the specified weight_function.

Diagonal entries are preserved as-is and not modified.

Parallelization is managed via **BiocParallel** for improved performance.

Value

A list of symmetric matrices, where for each matrix $A[i, j] = A[j, i]$ for all $i \neq j$.

Examples

```
data("count_matrices")

networks <- infer_networks(
  count_matrices_list = count_matrices,
  method = "GENIE3",
  nCores = 1
)
head(networks[[1]])

wadj_list <- generate_adjacency(networks)
swadj_list <- symmetrize(wadj_list, weight_function = "mean")
```

zinb_simdata	<i>Simulate Zero-Inflated Negative Binomial (ZINB) Count Matrices with Sequencing Depth</i>
--------------	---

Description

Simulates one or more count matrices following a zero-inflated negative binomial (ZINB) distribution, incorporating gene-gene interaction structures and cell-specific sequencing depth variation.

Usage

```
zinb_simdata(
  n,
  p,
  B,
  mu_range,
  mu_noise,
  theta,
  pi,
  kmat = 1,
  depth_range = NA
)
```

Arguments

n	Integer. Number of cells (samples) in each simulated matrix.
p	Integer. Number of genes (features) in each simulated matrix.
B	A symmetric binary adjacency matrix (0/1) defining gene-gene connectivity. Row and column names correspond to gene names.
mu_range	List of numeric vectors (length 2 each). Range of gene expression means for each simulated matrix.
mu_noise	Numeric vector. Mean of background noise for each matrix.
theta	Numeric vector. Dispersion parameters of the negative binomial distribution for each matrix. Smaller theta implies higher overdispersion.
pi	Numeric vector. Probability of excess zeros ($0 < \pi < 1$) for each matrix.
kmat	Integer. Number of count matrices to simulate. Default is 1.
depth_range	Numeric vector of length 2 or NA. Range of total sequencing depth per cell. If NA, no depth adjustment is performed.

Details

Each simulated matrix:

1. Generates gene expression values based on a ZINB model.
2. Modulates expression using the adjacency matrix B.

3. Applies random sequencing depth scaling if `depth_range` is provided.

Useful for benchmarking single-cell RNA-seq network inference methods with dropout events and network structure.

Value

A list containing `kmat` matrices. Each matrix has:

- Rows representing cells (`cell_1`, ..., `cell_n`).
- Columns representing genes (`rownames(B)`).
- Count values following a ZINB distribution.

Examples

```
data(adj_truth)
nodes <- nrow(adj_truth)
sims <- zinb_simdata(
  n = 50,
  p = nodes,
  B = adj_truth,
  mu_range = list(c(1, 4), c(1, 7), c(1, 10)),
  mu_noise = c(1, 3, 5),
  theta = c(1, 0.7, 0.5),
  pi = c(0.2, 0.2, 0.2),
  kmat = 3,
  depth_range = c(0.8 * nodes * 3, 1.2 * nodes * 3)
)
```

Index

* datasets

- adj_truth, [3](#)
- count_matrices, [8](#)

adj_truth, [3](#)

community_path, [3](#)
community_similarity, [5](#)
compare_consensus, [6](#)
consensusNet, [9](#)
count_matrices, [8](#)
create_consensus, [9](#)
cutoff_adjacency, [10](#)

download_Atlas, [12](#)

earlyj, [13](#)
edge_mining, [14](#)

generate_adjacency, [15](#)

infer_networks, [16](#)
init_py, [18](#)

plotg, [19](#)
plotROC, [20](#)
pscores, [21](#)

selgene, [22](#)
Seurat, [11](#), [13](#), [17](#), [23](#), [24](#)
SingleCellExperiment, [11](#), [13](#), [17](#), [23](#), [24](#)
stringdb_adjacency, [24](#)
symmetrize, [26](#)

zinb_simdata, [27](#)