Package 'scGraphVerse'

May 12, 2025

Title scGraphVerse: A Gene Regulatory Network Analysis Package

Version 0.1.0

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://github.com/francesco-cecere/scGraphVerse

BugReports https://github.com/francesco-cecere/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

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown,
tidyverse,
magick

VignetteBuilder knitr

Contents

community_path . . . 3
community_similarity . . . 4
compare_consensus . . . 6
create_consensus . . . 8
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
stringdb_adjacency . . . 24
symmetrize . . . 25
zinb_simdata . . . 26

Index **28**

community_path *Community Detection and Pathway Enrichment Analysis*

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

```
genes <- c(
    "TP53",   "BRCA1", "EGFR",   "MYC",     "CDKN1A",
    "BCL2",   "MDM2",  "PTEN",   "AKT1",    "MAPK1"
)

adj <- matrix(
    0,
    nrow = length(genes),
    ncol = length(genes),
    dimnames = list(genes, genes)
)

edge_list <- list(
    c("TP53", "MDM2"),
    c("TP53", "CDKN1A"),
    c("BRCA1", "BCL2"),
    c("PTEN", "AKT1"),
    c("EGFR", "MAPK1"),
    c("MYC", "CDKN1A"),
    c("MYC", "BCL2"),
    c("AKT1", "MAPK1")
)

for (e in edge_list) {
    adj[e[1], e[2]] <- 1
    adj[e[2], e[1]] <- 1
}

diag(adj) <- 0
result <- community_path(
    adj_matrix = adj,
    methods    = "louvain",
    pathway_db = "KEGG",
    genes_path = 2,
    plot       = TRUE,
    verbose    = TRUE
)
```

`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

```
genes <- c(
    "TP53",   "BRCA1", "EGFR",   "MYC",     "CDKN1A",
    "BCL2",   "MDM2",  "PTEN",   "AKT1",    "MAPK1"
)

adj <- matrix(
    0,
    nrow = length(genes),
    ncol = length(genes),
    dimnames = list(genes, genes)
)

edge_list <- list(
    c("TP53", "MDM2"),
    c("TP53", "CDKN1A"),
    c("BRCA1", "BCL2"),
    c("PTEN", "AKT1"),
```

```
    c("EGFR", "MAPK1"),
    c("MYC", "CDKN1A"),
    c("MYC", "BCL2"),
    c("AKT1", "MAPK1")
)

for (e in edge_list) {
    adj[e[1], e[2]] <- 1
    adj[e[2], e[1]] <- 1
}

diag(adj) <- 0
control <- community_path(
    adj_matrix = adj,
    methods    = "louvain",
    pathway_db = "KEGG",
    genes_path = 2,
    plot       = FALSE,
    verbose    = FALSE
)

pred1_adj <- adj
pred1_adj["TP53", "MDM2"] <- pred1_adj["MDM2", "TP53"] <- 0

pred2_adj <- adj
pred2_adj["MYC", "PTEN"] <- pred2_adj["PTEN", "MYC"] <- 1
pred1 <- community_path(
    adj_matrix = pred1_adj,
    methods = "louvain",
    pathway_db = "KEGG",
    genes_path = 2,
    plot = FALSE,
    verbose = FALSE
)

pred2 <- community_path(
    adj_matrix = pred2_adj,
    methods    = "louvain",
    pathway_db = "KEGG",
    genes_path = 2,
    plot       = FALSE,
    verbose    = FALSE
)

comparison <- community_similarity(
    control_output = control,
    predicted_list = list(pred1, pred2)
)
```

`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

`consensus_matrix`	A binary square adjacency matrix representing the consensus network. Row and column names should correspond to gene symbols.
`reference_matrix`	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.
`false_plot`	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

```
set.seed(42)

# Simulate small example matrices
original <- matrix(
    sample(0:1, 25, replace = TRUE, prob = c(0.8, 0.2)),
    5, 5
)
consensus <- matrix(
    sample(0:1, 25, replace = TRUE, prob = c(0.8, 0.2)),
    5, 5
)
diag(original) <- diag(consensus) <- 0
rownames(original) <- colnames(original) <- paste0("Gene", 1:5)
rownames(consensus) <- colnames(consensus) <- paste0("Gene", 1:5)

# Compare consensus network to original network
compare_consensus(
    consensus,
    reference_matrix = original,
    false_plot       = TRUE
)
```

`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:

- `"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

```
mat1 <- matrix(sample(0:1, 25, replace = TRUE), nrow = 5)
mat2 <- matrix(sample(0:1, 25, replace = TRUE), nrow = 5)
rownames(mat1) <- colnames(mat1) <- paste0("Gene", 1:5)
rownames(mat2) <- colnames(mat2) <- paste0("Gene", 1:5)

consensus_vote <- create_consensus(
    list(mat1, mat2),
    method    = "vote",
    threshold = 0.5
)

consensus_union <- create_consensus(
    list(mat1, mat2),
    method = "union"
)
```

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 × 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 (Petralia et al., 2015).

Value

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

Examples

```
set.seed(123)

# Simulate two small expression matrices
mat1 <- matrix(rpois(100, lambda = 5), nrow = 10, ncol = 10)
mat2 <- matrix(rpois(100, lambda = 5), nrow = 10, ncol = 10)
rownames(mat1) <- paste0("Gene", 1:10)
rownames(mat2) <- paste0("Gene", 1:10)

# Infer networks using GENIE3
network_list <- infer_networks(
    count_matrices_list = list(mat1, mat2),
    method = "GENIE3",
    nCores = 2
)

# Convert inferred networks into adjacency matrices
list_wadj <- generate_adjacency(network_list)
list_swadj <- symmetrize(
    list_wadj,
    weight_function = "mean"
)

# Apply cutoff based on shuffled quantiles
binary_adjm_list <- cutoff_adjacency(
    count_matrices     = list(mat1, mat2),
    weighted_adjm_list = list_swadj,
    n                  = 2,
    method             = "GENIE3",
    quantile_threshold = 0.95,
    nCores             = 2,
    debug              = TRUE
)
```

```
# Inspect one thresholded adjacency matrix
binary_adjm_list[[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

```
# Example with matrices where genes are rows (default behavior)
mat1 <- matrix(rpois(100, 5), nrow = 10, ncol = 10)
mat2 <- matrix(rpois(100, 5), nrow = 10, ncol = 10)
rownames(mat1) <- paste0("Gene", 1:10)
rownames(mat2) <- paste0("Gene", 1:10)

combined_matrix <- earlyj(list(mat1, mat2))

# Example with matrices where genes are columns
mat3 <- t(mat1)
mat4 <- t(mat2)

combined_matrix2 <- earlyj(list(mat3, mat4), rowg = FALSE)
```

edge_mining *Edge Mining of Gene Interactions Using PubMed*

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

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

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

```
set.seed(123)
predicted <- matrix(rbinom(100, 1, 0.1), nrow = 10)
rownames(predicted) <- colnames(predicted) <-
    paste0("Gene", 1:10)
ground_truth <- matrix(rbinom(100, 1, 0.05), nrow = 10)
rownames(ground_truth) <- colnames(ground_truth) <-
    paste0("Gene", 1:10)

results <- edge_mining(
    predicted_list = list(pred_net = predicted),
    ground_truth = ground_truth
)
head(results$pred_net)
```

---

generate_adjacency *Generate Adjacency Matrices from Gene Interaction Tables*

Description

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

Usage

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

Arguments

`df_list`	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`.
`nCores`	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

```
# Create two simple interaction tables
df1 <- data.frame(
    Gene1 = c("GeneA", "GeneB"),
    Gene2 = c("GeneB", "GeneC"),
    Weight = c(0.5, 0.8)
)

df2 <- data.frame(
    Gene1 = c("GeneC"),
    Gene2 = c("GeneA"),
    Weight = 1.0
)

# Generate adjacency matrices
adjacency_list <- generate_adjacency(list(df1, df2))

# View one of the matrices
adjacency_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 = "GENIE3",
  adjm = NULL,
  nCores = 1,
  grnboost_modules = NULL
)
```

Arguments

`count_matrices_list`
: A list of expression matrices (genes × 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 (Petralia 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

```
set.seed(123)

# Simulate two small expression matrices
mat1 <- matrix(rpois(100, lambda = 5), nrow = 10)
mat2 <- matrix(rpois(100, lambda = 5), nrow = 10)
rownames(mat1) <- paste0("Gene", 1:10)
rownames(mat2) <- paste0("Gene", 1:10)

# Infer networks using GENIE3
```

```
networks <- infer_networks(
    count_matrices_list = list(mat1, mat2),
    method = "GENIE3",
    nCores = 2
)

# Inspect first inferred network
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

`python_path`	Character string. Path to the Python executable, e.g., `"/usr/bin/python3"`.
`required`	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()
# Use numpy to generate random numbers
modules$numpy$random$rand(5)
```

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

```
adj1 <- matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), nrow = 3)
adj2 <- matrix(c(0, 1, 1, 1, 0, 0, 1, 0, 0), nrow = 3)
plotg(list(adj1, adj2))
```

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

matrices_list	A list of square matrices representing predicted interactions. Each must share dimnames with ground_truth; entries may be binary (0/1) or continuous weights.
ground_truth	A square binary matrix indicating true interactions (1) in the upper triangle. Must match dims and names of each element of matrices_list.
plot_title	Character string. Title for the ROC plot.
is_binary	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

```
mat1 <- matrix(runif(100), nrow = 10)
mat2 <- matrix(runif(100), nrow = 10)
gt <- matrix(sample(c(0, 1), 100, TRUE), nrow = 10)
diag(gt) <- 0
gt[lower.tri(gt)] <- 0
plotROC(
    matrices_list = list(mat1, mat2),
    ground_truth  = gt,
    plot_title    = "ROC for Network Inference",
    is_binary     = FALSE
)
```

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

```
# Simulate ground truth and predictions
ground_truth <- matrix(
    sample(0:1, 100, replace = TRUE),
    nrow = 10
)
diag(ground_truth) <- 0
pred1 <- ground_truth
```

```
pred2 <- matrix(
    sample(0:1, 100, replace = TRUE),
    nrow = 10
)

# Compute scores and generate radar plot
result <- pscores(
    ground_truth,
    list(pred1, pred2)
)
result$Statistics
```

`selgene` *Select Top Expressed Genes from Single-Cell Data*

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 × cells).

Usage

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

Arguments

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

```
library(SingleCellExperiment)

expr_mat <- matrix(rnorm(1000, mean = 3), nrow = 100, ncol = 10)
rownames(expr_mat) <- c(
    paste0("MT-", 1:5),
    paste0("RPL", 6:15),
    paste0("Gene", 16:100)
)
colnames(expr_mat) <- paste0("Cell", 1:10)

cell_md <- DataFrame(celltype = rep(c("A", "B"), each = 5))

sce <- SingleCellExperiment(
    assays   = list(logcounts = expr_mat),
    colData  = cell_md
)

top_genes <- selgene(
    object         = sce,
    top_n          = 10,
    cell_type      = "A",
    cell_type_col  = "celltype",
    remove_mt      = TRUE,
    remove_rib     = TRUE
)
```

`stringdb_adjacency` *Build Adjacency Matrices for Physical Interactions from STRING (POST API)*

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

```
large_gene_set <- c("TP53", "BRCA1", "MYC", "EGFR", "PTEN")
adjacency_list <- stringdb_adjacency(
    genes = large_gene_set,
    species = 9606,
    required_score = 700
)
weighted_mat <- adjacency_list$weighted
binary_mat <- adjacency_list$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

```
mat1 <- matrix(c(0, 2, 3, 4), nrow = 2)
mat2 <- matrix(c(0, 5, 6, 0), nrow = 2)
matrix_list <- list(mat1, mat2)

sym_list <- symmetrize(
    matrix_list,
    weight_function = "mean"
)

sym_list[[1]]
```

`zinb_simdata` *Simulate Zero-Inflated Negative Binomial (ZINB) Count Matrices with Sequencing Depth*

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

```
B <- matrix(0, nrow = 5, ncol = 5)
rownames(B) <- colnames(B) <- paste0("Gene", 1:5)
B[1, 2] <- B[2, 3] <- B[4, 5] <- 1
B <- B + t(B)
zinb_matrices <- zinb_simdata(
    n = 100, p = 5, B = B,
    mu_range = list(c(1, 5), c(2, 6)),
    mu_noise = c(0.5, 0.7),
    theta = c(1, 2),
    pi = c(0.2, 0.3),
    kmat = 2,
    depth_range = c(500, 5000)
)
print(zinb_matrices[[1]])
```

Index

community_path, 3
community_similarity, 4
compare_consensus, 6
consensusNet, *9*
create_consensus, 8
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, *10*, *13*, *16*, *22, 23*
SingleCellExperiment, *10*, *13*, *16*, *22, 23*
stringdb_adjacency, 24
symmetrize, 25

zinb_simdata, 26