Package 'SigBridgeR'

December 23, 2025

Title Multi-algorithm Integration of Phenotypic, scRNA-seq, and Bulk Data for Cell Screening

Version 2.5.4

Description SigBridgeR is an integrative toolkit designed to identify phenotype-associated cell subpopulations by combining phenotype(e.g. survival, drug sensitivity), bulk expression and single-cell RNA-seq data. It leverages multiple algorithms to robustly link cell features with clinical or functional phenotypes. The package provides a unified pipeline for cross-modal data analysis, enabling the discovery of biologically and clinically relevant cell states in heterogeneous samples.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/WangLabCSU/SigBridgeR

BugReports https://github.com/WangLabCSU/SigBridgeR/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports chk,
cli,
data.table,
DEGAS (>= 1.0.0),
dplyr,
edgeR,
glue,
IDConverter (>= 0.3.5),
LPSGL (>= 0.0.0.9000),
Matrix,
methods,
PIPET (>= 0.1.2),
processx,
purrr,
rlang,
scAB (>= 1.0.1),
Scissor (>= 2.0.0),
scPAS (>= 0.2.0),
ScPP (>= 0.0.0.9000),

Seurat (>= 5.0.0),
SeuratObject (>= 5.0.0),
SigBridgeRUtils (>= 0.1.0),
tibble

Depends R (>= 4.1.0)

Remotes bioc::edgeR,
github::Exceret/ALRA,
github::Exceret/DEGAS,
github::Exceret/LPSGL,
github::Exceret/PIPET,
github::Exceret/scAB,
github::Exceret/Scissor,
github::Exceret/scPAS,
github::Exceret/ScPP,
github::WangLabCSU/SigBridgeRUtils,
github::ShixiangWang/IDConverter

Suggests ALRA,
anndata,
carrier,
furrr,
future,
future.mirai,
ggforce,
ggplot2,
ggupset,
ggVennDiagram,
here,
KernSmooth,
knitr,
matrixStats,
matrixTests,
org.Hs.eg.db,
patchwork,
preprocessCore,
randomcoloR,
reticulate,
rmarkdown,
sparseMatrixStats,
testthat (>= 3.0.0),
tidyr,
zeallot

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`AddMetaFeature` *Add Gene-Level Metadata to Seurat Object (Vectorized, ...-based)*

Description

Add multiple feature-level metadata (vectors or 2D tables) to a Seurat object. Handles vectors (length-checked), matrices, data.frames, data.tables, etc. Columns/variables with duplicated names are suffixed (e.g., "type_1"). Gene alignment is auto-detected: rows or cols must match nrow(seurat_obj).

Usage

```
AddMetaFeature(seurat_obj, ..., assay = "RNA")
```

Arguments

seurat_obj	A Seurat object.
...	One or more metadata inputs: • Named/unnamed vectors (length = ngenes) • 2D objects (matrix, data.frame, data.table, etc.) where one dimension (rows or cols) has size = ngenes.
assay	Assay name (default: "RNA", fallback to first).

Value

Modified seurat_obj (invisibly).

AddMisc *Safely Add Miscellaneous Data to Seurat Object*

Description

Adds arbitrary data to the @misc slot of a Seurat object with automatic key conflict resolution. If the key already exists, automatically appends a numeric suffix to ensure unique key naming (e.g., "mykey_1", "mykey_2").

Usage

```
AddMisc(seurat_obj, ..., cover = TRUE)
```

Arguments

seurat_obj	A Seurat object to modify
...	key-value pairs to add to the @misc slot.
cover	Logical indicating whether to overwrite existing data. If (default TRUE).

Value

The modified Seurat object with added @misc data. The original object structure is preserved with no other modifications.

Key Generation Rules

1. If key doesn't exist: uses as-is
2. If key exists: appends the next available number (e.g., "key_1", "key_2")
3. If numbered keys exist (e.g., "key_2"): increments the highest number

Examples

```
## Not run:
# Basic usage
seurat_obj <- AddMisc(seurat_obj, "QC_stats" = qc_df)

# Auto-incrementing example
seurat_obj <- AddMisc(seurat_obj, markers = markers1)
seurat_obj <- AddMisc(seurat_obj, markers = markers2, cover=FALSE)
# Stores as "markers" and "markers_1"


## End(Not run)
```

---

aggregate-dups *Aggregate Rows or Columns with Duplicate Names*

Description

These functions collapse duplicated row names (e.g., gene symbols) or column names (e.g., sample IDs) in matrix-like objects by aggregating values using configurable methods. They support:

Rows `AggregateDupRows`: merges rows sharing the same row name.

Columns `AggregateDupCols`: merges columns sharing the same column name.

Both `AggregateDups`: convenience wrapper applying row-then-column aggregation.

Designed for expression matrices, count tables, or any numeric data where feature/sample duplication occurs. Handles `matrix`, `data.frame`, and S4 `Matrix` classes (e.g. `dgCMatrix`) robustly.

Convenience wrapper that first aggregates duplicated rows, then duplicated columns. Useful for cleaning matrices where both feature and sample duplication may occur.

Usage

```
AggregateDupRows(x, method = c("max", "sum", "mean", "median", "first"))

AggregateDupCols(x, method = c("max", "sum", "mean", "median", "first"))

AggregateDups(
  x,
  method = c("max", "sum", "mean", "median", "first"),
  row_method = NULL,
  col_method = NULL
)
```

Arguments

`x`	A numeric matrix-like object (see Details).
`method`	Character scalar. Aggregation method (see Methods below).
`row_method`	Aggregation method for rows. Defaults to `method`.
`col_method`	Aggregation method for columns. Defaults to `method`.

Value

An aggregated object of the same effective type as `x`, with unique row/column names.

Methods

Supported methods (applied column-wise for rows, row-wise for columns):

`"max"` Maximum value per group (default).

`"sum"` Sum of values per group.

`"mean"` Arithmetic mean (uses `na.rm = TRUE`).

`"median"` Median value.

`"first"` First occurrence in original order.

Input Types and Return Types

Input class	Output class (unless noted)
`matrix`	`matrix`
`data.frame`	`data.frame`
S4 `Matrix`	`matrix` (dense) — S4 attributes dropped for generality

Row/column order in output follows *first occurrence* of each unique name in `rownames(x)` / `colnames(x)`.

Examples

```
# Full deduplication in one step
mat <- matrix(1:16, nrow = 4,
              dimnames = list(c("TP53", "TP53", "BRCA1", "ACTB"),
                              c("S1", "S1", "S2", "S3")))
AggregateDups(mat, method = "sum")
#>       S1 S2 S3
#> TP53   5  7  9
#> BRCA1  3  7 11
#> ACTB   4  8 12
```

`BulkPreProcess` *Bulk RNA-seq Data Preprocessing and Quality Control Function*

Description

This function performs comprehensive preprocessing and quality control analysis for bulk RNA-seq data, including data validation, filtering, batch effect detection, principal component analysis, and visualization.

Usage

```
BulkPreProcess(
  data,
  sample_info = NULL,
  gene_symbol_conversion = FALSE,
  check = TRUE,
  min_count_threshold = 10L,
  min_gene_expressed = 3L,
  min_total_reads = 1000000L,
  min_genes_detected = 10000L,
  min_correlation = 0.8,
  n_top_genes = 500L,
  show_plot_results = TRUE,
  ...
)
```

Arguments

`data`	Expression matrix with genes as rows and samples as columns, or a list containing count_matrix and sample_info
`sample_info`	Sample information data frame (optional), ignored if data is a list. A qualified `sample_info` should contain both `sample` and `condition` columns (case-sensitive), and there are no specific requirements for the data type stored in the `condition` column. `batch` column is optional, which is used for batch effect detection.
`gene_symbol_conversion`	Whether to convert Ensembles version IDs and TCGA version IDs to genes with IDConverter, default: FALSE
`check`	Whether to perform detailed quality checks, default: TRUE
`min_count_threshold`	Minimum count threshold for gene filtering, default: 10
`min_gene_expressed`	Minimum number of samples a gene must be expressed in, default: 3
`min_total_reads`	Minimum total reads per sample, default: 1e6
`min_genes_detected`	Minimum number of genes detected per sample, default: 10000
`min_correlation`	Minimum correlation threshold between samples, default: 0.8
`n_top_genes`	Number of top variable genes for PCA analysis, default: 500
`show_plot_results`	Whether to generate visualization plots, default: TRUE
`...`	Additional arguments. Currently supports: • `verbose`: Logical indicating whether to print progress messages. Defaults to `TRUE`. • `seed`: For reproducibility, default is `123L`

Details

The function performs the following operations:

1. Data validation and format conversion
2. Basic statistics calculation (missing values, read depth, gene detection)
3. Optional detailed quality checks including:
   - Sample correlation analysis
   - Principal Component Analysis (PCA)
   - Outlier detection using Mahalanobis distance
   - Batch effect detection using ANOVA
4. Data filtering based on count thresholds and quality metrics
5. Optional gene symbol conversion
6. Visualization generation (PCA plots)

Value

Filtered count matrix

Quality Metrics

The function calculates and reports several quality metrics:

Data Integrity Number of missing values
Gene Count Total number of genes after filtering
Sample Read Depth Total reads per sample
Gene Detection Rate Number of genes detected per sample
Sample Correlation Pearson correlation between samples
PCA Variance Variance explained by first two principal components
Batch Effects Proportion of genes significantly affected by batch

Sample Information Format

The sample_info data frame should contain:

sample Character vector of unique sample identifiers
condition Character vector of experimental conditions
batch Optional character vector of batch identifiers

Filtering Criteria

Genes are retained if:

- They have counts >= min_count_threshold in >= min_gene_expressed samples

Samples are retained if:

- Total reads >= min_total_reads
- Detected genes >= min_genes_detected
- Mean correlation >= min_correlation (if check=TRUE)

See Also

cpm for counts per million calculation, prcomp for PCA analysis, cor for correlation analysis, SymbolConvert for gene symbol conversion

---

DoDEGAS *Run DEGAS Analysis for Single-Cell and Bulk RNA-seq Data Integration*

Description

This function performs DEGAS to integrate single-cell and bulk RNA-seq data, identifying phenotype-associated cells using a bootstrap aggregated multi-task learning approach.

Usage

```
DoDEGAS(
  select_fraction = 0.05,
  min_thresh = 0.4,
  matched_bulk,
  sc_data,
  phenotype = NULL,
  sc_data.pheno_colname = NULL,
  label_type = "DEGAS",
  phenotype_class = c("binary", "continuous", "survival"),
  tmp_dir = "tmp",
  env_params = list(),
  degas_params = list(),
  normality_test_method = c("jarque-bera", "d'agostino", "kolmogorov-smirnov"),
  ...
)
```

Arguments

`select_fraction`
: The top percentage of selected cells will be considered as Positive cells, without considering how much larger the possible correlation coefficient of the observation group is compared to that of the control group. Only usedl when `phenotype_class` is "binary" or "survival". (default: `0.05`)

`min_thresh`
: DEGAS will calculate the possible correlation coefficients for each cell related to the phenotype. When the coefficient of the observation group is at least `min_thresh` larger than that of the control group, it can be considered related to the phenotype and will be marked as Positive. The priority of `min_thresh` is higher than that of `select_fraction`. (default: `0.4`)

`matched_bulk`
: Bulk RNA-seq data as matrix or data.frame (rows=genes, columns=samples)

`sc_data`
: Single-cell data as Seurat object containing RNA assay

`phenotype` Bulk-level phenotype data. For classification: binary matrix with one-hot encoding. For survival: matrix with two columns (time and event status). Can be NULL, matrix, data.frame, or vector.

`sc_data.pheno_colname`
Column name for single-cell phenotype in metadata (if available), default: `NULL`

`label_type` Label type for DEGAS results (default: `"DEGAS"`)

`phenotype_class`
Type of phenotype: "binary" (classification), "continuous", or "survival"

`tmp_dir` Temporary directory for intermediate files (default: `"tmp"`)

`env_params` List of environment parameters for Python setup including:

- env.name: environment name (default: `"r-reticulate-degas"`)
- env.type: environment type "conda", "environment", or "venv" (default: `"environment"`)
- env.method: environment setup method "system", "conda" (default: `"system"`)
- env.file: path to environment file (default: system.file("conda/DEGAS_environment.yml", package = "SigBridgeR"))
- env.python_version: Python version (default: "3.9.15")
- env.packages: named vector of Python packages and versions (default: c("tensorflow" = "2.4.1", "protobuf" = "3.20" ,"numpy" = "any"))
- env.recreate: whether to recreate environment (default: `FALSE`)
- env.use_conda_forge: whether to use conda-forge channel (conda only, default: `TRUE`)
- env.verbose: verbose output (default: `FALSE`)

`degas_params` List of DEGAS algorithm parameters including:

- DEGAS.model_type: model type ("BlankClass", "ClassBlank", "ClassClass", "ClassCox", "BlankCox")
- DEGAS.architecture: "Standard" (feed forward) or "DenseNet" (dense net), default: `"DenseNet"`
- DEGAS.ff_depth: number of layers in model (>=1, default: `3`)
- DEGAS.pyloc: path to Python executable (default: `NULL`, automatic detection)
- DEGAS.bag_depth: bootstrap aggregation depth (>=1, default: `5`)
- DEGAS.train_steps: training steps (default: `2000`)
- DEGAS.scbatch_sz: single-cell batch size (default: `200`)
- DEGAS.patbatch_sz: patient batch size (default: `50`)
- DEGAS.hidden_feats: hidden features (default: `50`)
- DEGAS.do_prc: dropout percentage (default: `0.5`)
- DEGAS.lambda1: regularization parameter 1 (default: `3.0`)
- DEGAS.lambda2: regularization parameter 2 (default: `3.0`)
- DEGAS.lambda3: regularization parameter 3 (default: `3.0`)
- DEGAS.seed: random seed (default: `2`)

`normality_test_method`
Method for normality testing: "jarque-bera", "d'agostino", or "kolmogorov-smirnov"

... Additional arguments. Currently supports:

- `verbose`: Logical indicating whether to print progress messages. Defaults to `TRUE`.

Details

The function performs the following steps:

1. Validates input data and parameters
2. Sets up Python environment with required dependencies
3. Trains bootstrap aggregated DEGAS model using `runCCMTLBag`
4. Generates cell-level predictions using `predClassBag`
5. Applies statistical testing to identify phenotype-associated cells
6. Labels cells as "Positive" or "Other" based on selection criteria

Model type is automatically determined:

- BlankClass: only bulk phenotype specified (scLab = NULL)
- ClassBlank: only single-cell phenotype specified (patLab = NULL)
- ClassClass: both single-cell and bulk phenotypes specified
- ClassCox: single-cell phenotype + bulk survival data
- BlankCox: only bulk survival data specified

Value

A list containing:

- scRNA_data: Seurat object with DEGAS labels added to metadata
- model: The model trained using the input data, andit can be used for cell classification prediction.
- DEGAS_prediction: Data table with DEGAS predictions containing:
  - Predicted label probabilities for each cell
  - Cell labels ("Positive"/"Other") based on selection criteria
  - Difference scores for binary phenotypes
  - Cell identifiers

References

Johnson TS, Yu CY, Huang Z, Xu S, Wang T, Dong C, et al. Diagnostic Evidence GAuge of Single cells (DEGAS): a flexible deep transfer learning framework for prioritizing cells in relation to disease. Genome Med. 2022 Feb 1;14(1):11.

See Also

Other screen_method: `DoLP_SGL()`, `DoPIPET()`, `DoScissor()`, `DoscAB()`, `DoscPAS()`, `DoscPP()`

Other DEGAS: `DEGASEnvSet()`, `DEGASFindPy()`, `DEGASModelDetect()`, `DEGASParamSet()`

Examples

```
## Not run:
# Binary classification example
result <- DoDEGAS(
  select_fraction = 0.05, # `select_fraction` only used in binary and survival phenotyping
  matched_bulk = bulk_matrix,
  sc_data = seurat_obj,
  phenotype = bulk_phenotype,
  phenotype_class = "binary"
)

# Survival analysis example
result <- DoDEGAS(
  select_fraction = 0.05, # `select_fraction` only used in binary and survival phenotyping
  matched_bulk = bulk_matrix,
  sc_data = seurat_obj,
  phenotype = survival_data,
  phenotype_class = "survival"
)

## End(Not run)
```

DoLP_SGL *Perform LP-SGL Screening Analysis*

Description

Identifies phenotype-associated cell subpopulations using Lasso-Penalized Sparse Group Lasso (LP-SGL) with Leiden community detection. This method integrates bulk and single-cell RNA-seq data to identify cell subpopulations associated with phenotypic outcomes.

Usage

```
DoLP_SGL(
  matched_bulk,
  sc_data,
  phenotype,
  label_type = "LP_SGL",
  LPSGL_family = c("logit", "cox", "linear"),
  resolution = 0.6,
  alpha = 0.5,
  nfold = 5,
  dge_analysis = list(run = FALSE, logFC_threshold = 1, pval_threshold = 0.05),
  ...
)
```

Arguments

matched_bulk	Bulk expression matrix (features × samples)
sc_data	Single-cell RNA-seq data (Seurat object)
phenotype	Binary phenotype vector for bulk samples
label_type	Character specifying phenotype label type (default: "LP_SGL")
LPSGL_family	Type of regression model: "logit" (logistic), "cox" (Cox), or "linear" (linear regression)
resolution	Resolution parameter for Leiden clustering (default: 0.6)
alpha	Alpha parameter for SGL balancing L1 and L2 penalties (default: 0.5)
nfold	Number of folds for cross-validation (default: 5)
dge_analysis	List controlling differential expression analysis: • run: Whether to run DEG analysis (default: FALSE) • logFC_threshold: Log fold change threshold (default: 1) • pval_threshold: P-value threshold (default: 0.05)
...	Additional arguments passed to preprocessing functions, e.g.: • verbose: Whether to print progress messages (default: TRUE) • seed: Random seed for reproducibility (default: 123L)

Value

A list containing:

scRNA_data	Seurat object with LP-SGL results integrated
sgl_fit	Fitted SGL model object
cvfit	Cross-validation results
dge_res	Differential expression results if requested (NULL otherwise)

References

Li J, Zhang H, Mu B, Zuo H, Zhou K. Identifying phenotype-associated subpopulations through LP_SGL. Briefings in Bioinformatics. 2023 Nov 22;25(1):bbad424.

See Also

Other screen_method: DoDEGAS(), DoPIPET(), DoScissor(), DoscAB(), DoscPAS(), DoscPP()

Examples

```
## Not run:
# Example using simulated data
set.seed(123)

# Create simulated data
bulk_data <- matrix(rnorm(1000*50), nrow=1000, ncol=50)
sc_data <- matrix(rnorm(1000*500), nrow=1000, ncol=500)
```

```
phenotype <- rep(c(0, 1), each=25)

# Run LP-SGL analysis
results <- DoLP_SGL(
matched_bulk = bulk_data,
sc_data = sc_data,
phenotype = phenotype,
LPSGL_family = "logit",
resolution = 0.6,
dge_analysis = list(run = TRUE, logFC_threshold = 1, pval_threshold = 0.05)
)

# Access results
lpsgl_seurat <- results$scRNA_data
sgl_model <- results$sgl_fit
deg_results <- results$dge_res

## End(Not run)
```

DoPIPET *Perform PIPET Screening Analysis*

Description

Predicts cell subpopulations in single-cell data by matching expression profiles to predefined marker gene templates using various distance/similarity metrics. This function implements a template-based classification approach with permutation testing for significance assessment.

Usage

```
DoPIPET(
  matched_bulk,
  sc_data,
  phenotype,
  phenotype_class = c("binary", "continuous"),
  group = NULL,
  discretize_method = c("kmeans", "median", "custom"),
  cutoff = NULL,
  label_type = "PIPET",
  log2FC = 1L,
  p_adjust = 0.05,
  show_log2FC = TRUE,
  freq_counts = NULL,
  normalize = TRUE,
  scale = TRUE,
  nPerm = 1000L,
 distance = c("cosine", "pearson", "spearman", "kendall", "euclidean", "maximum"),
  ...
)
```

Arguments

matched_bulk	Normalized bulk expression matrix (features × samples). Column names must match phenotype identifiers.
sc_data	Seurat object containing single-cell RNA-seq data.
phenotype	Clinical outcome data. Can be: - Vector: named with sample IDs - Data frame: with row names matching bulk columns
phenotype_class	Analysis mode: - "binary": Case-control design (e.g., responder/non-responder) - "continuous": Continuous outcome (e.g.,)
group	A character, name of one metadata column to group cells by (for example, orig.ident). The default value is NULL. In this case, screening will be performed on each group separately.
discretize_method	c("median", "kmeans", "custom"). Discretization strategy for continuous phenotypes. Note: "median" is mapped internally to "quantile" (2-group quantile split). Default: "median".
cutoff	Numeric vector of length n_group - 1. Required only when discretize_method = "custom". Defines interior breakpoints on the *normalized, log2-transformed scale* (i.e., after scale(log2(x + 1))). Must be sorted in ascending order.
label_type	Character specifying phenotype label type (e.g., "SBS1", "time"), stored in scRNA_data@misc
log2FC	In the DESeq differential expression analysis results, the cutoff value of log2FC. The default value is 1L.
p_adjust	In the DESeq differential expression analysis results, the cutoff value of adjust P. The default value is 0.05.
show_log2FC	Select whether to show log2 fold changes. The default value is TRUE.
freq_counts	An integer, keep genes expressed in more than a certain number of cells. The default value is NULL, which means no filtering.
normalize	Select whether to perform normalization of count data. The default value is TRUE.
scale	Select whether to scale and center features in the dataset. The default value is TRUE.
nPerm	An integer, number of permutations to do. The default value is 1000L.
distance	A character, the distance algorithm must be included in "cosine", "pearson", "spearman", "kendall","euclidean","maximum". default value is NULL, which means "cosine".
...	Additional arguments to be passed to PIPET.optimized. • seed: Random seed for reproducibility • verbose: Whether to show progress messages • parallel: Whether to use parallel processing, default is FALSE. future::plan() must be set before calling this function.

See Also

Other screen_method: DoDEGAS(), DoLP_SGL(), DoScissor(), DoscAB(), DoscPAS(), DoscPP()

DoscAB *Perform scAB Screening Analysis*

Description

Implements the scAB algorithm to identify phenotype-associated cell subpopulations in single-cell RNA-seq data by integrating matched bulk expression and phenotype information. Uses non-negative matrix factorization (NMF) with dual regularization for phenotype association and cell-cell similarity.

Usage

```
DoscAB(
  matched_bulk,
  sc_data,
  phenotype,
  label_type = "scAB",
  phenotype_class = c("binary", "survival"),
  alpha = c(0.005, NULL),
  alpha_2 = c(0.005, NULL),
  maxiter = 2000L,
  tred = 2L,
  ...
)
```

Arguments

matched_bulk	Normalized bulk expression matrix (genes × samples) where: - Columns match phenotype row names - Genes match features in sc_data
sc_data	Seurat object containing preprocessed single-cell data:
phenotype	Data frame with clinical annotations where: - Rows correspond to matched_bulk columns - For survival: contains time and status columns
label_type	Character specifying phenotype label type (e.g., "SBS1", "time"), stored in scRNA_data@misc
phenotype_class	Analysis mode: - "binary": Case-control design (e.g., responder/non-responder) - "survival": Time-to-event analysis data.frame
alpha	Coefficient of phenotype regularization (default=0.005).
alpha_2	Coefficent of cell-cell similarity regularization (default=0.005).
maxiter	Maximum number of iterations for NMF (default=2000).
tred	Z-score threshold in finding subsets (default=2).
...	Additional arguments. Currently supports: • verbose: Logical indicating whether to print progress messages. Defaults to TRUE. • seed: For reproducibility, default is 123L

Value

A list containing:

scRNA_data Filtered Seurat object with selected cells

scAB_result scAB screening result

LICENSE

Licensed under the GNU General Public License version 3 (GPL-3.0). A copy of the license is available at https://www.gnu.org/licenses/gpl-3.0.en.html.

References

Zhang Q, Jin S, Zou X. scAB detects multiresolution cell states with clinical significance by integrating single-cell genomics and bulk sequencing data. Nucleic Acids Research. 2022 Nov 28;50(21):12112–30.

See Also

Other screen_method: DoDEGAS(), DoLP_SGL(), DoPIPET(), DoScissor(), DoscPAS(), DoscPP()

Examples

```
## Not run:
# Binary phenotype example
result <- DoscAB(
  matched_bulk = bulk_matrix,
  sc_data = seurat_obj,
  phenotype = clinical_df,
  label_type = "disease_status",
  phenotype_class = "binary",
  alpha = 0.005,
  alpha_2 = 0.005,
  maxiter = 2000,
  tred = 2
)

## End(Not run)
```

DoScissor *Perform Scissor Screening Analysis*

Description

Identifies phenotype-associated cell subpopulations in single-cell data using regularized regression on matched bulk expression profiles. Scissor integrates bulk and single-cell RNA-seq data to identify cells that are significantly associated with phenotypic outcomes.

Usage

```
DoScissor(
   path2load_scissor_cache = NULL,
   path2save_scissor_inputs = "Scissor_inputs.RData",
   matched_bulk,
   sc_data,
   phenotype,
   label_type = "scissor",
   alpha = c(0.05, NULL),
   cutoff = 0.2,
   scissor_family = c("gaussian", "binomial", "cox"),
   reliability_test = list(
     run = FALSE, # whether to run reliability test
     n = 10L, # permutation times
     nfold = 10L # cross validation folds
   ),
   cell_evaluation = list(
     run = FALSE, # whether to run cell evaluation
     benchmark_data = "path_to_file.RData", # path to benchmark data
     FDR_cutoff = 0.05,
     bootstrap_n = 100L
   ),
   ...
)
```

Arguments

path2load_scissor_cache	Path to precomputed Scissor inputs (RData file). If provided, skips recomputation (default: NULL).
path2save_scissor_inputs	Path to save intermediate files (default: "Scissor_inputs.RData").
matched_bulk	Normalized bulk expression matrix (features × samples). Column names must match phenotype identifiers.
sc_data	Seurat object containing single-cell RNA-seq data.
phenotype	Clinical outcome data. Can be: - Vector: named with sample IDs - Data frame: with row names matching bulk columns
label_type	Character specifying phenotype label type (e.g., "SBS1", "time"), stored in scRNA_data@misc
alpha	Parameter used to balance the effect of the l1 norm and the network-based penalties. It can be a number or a searching vector. If alpha = NULL, a default searching vector is used. The range of alpha is between 0 and 1. A larger alpha lays more emphasis on the l1 norm.
cutoff	(default: 0.2). When alpha=NULL, the cutoff is used to determine the optimal alpha. Higher values increase specificity.
scissor_family	Model family for outcome type: - "gaussian": Continuous outcomes - "binomial": Binary outcomes (default) - "cox": Survival outcomes

`reliability_test`
: List controlling reliability testing:
  - run: Whether to perform reliability test (default: FALSE)
  - n: Permutation times (default: `10L`)
  - nfold: Cross-validation folds (default: `10L`)

`cell_evaluation`
: List controlling cell evaluation:
  - run: Whether to perform cell evaluation (default: FALSE)
  - benchmark_data: Path to benchmark data (RData file)
  - FDR_cutoff: FDR threshold for evaluation (default: `0.05`)
  - bootstrap_n: Bootstrap iterations (default: `100L`)

`...`
: Additional arguments. Currently supports:
  - `verbose`: Logical indicating whether to print progress messages. Defaults to `TRUE`.
  - `seed`: For reproducibility, default is `123L`

Value

A list containing:

scRNA_data A Seurat object with screened cells containing metadata:

scissor "Positive"/"Negative"/"Neutral" classification

label_type Outcome label used

scissor_result Raw Scissor results

reliability_result If reliability_test=TRUE, contains:

statistic A value between 0 and 1

p p-value of the test statistic

AUC_test_real 10 values of AUC for real data

AUC_test_back A list of AUC for background data

cell_evaluation If cell_evaluation=TRUE, contains:

evaluation_res A data.frame with some supporting information for each Scissor selected cell

LICENSE

Licensed under the GNU General Public License version 3 (GPL-3.0). A copy of the license is available at https://www.gnu.org/licenses/gpl-3.0.en.html.

References

Sun D, Guan X, Moran AE, Wu LY, Qian DZ, Schedin P, et al. Identifying phenotype-associated subpopulations by integrating bulk and single-cell sequencing data. Nat Biotechnol. 2022 Apr;40(4):527–38.

See Also

Other screen_method: `DoDEGAS()`, `DoLP_SGL()`, `DoPIPET()`, `DoscAB()`, `DoscPAS()`, `DoscPP()`

Other scissor: `DoScissorCellEval()`, `DoScissorRelTest()`

Examples

```
## Not run:
# Binary outcome example
res <- DoScissor(
  matched_bulk = bulk_matrix,
  sc_data = seurat_obj,
  phenotype = a_named_vector,
  scissor_family = "binomial"
)

## End(Not run)
```

DoscPAS *Perform scPAS Screening Analysis*

Description

This function performs scPAS screening analysis by integrating bulk and single-cell RNA-seq data. It includes data filtering steps and wraps the core scPAS::scPAS function.

Usage

```
DoscPAS(
  matched_bulk,
  sc_data,
  phenotype,
  label_type = "scPAS",
  assay = "RNA",
  imputation = FALSE,
  imputation_method = c("KNN", "ALRA"),
  nfeature = 3000L,
  alpha = c(0.01, NULL),
  cutoff = 0.2,
  network_class = c("SC", "bulk"),
  scPAS_family = c("cox", "gaussian", "binomial"),
  permutation_times = 2000L,
  FDR_threshold = 0.05,
  independent = TRUE,
  ...
)
```

Arguments

matched_bulk	Bulk RNA-seq data (genes x samples)
sc_data	Single-cell RNA-seq data (Seurat object and preprocessed)
phenotype	Phenotype data frame with sample annotations

label_type — Character specifying phenotype label type (e.g., "SBS1", "time"), stored in scRNA_data@misc. (default: "scPAS")

assay — Assay to use from sc_data (default: 'RNA')

imputation — Logical, whether to perform imputation (default: FALSE)

imputation_method — Character. Name of alternative method for imputation. (options: "KNN", "ALRA")

nfeature — Number of features to select (default: 3000, indicating that the top 3000 highly variable genes are selected for model training

alpha — Numeric. Significance threshold. Parameter used to balance the effect of the l1 norm and the network-based penalties. It can be a number or a searching vector. If alpha = NULL, a default searching vector is used. The range of alpha is in [0,1]. A larger alpha lays more emphasis on the l1 norm. (default: 0.01)

cutoff — Numeric. Cutoff value for selecting the optimal alpha value when alpha = NULL. (default: 0.2)

network_class — Network class to use (default: 'SC', indicating gene-gene similarity networks derived from single-cell data. The other one is 'bulk'.)

scPAS_family — Model family for analysis (options: "cox", "gaussian", "binomial")

permutation_times — Number of permutations to perform (default: 2000)

FDR_threshold — Numeric. FDR value threshold for identifying phenotype-associated cells (default: 0.05)

independent — Logical. The background distribution of risk scores is constructed independently of each cell. (default: TRUE)

... — Additional arguments. Currently supports:

- verbose: Logical indicating whether to print progress messages. Defaults to TRUE.
- seed: For reproducibility, default is 123L

Value

A Seurat object from scPAS analysis

LICENSE

Licensed under the GNU General Public License version 3 (GPL-3.0). A copy of the license is available at https://www.gnu.org/licenses/gpl-3.0.en.html.

References

Xie A, Wang H, Zhao J, Wang Z, Xu J, Xu Y. scPAS: single-cell phenotype-associated subpopulation identifier. Briefings in Bioinformatics. 2024 Nov 22;26(1):bbae655.

See Also

Other screen_method: DoDEGAS(), DoLP_SGL(), DoPIPET(), DoScissor(), DoscAB(), DoscPP()

DoscPP *Perform scPP screening analysis*

Description

This function performs scPP screening on single-cell data using matched bulk data and phenotype information. It supports binary, continuous, and survival phenotype types.

Usage

```
DoscPP(
  matched_bulk,
  sc_data,
  phenotype,
  label_type = "scPP",
  phenotype_class = c("Binary", "Continuous", "Survival"),
  ref_group = 0,
  Log2FC_cutoff = 0.585,
  estimate_cutoff = 0.2,
  probs = c(0.2, NULL),
  ...
)
```

Arguments

matched_bulk	Bulk expression data (genes × samples) where: - Column names must match phenotype row names
sc_data	Seurat object containing preprocessed single-cell data: - Normalized counts in RNA assay
phenotype	Data frame or tibble or named vector with: - Rownames matching matched_bulk columns - For survival: must contain time and status columns
label_type	Character specifying phenotype label type (e.g., "SBS1"), stored in scRNA_data@misc
phenotype_class	Analysis type (case-sensitive): - "Binary": Case-control studies (e.g., tumor/normal) - "Continuous": Quantitative traits (e.g., drug response) - "Survival": Time-to-event data (requires time/status columns)
ref_group	Reference group or baseline for **binary** comparisons, e.g. "Normal" for Tumor/Normal studies and 0 for 0/1 case-control studies. (default: 0)
Log2FC_cutoff	Minimum log2 fold-change for binary markers (default: 0.585)
estimate_cutoff	Effect size threshold for **continuous** traits (default: 0.2)
probs	A numeric value indicating the quantile cutoff for cell classification. This parameter can also be a numeric vector, in which case an optimal threshold will be selected based on the AUC and enrichment score.(default: 0.2)

... Additional arguments. Currently supports:

- verbose: Logical indicating whether to print progress messages. Defaults to TRUE.
- seed: For reproducibility, default is 123L

Value

A list containing:

scRNA_data Seurat object with added metadata:

ScPP "Positive"/"Negative"/"Neutral" classification

gene_list List of genes used for screening

Algorithm Steps

1. Data Validation: Checks sample alignment between bulk and phenotype data
2. Marker Selection: Identifies phenotype-associated genes from bulk data
3. Single-cell Screening: Projects bulk markers onto single-cell data
4. Cell Classification: Categorizes cells based on phenotype association

Reference

WangX-Lab/ScPP [Internet]. [cited 2025 Aug 31]. Available from: https://github.com/WangX-Lab/ScPP

See Also

Other screen_method: DoDEGAS(), DoLP_SGL(), DoPIPET(), DoScissor(), DoscAB(), DoscPAS()

Examples

```
## Not run:
# Binary phenotype analysis
res <- DoscPP(
  matched_bulk = bulk_data,
  sc_data = seurat_obj,
  phenotype = ms_data,
  label_type = "SBS1",
  phenotype_class = "Binary"
)

# Survival analysis
surv_res <- DoscPP(
  sc_data = seurat_obj,
  matched_bulk = bulk_data,
  phenotype = surv_df,
  label_type = "OS_status",
  phenotype_class = "Survival"
)
```

```
## End(Not run)
```

`FindRobustElbow` *Automatically determine optimal PCA dimensions using multiple robust methods*

Description

This function combines multiple statistical approaches to automatically determine the optimal number of principal components (PCs) for downstream single-cell analysis. It integrates variance-based heuristics, elbow detection algorithms, and provides comprehensive visualization for result validation.

Usage

```
FindRobustElbow(
  obj,
  verbose = SigBridgeRUtils::getFuncOption("verbose"),
  ndims = 50L
)
```

Arguments

`obj`	A Seurat object that has PCA computed (after `RunPCA`)
`verbose`	Logical, if TRUE outputs detailed method results and creates visualization plot. If FALSE returns only the final dimension.
`ndims`	Integer, maximum number of dimensions to consider (default: `50L`)

Value

Integer, the recommended number of PCA dimensions for downstream analysis

See Also

Other single_cell_preprocess: `FilterTumorCell()`, `Pattern2Colname()`, `ProcessSeuratObject()`, `QCPatternDetect()`

Examples

```
## Not run:
# After running PCA on Seurat object
pbmc <- RunPCA(pbmc, npcs = 50)
optimal_dims <- FindRobustElbow(pbmc, verbose = TRUE)
pbmc <- FindNeighbors(pbmc, dims = 1:optimal_dims)

## End(Not run)
```

getFuncOption *Configuration Functions for SigBridgeR Package*

Description

These functions provide a centralized configuration system for the SigBridgeR package, allowing users to set and retrieve package-specific options with automatic naming conventions.

setFunOption sets one or more configuration options for the SigBridgeR package. Options are automatically prefixed with "SigBridgeR." if not already present, ensuring proper namespace isolation.

getFuncOption retrieves configuration options for the SigBridgeR package. The function automatically handles the "SigBridgeR." prefix, allowing users to reference options with or without the explicit prefix.

checkFuncOption validates configuration options for the SigBridgeR package. This function ensures that all options meet type and value requirements before they are set in the global options.

Usage

```
getFuncOption(option, default = NULL)
```

Arguments

option	Character string specifying the option name to check
default	The default value to return if the option is not set. Defaults to NULL.

Details

This function performs the following validations:

verbose, parallel Must be single logical values (TRUE/FALSE)

parallel.type Must be a single character string

workers, timeout, seed Must be single integer values

The function is automatically called by setFuncOption to ensure all configuration options are valid before they are set.

Value

Invisibly returns NULL. The function is called for its side effects of setting package options.

The value of the specified option if it exists, otherwise the provided default value.

No return value. The function throws an error if the value doesn't meet the required specifications for the given option.

Available Opions

- verbose: A logical value indicating whether to print verbose messages, defaults to TRUE
- timeout: An integer specifying the timeout in seconds for parallel processing, defaults to ‘180L“
- seed: An integer specifying the random seed for reproducible results, defaults to ‘123L“

Examples

```
# Set options with automatic prefixing
setFuncOption(verbose = TRUE)

# Retrieve options with automatic prefixing
getFuncOption("verbose")
```

ListPyEnv *List Available Python Environments*

Description

Discovers and lists available Python environments of various types on the system. This generic function provides a unified interface to find Conda environments and virtual environments (venv) through S3 method dispatch.

Default method that lists all Python environments by combining results from Conda and virtual environment discovery methods.

Discovers Conda environments using multiple detection strategies for maximum reliability. First attempts to use system Conda commands, then falls back to reticulate's built-in Conda interface if Conda command is unavailable or fails. Returns empty data frame if Conda is not available or no environments are found.

Discovers virtual environments by searching common venv locations including user directories (‘~/.virtualenvs‘, ‘~/.venvs‘) and project folders (‘./venv‘, ‘./.venv‘). Supports custom search paths through the ‘venv_locations‘ parameter. Returns empty data frame if no virtual environments are found in the specified locations.

Usage

```
ListPyEnv(
  env_type = c("all", "conda", "venv", "virtualenv"),
  timeout = 30000,
  venv_locations = c("~/.virtualenvs", "~/.venvs", "./venv", "./.venv"),
  verbose = TRUE,
  ...
)
```

Arguments

env_type	Character string specifying the type of environments to list. One of: ‘"all"‘, ‘"conda"‘, ‘"venv"‘. Defaults to ‘"all"‘.
timeout	The maximum timeout time when using system commands, only effective when ‘env_type=conda‘.
venv_locations	Character vector of directory paths to search for virtual environments. Default includes standard locations and common project directories.
verbose	Logical indicating whether to print verbose output.
...	For future use.

Details

The function uses S3 method dispatch to handle different environment types:

- **‘"all"‘**: Combines results from all environment types using ‘rbind()‘ - **‘"conda"‘**: Searches for Conda environments using multiple methods: - Primary: ‘reticulate::conda_list()‘ for reliable environment detection - Fallback: System ‘conda info –envs‘ command for broader compatibility - **‘"venv"‘**: Searches common virtual environment locations including user directories and project folders

Each method includes comprehensive error handling and will return empty results with informative warnings if no environments are found or if errors occur during discovery.

Value

A data frame with the following columns:

- ‘name‘ - Character vector of environment names
- ‘python‘ - Character vector of paths to Python executables
- ‘type‘ - Character vector indicating environment type (‘"conda"‘ or ‘"venv"‘)

Returns an empty data frame with these columns if no environments are found.

Examples

```
## Not run:
# List all Python environments
ListPyEnv("all")

# List only Conda environments
ListPyEnv("conda")

# List only virtual environments with custom search paths
ListPyEnv("venv", venv_locations = c("~/my_envs", "./project_env"))

## End(Not run)
```

`LoadRefData` *Download & Load Reference Data*

Description

This function checks if the data already exists in a local cache. If not, it downloads the file from the remote repository using multiple sources with fallback.

Usage

```
LoadRefData(
  data_type = c("survival", "binary", "continuous"),
  path = NULL,
  timeout = SigBridgeRUtils::getFuncOption("timeout"),
  ...
)
```

Arguments

`data_type`	The type of data to download. Must be one of "survival", "binary", or "continuous".
`path`	Optional path to save the downloaded file, default: NULL, saving in package.
`timeout`	Integer. Connection timeout in seconds (default: 60)
`...`	Additional arguments. Currently supports: • `verbose`: Logical indicating whether to print progress messages. Defaults to `TRUE`.

Value

The requested datasets, stored in a list.

Examples

```
## Not run:
ref_data <- LoadRefData(data_type = c("survival"))

## End(Not run)
```

`MergeResult` *Merge Multiple Screening Analysis Results*

Description

Combines results from multiple single-cell screening analyses (Scissor, scPAS, scPP, or scAB) by merging their metadata and miscellaneous information while preserving the original expression data. Performs an inner join on cell barcodes to ensure only cells present in all inputs are retained.

Usage

```
MergeResult(..., verbose = SigBridgeRUtils::getFuncOption("verbose"))
```

Arguments

`...`	Input objects to merge. Can be: - Seurat objects - Lists containing `scRNA_data` (Seurat objects) - Mixed combinations of the above - The first one will be used as base object for merging, priority given to first one when duplicate columns are found
`verbose`	Logical, whether to print a message

Value

A merged Seurat object containing:

- Expression data from the first input object
- Combined metadata from all input objects
- Miscellaneous information from all input objects
- Only cells present in all input objects (inner join)

Processing Details

1. Input Validation: Checks for valid Seurat objects or lists containing Seurat objects
2. Metadata Extraction: Collects metadata from all objects
3. Cell Intersection: Retains only cells present in all datasets
4. Object Merging: Creates new Seurat object with combined metadata
5. Miscellaneous: Adds miscellaneous information to the merged object

Examples

```
## Not run:
# Merge mixed analysis types
combined <- MergeResult(scissor_output, scAB_output, scPP_output)

# Merge list-containing objects
merged_list <- MergeResult(list1, list2, seurat_obj)
```

```
## End(Not run)
```

`QCFilter` *Filter Seurat object cells by QC metrics*

Description

Filters cells based on nFeature_RNA and optional QC metrics (e.g. percent.mt, percent.rp), defined in `seurat_obj@misc$qc_colnames` (See `QCPatternDetect`). Only metrics with non-constant, non-all-zero values are used.

Usage

```
QCFilter(
  seurat_obj,
 data_filter.thresh = list(nFeature_RNA_thresh = c(200L, 6000L), percent.mt = 20L,
    percent.rp = 60L),
  verbose = TRUE,
  ...
)
```

Arguments

`seurat_obj`	A `Seurat` object.
`data_filter.thresh`	A named list with thresholds. Default: `list(nFeature_RNA_thresh = c(200L, 6000L), percent.mt = 20L, percent.rp = 60L)`. Keys not in default are treated as QC column names.
`verbose`	Logical; whether to print progress messages via `cli`.
`...`	No use

Value

A filtered `Seurat` object.

QCPatternDetect *Calculate Percentage of Features Matching Patterns*

Description

This function calculates the percentage of counts coming from features matching specified patterns (e.g., mitochondrial genes, ribosomal genes) and adds them as metadata columns to the Seurat object.

Usage

```
QCPatternDetect(
  obj,
  pattern = c("^MT-", "^mt-", "^RP[SL]", "^MT-|^RP[SL]"),
  verbose = TRUE,
  ...
)
```

Arguments

obj	A seurat object.
pattern	A character vector or list containing regex patterns to identify mitochondrial genes, ribosomal protein genes, or other unwanted genes, as well as combinations of these genes. Customized patterns are supported.
verbose	logical, whether to print progress messages
...	Additional arguments passed to PercentageFeatureSet

Details

The function automatically generates friendly column names based on the patterns:

- "mt" for mitochondrial patterns
- "rp" for ribosomal patterns
- "rrna" for ribosomal RNA patterns
- For combined patterns (using |), creates names like "mt_rp"
- For other patterns, creates cleaned lowercase names

See Also

Other single_cell_preprocess: FilterTumorCell(), FindRobustElbow(), Pattern2Colname(), ProcessSeuratObject()

SCPreProcess *Single-Cell RNA-seq Preprocessing Pipeline*

Description

A generic function for standardized preprocessing of single-cell RNA-seq data from multiple sources. Handles data.frame/matrix, AnnData, and Seurat inputs with tumor cell filtering. Implements a complete analysis pipeline from raw data to clustered embeddings.

Usage

```
SCPreProcess(sc, ...)

## Default S3 method:
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = "SC_Screening_Proj",
  min_cells = 400L,
  min_features = 0L,
  quality_control = TRUE,
  quality_control.pattern = c("^MT-"),
  data_filter = TRUE,
 data_filter.thresh = list(nFeature_RNA_thresh = c(200L, 6000L), percent.mt = 20L,
    percent.rp = 60L),
  normalization_method = "LogNormalize",
  scale_factor = 10000L,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
  dims = NULL,
  ...
)

## S3 method for class 'R6'
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = "SC_Screening_Proj",
  min_cells = 400L,
  min_features = 0L,
  quality_control = TRUE,
  quality_control.pattern = c("^MT-"),
  data_filter = TRUE,
 data_filter.thresh = list(nFeature_RNA_thresh = c(200L, 6000L), percent.mt = 20L,
```

```
    percent.rp = 60L),
  normalization_method = "LogNormalize",
  scale_factor = 10000L,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
  dims = NULL,
  ...
)

## S3 method for class 'Seurat'
SCPreProcess(sc, column2only_tumor = NULL, ...)
```

Arguments

sc	Input data, one of: • data.frame/matrix/dgCMatrix: Raw count matrix (features x cells) • R6: Python AnnData object, obtained via package anndata or anndataR • Seurat: Preprocessed Seurat object
...	Additional arguments passed to specific methods. Currently supports: • verbose: Logical indicating whether to print progress messages. Defaults to TRUE. • dims_Neighbors: Dimensions to use for FindNeighbors. Defaults to NULL, using dims. • dims_TSNE: Dimensions to use for RunTSNE. Defaults to NULL, using dims. • dims_UMAP: Dimensions to use for RunUMAP. Defaults to NULL, using dims.
meta_data	A data.frame containing metadata for each cell. It will be added to the Seurat object as @meta.data. If sc is an anndata object, obs will be automatically used.
column2only_tumor	A character of column names in meta_data, used to filter the Seurat object to only tumor cells. If NULL, no filtering is performed.
project	A character of project name, used to name the Seurat object.
min_cells	Minimum number of cells that must express a feature for it to be included in the analysis. Defaults to 400L.
min_features	Minimum number of features that must be detected in a cell for it to be included in the analysis. Defaults to 0L.
quality_control	Logical, for identification of the proportion of mitochondrial genes, ribosomal protein genes, or other types of genes (without filtering), the results will be stored in meta.data. Defaults to TRUE.
quality_control.pattern	A vector or character containing regex pattern(s) to identify mitochondrial genes, ribosomal protein genes, or other unwanted genes, as well as combinations of these genes. Customized patterns are supported. Defaults to "^MT-".
data_filter	Logical indicating whether to filter cells based on quality metrics. Defaults to TRUE.

`data_filter.thresh`
: A list containing filtering thresholds for different quality metrics:
  - `nFeature_RNA_thresh`: Numeric vector of length 2 specifying the minimum and maximum number of features per cell. Defaults to `c(200L, 6000L)`
  - `percent.mt`: Maximum mitochondrial percentage allowed. Defaults to `20L`
  - `percent.rp`: Maximum ribosomal protein percentage allowed. Not used in default unless ribosomal protein genes filter pattern (like `^RP[LS]`) is added to `quality_control.pattern`

`normalization_method`
: Method for normalization: "LogNormalize", "CLR", or "RC". Defaults to `"LogNormalize"`.

`scale_factor`
: Scaling factor for normalization. Defaults to `10000L`.

`scale_features`
: Features to use for scaling. If NULL, uses all variable features. Defaults to `NULL`.

`selection_method`
: Method for variable feature selection: "vst", "mvp", or "disp". Defaults to `"vst"`.

`resolution`
: Resolution parameter for clustering. Higher values lead to more clusters. Defaults to `0.6`.

`dims`
: Dimensions to use for clustering and dimensionality reduction. If NULL, automatically determined by elbow method. Defaults to `NULL`.

Details

Quality Control Patterns: The function supports flexible pattern matching for quality control, for example:

- `"^MT-"` - Mitochondrial genes (default)
- `"^RP\[LS\]"` - Ribosomal protein genes
- `"^\[rt\]rna"` - rRNA and tRNA genes
- Custom patterns using regular expressions
- Combined patterns: `"^MT-|^RP\[LS\]"` for both mitochondrial and ribosomal genes

Flexible Filtering: The filtering system dynamically adapts to detected quality control patterns:

- Column names are automatically generated from patterns
- Multiple thresholds can be specified in `data_filter.thresh`
- Use `SigBridgeR:::Pattern2Colname` to determine correct column names for custom patterns if still confused

Value

A Seurat object containing:

- Normalized and scaled expression data

- Variable features identified by selection method
- PCA, t-SNE, and UMAP dimensionality reductions
- Cluster identities at specified resolution
- Quality control metrics in `@meta.data`
- When tumor cells filtered: original dimensions in `@misc$raw_dim`
- Final dimensions in `@misc$self_dim`
- Quality control column names in `@misc$qc_colnames`

See Also

CreateSeuratObject, NormalizeData, ScaleData, FindVariableFeatures, RunPCA, RunTSNE, RunUMAP, FindNeighbors, FindClusters

Examples

```
## Not run:
# Example with matrix input
counts_matrix <- matrix(rpois(1000, 5), nrow = 100, ncol = 10)
rownames(counts_matrix) <- paste0("Gene", 1:100)
colnames(counts_matrix) <- paste0("Cell", 1:10)

seurat_obj <- SCPreProcess(
  sc = counts_matrix,
  project = "TestProject",
  min_features = 50,
  resolution = 0.8
)

# Example with tumor cell filtering
metadata <- data.frame(
  cell_type = c(rep("Tumor", 5), rep("Normal", 5)),
  row.names = paste0("Cell", 1:10)
)

tumor_seurat <- SCPreProcess(
  sc = counts_matrix,
  meta_data = metadata,
  column2only_tumor = "cell_type",
  project = "TumorAnalysis"
)

## End(Not run)
```

Screen *Single-Cell Data Screening*

Description

Integrates matched bulk expression data and phenotype information to identify phenotype-associated cell populations in single-cell RNA-seq data using one of four computational methods. Ensures consistency between bulk and phenotype data before analysis.

Usage

```
Screen(
  matched_bulk,
  sc_data,
  phenotype,
  label_type = NULL,
  phenotype_class = c("binary", "survival", "continuous"),
 screen_method = c("Scissor", "scPP", "scPAS", "scAB", "DEGAS", "LP_SGL", "PIPET"),
  ...
)
```

Arguments

matched_bulk	Matrix or data frame of preprocessed bulk RNA-seq expression data (genes x samples). Column names must match names/IDs in phenotype.
sc_data	A Seurat object containing scRNA-seq data to be screened.
phenotype	Phenotype data, either: - Named vector (names match matched_bulk columns), or - Patient survival Data frame with row names matching matched_bulk columns, colnames named "time" and "status"
label_type	Character specifying phenotype label type (e.g., "SBS1", "time")
phenotype_class	Type of phenotypic outcome (must be consistent with input data): - "binary": Binary traits (e.g., case/control) - "continuous": Continuous measurements - "survival": Survival infomation
screen_method	Screening algorithm to use, there are six options: • "Scissor": see also DoScissor() • "scPP": see also DoscPP() • "scPAS": see also DoscPAS() • "scAB": see also DoscAB(), continuous phenotype is not supported • "DEGAS": see also DoDEGAS() • "LP_SGL": see also DoLP_SGL()
...	Additional method-specific parameters: **Scissor alpha** (numeric or NULL) Significance threshold. When NULL, alpha will keep increasing iteratively until the corresponding cells are screened out, default 0.05

cutoff (numeric) A threshold for terminating the iteration of alpha, only work when alpha is NULL, default 0.2

path2load_scissor_cache (character) default NULL

path2save_scissor_inputs (character) A path to save the intermediary data. By using path2load_scissor_cache, the intermediary data can be loaded from the specified path. default "Scissor_inputs.RData"

reliability_test (logical) Whether to perform reliability test, default FALSE

reliability_test.nfold (integer) Cross-validation folds for reliability test, default 10

reliability_test.n (integer) Number of cells to use for reliability test, default 10

cell_evaluation (logical) Whether to perform cell evaluation, default FALSE

cell_evaluation.benchmark_data .RData Benchmark data for cell evaluation, default NULL

cell_evaluation.FDR (numeric) FDR threshold for cell evaluation, default 0.05

cell_evaluation.bootstrap_n (integer) Number of bootstrap samples for cell evaluation, default 10

scPP ref_group (integer or character) Reference group or baseline for binary comparisons, e.g. "Normal" for Tumor/Normal studies and 0 for 0/1 case-control studies. default: 0

Log2FC_cutoff (numeric) Minimum log2 fold-change for binary markers, default 0.585

estimate_cutoff (numeric) Effect size threshold for continuous traits, default 0.2

probs (numeric) Quantile cutoff for cell classification, default 0.2

scPAS assay (character) Assay to use from sc_data, default "RNA"

imputation (logical) Whether to perform imputation, default FALSE

nfeature (integer) Number of features to select, default 3000

alpha (numeric or NULL) Significance threshold, When NULL, alpha will keep increasing iteratively until the corresponding cells are screened out, default 0.01

independent (logical) The background distribution of risk scores is constructed independently of each cell. default: TRUE

network_class (character) Network class to use. default: 'SC', indicating gene-gene similarity networks derived from single-cell data. The other one is 'bulk'.

permutation_times (integer) Number of permutations, default 2000

FDR_threshold (numeric) FDR value threshold for identifying phenotype-associated cells default 0.05

scAB alpha (numeric) Coefficient of phenotype regularization ,default 0.005

alpha_2 (numeric) Coefficent of cell-cell similarity regularization, default 0.005

maxiter (integer) NMF optimization iterations, default 2000

tred (integer) Z-score threshold, default 2

DEGAS **sc_data.pheno_colname** (character) Phenotype column name in sc_data, default "NULL"

select_fraction (numeric) Fraction of cells to select for DEGAS, default 0.05

tmp_dir (character) Temporary directory for DEGAS, default "NULL"

env_params (list) Environment parameters for DEGAS, default "list()"

degas_params (list) DEGAS parameters, default "list()"

normality_test_method (character) Normality test method for DEGAS, default "jarque-bera"

LP_SGL **resolution** (numeric) Resolution parameter for Leiden clustering, default 0.6

alpha (numeric) Alpha parameter for SGL balancing L1 and L2 penalties, default 0.5

nfold (integer) Number of folds for cross-validation, default 5

dge_analysis (list) Differential expression analysis settings:

- run: (logical) Whether to run DEG analysis, default FALSE
- logFC_threshold: (numeric) Log fold change threshold, default 1
- pval_threshold: (numeric) P-value threshold, default 0.05

Value

A list containing:

scRNA_data A Seurat object with phenotype-associated cells labelled in `meta.data` column

Some screen_result Important information about the screened result related to the selected method

Data Matching Requirements

- matched_bulk column names and phenotype names/rownames must be identical
- Phenotype values must correspond to bulk samples (not directly to single cells)
- Mismatches will trigger an error before analysis begins, and there is a built-in pre-run check.

Method Compatibility

Method	Supported Phenotypes	Additional Parameters
Scissor	All three types	alpha, cutoff, path2load_scissor_cache, path2save_scissor_inputs, reliability_test, reliabil
scPP	All three types	ref_group, Log2FC_cutoff, estimate_cutoff, probs
scPAS	All three types	n_components ,assay, imputation,nfeature, alpha,network_class,permutation_times,FDR_
scAB	Binary/Survival	alpha, alpha_2, maxiter, tred
DEGAS	All three types	sc_data.pheno_colname,select_fraction,tmp_dir,env_params,degas_params,normality_tes
LP_SGL	All three types	resolution, alpha, nfold, dge_analysis

See Also

Associated functions:

- DoScissor
- DoscPP
- DoscPAS
- DoscAB
- DoDEGAS
- DoLP_SGL

`ScreenFractionPlot` *Visualization of Cell Screening Fractions*

Description

Generates stacked bar plots showing the proportion of cells classified as Positive/Negative/Neutral by single-cell screening algorithms (Scissor, scPAS, scPP, or scAB) across different sample groups. Supports both single and multiple screen types visualization.

Usage

```
ScreenFractionPlot(
  screened_seurat,
  group_by = "Source", # x-axis label
  screen_type = NULL, # default: search all available screens
  show_null = FALSE,
  plot_color = NULL, # stack bar color of each status
  show_plot = FALSE,
  plot_title = "Screen Fraction",
  stack_width = 0.85,
  x_text_angle = 45,
  axis_linewidth = 0.8,
  legend_position = "right",
  x_lab = NULL,
  y_lab = "Fraction of Status",
  ncol = 2, # number of columns for facet wrap
  nrow = NULL, # number of rows for facet wrap
  verbose = SigBridgeRUtils::getFuncOption("verbose"),
  ... # ggplot2 theme arguments
)
```

Arguments

screened_seurat	A Seurat object containing screening results in metadata. Must contain columns corresponding to screen_type.
group_by	Metadata column name for grouping samples (default: "Source").
screen_type	Screening algorithm(s) used. Can be a single value or vector. Must match metadata column(s) (case-sensitive, e.g., "scissor" for Scissor results).
show_null	Logical whether to show groups with zero cells (default: FALSE).
plot_color	Custom color palette (named vector format): - Required names: "Positive", "Negative", "Neutral" - Default: c("Neutral"="#CECECE", "Other"="#CECECE", Positive"="#ff3333", "Negative"="#386c9b")
show_plot	Logical to immediately display plot (default: TRUE).
plot_title	Plot title (default: "Screen Fraction"). When multiple screen types, can be a vector of titles or single title (will append screen type).
stack_width	Bar width (default: 0.85).
x_text_angle	X-axis label angle (default: 45).
axis_linewidth	Axis line thickness (default: 0.8).
legend_position	Legend position (default: "right").
x_lab	X-axis label (default: NULL).
y_lab	Y-axis label (default: "Fraction of Status").
ncol	Number of columns for facet wrap when multiple screen types (default: 2).
nrow	Number of rows for facet wrap when multiple screen types (default: NULL).
verbose	Logical, whether to print a message
...	Other arguments passed to ggplot2::theme()

Value

A list containing:

- stats: A data frame (single screen) or list of data frames (multiple screens) with screening statistics including:
  - Grouping variable counts
  - Raw cell counts
  - Percentage fractions
- plot: A ggplot2 object (single screen) or list of ggplot2 objects (multiple screens)
- combined_plot: A combined plot using patchwork (only for multiple screens)

Visualization Details

- Bars are ordered by descending Positive fraction
- Y-axis shows percentage (0-100%)
- Zero-fraction groups are automatically hidden unless show_null=TRUE
- For multiple screen types, plots can be combined using patchwork

See Also

Other visualization_function: ScreenUpset()

Examples

```
## Not run:
# Single screen type usage
res <- ScreenFractionPlot(
  screened_seurat = scissor_result,
  group_by = "PatientID",
  screen_type = "scissor"
)

# Multiple screen types at once
multi_res <- ScreenFractionPlot(
  screened_seurat = multi_screened_result,
  group_by = "TissueType",
  screen_type = c("scissor", "scPAS", "scPP", "scAB", "DEGAS"),
  ncol = 2
)

## End(Not run)
```

ScreenUpset *ScreenUpset - Visualize cell type intersections from screened Seurat object*

Description

This function creates an UpSet plot to visualize intersections between different screening methods (e.g., scissor, scAB, scPAS, scPP) in a Seurat object metadata. It calculates all possible combinations of screening types and displays the number of cells positive for each combination.

Usage

```
ScreenUpset(
  screened_seurat,
  screen_type = NULL,
  show_plot = FALSE,
  n_intersections = 20,
  x_lab = "Screen Set Intersections",
  y_lab = "Number of Cells",
  title = "Cell Counts Across Screen Set Intersections",
  bar_color = "#4E79A7",
  combmatrix_point_color = "black",
  verbose = SigBridgeRUtils::getFuncOption("verbose"),
  ...
)
```

Arguments

screened_seurat	A Seurat object containing screening results in metadata. Must contain columns with screening types marked as "Positive".
screen_type	Character vector of screening types to analyze. Default: NULL, indicating that a self-search pattern will be used.
show_plot	Whether to show the upset plot. Default: FALSE
n_intersections	Number of intersections to display in the plot. Default: 20.
x_lab	Label for the x-axis. Default: "Screen Set Intersections".
y_lab	Label for the y-axis. Default: "Number of Cells".
title	Plot title. Default: "Cell Counts Across Screen Set Intersections".
bar_color	Color for the bars in the plot. Default: "#4E79A7".
combmatrix_point_color	Color for points in the combination matrix. Default: "black".
verbose	Logical, whether to print a message
...	Additional arguments passed to `ggplot2::theme()` for customizing the plot appearance.

Details

The function performs the following steps:

1. Validates input parameters and checks if specified screening types exist in metadata
2. Generates all possible combinations of screening types (from 1 to the total number of types)
3. Creates a logical matrix of positive cells for efficient computation
4. Calculates cell counts for each intersection using vectorized operations
5. Creates an UpSet plot visualization with customizable appearance

Value

A list containing two elements:

- `plot`: A ggplot object displaying the UpSet plot
- `stats`: A data frame with intersection statistics including:
  - `intersection`: Name of the intersection
  - `sets`: List of screening types in the intersection
  - `count`: Number of cells positive for all screening types in the intersection

See Also

Other visualization_function: `ScreenFractionPlot()`

Examples

```
## Not run:
# Basic usage with default parameters
result <- ScreenUpset(screened_seurat = my_seurat_obj)

# Customize screening types and appearance
result <- ScreenUpset(
  screened_seurat = my_seurat_obj,
  screen_type = c("scissor", "scAB"),
  n_intersections = 15,
  title = "Custom Title",
  bar_color = "darkred"
)

## End(Not run)
```

`setFuncOption` *Configuration Functions for SigBridgeR Package*

Description

These functions provide a centralized configuration system for the SigBridgeR package, allowing users to set and retrieve package-specific options with automatic naming conventions.

setFunOption sets one or more configuration options for the SigBridgeR package. Options are automatically prefixed with "SigBridgeR." if not already present, ensuring proper namespace isolation.

getFuncOption retrieves configuration options for the SigBridgeR package. The function automatically handles the "SigBridgeR." prefix, allowing users to reference options with or without the explicit prefix.

checkFuncOption validates configuration options for the SigBridgeR package. This function ensures that all options meet type and value requirements before they are set in the global options.

Usage

```
setFuncOption(...)
```

Arguments

`...`	Named arguments representing option-value pairs. Options can be provided with or without the "SigBridgeR." prefix. If the prefix is missing, it will be automatically added.

Details

This function performs the following validations:

verbose, parallel Must be single logical values (TRUE/FALSE)

parallel.type Must be a single character string

workers, timeout, seed Must be single integer values

The function is automatically called by setFuncOption to ensure all configuration options are valid before they are set.

Value

Invisibly returns NULL. The function is called for its side effects of setting package options.

The value of the specified option if it exists, otherwise the provided default value.

No return value. The function throws an error if the value doesn't meet the required specifications for the given option.

Available Opions

- `verbose`: A logical value indicating whether to print verbose messages, defaults to `TRUE`
- `timeout`: An integer specifying the timeout in seconds for parallel processing, defaults to '180L"
- `seed`: An integer specifying the random seed for reproducible results, defaults to '123L"

Examples

```
# Set options with automatic prefixing
setFuncOption(verbose = TRUE)

# Retrieve options with automatic prefixing
getFuncOption("verbose")
```

`SetupPyEnv` *Create or Use Python Environment with Required Packages*

Description

Sets up a Python environment with specified packages for DEGAS screening methods. This function can create new environments or reuse existing ones, supporting both Conda and venv environment types. It ensures all required dependencies are properly installed and verified.

Default method for unsupported environment types. Throws an informative error with supported environment types.

Usage

```
SetupPyEnv(env_type = c("conda", "venv"), ...)
```

Arguments

env_type	Character string specifying the type of Python environment to create or use. One of: ‘"conda"‘, ‘"venv"‘.
...	Additional parameters passed to specific environment methods.

Details

This function provides a comprehensive solution for Python environment management in R projects, particularly for machine learning workflows requiring TensorFlow. Key features include:

- **Environment Creation**: Automatically creates new environments or reuses existing ones with the same name - **Package Management**: Installs specified Python packages with version pinning support - **Verification**: Validates environment setup and package installations - **Flexible Methods**: Supports different backend methods for environment creation (reticulate vs system calls)

The function uses S3 method dispatch to handle different environment types, allowing for extensible support of additional environment managers in the future.

Value

A data frame containing verification results for the environment setup, including installation status of all required packages. Invisibly returns the verification results.

See Also

[reticulate::conda_create()], [reticulate::virtualenv_create()] for underlying environment creation functions.

Examples

```
## Not run:
# Setup a Conda environment with default parameters
SetupPyEnv("conda")

# Setup a venv environment
SetupPyEnv("venv")

## End(Not run)
```

SymbolConvert *Convert Ensembles Version IDs & TCGA Version IDs to Genes in Bulk Expression Data*

Description

Preprocess bulk expression data: convert Ensembles version IDs and TCGA version IDs to genes. NA values are replaced with unknown_k format (k stands for the position of the NA value in the row).

Usage

```
SymbolConvert(data, unknown_format = "unknown_{k}")
```

Arguments

data	bulk expression data (matrix or data.frame)
unknown_format	A glue pattern containing {k} for replace the NA value during conversion. k must be wrapped in curly braces, stands for the position of the NA value in the row. Default: "unknown_{k}".

Index