

# Package ‘SigBridgeR’

October 15, 2025

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

**Version** 2.5.2

**Description** SigBridgeR is an integrative toolkit designed to identify phenotype-associated cell sub-populations by combining phenotype(e.g. survival, drug sensitivity), bulk expression and single-cell RNA-seq data. It leverages multiple algorithms (including 'Scissor', 'sc-PAS', 'scPP', 'scAB' and 'DEGAS') 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** AUCell (>= 1.26.0),

  chk (>= 0.10.0),

  cli (>= 3.6.3),

  data.table (>= 1.17.8),

  DEGAS (>= 1.0.0),

  dplyr (>= 1.1.4),

  edgeR (>= 4.2.2),

  ggforce (>= 0.5.0),

  ggplot2 (>= 4.0.0),

  ggupset (>= 0.4.1),

  glue (>= 1.8.0),

  IDConverter (>= 0.3.5),

  magrittr (>= 2.0.4),

  Matrix (>= 1.7.3),

  matrixStats (>= 1.5.0),

  methods (>= 4.4.1),

  patchwork (>= 1.3.2),

  preprocessCore (>= 1.66.0),

  processx (>= 3.8.4),

purrr (>= 1.1.0),  
 reticulate (>= 1.43.0),  
 rlang (>= 1.1.6),  
 scAB (>= 1.0.0),  
 scales (>= 1.4.0),  
 Scissor (>= 2.0.0),  
 scPAS (>= 0.2.0),  
 ScPP (>= 0.0.0.9000),  
 Seurat (>= 5.0.0),  
 SeuratObject (>= 5.0.0),  
 tibble (>= 3.3.0),  
 tidyR (>= 1.3.1),  
 tools (>= 4.4.1)

**Depends** R (>= 4.1.0)

**Remotes** tsteeljohnson91/DEGAS,  
 ShixiangWang/IDConverter,  
 Qinran-Zhang/scAB,  
 sunduanchen/Scissor,  
 aiminXie/scPAS,  
 WangX-Lab/ScPP

**Suggests** annData,  
 ggVennDiagram,  
 here,  
 knitr,  
 org.Hs.eg.db,  
 randomcoloR,  
 rmarkdown,  
 zeallot

**VignetteBuilder** knitr

## Contents

AddMisc . . . . .	3
BulkPreProcess . . . . .	4
FindRobustElbow . . . . .	6
ListPyEnv . . . . .	7
LoadRefData . . . . .	9
MergeResult . . . . .	10
SCPreProcess . . . . .	11
Screen . . . . .	14
ScreenFractionPlot . . . . .	17
ScreenUpset . . . . .	19
SetupPyEnv . . . . .	21
SetupPyEnv.conda . . . . .	22
SetupPyEnv.venv . . . . .	24
SymbolConvert . . . . .	26

---

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,
  ...,
  # key = value
  cover = TRUE # overwrite existing data
)
```

**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)
```

---

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 = 10,
  min_gene_expressed = 3,
  min_total_reads = 1e+06,
  min_genes_detected = 10000,
  min_correlation = 0.8,
  n_top_genes = 500,
  show_plot_results = TRUE,
  verbose = TRUE
)
```

**Arguments**

<code>data</code>	Expression matrix with genes as rows and samples as columns, or a list containing <code>count_matrix</code> and <code>sample_info</code>
<code>sample_info</code>	Sample information data frame (optional), ignored if <code>data</code> is a list. A qualified <code>sample_info</code> should contain both <code>sample</code> and <code>condition</code> columns (case-sensitive), and there are no specific requirements for the data type stored in the <code>condition</code> column.
<code>gene_symbol_conversion</code>	Whether to convert Ensembls version IDs and TCGA version IDs to genes with <code>IDCConverter</code> , default: <code>FALSE</code>
<code>check</code>	Whether to perform detailed quality checks, default: <code>TRUE</code>
<code>min_count_threshold</code>	Minimum count threshold for gene filtering, default: 10
<code>min_gene_expressed</code>	Minimum number of samples a gene must be expressed in, default: 3
<code>min_total_reads</code>	Minimum total reads per sample, default: <code>1e6</code>
<code>min_genes_detected</code>	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
verbose                    Whether to output detailed information, default: TRUE

```

## 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  $\geq$  min\_count\_threshold in  $\geq$  min\_gene\_expressed samples

Samples are retained if:

- Total reads  $\geq$  min\_total\_reads
- Detected genes  $\geq$  min\_genes\_detected
- Mean correlation  $\geq$  min\_correlation (if check=TRUE)

## See Also

[cpm](#) for counts per million calculation, [prcomp](#) for PCA analysis, [cor](#) for correlation analysis, [RowVars](#) for variance calculation of each row, [SymbolConvert](#) for gene symbol conversion

[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 = TRUE, ndims = 50)
```

## 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: 50)

## Value

Integer, the recommended number of PCA dimensions for downstream analysis

## See Also

Other single\_cell\_preprocess: [ClusterAndReduce\(\)](#), [FilterTumorCell\(\)](#), [ProcessSeuratObject\(\)](#)

## 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)
```

### 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,
  ...
)

## Default S3 method:
ListPyEnv(
  env_type = c("all", "conda", "venv", "virtualenv"),
  timeout = 30000,
  venv_locations = c("~/virtualenvs", "~/venvs", "./venv", "./venv"),
  verbose = TRUE,
  ...
```

```

)
## S3 method for class 'conda'
ListPyEnv(
  env_type = c("all", "conda", "venv", "virtualenv"),
  timeout = 30000,
  venv_locations = c("~/virtualenvs", "~/venvs", "./venv", "./.venv"),
  verbose = TRUE,
  ...
)

## S3 method for class 'venv'
ListPyEnv(
  env_type = c("all", "conda", "venv", "virtualenv"),
  timeout = 30000,
  venv_locations = c("~/virtualenvs", "~/venvs", "./venv", "./.venv"),
  verbose = TRUE,
  ...
)

```

## Arguments

<code>env_type</code>	Character string specifying the type of environments to list. One of: "all", "conda", "venv". Defaults to "all".
<code>timeout</code>	The maximum timeout time when using system commands, only effective when <code>env_type=conda</code> .
<code>venv_locations</code>	Character vector of directory paths to search for virtual environments. Default includes standard locations and common project directories.
<code>verbose</code>	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)
```

---

**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,  
  cache = TRUE,  
  timeout = 60  
)
```

**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.
cache	Logical. If TRUE (default), saves the data for future sessions.
timeout	Integer. Connection timeout in seconds (default: 60).

**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(...)
```

**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

**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)
```

## 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, ...)

SCPreProcess(sc, ...)

## Default S3 method:
SCPreProcess(sc, ...)

## S3 method for class 'matrix'
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = glue::glue("SC_Screening_Proj"),
  min_cells = 400,
  min_features = 0,
  quality_control = TRUE,
  quality_control.pattern = c("^MT-", "^mt-"),
  data_filter = TRUE,
  data_filter.nFeature_RNA_thresh = c(200, 6000),
  data_filter.percent.mt = 20,
  normalization_method = "LogNormalize",
  scale_factor = 10000,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
```

```
dims = 1:10,
verbose = TRUE,
...
)

## S3 method for class 'data.frame'
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = glue::glue("SC_Screening_Proj"),
  min_cells = 400,
  min_features = 0,
  quality_control = TRUE,
  quality_control.pattern = c("^MT-", "^mt-"),
  data_filter = TRUE,
  data_filter.nFeature_RNA_thresh = c(200, 6000),
  data_filter.percent.mt = 20,
  normalization_method = "LogNormalize",
  scale_factor = 10000,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
  dims = 1:10,
  verbose = TRUE,
  ...
)

## S3 method for class 'dgCMatrix'
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = glue::glue("SC_Screening_Proj"),
  min_cells = 400,
  min_features = 0,
  quality_control = TRUE,
  quality_control.pattern = "^MT-",
  data_filter = TRUE,
  data_filter.nFeature_RNA_thresh = c(200, 6000),
  data_filter.percent.mt = 20,
  normalization_method = "LogNormalize",
  scale_factor = 10000,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
  dims = 1:10,
  verbose = TRUE,
```

```

    ...
)

## S3 method for class 'AnnDataR6'
SCPreProcess(
  sc,
  meta_data = NULL,
  column2only_tumor = NULL,
  project = glue::glue("SC_Screening_Proj"),
  min_cells = 400,
  min_features = 0,
  quality_control = TRUE,
  quality_control.pattern = c("^MT-", "^mt-"),
  data_filter = TRUE,
  data_filter.nFeature_RNA_thresh = c(200, 6000),
  data_filter.percent.mt = 20,
  normalization_method = "LogNormalize",
  scale_factor = 10000,
  scale_features = NULL,
  selection_method = "vst",
  resolution = 0.6,
  dims = 1:10,
  verbose = TRUE,
  ...
)
## S3 method for class 'Seurat'
SCPreProcess(sc, column2only_tumor = NULL, verbose = TRUE, ...)

```

## Arguments

<code>sc</code>	Input data, one of: <ul style="list-style-type: none"> <li>• <code>data.frame/matrix/dgCMatrix</code>: Raw count matrix (features x cells)</li> <li>• <code>AnnDataR6</code>: Python AnnData object via reticulate</li> <li>• <code>Seurat</code>: Preprocessed Seurat object</li> </ul>
<code>...</code>	Method-specific arguments (see below)
<code>meta_data</code>	Optional metadata dataframe (rows = cells, columns = attributes)
<code>column2only_tumor</code>	Metadata column used for filtering tumor cells, matching patterns such as "tumor" (or "tumour"), "cancer", "malignant", or "neoplasm" (case-insensitive).
<code>project</code>	Project name for Seurat object
<code>min_cells</code>	Minimum cells per gene to retain (features present in at least this many cells)
<code>min_features</code>	Minimum features per cell to retain (cells with at least this many features)
<code>quality_control</code>	Logical indicating whether to perform mitochondrial QC
<code>quality_control.pattern</code>	Regex pattern to identify mitochondrial genes (e.g., " <code>^MT-</code> " for human)

```

data_filter      Logical indicating whether to filter low-quality cells
data_filter.nFeature_RNA_thresh
                  Numeric vector of length 2 specifying (min, max) features per cell
data_filter.percent.mt
                  Maximum mitochondrial percentage allowed (0-100)
normalization_method
                  Normalization method ("LogNormalize", "CLR", or "RC")
scale_factor     Scaling factor for normalization
scale_features   Scale features to unit variance
selection_method
                  Variable feature selection method ("vst", "mvp", or "disp")
resolution       Cluster resolution (higher for more clusters)
dims             PCA dimensions to use
verbose          Print progress messages

```

### Value

A Seurat object containing:

- Data filter and quality control
- Normalized and scaled expression data
- Variable features
- PCA/tSNE/UMAP reductions
- Cluster identities
- When tumor cells filtered: original dimensions in @misc\$raw\_dim
- Final dimensions in @misc\$self\_dim

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"),
  ...
)

```

## Arguments

<code>matched_bulk</code>	Matrix or data frame of preprocessed bulk RNA-seq expression data (genes x samples). Column names must match names/IDs in phenotype.
<code>sc_data</code>	A Seurat object containing scRNA-seq data to be screened.
<code>phenotype</code>	Phenotype data, either: - Named vector (names match <code>matched_bulk</code> columns), or - Data frame with row names matching <code>matched_bulk</code> columns
<code>label_type</code>	Character specifying phenotype label type (e.g., "SBS1", "time")
<code>phenotype_class</code>	Type of phenotypic outcome (must be consistent with input data): - "binary": Binary traits (e.g., case/control) - "continuous": Continuous measurements - "survival": Survival objects
<code>screen_method</code>	Screening algorithm to use, there are four options: - "Scissor": see also <code>DoScissor()</code> - "scPP": see also <code>DoscPP()</code> - "scPAS": see also <code>DoscPAS()</code> - "scAB": see also <code>DoscAB()</code> , no continuous support - "DEGAS": see also <code>DoDEGAS()</code>
<code>...</code>	Additional method-specific parameters:
<b>Scissor alpha</b> (numeric or NULL)	Significance threshold. When NULL, alpha will keep increasing iteratively until the corresponding cells are screened out, default 0.05
<b>cutoff</b> (numeric)	A threshold for terminating the iteration of alpha, only work when alpha is NULL, default 0.2
<b>path2load_scissor_cache</b> (character)	default NULL
<b>path2save_scissor_inputs</b> (character)	A path to save the intermediary data. By using <code>path2load_scissor_cache</code> , the intermediary data can be loaded from the specified path. default "Scissor_inputs.RData"
<b>reliability_test</b> (logical)	Whether to perform reliability test, default FALSE
<b>reliability_test.nfold</b> (integer)	Cross-validation folds for reliability test, default 10
<b>reliability_test.n</b> (integer)	Number of cells to use for reliability test, default 10
<b>cell_evaluation</b> (logical)	Whether to perform cell evaluation, default FALSE
<b>cell_evaluation.benchmark_data</b> .RData	Benchmark data for cell evaluation, default NULL
<b>cell_evaluation.FDR</b> (numeric)	FDR threshold for cell evaluation, default 0.05
<b>cell_evaluation.bootstrap_n</b> (integer)	Number of bootstrap samples for cell evaluation, default 10
<b>scPP ref_group</b> (integer or character)	Reference group or baseline for <b>binary</b> comparisons, e.g. "Normal" for Tumor/Normal studies and 0 for 0/1 case-control studies. default: 0
<b>Log2FC_cutoff</b> (numeric)	Minimum log2 fold-change for binary markers, default 0.585
<b>estimate_cutoff</b> (numeric)	Effect size threshold for <b>continuous</b> traits, default 0.2
<b>probs</b> (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 5e-05  
**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"

### Value

A list containing:

**scRNA\_data** Filtered Seurat object with phenotype-associated cells

**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
scPP	All three types	ref_group, Log2FC_cutoff, estimate_cutoff, probs
scPAS	All three types	n_components ,assay, imputation,nfeature, alpha, network_class, permutation_
scAB	Binary/Survival	alpha, alpha_2, maxiter, tred
DEGAS	All three types	sc_data.pheno_colname,select_fraction,tmp_dir,env_params,degas_params,no

## See Also

Associated functions:

- [DoScissor](#)
- [DoScPP](#)
- [DoScPAS](#)
- [DoScAB](#)
- [DoDEGAS](#)

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",
  screen_type = c("scissor", "scPAS", "scPP", "scAB", "DEGAS"),
  show_null = FALSE,
  plot_color = NULL,
  show_plot = TRUE,
  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,
nrow = NULL,
scales = "fixed"
)

```

## 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).
scales	Should scales be fixed ("fixed"), free ("free"), or free in one dimension ("free_x", "free_y") for faceted plots (default: "fixed").

## 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 = TRUE,
```

```

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",
...
)

```

## 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: TRUE.
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".
...	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)
```

---

## 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"), ...)  
  
## Default S3 method:  
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)
```

## Description

Creates and configures a Conda environment specifically designed for screening workflows. This function provides multiple methods for environment creation and package installation, including support for environment files, with comprehensive verification and error handling.

## Usage

```
## S3 method for class 'conda'
SetupPyEnv(
  env_type = "conda",
  env_name = "r-reticulate-degas",
  method = c("reticulate", "system", "environment"),
  env_file = NULL,
  python_version = "3.9.15",
  packages = c(tensorflow = "2.4.1", protobuf = "3.20.3"),
  recreate = FALSE,
  use_conda_forge = TRUE,
  verbose = TRUE,
  timeout = 1800000,
  ...
)
```

## Arguments

env_type	Character string specifying the environment type. For this method, must be "conda".
env_name	Character string specifying the Conda environment name. Default: "r-reticulate-degas".
method	Character string specifying the method for environment creation and package installation. One of: "reticulate" (uses reticulate package), "system" (uses system conda commands), or "environment" (uses YAML environment file). Default: "reticulate".
env_file	Character string specifying the path to a Conda environment YAML file. Used when method = "environment". Default: NULL
python_version	Character string specifying the Python version to install. Default: "3.9.15".
packages	Named character vector of Python packages to install. Package names as names, versions as values. Use "any" for version to install latest available. Default includes tensorflow, protobuf, and numpy.
recreate	Logical indicating whether to force recreation of the environment if it already exists. Default: FALSE.
use_conda_forge	Logical indicating whether to use the conda-forge channel for package installation. Default: TRUE.
verbose	Logical indicating whether to display detailed progress messages and command output. Default: TRUE.
timeout	The maximum timeout time when using system commands, default: 30 minutes
...	For future compatibility.

## Value

Invisibly returns NULL.

### Note

The function requires Conda to be installed and accessible on the system PATH or through reticulate. For method = "environment", the specified YAML file must exist and be properly formatted. The function includes extensive error handling but may fail if Conda is not properly configured.

### See Also

[reticulate::conda\\_create\(\)](#), [reticulate::py\\_install\(\)](#) for the underlying functions used in reticulate method.

### Examples

```
## Not run:
# Setup using reticulate method (default)
SetupPyEnv.conda(
  env_name = "my-degas-env",
  python_version = "3.9.15"
)

# Setup using environment file
SetupPyEnv.conda(
  method = "environment",
  env_file = "path/to/environment.yml"
)

# Setup with custom packages
SetupPyEnv.conda(
  packages = c(
    "tensorflow" = "2.4.1",
    "scikit-learn" = "1.0.2",
    "pandas" = "any"
  )
)

## End(Not run)
```

### Description

Creates and configures a Python virtual environment (venv) specifically designed for screening workflows. This function provides a lightweight, isolated Python environment alternative to Conda environments with similar package management capabilities.

**Usage**

```
## S3 method for class 'venv'
SetupPyEnv(
  env_type = "venv",
  env_name = "r-reticulate-degas",
  python_version = "3.9.15",
  packages = c(tensorflow = "2.4.1", protobuf = "3.20.3"),
  python_path = NULL,
  recreate = FALSE,
  verbose = TRUE,
  ...
)
```

**Arguments**

<code>env_type</code>	Character string specifying the environment type. For this method, must be "venv".
<code>env_name</code>	Character string specifying the virtual environment name. Default: "r-reticulate-degas".
<code>python_version</code>	Character string specifying the Python version to use. Default: "3.9.15".
<code>packages</code>	Named character vector of Python packages to install. Package names as names, versions as values. Use "any" for version to install latest available. Default includes tensorflow, protobuf, and numpy.
<code>python_path</code>	Character string specifying the path to a specific Python executable. If NULL, uses the system default or installs the specified version. Default: NULL.
<code>recreate</code>	Logical indicating whether to force recreation of the virtual environment if it already exists. Default: FALSE.
<code>verbose</code>	Logical indicating whether to display detailed progress messages and command output. Default: TRUE.
<code>...</code>	For future compatibility.

**Value**

Invisibly returns NULL.

**Note**

Virtual environments require a base Python installation. If the specified Python version is not available, the function will attempt to install it using reticulate. Virtual environments are generally faster to create than Conda environments but may have more limited package availability compared to Conda-forge.

**See Also**

[reticulate::virtualenv\\_create\(\)](#), [reticulate::virtualenv\\_remove\(\)](#), [reticulate::use\\_virtualenv\(\)](#)  
for the underlying virtual environment management functions.

## Examples

```
## Not run:
# Setup virtual environment with default parameters
SetupPyEnv.venv()

# Setup with custom Python version and packages
SetupPyEnv.venv(
  env_name = "my-degas-venv",
  python_version = "3.8.12",
  packages = c(
    "tensorflow" = "2.4.1",
    "scikit-learn" = "1.0.2",
    "pandas" = "any"
  )
)

# Force recreate existing environment
SetupPyEnv.venv(
  env_name = "existing-env",
  recreate = TRUE
)
## 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)
```

## Arguments

data	bulk expression data (matrix or data.frame)
------	---

# Index

- \* **single\_cell\_preprocess**
  - FindRobustElbow, 6
- \* **visualization\_function**
  - ScreenFractionPlot, 17
  - ScreenUpset, 19
- AddMisc, 3
- BulkPreProcess, 4
- ClusterAndReduce, 6
- cor, 6
- cpm, 6
- DoDEGAS, 17
- DoscAB, 17
- DoScissor, 17
- DoscPAS, 17
- DoscPP, 17
- FilterTumorCell, 6
- FindRobustElbow, 6
- ListPyEnv, 7
- LoadRefData, 9
- MergeResult, 10
- prcomp, 6
- ProcessSeuratObject, 6
- reticulate::conda\_create(), 22, 24
- reticulate::py\_install(), 24
- reticulate::use\_virtualenv(), 25
- reticulate::virtualenv\_create(), 22, 25
- reticulate::virtualenv\_remove(), 25
- RowVars, 6
- SCPreProcess, 11
- Screen, 14
- ScreenFractionPlot, 17, 21
- ScreenUpset, 19, 19
- SetupPyEnv, 21
- SetupPyEnv.conda, 22
- SetupPyEnv.venv, 24
- SymbolConvert, 6, 26