Package: scTypeEval (via r-universe)

May 15, 2026

Title Evaluation of cell type classifications in single-cell transcriptomics

Version 1.1.0

Description scTypeEval provides tools to evaluate and validate cell type classifications in single-cell transcriptomics when ground truth labels are limited or unavailable. Results are organized in an S4 object that integrates processed data, dimensional reductions, dissimilarity assays, and consistency metrics computed across samples. The workflow includes preprocessing and feature selection, principal component analysis, computation of dissimilarity matrices, internal validation metrics (for example, silhouette-based summaries), and visualization utilities to inspect heatmaps and PCA plots. Functions support common single-cell containers and enable comparison of clustering and labeling strategies across datasets.

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

URL https://github.com/carmonalab/scTypeEval

BugReports https://github.com/carmonalab/scTypeEval/issues

License GPL-3 + file LICENSE

LazyData false

biocViews SingleCell, Transcriptomics, GeneExpression, CellBasedAssays, DimensionReduction, Preprocessing, PrincipalComponent

Depends R (>= 4.5.0)

Imports Matrix (>= 1.6-5), BiocParallel (>= 1.34.2), dplyr (>= 1.1.4), tidyr (>= 1.3.1), scran (>= 1.30.2), bluster (>= 1.12.0), ggplot2 (>= 3.5.1), ggrepel (>= 0.9.6), cluster (>= 2.1.4), SingleR (>= 2.4.1), irlba (>= 2.3.5.1), transport (>= 0.15-4), grDevices, methods, stats, utils

Suggests testthat (>= 3.0.0), transformGamPoi (>= 1.8.0), glmGamPoi (>= 1.14.3), anndata (>= 0.8.0), SummarizedExperiment (>= 1.32.0), igraph (>= 2.1.1), Seurat, SingleCellExperiment, knitr, rmarkdown, BiocStyle, BiocManager, SeuratObject, ggpubr, rlang, stringr, tibble

Config/testthat/edition 3

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-05-15 13:20:47 UTC

RemoteUrl https://github.com/bioc/scTypeEval

RemoteRef HEAD

RemoteSha 33e37ac9513de2bd02ebdd1ff20449f1c7d2ae36

Contents

`add_dim_reduction` *Add a Custom Dimensionality Reduction to an scTypeEval Object*

Description

This function allows the user to insert pre-computed dimensionality reduction (e.g., PCA, UMAP, t-SNE, or any embedding) into an `scTypeEval` object. The embeddings are stored in the `reductions` slot as a `dim_red` object, enabling integration with the scTypeEval workflow for downstream analysis.

Usage

```
add_dim_reduction(
  scTypeEval,
  embeddings,
  aggregation,
  ident = NULL,
  ident_name = "custom",
  sample = NULL,
  key = NULL,
  gene_list = NULL,
  black_list = NULL,
  feature_loadings = NULL,
  filter = FALSE,
  min_samples = 5,
  min_cells = 10,
  verbose = TRUE
)
```

Arguments

scTypeEval	A scTypeEval object where the dimensionality reduction will be stored.
embeddings	A numeric matrix of dimension-reduced embeddings (cells/samples x components).
aggregation	Character. Aggregation level of the embeddings. Options: "single-cell" (no aggregation) or "pseudobulk" (aggregated per sample and cell type).
ident	Required. A vector of cell identities (e.g., cell type annotation). If NULL, the function attempts to infer it.
ident_name	Character. Name assigned to the provided ident grouping (default: "custom").
sample	Required. A vector indicating sample identity of each observation. If NULL, the function attempts to infer it.
key	Optional. Character. Key or label assigned to this dimensionality reduction (e.g., "PCA", "UMAP").
gene_list	Optional. Character vector or named list of genes associated with the embeddings (e.g., input features used for the dimensionality reduction). Default: NULL.
black_list	Optional. Character vector of genes excluded from the dimensionality reduction. Default: NULL.
feature_loadings	Optional. Matrix of feature loadings corresponding to the embeddings (e.g., PCA rotation matrix). Default: NULL.
filter	Logical. If TRUE, filters cells before adding the embeddings based on min_samples and min_cells. Only supported for "single-cell" aggregation. Default: FALSE.
min_samples	Integer. Minimum number of samples required for retaining a cell type in single-cell filtering. Default: 5.
min_cells	Integer. Minimum number of cells required per group for filtering. Default: 10.
verbose	Logical. Whether to print messages during execution. Default: TRUE.

Value

The modified `scTypeEval` object with the new dimensionality reduction stored in the `reductions` slot.

See Also

run_pca

Examples

```
# Create test data with enough samples
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

# Create scTypeEval object from matrix
sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

sceval <- create_scTypeEval(
  matrix = counts,
  metadata = metadata,
  active_ident = "celltype"
)

#' # Process data with filtering
sceval <- run_processing_data(
  sceval,
  ident = "celltype",
  sample = "sample",
  min_samples = 3,
  min_cells = 3,
  verbose = FALSE
)

# Create mock embeddings
n_cells <- ncol(sceval@data[["single-cell"]]@matrix)
embeddings <- matrix(rnorm(n_cells * 10), nrow = 10, ncol = n_cells)

ident <- sceval@data[["single-cell"]]@ident[[1]]
sample <- sceval@data[["single-cell"]]@sample

sceval <- add_dim_reduction(
   sceval,
   embeddings = embeddings,
   aggregation = "single-cell",
```

```
   ident = ident,
   sample = sample,
   key = "custom_embedding",
  verbose = FALSE
)
```

`add_gene_list` *Add a gene list to an scTypeEval object.*

Description

This function appends a new gene list to an existing `scTypeEval` object. It ensures that the input list is valid and assigns names if they are missing.

Usage

```
add_gene_list(scTypeEval, gene_list = NULL)
```

Arguments

`scTypeEval`	An `scTypeEval` object to which the gene list will be added.
`gene_list`	A named list of gene sets to append. If the list is unnamed, names will be assigned automatically.

Details

The function verifies that `gene_list` is provided and is a valid list. If any elements in `gene_list` lack names, they are automatically renamed.

Value

An updated `scTypeEval` object with the new gene list added.

Examples

```
#' # Create synthetic test data
library(Matrix)
counts <- Matrix(rpois(1000, 5), nrow = 50, ncol = 20, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(50))
colnames(counts) <- paste0("Cell", seq_len(20))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 10),
  sample = rep(paste0("Sample", seq_len(4)), each = 5),
  row.names = colnames(counts)
)

# Create scTypeEval object from matrix
```

```
sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

sceval <- create_scTypeEval(
  matrix = counts,
  metadata = metadata,
  active_ident = "celltype"
)

sceval <- add_gene_list(sceval,
                        gene_list = list("cytokines" = c("IL10", "IL6", "IL4")))
```

`add_processed_data` *Add externally processed data to an scTypeEval object*

Description

Adds a user-supplied processed dataset (e.g. single-cell or pseudobulk) into an `scTypeEval` object as a `data_assay`. Supports filtering and consistency checks on cell type and sample annotations.

Usage

```
add_processed_data(
  scTypeEval,
  data,
  aggregation,
  ident = NULL,
  ident_name = "custom",
  sample = NULL,
  filter = FALSE,
  min_samples = 5,
  min_cells = 10,
  verbose = TRUE
)
```

Arguments

`scTypeEval`	An `scTypeEval` object generated by `create_scTypeEval`.
`data`	A count matrix (dense or sparse) containing processed expression values (either single-cell or pseudobulk aggregated).
`aggregation`	A string specifying the aggregation type. Must be one of the supported: `"single-cell"` or `"pseudobulk"`.
`ident`	A vector of cell identities corresponding to the columns of `data`. Used to define grouping (e.g. cell type). Required.
`ident_name`	A string specifying the name under which the provided `ident` will be stored (default: `"custom"`).

sample	A vector of sample identifiers corresponding to the columns of data. Required unless already encoded in the object.
filter	Logical indicating whether to filter the data based on min_samples and min_cells. Only allowed when aggregation = "single-cell" (default: FALSE).
min_samples	Minimum number of samples required to retain a feature (default: 5).
min_cells	Minimum number of cells required to retain a feature (default: 10).
verbose	Logical indicating whether to print progress messages (default: TRUE).

Details

- The function validates that identity and sample annotations match the dimensions of the input data.
- For aggregation = "single-cell", optional filtering removes groups with too few samples or cells.
- For aggregation = "pseudobulk", the function checks that each (sample, identity) pair occurs exactly once (i.e., fully aggregated).
- Processed data is wrapped in a data_assay object and added to the scTypeEval.

Value

An updated scTypeEval object containing:

- data: A new data_assay object stored under the specified aggregation type.

See Also

run_processing_data

Examples

```
# Create test data with enough samples
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

sceval <- add_processed_data(
  sceval,
  data = counts,
  aggregation = "single-cell",
  ident = metadata$celltype,
```

```
  sample = metadata$sample,
  filter = FALSE
)
```

`black_list` *Default Gene Blacklist for scTypeEval*

Description

A curated list of genes typically excluded from single-cell RNA-seq analysis to reduce technical artifacts and improve cell type annotation quality.

Format

A character vector with 7,775 gene symbols including:

Mitochondrial genes MT- prefix genes (MT-RNR1, MT-RNR2, MT-TA, etc.)

Ribosomal genes RPL and RPS prefix genes

Long non-coding RNAs Genes with -AS1, -AS2, -IT1, -DT suffixes

MicroRNAs MIR prefix genes (MIR1-1, MIR1-2, etc.)

Small nuclear RNAs SNORA, SNORD, RNU prefix genes

Cell cycle genes G1/S and G2/M phase markers

Sex chromosome genes X and Y chromosome specific genes

Other technical artifacts Heat shock proteins, immediate early genes

Details

This blacklist is automatically loaded and used by default in scTypeEval when `black_list = NULL` is specified in functions like `run_hvg` and `run_gene_markers`. Users can override this default by providing a custom character vector of gene symbols to exclude.

The blacklist helps improve downstream analysis by removing:

- Genes with high technical variance unrelated to cell type identity
- Genes that may confound clustering (e.g., cell cycle genes)
- Genes with batch-specific expression patterns
- Non-coding RNAs that may not be informative for cell type annotation

Value

A character vector containing gene symbols to be excluded from analysis, including cell cycle genes (G1/S and G2/M), mitochondrial genes, ribosomal genes, TCR and immunoglobulin genes, pseudogenes, heat shock proteins, non-coding RNAs, and sex chromosome genes (X and Y).

Source

Generated using the SignatuR package (Andreatta et al., https://github.com/carmonalab/SignatuR) with the following categories:

- Cell cycle genes: G1/S and G2/M phase markers from SignatuR::Programs
- Technical artifacts: Mitochondrial, ribosomal, TCR, immunoglobulins, pseudogenes, HSP, and non-coding RNAs from SignatuR::Blocklists
- Sex chromosome genes: X-inactivation escapees and Y-chromosome specific genes from GenderGenes database (http://bioinf.wehi.edu.au/software/GenderGenes/)

See Also

run_hvg, run_gene_markers

Examples

```
# Load the default gene blacklist
data(black_list)

# Inspect the first few entries
head(black_list)
```

---

create_scTypeEval *Create an scTypeEval object for evaluating cell type classification.*

Description

This function initializes an scTypeEval object from various input formats, including Seurat, SingleCellExperiment, or raw count matrices. It ensures compatibility by validating input types and structures before constructing the object.

Usage

```
create_scTypeEval(
  matrix = NULL,
  metadata = NULL,
  gene_lists = list(),
  black_list = NULL,
  active_ident = NULL
)
```

Arguments

matrix	A Seurat object, SingleCellExperiment object, dense/sparse count matrix, or NULL. If NULL, an empty object is initialized.
metadata	A metadata dataframe. Required if matrix is a raw matrix or NULL. Must have as many rows as the number of cells (columns of the count matrix).
gene_lists	A named list of gene sets to use in the evaluation (default: empty list).
black_list	A character vector of genes to exclude from analysis (default: empty).
active_ident	The active identity class or cluster label (optional). If provided, it is validated against the metadata.

Details

- If matrix = NULL, the function initializes an empty dgCMatrix with 0 rows and 0 columns, and requires a metadata dataframe.
- For Seurat and SingleCellExperiment objects, counts and metadata are extracted automatically.
- For raw matrices, metadata must be provided explicitly.
- The function validates that the number of metadata rows matches the number of cells (matrix columns).
- If active_ident is provided, it is checked for consistency with the metadata.

Value

An scTypeEval object containing:

- counts: A sparse count matrix (dgCMatrix).
- metadata: A dataframe with metadata for each cell.
- gene_lists: A list of gene sets used in classification.
- black_list: A vector of excluded genes.
- active_ident: Active cluster identity (if provided).

Examples

```
# Create synthetic test data
library(Matrix)
counts <- Matrix(rpois(1000, 5), nrow = 50, ncol = 20, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(50))
colnames(counts) <- paste0("Cell", seq_len(20))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 10),
  sample = rep(paste0("Sample", seq_len(4)), each = 5),
  row.names = colnames(counts)
)

# Create scTypeEval object from matrix
```

```
sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

# With custom gene lists and active identity
gene_list <- list(markers = rownames(counts)[seq_len(10)])
sceval <- create_scTypeEval(
  matrix = counts,
  metadata = metadata,
  gene_lists = gene_list,
  active_ident = "celltype"
)
```

`get_consistency` *Compute Consistency of Cell Type Annotations Across samples*

Description

Computes internal validation metrics (consistency measures) for cell type annotations based on dissimilarity assays stored in a `scTypeEval` object. For each indicted dissimilarity representation, one or more internal validation metrics are calculated per cell type and returned in a tidy `data.frame`.

Usage

```
get_consistency(
  scTypeEval,
  dissimilarity_slot = "all",
 consistency_metric = c("silhouette", "2label_silhouette", "NeighborhoodPurity",
    "ward_PropMatch", "Orbital_medoid", "Average_similarity"),
  knn_graph_k = 5,
  hclust_method = "ward.D2",
  normalize = FALSE,
  return_scTypeEval = FALSE,
  verbose = TRUE
)
```

Arguments

`scTypeEval` A `scTypeEval` object containing one or more dissimilarity assays (see `run_dissimilarity`).

`dissimilarity_slot`
Character. Which dissimilarity assay(s) to use. Can be `"all"` (default) or the name(s) of specific slots stored in `scTypeEval@dissimilarity`.

`consistency_metric`
Character vector. Internal validation metrics to compute. Supported options include:

- `"silhouette"` – cohesion/separation score

- `"2label_silhouette"` – Variant of the silhouette score where, for each cell, the within-cluster distance is compared against the average distance to all cells outside its cluster (rather than just the closest other cluster). This effectively treats the clustering as a two-group problem: "own cell type" vs. "all others." Scores are then averaged per cell type. Higher values indicate stronger separation of a cell type from the rest of the dataset as a whole.
- `"NeighborhoodPurity"` – For each cell, computes the fraction of its K nearest neighbors that belong to the same cell type. Scores are then averaged per cell type. Optionally, values can be normalized relative to the expected proportion by chance. Higher scores indicate that cells are surrounded by neighbors of the same type, reflecting strong local label consistency.
- `"ward_PropMatch"` – For each true cell type, identifies the cluster that contains the largest number of its cells (the dominant cluster) and computes the proportion of cells from that cell type that fall into this cluster. Optionally, this proportion can be normalized relative to the expected proportion by chance. Higher values indicate better alignment between true labels and cluster assignments.
- `"Orbital_medoid"` – For each cell type, identifies a representative medoid cell (the cell minimizing the total distance to all other cells of the same type). Then, for each non-medoid cell, it checks whether the cell is closer to its own medoid cell type than to the medoids of other cell types. The metric for each cell type is the proportion of cells that are closer to their own medoid than to any other medoid. Optionally, this proportion can be normalized relative to the expected proportion by chance. Higher values indicate that cells are well-clustered around their medoid.
- `"Average_similarity"` – Measures how similar cells are within the same cell type relative to cells of other types. For each cell, it computes the average distance to other cells in its group and to cells outside its group, then combines these into a normalized score (higher = better). Scores are then averaged per cell type. This metric is similar in spirit to a `"2label_silhouette"`.

Default: all supported metrics.

`knn_graph_k` Integer. Number of nearest neighbors to use for graph-based metrics (`"NeighborhoodPurity"`). Default is `5`.

`hclust_method` Character. Agglomeration method passed to `hclust` for Ward-based metrics. Default is `"ward.D2"`.

`normalize` Logical. Whether to normalize metric values for expected proportions by chance. Default is `FALSE`.

`return_scTypeEval` Logical. Whether to return data frame with inter-sample consistencies or store within scTypeEval@consistency slot. Default is `FALSE`.

`verbose` Logical. Whether to print progress messages. Default is `TRUE`.

Details

This function builds upon the dissimilarity assays generated by `run_dissimilarity`. For each selected dissimilarity representation, the chosen internal validation metrics are computed and stored

in a long-format data frame, allowing downstream comparison across cell types, metrics, and dissimilarity methods.

Value

A `data.frame` with the following columns:

- `celltype` – the annotation/group label
- `measure` – numeric consistency score
- `consistency_metric` – the metric name
- `dissimilarity_method` – the dissimilarity method used
- `ident` – the identity class (from `scTypeEval@ident`)

See Also

`run_dissimilarity`

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)

# Obtain consistency
cons <- get_consistency(sceval, verbose = FALSE)
```

`get_hierarchy` *Perform Hierarchical Clustering on scTypeEval Dissimilarity Matrices*

Description

This function performs hierarchical clustering using precomputed dissimilarity matrices stored in the `dissimilarity` slot of an `scTypeEval` object. Each dissimilarity assay is clustered independently using the specified hierarchical clustering method.

Usage

```
get_hierarchy(
  scTypeEval,
  dissimilarity_slot = "all",
  hierarchy_method = "ward.D2",
  verbose = TRUE
)
```

Arguments

`scTypeEval`	An `scTypeEval` object containing dissimilarity matrices in the `dissimilarity` slot.
`dissimilarity_slot`	Character string. Specifies which dissimilarity assays to cluster. Use `"all"` (default) to include all available assays.
`hierarchy_method`	Character string specifying the hierarchical clustering method (default: `"ward.D2"`). See hclust for available options.
`verbose`	Logical. If `TRUE`, prints messages about progress (default: `TRUE`).

Details

For each dissimilarity assay, the function applies hclust to the stored dissimilarity matrix. The number of clusters is set equal to the number of unique identities provided in the assay metadata.

Value

A list of clustering results, one per dissimilarity assay. Each element contains a contingency table of cluster assignments versus input labels.

See Also

run_dissimilarity, hclust

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)

# Compute Pseudobulk Euclidean dissimilarity
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)
# Perform hierarchical clustering on all available dissimilarity matrices
hier_results <- get_hierarchy(sceval, verbose = FALSE)
```

---

get_nn *Compute K-Nearest Neighbor (KNN) Composition from Dissimilarity Assays*

Description

This function computes KNN-based neighborhood composition scores from dissimilarity matrices stored in a `scTypeEval` object. For each dissimilarity assay, it constructs a KNN graph, computes the cell type composition among neighbors and aggregates results at the group level.

Usage

```
get_nn(
  scTypeEval,
  dissimilarity_slot = "all",
  knn_graph_k = 5,
  normalize = FALSE,
  verbose = TRUE
)
```

Arguments

scTypeEval	An object containing dissimilarity matrices and metadata.
dissimilarity_slot	A character string specifying which dissimilarity assay(s) to use. If "all", all available dissimilarity assays are processed (default: "all").
knn_graph_k	Integer; the number of neighbors to consider in the KNN graph (default: 5).
normalize	Logical; if TRUE, normalizes neighbor proportions relative to expected frequencies of each cell type (default: FALSE).
verbose	Logical; if TRUE, prints progress messages during computation (default: TRUE).

Details

The function extracts dissimilarity matrices from the scTypeEval object and applies a KNN graph construction. For each assay, it computes neighbor cell-type proportions per cell and then aggregate them by cell type. If normalization is enabled, the observed neighbor proportions are scaled relative to the expected global frequency of each cell type.

Value

A list (or a single data frame if only one assay is processed) where each element corresponds to a dissimilarity assay. Each element contains a data frame with rows corresponding to reference cell types and columns representing the mean proportion of neighbors belonging to each cell type.

See Also

run_dissimilarity

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)
```

```
# Compute Pseudobulk Euclidean dissimilarity
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)
# Get nearest neighbors
result <- get_nn(scTypeEval = sceval,
                 knn_graph_k = 5,
                 normalize = TRUE,
                 verbose = FALSE)
```

`load_single_cell_object` *Load a Single-Cell Object from File*

Description

Loads single-cell datasets from `.rds` or `.h5ad` files into R. Supports **Seurat**, **SingleCellExperiment**, and **anndata** objects. Depending on the input type and the `split` parameter, the function either returns the original object or extracts and returns the raw counts matrix and metadata.

Usage

```
load_single_cell_object(path, split = TRUE)
```

Arguments

`path`	Character string. Path to the file to load. Supported formats: `.rds` (Seurat or SingleCellExperiment objects) and `.h5ad` (AnnData objects).
`split`	Logical (default: `TRUE`). If `TRUE`, returns a list with two elements: • `counts`: A sparse `dgCMatrix` of raw counts. • `metadata`: A `data.frame` of cell metadata. If `FALSE`, returns the loaded object as-is (Seurat, SingleCellExperiment, or AnnData).

Details

- `.rds` **input**:
  - If the object is a **Seurat** object, the raw counts are extracted from the "RNA" assay, and cell metadata is taken from `object@meta.data`.
  - If the object is a **SingleCellExperiment**, counts are extracted from the `"counts"` assay, and cell metadata is taken from `colData(object)`.
  - Other `.rds` object types are not supported.
- `.h5ad` **input**: Requires the **anndata** package. The `"counts"` layer must be present, otherwise the function will stop with an error. Counts are transposed to match R's cell-by-gene convention.

Value

If `split = TRUE`, a list containing:

- `counts` – sparse counts matrix
- `metadata` – cell metadata

If `split = FALSE`, returns the loaded single-cell object directly.

Examples

```
# Set a temporary location for temporary file
filepath <- file.path(tempdir(), "sce_test.rds")

# Create small SCE object with sparse matrix
counts <- Matrix::Matrix(rpois(1000, 5), nrow = 50, ncol = 20, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(50))
colnames(counts) <- paste0("Cell", seq_len(20))
sce_obj <- SingleCellExperiment::SingleCellExperiment(
   assays = list(counts = counts),
   colData = data.frame(
      cell_type = rep(c("TypeA", "TypeB"), each = 10),
      row.names = colnames(counts)
  )
)

saveRDS(sce_obj, filepath)

obj <- load_single_cell_object(filepath, split = TRUE)
```

---

plot_heatmap *Plot Heatmaps of Dissimilarity Matrices*

Description

Visualize dissimilarity matrices stored in an `scTypeEval` object as annotated heatmaps. Each selected dissimilarity assay is shown as a heatmap where rows and columns represent cell type and sample, ordered by cell type and optionally sorted by similarity or consistency metrics. Group boundaries are marked to highlight cell-type consistency.

Usage

```
plot_heatmap(
  scTypeEval,
  dissimilarity_slot = "all",
  sort_similarity = NULL,
  sort_consistency = NULL,
  low_color = "black",
  high_color = "white",
```

```
  hclust_method = "ward.D2",
  verbose = TRUE,
  ...
)
```

Arguments

`scTypeEval`	An `scTypeEval` object containing one or more dissimilarity assays.
`dissimilarity_slot`	Character string specifying which dissimilarity assay(s) to plot. If `"all"`, all available dissimilarity assays are included (default: `"all"`).
`sort_similarity`	Optional. Character string naming a dissimilarity assay to use for ordering cells by similarity (hierarchical clustering within each cell type).
`sort_consistency`	Optional. Character string specifying a consistency metric (passed to `get_consistency()`) for ordering cell types by overall consistency.
`low_color`	Color for the low dissimilarity end of the heatmap gradient (default: `"black"`).
`high_color`	Color for the high dissimilarity end of the heatmap gradient (default: `"white"`).
`hclust_method`	Clustering method to use when `sort_similarity` is provided (default: `"ward.D2"`).
`verbose`	Logical. Whether to print progress and diagnostic messages (default: `TRUE`).
`...`	Additional arguments passed to `get_consistency()`.

Details

Ordering logic:

- If both `sort_similarity` and `sort_consistency` are `NULL`, cell types are ordered alphabetically, and cells within each type are ordered alphabetically.
- If only `sort_consistency` is provided, cell types are ordered by the selected consistency metric, and cells within each type are ordered alphabetically.
- If only `sort_similarity` is provided, cell types are ordered by hierarchical clustering of average similarities, and cells within each type are clustered.
- If both are provided, cell types are ordered by consistency, while cells within each type are ordered by similarity clustering.

Each heatmap is rendered with ggplot2, with group boundaries and axis labels indicating cell-type structure.

Value

A named list of `ggplot2` objects, one per dissimilarity assay. If only a single assay is selected, the corresponding heatmap plot is returned directly.

See Also

run_dissimilarity, hclust, get_consistency

Examples

```
#' # Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)

# Compute Pseudobulk Euclidean dissimilarity
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)
# Plot heatmaps for all dissimilarity assays
heatmap <- plot_heatmap(sceval)
```

`plot_mds` *Plot MDS Results from scTypeEval Object*

Description

This function visualizes Multidimensional Scaling (MDS) results computed from dissimilarity assays stored in an `scTypeEval` object. For each dissimilarity assay, MDS is performed using `stats::cmdscale()` and results are displayed as scatterplots.

Usage

```
plot_mds(
  scTypeEval,
  dissimilarity_slot = "all",
  label = TRUE,
  dims = c(1, 2),
  show_legend = FALSE
)
```

Arguments

scTypeEval	An scTypeEval object containing one or more dissimilarity assays.
dissimilarity_slot	Character string specifying which dissimilarity assay(s) to use. If "all", all available dissimilarity assays are plotted (default: "all").
label	Logical; whether to add medoid labels to the MDS plot (default: TRUE).
dims	Integer vector of length 2; the MDS dimensions to plot (default: c(1, 2)).
show_legend	Logical; whether to display a legend (default: FALSE).

Details

For each selected dissimilarity assay, the function:

1. Extracts the dissimilarity matrix and cell-type identities.
2. Performs classical multidimensional scaling (MDS) with cmdscale().
3. Generates a 2D scatterplot with group labels and optional legends using an internal plotting helper.

The plot axes correspond to the requested MDS dimensions (dims).

Value

A named list of MDS plots (ggplot2 objects), one per dissimilarity assay. If only a single assay is processed, a single ggplot2 object is returned.

See Also

run_dissimilarity

Examples

```
#' # Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
```

```
                  ngenes = 100,
                  verbose = FALSE)

# Compute Pseudobulk Euclidean dissimilarity
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)
# Plot MDS for all dissimilarity assays
mds_plots <- plot_mds(sceval)
```

`plot_pca` *Plot PCA Results from scTypeEval Object*

Description

This function visualizes Principal Component Analysis (PCA) results stored in the `reductions` slot of an `scTypeEval` object.

Usage

```
plot_pca(
  scTypeEval,
  reduction_slot = "all",
  label = TRUE,
  dims = c(1, 2),
  show_legend = FALSE
)
```

Arguments

`scTypeEval`	An `scTypeEval` object containing PCA results in the `reductions` slot.
`reduction_slot`	Character. Name(s) of the reduction(s) to plot. If `"all"` (default), all available PCA reductions in the object are plotted.
`label`	Logical. Whether to add medoid labels to the PCA plot (default: `TRUE`).
`dims`	Integer vector of length 2. The principal component (PC) dimensions to plot (default: `c(1,2)`).
`show_legend`	Logical. Whether to display a legend (default: `FALSE`).

Value

A named list of PCA plots (ggplot objects) corresponding to the PCA analyses stored in the `reductions` slot of an `scTypeEval` object. If only one reduction is selected, a single ggplot object is returned.

See Also

`add_dim_reduction`, `run_pca`

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)

# Run PCA on HVG genes
sceval <- run_pca(sceval,
                 gene_list = "HVG",
                 ndim = 4,
                 verbose = FALSE)

# plot PCA
plot_pca(sceval)
```

`run_dissimilarity` *Run Dissimilarity Analysis*

Description

Computes dissimilarity between cell populations or pseudobulk profiles stored in a `scTypeEval` object, using one of the supported methods. Dissimilarity can be computed either on dimensional reduction embeddings (if available) or on processed gene expression data.

Usage

```
run_dissimilarity(
  scTypeEval,
  method = "Pseudobulk:Euclidean",
  reduction = TRUE,
  gene_list = NULL,
  black_list = NULL,
```

```
  reciprocal_classifier = "SingleR",
  ncores = 1,
  bparam = NULL,
  progressbar = FALSE,
  verbose = TRUE
)
```

Arguments

scTypeEval	A scTypeEval object containing processed data and/or dimensional reduction assays.
method	Character. Dissimilarity method to use. Must be one of "WasserStein", "Pseudobulk:Euclidean", "Pseudobulk:Cosine", "Pseudobulk:Pearson", "recip_classif:Match", or "recip_classif:Score". Default is "Pseudobulk:Euclidean".
reduction	Logical. Whether to compute dissimilarity on dimensional reduction embeddings (if available). No supported for "recip_classif" dissimilarity methods. Default is TRUE.
gene_list	Optional. Character vector of genes to include. If NULL, first gene list stored in scTypeEval is used.
black_list	Optional. Character vector of genes to exclude. If NULL, the method will use the default or inherited blacklist.
reciprocal_classifier	Character. Classifier to use for recip_classif dissimilarity methods. Default is "SingleR".
ncores	Integer. Number of cores for parallelization. Default is 1.
bparam	Optional. BiocParallel parameter object to control parallelization. Default is NULL.
progressbar	Logical. Whether to display a progress bar during computation. Default is FALSE.
verbose	Logical. Whether to print progress messages. Default is TRUE.

Details

The function supports multiple dissimilarity strategies:

- "Pseudobulk:<distance>" – computes pairwise distances between pseudobulk profiles using the specified distance metric. Supported distances are euclidean, cosine, and pearson.
- "WasserStein" – computes Wasserstein distances between groups of embeddings or cells.
- "recip_classif:<method>" – assigns cells pairwise between samples using the specified classifier, then computes dissimilarity between assignments. Supported methods are 'match' (binary) and 'score'.

If reduction = TRUE, the function expects that dimensional reduction embeddings have been added previously via run_pca() or add_dim_reduction(). If unavailable, set reduction = FALSE to compute dissimilarity on processed expression data instead.

Value

An updated scTypeEval object with a new dissimilarity_assay stored in scTypeEval@dissimilarity[[method]].

See Also

add_dim_reduction, run_pca, run_processing_data

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)

# Compute Pseudobulk Euclidean dissimilarity
sceval <- run_dissimilarity(sceval, method = "Pseudobulk:Euclidean",
                            reduction = FALSE, verbose = FALSE)
```

---

run_gene_markers *Identify and add marker genes to an scTypeEval object*

Description

Identifies cell type marker genes from normalized single-cell data stored in an scTypeEval object. The identified markers are stored in the gene_lists slot under the chosen method.

Usage

```
run_gene_markers(
  scTypeEval,
  method = c("scran.findMarkers"),
  ngenes_celltype = 50,
```

```
  aggregation = "single-cell",
  black_list = NULL,
  ncores = 1,
  bparam = NULL,
  progressbar = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

scTypeEval	An scTypeEval object containing normalized data (see run_processing_data).
method	A character string specifying the marker gene identification method. Currently supported: • "scran.findMarkers" — Uses **scran**'s findMarkers for differential expression analysis. Default: "scran.findMarkers".
ngenes_celltype	Integer specifying the max number of marker genes to retain per cell type (default: 50).
aggregation	Method to group cells stored in scTypeEval@data, either "single-cell" or "pseudobulk". Default is "single-cell".
black_list	A character vector of genes to exclude from marker selection. If NULL, uses the object's internal blacklist (scTypeEval@black_list).
ncores	Integer specifying the number of cores to use for parallel processing (default: 1).
bparam	Optional. A BiocParallel parameter object for controlling parallel computation. If provided, overrides ncores.
progressbar	Logical, whether to display a progress bar during computation (default: FALSE).
verbose	Logical, whether to print messages during execution (default: TRUE).
...	Additional arguments passed to the underlying marker detection function.

Details

- Requires that normalized single-cell data has been generated with run_processing_data.
- Both cell identities and sample annotations are automatically extracted from the normalized data.
- Genes present in the blacklist (black_list) are removed before marker selection.
- For "scran.findMarkers", the **scran** method findMarkers is applied to identify differentially expressed genes per cell type while adjusting for sample effects.

Value

The modified scTypeEval object with marker genes added to scTypeEval@gene_lists[[method]].

See Also

run_processing_data, add_processed_data

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              aggregation = "single-cell",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)

# Identify marker genes per cell type
sceval <- run_gene_markers(sceval,
 method = "scran.findMarkers",
 ngenes_celltype = 10,
 verbose = FALSE)
```

---

run_hvg *Identify and add highly variable genes (HVG) to an scTypeEval object*

Description

Detects highly variable genes (HVGs) from the normalized single-cell data stored in an `scTypeEval` object. The identified HVGs are stored in the `gene_lists` slot under `"HVG"`.

Usage

```
run_hvg(
  scTypeEval,
  var_method = "scran",
  ngenes = 2000,
  sample = TRUE,
  aggregation = "single-cell",
  black_list = NULL,
  ncores = 1,
  bparam = NULL,
```

```
  progressbar = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

scTypeEval	An scTypeEval object containing normalized data (see run_processing_data).
var_method	Character string specifying the method for identifying highly variable genes. Options: "scran" (default) or "basic".
ngenes	Integer specifying the number of highly variable genes to retain (default: 2000).
sample	Logical indicating whether to leverage sample information when computing HVGs. If TRUE, the sample annotation stored in data is used. If FALSE, HVGs are computed without sample grouping.
aggregation	Method to group cells stored in scTypeEval@data, either "single-cell" or "pseudobulk". Default is "single-cell".
black_list	A character vector of genes to exclude from HVG selection. If NULL, uses the object's internal blacklist (scTypeEval@black_list).
ncores	Integer specifying the number of CPU cores to use for parallel processing (default: 1).
bparam	A BiocParallel backend parameter object for parallelization. If provided, overrides ncores.
progressbar	Logical, whether to display a progress bar during computation (default: FALSE).
verbose	Logical, whether to print progress messages (default: TRUE).
...	Additional arguments passed to internal HVG computation functions.

Details

- Requires that normalized single-cell data has been generated with run_processing_data.
- Genes present in the blacklist (black_list) are removed before HVG selection.
- Available HVG methods:
  - "scran": Uses the **scran** modelGeneVar function to model gene-specific variance and identify biologically variable genes.
  - "basic": A simple variance-to-mean approach ranking genes by coefficient of variation, selecting the top ngenes.

Value

The modified scTypeEval object with HVGs added to scTypeEval@gene_lists[["HVG"]].

See Also

run_processing_data, add_processed_data

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)

# Compute HVGs using basic method
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
                  verbose = FALSE)
```

`run_pca` *Perform PCA on Processed Data and Store Results in scTypeEval Object*

Description

This function computes Principal Component Analysis (PCA) for each processed data aggregation stored in the `scTypeEval` object (e.g., single-cell and/or pseudobulk). The resulting PCA embeddings and loadings are stored in the `reductions` slot of the object.

Usage

```
run_pca(
  scTypeEval,
  gene_list = NULL,
  black_list = NULL,
  ndim = 30,
  verbose = TRUE
)
```

Arguments

`scTypeEval`	A `scTypeEval` object containing processed expression data. See `run_processing_data` or `add_processed_data`).

gene_list	Named list of features defining the gene set for PCA analysis. If NULL, first gene list stored in scTypeEval is used.
black_list	Character vector of genes to exclude from PCA. If NULL, the blacklist stored in scTypeEval is used.
ndim	Integer. Number of principal components to compute (default: 30).
verbose	Logical. Whether to print progress messages during computation (default: TRUE).

Details

This function runs PCA on all processed data slots within the scTypeEval object. Each PCA result is stored as a dim_red assay containing:

- embeddings: PCA coordinates of samples/cells.
- feature_loadings: Loadings of features (genes) on each PC.
- gene_list: The gene sets used for PCA.
- black_list: The genes excluded from PCA.
- aggregation: The aggregation type (e.g., "single-cell", "pseudobulk").
- ident, sample, and group: Metadata carried over from processed data.

Value

The modified scTypeEval object with PCA results stored in the reductions slot for each processed data aggregation.

See Also

run_processing_data

Examples

```
# Create and process test data
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)
sceval <- run_processing_data(sceval, ident = "celltype",
                              sample = "sample", min_samples = 3,
                              min_cells = 3, verbose = FALSE)
sceval <- run_hvg(sceval,
                  var_method = "basic",
                  ngenes = 100,
```

```
                  verbose = FALSE)

# Run PCA on HVG genes
sceval <- run_pca(sceval, gene_list = "HVG", ndim = 4, verbose = FALSE)
```

`run_processing_data` *Process and normalize data within an scTypeEval object*

Description

Runs processing on the count matrix stored in an `scTypeEval` object by aggregating, filtering, and normalizing data. Results are stored as `data_assay` objects within the `scTypeEval`.

Usage

```
run_processing_data(
  scTypeEval,
  ident = NULL,
  sample = NULL,
  aggregation = c("single-cell", "pseudobulk"),
  normalization_method = "Log1p",
  min_samples = 5,
  min_cells = 10,
  verbose = TRUE
)
```

Arguments

`scTypeEval`	An `scTypeEval` object generated by `create_scTypeEval`.
`ident`	A column name from metadata to use as the identity class (e.g. cell type, cluster). If `NULL`, defaults to `scTypeEval@active_ident`.
`sample`	A column name from metadata indicating sample identifiers, required.
`aggregation`	Method to group cells, either `"single-cell"` or `"pseudobulk"`. Default is both.
`normalization_method`	A string specifying the normalization method to apply (default: `"Log1p"`).
`min_samples`	Minimum number of samples required to retain a cell type (default: 5).
`min_cells`	Minimum number of cells required to retain a cell type in a sample (default: 10).
`verbose`	Logical indicating whether to print progress messages (default: TRUE).

Details

The function performs the following steps:

1. Validates and sets the identity (`ident`) and sample grouping (`sample`).
2. Iterates over `aggregation_types`: single-cell and/or pseudobulk.
3. Extracts and filters the count matrix using `min_samples` and `min_cells` thresholds.
4. Normalizes the resulting matrix.
5. Wraps the processed data into `data_assay` objects.
6. Stores the list of processed assays inside the `scTypeEval` object.

Value

An updated `scTypeEval` object containing:

- `data`: A list of `data_assay` objects, one for each aggregation method.

Examples

```
# Create test data with enough samples
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)

sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

# Process data with filtering
sceval <- run_processing_data(
  sceval,
  ident = "celltype",
  sample = "sample",
  min_samples = 3,
  min_cells = 3,
  verbose = FALSE
)

# Check processed data
length(sceval@data)
```

set_active_ident *Set the active ident or annotation labels for an scTypeEval object.*

Description

This function assigns a specific cell type annotation from the metadata as the active identity in an scTypeEval object, allowing for downstream analysis based on the selected classification.

Usage

```
set_active_ident(scTypeEval, ident = NULL)
```

Arguments

scTypeEval	An scTypeEval object.
ident	A character string specifying the column in scTypeEval@metadata to set as the active identity.

Details

The function ensures that the provided identity exists within the metadata before setting it. If no valid identity is provided, an error is raised.

Value

The modified scTypeEval object with the active identity set.

Examples

```
# Create synthetic test data
library(Matrix)
counts <- Matrix(rpois(1000, 5), nrow = 50, ncol = 20, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(50))
colnames(counts) <- paste0("Cell", seq_len(20))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 10),
  sample = rep(paste0("Sample", seq_len(4)), each = 5),
  row.names = colnames(counts)
)

# Create scTypeEval object from matrix
sceval <- create_scTypeEval(matrix = counts, metadata = metadata)

# add active ident
sceval <- set_active_ident(sceval, ident = "celltype")
```

wrapper_scTypeEval *Wrapper Function to Compute Dissimilarities from Single-Cell Data*

Description

A high-level convenience function that initializes an scTypeEval object from a count matrix and metadata, performs preprocessing (normalization, filtering, optional dimensionality reduction), defines gene lists, and computes one or more dissimilarity metrics between cell populations.

This function integrates multiple internal **scTypeEval** pipeline steps, including data preparation, HVG selection, PCA reduction, and dissimilarity computation, providing a streamlined workflow for single-cell data annotation evaluation.

Usage

```
wrapper_scTypeEval(
  scTypeEval = NULL,
  count_matrix,
  metadata,
  ident,
  sample,
  aggregation = c("single-cell", "pseudobulk"),
  gene_list = NULL,
  reduction = TRUE,
  ndim = 30,
  black_list = NULL,
  normalization_method = "Log1p",
 dissimilarity_method = c("WasserStein", "Pseudobulk:Euclidean", "Pseudobulk:Cosine",
    "Pseudobulk:Pearson", "recip_classif:Match", "recip_classif:Score"),
  min_samples = 5,
  min_cells = 10,
  ncores = 1,
  bparam = NULL,
  progressbar = FALSE,
  verbose = TRUE
)
```

Arguments

scTypeEval	An scTypeEval object generated by create_scTypeEval. If null count_matrix and metadata must be provided to build scTypeEval object internally.
count_matrix	A numeric or sparse dgCMatrix of raw counts (genes as rows, cells as columns).
metadata	A data.frame containing cell-level metadata.
ident	Character string indicating the metadata column specifying cell identities (e.g., cell types or clusters).
sample	Character string specifying the metadata column containing sample identifiers (used for pseudobulk aggregation).

`aggregation` Method to group cells, either `"single-cell"` or `"pseudobulk"`, or both. Default is both.

`gene_list` Optional named list of gene sets to include in the analysis. If `NULL`, highly variable genes (HVGs) are automatically computed.

`reduction` Logical; if `TRUE`, performs PCA dimensionality reduction prior to dissimilarity computation (default: `TRUE`).

`ndim` Integer; number of principal components to retain when `reduction = TRUE` (default: `30`).

`black_list` Optional character vector of genes to exclude from analysis. If `NULL`, no genes are blacklisted.

`normalization_method`
Character string specifying the normalization method to apply. Options include `"Log1p"`, `"CLR"`, and `"pearson"` (default: `"Log1p"`).

`dissimilarity_method`
Character vector of dissimilarity metrics to compute. Available options include:

- `"WasserStein"`
- `"Pseudobulk:Euclidean"`
- `"Pseudobulk:Cosine"`
- `"Pseudobulk:Pearson"`
- `"recip_classif:Match"`
- `"recip_classif:Score"`

By default all supported methods are run, and multiple methods can be provided.

`min_samples` Integer; minimum number of samples required for pseudobulk analysis (default: `5`).

`min_cells` Integer; minimum number of cells required per group for inclusion (default: `10`).

`ncores` Integer; number of CPU cores for parallel execution (default: `1`).

`bparam` Optional `BiocParallelParam` object for fine-grained parallelization control. Overrides `ncores` if provided.

`progressbar` Logical; if `TRUE`, displays a progress bar (default: `FALSE`).

`verbose` Logical; if `TRUE`, prints progress messages (default: `TRUE`).

Details

This wrapper combines multiple pipeline steps from **scTypeEval**:

1. `create_scTypeEval()` — initializes the object.
2. `run_processing_data()` — performs normalization and filtering.
3. `run_hvg()` or `add_gene_list()` — defines gene sets.
4. `run_pca()` — performs PCA if `reduction = TRUE`.
5. `run_dissimilarity()` — computes dissimilarities across methods.

This provides a simple entry point for end-to-end setup and dissimilarity computation from raw single-cell data with minimal manual steps.

Value

An updated `scTypeEval` object containing:

- Normalized and filtered data
- HVG gene sets or user-provided gene lists
- PCA reductions (if enabled)
- Computed dissimilarity matrices for each selected method

See Also

`create_scTypeEval`, `run_processing_data`, `run_dissimilarity`, `run_pca`, `run_hvg`

Examples

```
#' # Create test data with enough samples
library(Matrix)
counts <- Matrix(rpois(6000, 5), nrow = 100, ncol = 60, sparse = TRUE)
rownames(counts) <- paste0("Gene", seq_len(100))
colnames(counts) <- paste0("Cell", seq_len(60))

metadata <- data.frame(
  celltype = rep(c("TypeA", "TypeB"), each = 30),
  sample = rep(paste0("Sample", seq_len(6)), times = 10),
  row.names = colnames(counts)
)
sc_res <- wrapper_scTypeEval(
  count_matrix = counts,
  metadata = metadata,
  ident = "celltype",
  sample = "sample",
  min_samples = 3,
  min_cells = 3,
  normalization_method = "Log1p",
  dissimilarity_method = c("Pseudobulk:Euclidean"),
  reduction = TRUE,
  ndim = 4,
  verbose = FALSE
)
```

Index