Package: scClustEval (via r-universe)

May 26, 2026

Type Package

Title Single Cell Clustering Evaluation and Optimization Framework

Version 1.0.0

Date 2026-01-26

Author Zaoqu Liu [aut, cre], Chichau Miao [ctb] (Original SCCAF Python implementation)

Maintainer Zaoqu Liu <liuzaoqu@163.com>

Description A comprehensive framework for evaluating and optimizing single-cell RNA-seq clustering results using self-projection machine learning approaches. The package implements an iterative optimization strategy that merges poorly discriminated clusters based on confusion matrix analysis, achieving robust and reliable cell type identification. Features include multiple classifier support (logistic regression, random forest, SVM, etc.), ROC curve analysis, confusion matrix visualization, and seamless integration with Seurat objects. This is an R implementation inspired by the SCCAF Python package.

License MIT + file LICENSE

URL https://github.com/Zaoqu-Liu/scClustEval

BugReports https://github.com/Zaoqu-Liu/scClustEval/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports Matrix, Rcpp (>= 1.0.0), glmnet, caret, rpart, igraph, pROC, ggplot2, methods, stats, utils, parallel, future, rlang

Suggests Seurat, SeuratObject, randomForest, ranger, e1071, xgboost, leiden, patchwork, ComplexHeatmap, pheatmap, ggalluvial, testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'RcppExports.R' 'scClustEval-package.R' 'utils.R' 'classifiers.R' 'confusion_matrix.R' 'clustering.R' 'assessment.R' 'optimization.R' 'seurat_integration.R' 'visualization.R' 'zzz.R'

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

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-26 03:25:02 UTC

RemoteUrl https://github.com/Zaoqu-Liu/scClustEval

RemoteRef main

RemoteSha dabff729e1d04edfedf119d6d2c1617aa99de218

Contents

`scClustEval-package` *scClustEval: Single Cell Clustering Evaluation and Optimization Framework*

Description

A comprehensive framework for evaluating and optimizing single-cell RNA-seq clustering results using self-projection machine learning approaches.

Details

The scClustEval package provides tools for:

- **Clustering Assessment**: Evaluate the quality of cell clustering using self-projection with various machine learning classifiers
- **Clustering Optimization**: Iteratively merge poorly discriminated clusters to achieve robust cell type identification
- **Visualization**: ROC curves, confusion matrices, Sankey diagrams, and comprehensive assessment plots

- **Seurat Integration**: Seamless workflow with Seurat objects

The core algorithm works by:

1. Training a classifier to distinguish between clusters
2. Evaluating prediction accuracy via cross-validation and hold-out testing
3. Identifying cluster pairs that are difficult to discriminate
4. Merging confused clusters and iterating until target accuracy is reached

Main Functions

`sc_assessment` Core function for clustering assessment

`sc_optimize` Single round of clustering optimization

`sc_optimize_all` Full iterative optimization pipeline

`RunAssessment` Seurat-style assessment function

`RunOptimization` Seurat-style optimization function

Classifiers

The package supports multiple classifiers:

- LR: Logistic Regression (L1/L2 regularization)
- RF: Random Forest
- SVM: Support Vector Machine
- NB: Naive Bayes
- DT: Decision Tree
- XGB: XGBoost (if installed)

Author(s)

Zaoqu Liu <liuzaoqu@163.com>

References

This package is an R implementation inspired by the SCCAF Python package: https://github.com/SCCAF/sccaf

Miao, Z., et al. (2020). Putative cell type discovery from single-cell gene expression data. Nature Methods.

See Also

Useful links:

- https://github.com/Zaoqu-Liu/scClustEval
- Report bugs at https://github.com/Zaoqu-Liu/scClustEval/issues

AddClusterReliability *Add cluster reliability scores to Seurat object*

Description

Calculate and add per-cluster reliability scores

Usage

```
AddClusterReliability(
  object,
  result,
  cluster_col = NULL,
  reliability_col = NULL
)
```

Arguments

`object`	Seurat object
`result`	scClustEval assessment result
`cluster_col`	Cluster column that was assessed
`reliability_col`	Name for reliability column

Value

Seurat object with reliability scores added

assessment *Assessment Functions for scClustEval*

Description

Core functions for clustering quality assessment using self-projection

bhattacharyya_distance *Bhattacharyya distance*

Description

Calculate Bhattacharyya distance between two probability distributions

Usage

```
bhattacharyya_distance(p, q)
```

Arguments

p	First probability distribution
q	Second probability distribution

Value

Bhattacharyya distance (non-negative value)

bhattacharyya_matrix *Bhattacharyya distance matrix*

Description

Compute pairwise Bhattacharyya distances between distributions

Usage

```
bhattacharyya_matrix(prob_matrix, flags = NULL)
```

Arguments

prob_matrix	Matrix where each column is a probability distribution
flags	Optional logical vector to filter rows

Value

Distance matrix

calc_confusion_matrix *Calculate confusion matrix*

Description

Compute confusion matrix from predictions and true labels

Usage

```
calc_confusion_matrix(y_true, y_pred, labels = NULL)
```

Arguments

y_true	True class labels
y_pred	Predicted class labels
labels	Optional vector of labels to use (in order)

Value

A matrix with rows as true labels and columns as predicted labels

Examples

```
y_true <- c("A", "A", "B", "B", "C", "C")
y_pred <- c("A", "B", "B", "B", "C", "A")
calc_confusion_matrix(y_true, y_pred)
```

classifiers *Classifier Functions for scClustEval*

Description

Unified interface for multiple machine learning classifiers

`cluster_adjacency_matrix` *Cluster adjacency matrix*

Description

Cluster groups based on an adjacency/confusion matrix using Louvain

Usage

```
cluster_adjacency_matrix(
  adj_matrix,
  cutoff = 0.1,
  resolution = 1,
  algorithm = "louvain"
)
```

Arguments

`adj_matrix`	Adjacency matrix (e.g., normalized confusion matrix)
`cutoff`	Threshold for binarizing the matrix (default: 0.1)
`resolution`	Resolution parameter for Louvain clustering (default: 1.0)
`algorithm`	Clustering algorithm: "louvain" or "leiden" (if igraph supports it)

Details

The function:

1. Binarizes the adjacency matrix using the cutoff
2. Creates a graph from the binary matrix
3. Applies Louvain/Leiden clustering to identify groups of connected clusters

Value

Integer vector of cluster assignments

Examples

```
# Create a sample adjacency matrix
adj <- matrix(c(0, 0.3, 0.05, 0.3, 0, 0.02, 0.05, 0.02, 0), nrow = 3)
rownames(adj) <- colnames(adj) <- c("A", "B", "C")
cluster_adjacency_matrix(adj, cutoff = 0.1)
```

clustering *Clustering Functions for scClustEval*

Description

Functions for graph-based clustering and cluster manipulation

confusion_matrix *Confusion Matrix Functions for scClustEval*

Description

Functions for computing and normalizing confusion matrices

create_classifier *Create a classifier*

Description

Create a unified classifier object that wraps various ML algorithms

Usage

```
create_classifier(
  type = "LR",
  penalty = "l1",
  alpha = NULL,
  lambda = NULL,
  n_trees = 500,
  max_depth = NULL,
  kernel = "radial",
  seed = NULL,
  ...
)
```

Arguments

type	Classifier type: "LR", "RF", "SVM", "NB", "DT", "XGB", "RANGER"
penalty	For LR: "l1" (lasso), "l2" (ridge), or "elasticnet"
alpha	For LR: elasticnet mixing parameter (1=lasso, 0=ridge). Default: 1 for L1
lambda	For LR: regularization strength (smaller = more regularization). If NULL, uses cross-validation to select

n_trees	For RF/RANGER/XGB: number of trees
max_depth	For DT/XGB: maximum tree depth
kernel	For SVM: kernel type ("linear", "radial", "polynomial")
seed	Random seed for reproducibility
...	Additional arguments passed to the underlying classifier

Details

The returned classifier object has the following methods:

$fit(X, y) Train the classifier on data

$predict(X) Get class predictions

$predict_prob(X) Get class probabilities

$get_coefficients() Get model coefficients (for LR)

$get_importance() Get feature importance (for tree-based)

Value

A classifier object with fit, predict, and predict_prob methods

Examples

```
## Not run:
# Create a logistic regression classifier
clf <- create_classifier("LR", penalty = "l1")

# Train
clf$fit(X_train, y_train)

# Predict
predictions <- clf$predict(X_test)
probabilities <- clf$predict_prob(X_test)

## End(Not run)
```

---

get_available_classifiers

Get available classifiers

Description

List all available classifiers and their requirements

Usage

```
get_available_classifiers()
```

Value

A data.frame with classifier information

Examples

```
get_available_classifiers()
```

---

get_connection_matrix *Get connection matrix between clusterings*

Description

Compute connection matrix showing overlap between two clusterings

Usage

```
get_connection_matrix(labels1, labels2, min_percent = 0.1)
```

Arguments

labels1	First clustering labels (e.g., low resolution)
labels2	Second clustering labels (e.g., high resolution)
min_percent	Minimum percentage threshold to consider a connection (default: 0.1)

Value

Binary connection matrix indicating which clusters from labels2 are connected

---

get_distance_matrix *Get distance matrix between clusters*

Description

Compute pairwise distances between cluster centroids

Usage

```
get_distance_matrix(X, clusters, labels = NULL, method = "euclidean")
```

Arguments

X	Expression matrix (cells x features)
clusters	Cluster assignments
labels	Optional: specific labels to include (in order)
method	Distance method: "euclidean", "manhattan", "cosine"

Value

Distance matrix between cluster centroids

`get_top_markers` *Get top markers from classifier*

Description

Extract top weighted features from a logistic regression classifier

Usage

```
get_top_markers(result, feature_names = NULL, top_n = 10)
```

Arguments

`result`	scClustEval result object (from self_projection)
`feature_names`	Names of features (genes). If NULL, uses column indices
`top_n`	Number of top features per class (default: 10)

Value

Data frame with columns: class, feature, weight

`GetExpressionMatrix` *Extract feature matrix from Seurat (exported helper)*

Description

Helper function to extract feature matrix with V4/V5 compatibility

Usage

```
GetExpressionMatrix(object, assay = NULL, slot = "data", features = NULL)
```

Arguments

`object`	Seurat object
`assay`	Assay name
`slot`	Slot name: "data", "counts", "scale.data"
`features`	Features to extract

Value

Matrix (cells x features)

make_unique_names *Make unique names*

Description

Make a name vector unique by adding suffix "_n"

Usage

```
make_unique_names(x)
```

Arguments

x	Character vector with potential duplicates

Value

Character vector with unique names

Examples

```
make_unique_names(c("A", "B", "A", "C", "A"))
# Returns: c("A", "B", "A_1", "C", "A_2")
```

merge_clusters *Merge clusters*

Description

Merge clusters based on group assignments

Usage

```
merge_clusters(labels, groups, label_map = NULL)
```

Arguments

labels	Original cluster labels
groups	New group assignments (from cluster_adjacency_matrix)
label_map	Optional: named vector mapping old labels to new labels

Value

Factor with merged cluster labels

`normalize_confmat_r1` *Normalize confusion matrix (R1 norm)*

Description

Normalize confusion matrix relative to correctly classified cells

Usage

```
normalize_confmat_r1(cmat, mode = "1")
```

Arguments

`cmat`	Confusion matrix (from calc_confusion_matrix). Rows represent true labels, columns represent predicted labels.
`mode`	Normalization mode: "1" (default, as in SCCAF) or "2"

Details

R1 normalization measures the confusion rate between cluster pairs relative to the number of correctly classified cells.

For each pair (i, j), compute:

$$R1(i,j) = \max\left(\frac{C_{ij}}{C_{jj}}, \frac{C_{ji}}{C_{ii}}\right)$$

where:

- C_{ij} = cells truly in cluster i but predicted as cluster j
- C_{jj} = cells truly in cluster j and correctly predicted (diagonal)

The ratio C_{ij}/C_{jj} represents how many cells from cluster i are misclassified as j, relative to the correctly classified cells in j. A high R1 value indicates substantial confusion between the cluster pair.

Value

Symmetric matrix of pairwise R1-normalized confusion values. Values typically range from 0 to >1 (can exceed 1 when misclassifications outnumber correct classifications).

References

Miao, Z., et al. (2020). Putative cell type discovery from single-cell gene expression data. Nature Methods.

Examples

```
cmat <- matrix(c(90, 5, 5, 3, 85, 12, 2, 10, 88), nrow = 3)
rownames(cmat) <- colnames(cmat) <- c("A", "B", "C")
normalize_confmat_r1(cmat)
```

`normalize_confmat_r2` *Normalize confusion matrix (R2 norm)*

Description

Normalize confusion matrix relative to total cell count

Usage

```
normalize_confmat_r2(cmat)
```

Arguments

`cmat`	Confusion matrix (from calc_confusion_matrix). Rows represent true labels, columns represent predicted labels.

Details

R2 normalization measures the overall impact of confusion between cluster pairs on the entire dataset.

For each pair (i, j), compute:

$$R2(i,j) = \frac{C_{ij} + C_{ji}}{N}$$

where:

- C_{ij} = cells truly in cluster i but predicted as cluster j
- C_{ji} = cells truly in cluster j but predicted as cluster i
- N = total number of cells in the test set

This gives the fraction of total cells that are confused between clusters i and j. Unlike R1, R2 values are always between 0 and 1.

Value

Symmetric matrix of pairwise R2-normalized confusion values. Values range from 0 to 1, representing the fraction of total cells misclassified between each cluster pair.

References

Miao, Z., et al. (2020). Putative cell type discovery from single-cell gene expression data. Nature Methods.

Examples

```
cmat <- matrix(c(90, 5, 5, 3, 85, 12, 2, 10, 88), nrow = 3)
rownames(cmat) <- colnames(cmat) <- c("A", "B", "C")
normalize_confmat_r2(cmat)
```

normalize_confusion_matrix *Normalize confusion matrix*

Description

Wrapper function for confusion matrix normalization

Usage

```
normalize_confusion_matrix(cmat, method = "R1", mode = "1")
```

Arguments

cmat	Confusion matrix
method	Normalization method: "R1" or "R2"
mode	For R1: normalization mode "1" or "2"

Value

Normalized confusion matrix

optimization *Optimization Functions for scClustEval*

Description

Functions for iterative clustering optimization

per_cell_accuracy *Compute per-cell accuracy*

Description

Calculate the prediction confidence for each cell

Usage

```
per_cell_accuracy(X, y, clf)
```

Arguments

X	Feature matrix (cells x features)
y	True cell labels
clf	Trained classifier object

Value

Numeric vector of per-cell accuracy scores

---

per_cluster_accuracy *Compute per-cluster accuracy*

Description

Calculate the classification accuracy for each cluster

Usage

```
per_cluster_accuracy(cmat)
```

Arguments

cmat	Confusion matrix (rows = true, cols = predicted)

Value

Named numeric vector of per-cluster accuracies

Examples

```
cmat <- matrix(c(90, 5, 5, 3, 85, 12, 2, 10, 88), nrow = 3)
rownames(cmat) <- colnames(cmat) <- c("A", "B", "C")
per_cluster_accuracy(cmat)
```

plot.scClustEval *Plot method for scClustEval objects*

Description

Plot method for scClustEval objects

Usage

```
## S3 method for class 'scClustEval'
plot(x, type = "roc", ...)
```

Arguments

x	scClustEval object
type	Plot type: "roc", "confusion", "accuracy", "summary"
...	Additional arguments passed to specific plot functions

plot_assessment_summary *Plot assessment summary*

Description

Create a comprehensive summary plot of assessment results

Usage

```
plot_assessment_summary(result, include = c("roc", "confusion", "accuracy"))
```

Arguments

result	scClustEval result object
include	Which plots to include: combination of "roc", "confusion", "accuracy"

Value

A combined ggplot2 object

plot_cluster_centers *Plot cluster centroids on embedding*

Description

Mark cluster centroids on an embedding plot

Usage

```
plot_cluster_centers(
  embeddings,
  labels,
  point_color = "white",
  point_size = 8,
  point_alpha = 0.6,
  add_to_plot = NULL
)
```

Arguments

embeddings	Matrix of 2D embeddings (cells x 2)
labels	Cluster labels for each cell
point_color	Color for centroid markers (default: "white")
point_size	Size of centroid markers (default: 8)
point_alpha	Transparency of markers (default: 0.6)
add_to_plot	If provided, add to existing ggplot object

Value

A ggplot2 object with cluster centroids marked

plot_cluster_links *Plot cluster connections on embedding*

Description

Draw lines between cluster centroids on embedding space based on confusion matrix values. This visualizes which clusters are frequently confused.

Usage

```
plot_cluster_links(
  embeddings,
  labels,
  confusion_matrix,
  threshold = 0,
  line_color = "#ffa500",
  line_scale = 10,
  show_labels = TRUE,
  label_size = 4,
  point_color = "white",
  point_size = 5,
  point_alpha = 0.7,
  add_to_plot = NULL
)
```

Arguments

`embeddings`	Matrix of 2D embeddings (cells x 2), e.g., UMAP or t-SNE coordinates
`labels`	Cluster labels for each cell
`confusion_matrix`	Confusion/connection matrix between clusters (e.g., R1-normalized)
`threshold`	Only draw lines for values above this threshold (default: 0)
`line_color`	Color for connection lines (default: "#ffa500", orange)
`line_scale`	Scale factor for line width (default: 10)
`show_labels`	Show cluster labels at centroids (default: TRUE)
`label_size`	Size of cluster labels (default: 4)
`point_color`	Color for centroid points (default: "white")
`point_size`	Size of centroid points (default: 5)
`point_alpha`	Transparency of centroid points (default: 0.7)
`add_to_plot`	If provided, add to existing ggplot object

Details

This function computes the median position (centroid) of each cluster in the embedding space, then draws lines between centroids where the confusion matrix value exceeds the threshold. Line width is proportional to the confusion value.

Value

A ggplot2 object showing cluster connections

Examples

```
## Not run:
# Get UMAP coordinates from Seurat object
embeddings <- Seurat::Embeddings(seurat_obj, "umap")
labels <- seurat_obj$seurat_clusters

# Run assessment and get R1 matrix
result <- sc_assessment(X, labels)

# Plot connections
plot_cluster_links(embeddings, labels, result$r1_normalized, threshold = 0.1)

## End(Not run)
```

`plot_cluster_sankey` *Plot Sankey diagram of cluster changes*

Description

Visualize cluster reassignment between rounds or annotations

Usage

```
plot_cluster_sankey(
  labels_from,
  labels_to,
  title = "Cluster Reassignment",
  colors = NULL,
  alpha = 0.6
)
```

Arguments

`labels_from`	Original cluster labels
`labels_to`	New cluster labels
`title`	Plot title
`colors`	Color palette
`alpha`	Transparency of flows

Value

A ggplot2 object (using ggalluvial if available)

`plot_confusion_heatmap` *Plot confusion matrix heatmap*

Description

Visualize confusion matrix as a heatmap

Usage

```
plot_confusion_heatmap(
  result,
  normalized = "R1",
  title = NULL,
  colors = c("white", "gray20"),
  show_values = TRUE,
  text_size = 3
)
```

Arguments

`result`	scClustEval result object, or a confusion matrix directly
`normalized`	Which matrix to plot: "raw", "R1", or "R2"
`title`	Plot title
`colors`	Color gradient (low to high)
`show_values`	Show values in cells
`text_size`	Size of text in cells

Value

A ggplot2 object

`plot_embedding_with_links` *Plot embedding with cluster links*

Description

Combined plot showing cells colored by cluster with confusion-based connections

Usage

```
plot_embedding_with_links(
  embeddings,
  labels,
  confusion_matrix,
  threshold = 0.1,
  title = NULL,
  colors = NULL,
  point_size = 0.5,
  line_color = "#ffa500",
  line_scale = 10,
  show_legend = TRUE
)
```

Arguments

embeddings	Matrix of 2D embeddings (cells x 2)
labels	Cluster labels for each cell
confusion_matrix	Confusion matrix between clusters
threshold	Threshold for drawing connection lines (default: 0.1)
title	Plot title
colors	Optional color palette for clusters
point_size	Size of cell points (default: 0.5)
line_color	Color for connection lines (default: "#ffa500")
line_scale	Scale factor for line widths (default: 10)
show_legend	Show cluster legend (default: TRUE)

Details

This function creates a scatter plot of cells colored by cluster assignment, with lines connecting cluster centroids based on confusion matrix values. Useful for visualizing which clusters are transcriptionally similar and frequently confused by the classifier.

Value

A ggplot2 object

Examples

```
## Not run:
# With Seurat object
embeddings <- Seurat::Embeddings(seurat_obj, "umap")
labels <- seurat_obj$seurat_clusters
result <- sc_assessment(GetAssayData(seurat_obj), labels)

plot_embedding_with_links(
```

```
  embeddings, labels, result$r1_normalized,
  threshold = 0.1, title = "UMAP with Confusion Links"
)

## End(Not run)
```

`plot_optimization_history` *Plot optimization history*

Description

Visualize the progression of optimization rounds

Usage

```
plot_optimization_history(result, metric = "both")
```

Arguments

`result`	scClustEval_optim result object from sc_optimize_all
`metric`	Which metric to plot: "accuracy", "clusters", or "both"

Value

A ggplot2 object

`plot_roc` *Plot ROC curves*

Description

Plot ROC and Precision-Recall curves for assessment results

Usage

```
plot_roc(
  result,
  plot_type = "both",
  show_auc = TRUE,
  show_cv = TRUE,
  show_acc = TRUE,
  colors = NULL,
  title = NULL,
  legend_position = "right"
)
```

Arguments

result	scClustEval result object from self_projection/sc_assessment
plot_type	Type of plot: "both", "roc", or "prc"
show_auc	Show AUC values on plot
show_cv	Show cross-validation accuracy
show_acc	Show test accuracy
colors	Custom color palette
title	Plot title
legend_position	Legend position: "right", "bottom", or "none"

Value

A ggplot2 object

Examples

```
## Not run:
result <- sc_assessment(X, labels)
plot_roc(result)
plot_roc(result, plot_type = "roc")

## End(Not run)
```

PlotConfusionLinks *Plot Seurat embedding with cluster confusion links*

Description

Visualize clustering confusion on Seurat UMAP/t-SNE embedding

Usage

```
PlotConfusionLinks(
  object,
  result,
  reduction = "umap",
  cluster_col = NULL,
  matrix_type = "R1",
  threshold = 0.1,
  line_color = "#ffa500",
  line_scale = 10,
  point_size = 0.5,
  title = NULL,
  show_legend = TRUE
)
```

Arguments

object	Seurat object
result	scClustEval assessment result
reduction	Name of reduction to plot (default: "umap")
cluster_col	Cluster column to use (default: uses result info or Idents)
matrix_type	Which confusion matrix to use: "R1" or "R2"
threshold	Threshold for drawing lines (default: 0.1)
line_color	Color for connection lines
line_scale	Scale factor for line widths
point_size	Size of cell points
title	Plot title
show_legend	Show cluster legend

Details

This function creates a UMAP/t-SNE plot showing cells colored by cluster, with lines connecting cluster centroids based on confusion matrix values. This helps identify which clusters are transcriptionally similar.

Value

A ggplot2 object

Examples

```
## Not run:
# Run assessment
result <- RunAssessment(seurat_obj)

# Plot UMAP with confusion links
PlotConfusionLinks(seurat_obj, result, threshold = 0.1)

# Use t-SNE instead
PlotConfusionLinks(seurat_obj, result, reduction = "tsne")

## End(Not run)
```

print.scClustEval *Print method for scClustEval*

Description

Print method for scClustEval

Usage

```
## S3 method for class 'scClustEval'
print(x, ...)
```

Arguments

x	scClustEval object
...	Additional arguments (ignored)

print.scClustEval_classifier *Print method for classifier*

Description

Print method for classifier

Usage

```
## S3 method for class 'scClustEval_classifier'
print(x, ...)
```

Arguments

x	Classifier object
...	Additional arguments

print.scClustEval_optim *Print method for optimization result*

Description

Print method for optimization result

Usage

```
## S3 method for class 'scClustEval_optim'
print(x, ...)
```

Arguments

x	scClustEval_optim object
...	Additional arguments (ignored)

QuickAssess *Quick assessment from Seurat object*

Description

One-liner for quick clustering quality check

Usage

```
QuickAssess(object, ...)
```

Arguments

object	Seurat object
...	Additional arguments passed to RunAssessment

Value

Prints summary and returns accuracy

Examples

```
## Not run:
# Quick check
accuracy <- QuickAssess(seurat_obj)

## End(Not run)
```

RunAssessment *Run clustering assessment on Seurat object*

Description

Assess clustering quality directly on a Seurat object

Usage

```
RunAssessment(
  object,
  cluster_col = NULL,
  assay = NULL,
  use = "pca",
  dims = 1:50,
  features = NULL,
  classifier = "LR",
  penalty = "l1",
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	Seurat object
cluster_col	Column name in meta.data containing cluster labels (default: uses Idents)
assay	Assay to use (default: DefaultAssay)
use	Feature space to use: "raw" (normalized data), "pca", or other reduction name
dims	Dimensions to use for PCA/reduction (default: 1:50)
features	Optional: specific features to use. If NULL, uses all (for raw) or VariableFeatures
classifier	Classifier type: "LR", "RF", etc.
penalty	For LR: "l1" or "l2"
test_size	Fraction for test set
n_per_class	Max samples per class
cv	Cross-validation folds
seed	Random seed
n_cores	Number of cores
verbose	Print progress
...	Additional arguments passed to self_projection

Details

This function extracts the appropriate data from the Seurat object and runs self_projection assessment. By default, it uses PCA coordinates if available, otherwise normalized data.

For Seurat V4, uses GetAssayData with slots. For Seurat V5, automatically uses LayerData when appropriate.

Value

An scClustEval result object

See Also

self_projection, RunOptimization

Examples

```
## Not run:
# Assess default clustering
result <- RunAssessment(seurat_obj)

# Assess specific clustering with PCA
result <- RunAssessment(
  seurat_obj,
  cluster_col = "seurat_clusters",
  use = "pca",
  dims = 1:30
)

# Assess with raw expression
result <- RunAssessment(
  seurat_obj,
  cluster_col = "manual_annotation",
  use = "raw"
)

## End(Not run)
```

RunOptimization *Run clustering optimization on Seurat object*

Description

Optimize clustering directly on a Seurat object

Usage

```
RunOptimization(
  object,
  cluster_col,
  result_col = "scClustEval_clusters",
  prefix = "scClustEval",
  store_rounds = TRUE,
  assay = NULL,
  use = "pca",
  dims = 1:50,
  features = NULL,
  min_accuracy = 0.9,
  max_rounds = 10,
  classifier = "LR",
  penalty = "l1",
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  n_iter = 3,
  r1_cutoff = 0.5,
  r2_cutoff = 0.05,
  under_cluster_col = NULL,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	Seurat object
cluster_col	Column name containing initial over-clustering
result_col	Column name to store final optimized clustering
prefix	Prefix for intermediate round columns (default: "scClustEval")
store_rounds	Whether to store intermediate round results
assay	Assay to use
use	Feature space: "raw", "pca", etc.
dims	Dimensions for PCA/reduction
features	Specific features to use
min_accuracy	Target accuracy
max_rounds	Maximum rounds
classifier	Classifier type
penalty	For LR: regularization type
test_size	Test fraction

n_per_class	Max samples per class
cv	CV folds
n_iter	Confusion matrix iterations
r1_cutoff	Initial R1 cutoff
r2_cutoff	Initial R2 cutoff
under_cluster_col	Optional: column with under-clustering constraint
seed	Random seed
n_cores	Number of cores
verbose	Print progress
...	Additional arguments

Details

The function modifies the Seurat object by adding:

- Final optimized clustering in result_col (default: "scClustEval_clusters")
- Intermediate round results (if store_rounds = TRUE)
- Cluster reliability scores (optional)

Value

Modified Seurat object with optimized clustering added to meta.data

See Also

sc_optimize_all, RunAssessment

Examples

```
## Not run:
# Basic optimization
seurat_obj <- RunOptimization(
  seurat_obj,
  cluster_col = "seurat_clusters",
  min_accuracy = 0.9
)

# With under-clustering constraint
seurat_obj <- RunOptimization(
  seurat_obj,
  cluster_col = "high_res_clusters",
  under_cluster_col = "low_res_clusters",
  min_accuracy = 0.95
)

## End(Not run)
```

sc_assessment *Single Cell Clustering Assessment*

Description

Main assessment function with user-friendly interface

Usage

```
sc_assessment(
  X,
  labels,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features). Can be sparse.
labels	Cluster labels for each cell
classifier	Classifier type: "LR", "RF", "SVM", "NB", "DT", "XGB", "RANGER"
penalty	For LR: regularization type "l1", "l2", or "elasticnet"
lambda	For LR: regularization strength. If NULL, uses CV to select
test_size	Fraction of data for testing (default: 0.5)
n_per_class	Maximum samples per class in training set. If NULL, uses test_size
cv	Number of cross-validation folds on training set (0 to skip CV)
seed	Random seed for reproducibility
n_cores	Number of cores for parallel processing (NULL = auto-detect)
verbose	Print progress messages

Value

An scClustEval result object

See Also

self_projection, RunAssessment

Examples

```
## Not run:
# Assess clustering quality
result <- sc_assessment(
  X = expression_matrix,
  labels = seurat_object$seurat_clusters
)

# Print summary
print(result)

# Plot ROC curves
plot_roc(result)

## End(Not run)
```

`sc_optimize` *Single round of clustering optimization*

Description

Perform one round of confusion-matrix-guided cluster merging

Usage

```
sc_optimize(
  X,
  labels,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  n_iter = 3,
  r1_cutoff = 0.1,
  r2_cutoff = 0.05,
  r1_mode = "1",
  use_r1_only = FALSE,
  use_r2_only = FALSE,
  use_distance = FALSE,
  dist_cutoff = 8,
  use_projection = FALSE,
  connection_matrix = NULL,
  resolution = 1,
  seed = 1,
  n_cores = NULL,
```

```
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features)
labels	Current cluster labels
classifier	Classifier type: "LR", "RF", "SVM", etc.
penalty	For LR: regularization type
lambda	For LR: regularization strength
test_size	Fraction for test set
n_per_class	Max samples per class in training
cv	Cross-validation folds (0 to skip)
n_iter	Number of sampling iterations for confusion matrix (default: 3)
r1_cutoff	Threshold for R1-normalized confusion (default: 0.1)
r2_cutoff	Threshold for R2-normalized confusion (default: 0.05)
r1_mode	R1 normalization mode: "1" or "2" (default: "1", as in SCCAF)
use_r1_only	Use only R1 normalization for merging decisions
use_r2_only	Use only R2 normalization for merging decisions
use_distance	Use distance matrix in merging decision (default: FALSE)
dist_cutoff	Distance cutoff for merging (default: 8.0)
use_projection	Use self-projection labels for subsequent iterations (default: FALSE)
connection_matrix	Optional connection matrix for constrained merging
resolution	Louvain resolution for merging (default: 1.0)
seed	Random seed
n_cores	Number of cores for parallel processing
verbose	Print progress

Value

A list containing:

new_labels Merged cluster labels

old_labels Original cluster labels

group_map Mapping from old clusters to new groups

accuracy Test accuracy from assessment

cv_accuracy CV accuracy

max_r1 Maximum R1 confusion after this round

max_r2 Maximum R2 confusion after this round

r1_matrix Aggregated R1-normalized confusion matrix

r2_matrix Aggregated R2-normalized confusion matrix

n_clusters_before Number of clusters before merging

n_clusters_after Number of clusters after merging

converged Whether optimization has converged (no merging possible)

assessments List of assessment results from iterations

self_projection_labels Self-projection predicted labels (if use_projection=TRUE)

`sc_optimize_all` *Full iterative optimization pipeline*

Description

Iteratively optimize clustering until target accuracy is reached

Usage

```
sc_optimize_all(
  X,
  labels,
  min_accuracy = 0.9,
  max_rounds = 10,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  n_iter = 3,
  r1_cutoff = 0.5,
  r2_cutoff = 0.05,
  r1_step = 0.01,
  r2_step = 0.001,
  r1_mode = "1",
  use_r1_only = FALSE,
  use_r2_only = FALSE,
  use_distance = FALSE,
  dist_cutoff = 8,
  use_projection = FALSE,
  under_cluster_labels = NULL,
  min_outer_iter = 3,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features)
labels	Initial cluster labels (should be over-clustered)
min_accuracy	Target minimum accuracy to achieve (default: 0.9)
max_rounds	Maximum optimization rounds (default: 10)
classifier	Classifier type
penalty	For LR: regularization type
lambda	For LR: regularization strength
test_size	Fraction for test set
n_per_class	Max samples per class
cv	CV folds
n_iter	Iterations per round for confusion matrix
r1_cutoff	Initial R1 cutoff (default: 0.5)
r2_cutoff	Initial R2 cutoff (default: 0.05)
r1_step	Step to reduce R1 cutoff each outer iteration (default: 0.01)
r2_step	Step to reduce R2 cutoff each outer iteration (default: 0.001)
r1_mode	R1 normalization mode: "1" or "2" (default: "1")
use_r1_only	Use only R1 for merging
use_r2_only	Use only R2 for merging
use_distance	Use distance matrix in merging decisions (default: FALSE)
dist_cutoff	Distance cutoff for merging (default: 8.0)
use_projection	Use self-projection labels for subsequent iterations (default: FALSE)
under_cluster_labels	Optional: under-clustering labels as constraint
min_outer_iter	Minimum outer iterations before allowing convergence
seed	Random seed
n_cores	Number of cores
verbose	Print progress

Details

The optimization proceeds in two levels:

1. **Outer iterations**: Progressively lower the R1/R2 cutoffs
2. **Inner rounds**: Merge clusters based on current cutoffs

The process continues until:

- Target accuracy is reached
- Maximum rounds are exceeded
- No more clusters can be merged

Value

A list containing:

final_labels Final optimized cluster labels

initial_labels Initial cluster labels

round_history List of results from each round

accuracy_history Vector of accuracies per round

n_clusters_history Vector of cluster counts per round

final_accuracy Final achieved accuracy

total_rounds Number of rounds performed

converged Whether target accuracy was reached

See Also

sc_optimize, RunOptimization

Examples

```
## Not run:
# Optimize over-clustered result
result <- sc_optimize_all(
  X = expression_matrix,
  labels = over_clustered_labels,
  min_accuracy = 0.9,
  classifier = "LR"
)

# Get final labels
final_clusters <- result$final_labels

## End(Not run)
```

SCCAF_assessment *SCCAF Assessment*

Description

Alias for self_projection for compatibility with SCCAF Python package

Usage

```
SCCAF_assessment(...)
```

Arguments

...	Arguments passed to self_projection

Value

A list (class "scClustEval") containing:

accuracy Overall accuracy on test set

cv_accuracy Mean cross-validation accuracy (if cv > 0)

train_accuracy Accuracy on training set

y_pred Predicted labels for test set

y_test True labels for test set

y_prob Prediction probabilities (matrix)

confusion_matrix Confusion matrix

r1_normalized R1-normalized confusion matrix

r2_normalized R2-normalized confusion matrix

per_class_accuracy Per-cluster accuracy

classifier Trained classifier object

classes Unique class labels

max_r1 Maximum R1 confusion value

max_r2 Maximum R2 confusion value

See Also

self_projection

SCCAF_optimize *SCCAF Optimize*

Description

Alias for sc_optimize for compatibility

Usage

```
SCCAF_optimize(
  X,
  labels,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  n_iter = 3,
  r1_cutoff = 0.1,
  r2_cutoff = 0.05,
```

```
  r1_mode = "1",
  use_r1_only = FALSE,
  use_r2_only = FALSE,
  use_distance = FALSE,
  dist_cutoff = 8,
  use_projection = FALSE,
  connection_matrix = NULL,
  resolution = 1,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features)
labels	Current cluster labels
classifier	Classifier type: "LR", "RF", "SVM", etc.
penalty	For LR: regularization type
lambda	For LR: regularization strength
test_size	Fraction for test set
n_per_class	Max samples per class in training
cv	Cross-validation folds (0 to skip)
n_iter	Number of sampling iterations for confusion matrix (default: 3)
r1_cutoff	Threshold for R1-normalized confusion (default: 0.1)
r2_cutoff	Threshold for R2-normalized confusion (default: 0.05)
r1_mode	R1 normalization mode: "1" or "2" (default: "1", as in SCCAF)
use_r1_only	Use only R1 normalization for merging decisions
use_r2_only	Use only R2 normalization for merging decisions
use_distance	Use distance matrix in merging decision (default: FALSE)
dist_cutoff	Distance cutoff for merging (default: 8.0)
use_projection	Use self-projection labels for subsequent iterations (default: FALSE)
connection_matrix	Optional connection matrix for constrained merging
resolution	Louvain resolution for merging (default: 1.0)
seed	Random seed
n_cores	Number of cores for parallel processing
verbose	Print progress

SCCAF_optimize_all *SCCAF Optimize All*

Description

Alias for sc_optimize_all for compatibility

Usage

```
SCCAF_optimize_all(
  X,
  labels,
  min_accuracy = 0.9,
  max_rounds = 10,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = 100,
  cv = 5,
  n_iter = 3,
  r1_cutoff = 0.5,
  r2_cutoff = 0.05,
  r1_step = 0.01,
  r2_step = 0.001,
  r1_mode = "1",
  use_r1_only = FALSE,
  use_r2_only = FALSE,
  use_distance = FALSE,
  dist_cutoff = 8,
  use_projection = FALSE,
  under_cluster_labels = NULL,
  min_outer_iter = 3,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features)
labels	Initial cluster labels (should be over-clustered)
min_accuracy	Target minimum accuracy to achieve (default: 0.9)
max_rounds	Maximum optimization rounds (default: 10)
classifier	Classifier type

penalty	For LR: regularization type
lambda	For LR: regularization strength
test_size	Fraction for test set
n_per_class	Max samples per class
cv	CV folds
n_iter	Iterations per round for confusion matrix
r1_cutoff	Initial R1 cutoff (default: 0.5)
r2_cutoff	Initial R2 cutoff (default: 0.05)
r1_step	Step to reduce R1 cutoff each outer iteration (default: 0.01)
r2_step	Step to reduce R2 cutoff each outer iteration (default: 0.001)
r1_mode	R1 normalization mode: "1" or "2" (default: "1")
use_r1_only	Use only R1 for merging
use_r2_only	Use only R2 for merging
use_distance	Use distance matrix in merging decisions (default: FALSE)
dist_cutoff	Distance cutoff for merging (default: 8.0)
use_projection	Use self-projection labels for subsequent iterations (default: FALSE)
under_cluster_labels	Optional: under-clustering labels as constraint
min_outer_iter	Minimum outer iterations before allowing convergence
seed	Random seed
n_cores	Number of cores
verbose	Print progress

`self_projection` *Self-projection assessment*

Description

Core function for evaluating clustering quality using self-projection

Usage

```
self_projection(
  X,
  labels,
  classifier = "LR",
  penalty = "l1",
  lambda = NULL,
  test_size = 0.5,
  n_per_class = NULL,
  cv = 5,
  seed = 1,
  n_cores = NULL,
  verbose = TRUE
)
```

Arguments

X	Expression/feature matrix (cells x features). Can be sparse.
labels	Cluster labels for each cell
classifier	Classifier type: "LR", "RF", "SVM", "NB", "DT", "XGB", "RANGER"
penalty	For LR: regularization type "l1", "l2", or "elasticnet"
lambda	For LR: regularization strength. If NULL, uses CV to select
test_size	Fraction of data for testing (default: 0.5)
n_per_class	Maximum samples per class in training set. If NULL, uses test_size
cv	Number of cross-validation folds on training set (0 to skip CV)
seed	Random seed for reproducibility
n_cores	Number of cores for parallel processing (NULL = auto-detect)
verbose	Print progress messages

Details

The self-projection method works by:

1. Splitting data into training and test sets (stratified by cluster)
2. Training a classifier on the training set
3. Evaluating prediction accuracy on the held-out test set
4. Computing confusion matrices to identify poorly discriminated clusters

High accuracy indicates that clusters are well-separated. Pairs of clusters that are frequently confused may need to be merged.

Value

A list (class "scClustEval") containing:

accuracy Overall accuracy on test set

cv_accuracy Mean cross-validation accuracy (if cv > 0)

train_accuracy Accuracy on training set

y_pred Predicted labels for test set

y_test True labels for test set

y_prob Prediction probabilities (matrix)

confusion_matrix Confusion matrix

r1_normalized R1-normalized confusion matrix

r2_normalized R2-normalized confusion matrix

per_class_accuracy Per-cluster accuracy

classifier Trained classifier object

classes Unique class labels

max_r1 Maximum R1 confusion value

max_r2 Maximum R2 confusion value

See Also

sc_assessment, plot_roc

Examples

```
## Not run:
# Basic usage with expression matrix
result <- self_projection(
  X = expression_matrix,
  labels = cluster_assignments,
  classifier = "LR"
)
print(result$accuracy)

# With random forest
result <- self_projection(
  X = expression_matrix,
  labels = cluster_assignments,
  classifier = "RF",
  n_per_class = 100
)

## End(Not run)
```

---

seurat_integration *Seurat Integration Functions for scClustEval*

Description

Seamless integration with Seurat objects

---

summary.scClustEval *Summary method for scClustEval*

Description

Summary method for scClustEval

Usage

```
## S3 method for class 'scClustEval'
summary(object, ...)
```

Arguments

object	scClustEval object
...	Additional arguments (ignored)

summary.scClustEval_optim *Summary method for optimization result*

Description

Summary method for optimization result

Usage

```
## S3 method for class 'scClustEval_optim'
summary(object, ...)
```

Arguments

object	scClustEval_optim object
...	Additional arguments (ignored)

train_test_split *Simple train-test split*

Description

Wrapper around train_test_split_stratified for simple usage

Usage

```
train_test_split(X, y, test_size = 0.5, n_per_class = NULL, seed = NULL)
```

Arguments

X	Feature matrix (cells x features)
y	Class labels (factor or character vector)
test_size	Fraction of data for testing (default: 0.5)
n_per_class	Maximum number of samples per class in training set. If NULL, uses test_size fraction
seed	Random seed for reproducibility

train_test_split_stratified *Stratified train-test split*

Description

Split data into training and test sets while maintaining class proportions

Usage

```
train_test_split_stratified(
  X,
  y,
  test_size = 0.5,
  n_per_class = NULL,
  seed = NULL
)
```

Arguments

X	Feature matrix (cells x features)
y	Class labels (factor or character vector)
test_size	Fraction of data for testing (default: 0.5)
n_per_class	Maximum number of samples per class in training set. If NULL, uses test_size fraction
seed	Random seed for reproducibility

Value

A list with components:

X_train Training feature matrix

X_test Test feature matrix

y_train Training labels

y_test Test labels

train_idx Indices of training samples

test_idx Indices of test samples

utils *Utility Functions for scClustEval*

Description

Internal utility functions for data processing and manipulation

`visualization` *Visualization Functions for scClustEval*

Description

Functions for plotting assessment and optimization results

`zzz` *Package Startup Functions*

Description

Functions called when the package is loaded

Index