Package: STRIDER (via r-universe)

May 24, 2026

Type Package

Title Spatial Transcriptomics Deconvolution and Integration in R

Version 1.0.0

Date 2026-01-25

Author Zaoqu Liu [aut, cre] (<https://orcid.org/0000-0002-4386-2787>)

Maintainer Zaoqu Liu <liuzaoqu@163.com>

Description STRIDER (Spatial Transcriptomics deconvolutIon and integration in R) provides comprehensive tools for analyzing spatial transcriptomics data. The package implements topic modeling based deconvolution using Latent Dirichlet Allocation (LDA) to decompose spatial spots into cell type proportions. It also provides multi-sample integration using Fused Gromov-Wasserstein (FGW) optimal transport, spatial clustering with neighborhood awareness, and visualization functions. STRIDER is designed to work seamlessly with Seurat objects (v4 and v5) and supports various input formats including 10X Genomics HDF5 files.

License MIT + file LICENSE

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

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

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports Matrix, methods, stats, rlang (>= 1.0.0), data.table, ggplot2 (>= 3.4.0), viridis, Rcpp (>= 1.0.0)

Suggests text2vec (>= 0.6.0), FNN, scatterpie, scales, hdf5r, rhdf5, SeuratObject, patchwork, future, future.apply, igraph, cluster, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

SystemRequirements GNU make

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs make

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

Date/Publication 2026-01-24 19:14:08 UTC

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

RemoteRef main

RemoteSha 8fbfecb48f36d48d6536524fcc066393fb2052f4

Contents

`align_samples` *Align Two Samples Using FGW*

Description

Align two spatial transcriptomics samples using Fused Gromov-Wasserstein optimal transport.

Usage

```
align_samples(
  topic1,
  topic2,
  coords1,
  coords2,
  alpha = 0.5,
  reg = 0.1,
  max.iter = 100
)
```

Arguments

`topic1`	Topic distribution for sample 1 (topics x spots)
`topic2`	Topic distribution for sample 2 (topics x spots)
`coords1`	Spatial coordinates for sample 1
`coords2`	Spatial coordinates for sample 2
`alpha`	Trade-off parameter (default: 0.5)
`reg`	Regularization parameter (default: 0.1)
`max.iter`	Maximum iterations (default: 100)

Value

A list containing:

- transportOptimal transport plan matrix
- rotationRotation matrix for alignment
- fgw_distFGW distance value

Examples

```
## Not run:
alignment <- align_samples(topic1, topic2, coords1, coords2)

## End(Not run)
```

---

as_seurat *Convert STRIDER Object to Seurat*

Description

Add STRIDER deconvolution results to a Seurat object.

Usage

```
as_seurat(strider, seurat, assay.name = "STRIDER")
```

Arguments

strider	STRIDER result object
seurat	Seurat object
assay.name	Name for new assay (default: "STRIDER")

Value

Seurat object with deconvolution results

---

benchmark_strider *Benchmark STRIDER Against Known Ground Truth*

Description

Evaluate STRIDER performance using simulated data with known cell type proportions.

Usage

```
benchmark_strider(strider_result, true_proportions)
```

Arguments

strider_result	STRIDER result
true_proportions	True cell type proportions (spots x celltypes)

Value

Benchmark metrics

calculate_celltype_topic_bayes *Calculate Bayesian Celltype-Topic Matrix*

Description

Calculate celltype-topic association using Bayes' theorem, incorporating prior probabilities of cell types.

Usage

```
calculate_celltype_topic_bayes(topic_celltype, celltype_counts, total_cells)
```

Arguments

`topic_celltype`	Topic-celltype matrix (topics x celltypes)
`celltype_counts`	Named vector of cell counts per cell type
`total_cells`	Total number of cells

Details

Applies Bayes' theorem: P(celltype|topic) = P(topic|celltype) * P(celltype) / P(topic)

This accounts for differences in cell type abundance in the reference data, preventing overrepresentation of abundant cell types in deconvolution.

Value

Celltype-topic Bayes matrix (celltypes x topics)

calculate_coherence *Calculate LDA Coherence Score*

Description

Calculate topic coherence scores to evaluate model quality. Implements UMass and optionally C_v coherence metrics.

Usage

```
calculate_coherence(
  model,
  matrix,
  n.words = 10,
  type = "both",
  cell_gene_list = NULL
)
```

Arguments

model	LDA model from train_lda()
matrix	Original count matrix used for training
n.words	Number of top words per topic (default: 10)
type	Coherence type: "umass", "cv", or "both" (default: "both")
cell_gene_list	List of genes expressed per cell (for c_v, auto-generated if NULL)

Details

UMass coherence measures how often top words co-occur in the corpus: C_UMass = sum_j=2^M sum_i=1^j-1 log((D(w_i, w_j) + 1) / D(w_i))

C_v coherence uses normalized pointwise mutual information (NPMI) with a sliding window and cosine similarity, providing better correlation with human judgments.

Value

A list containing:

- umassUMass coherence (mean across topics)
- cvC_v coherence (mean across topics), NA if type="umass"
- umass_per_topicUMass score for each topic
- cv_per_topicC_v score for each topic

References

Mimno, D., et al. (2011). Optimizing semantic coherence in topic models. EMNLP. Röder, M., et al. (2015). Exploring the Space of Topic Coherence Measures. WSDM.

---

calculate_neighbor_fractions

Calculate Neighbor Average Fractions

Description

Calculate the average cell type fractions from neighboring spots.

Usage

```
calculate_neighbor_fractions(deconv, neighbors)
```

Arguments

deconv	Cell type fraction matrix (spots x celltypes)
neighbors	List of neighbor indices from find_neighbors()

Value

Matrix of neighbor-averaged fractions (spots x celltypes)

---

calculate_nmi *Calculate Normalized Mutual Information*

Description

Calculate NMI between predicted clusters and true cell types.

Usage

```
calculate_nmi(pred, truth)
```

Arguments

pred	Predicted labels
truth	True labels

Value

NMI score between 0 and 1

---

calculate_qc_metrics *Calculate Quality Control Metrics*

Description

Calculate quality control metrics for cells/spots.

Usage

```
calculate_qc_metrics(matrix, mito.pattern = "^MT-|^mt-")
```

Arguments

matrix	Count matrix (genes x cells/spots)
mito.pattern	Pattern to match mitochondrial genes (default: "^MT-|^mt-")

Value

Data frame with QC metrics

Examples

```
## Not run:
qc <- calculate_qc_metrics(mat)

## End(Not run)
```

`calculate_topic_celltype` *Calculate Topic-Celltype Association Matrix*

Description

Calculate the association between topics and cell types by aggregating topic distributions across cells of each type.

Usage

```
calculate_topic_celltype(topic_cell, celltypes, method = "mean")
```

Arguments

`topic_cell`	Topic-cell matrix (topics x cells)
`celltypes`	Named vector of cell type annotations
`method`	Aggregation method: "mean" (default), "median", or "sum"

Details

This matrix represents P(topic|celltype), the probability of each topic given a cell type. It is computed by averaging topic distributions across all cells of each type.

Value

Topic-celltype matrix (topics x celltypes)

Examples

```
## Not run:
topic_ct <- calculate_topic_celltype(topic_cell, celltypes)

## End(Not run)
```

center_coords *Center Coordinates*

Description

Center coordinates by weighted mean.

Usage

```
center_coords(coords, weights = NULL)
```

Arguments

coords	Coordinate data frame or matrix (n x 2)
weights	Point weights (default: uniform)

Value

Centered coordinates as data frame

center_points_cpp *Center Points by Weighted Mean*

Description

Center point coordinates by subtracting the weighted centroid.

Usage

```
center_points_cpp(X, weights)
```

Arguments

X	Points (n x d)
weights	Point weights (n), should sum to 1

Value

Centered points (n x d)

cluster_composition *Calculate Cluster Statistics*

Description

Calculate cell type composition statistics for each cluster.

Usage

```
cluster_composition(deconv, clusters)
```

Arguments

deconv	Deconvolution matrix (spots x celltypes)
clusters	Cluster assignments (vector or data frame with 'cluster' column)

Value

Data frame with mean composition per cluster

cluster_in_topic_space *Perform K-Means Clustering on Topic Space*

Description

Cluster cells in topic space and evaluate clustering quality.

Usage

```
cluster_in_topic_space(
  topic_cell,
  true_celltypes,
  n_clusters = NULL,
  n_iter = 1000,
  seed = 42
)
```

Arguments

topic_cell	Topic-cell matrix (topics x cells)
true_celltypes	Named vector of true cell type annotations
n_clusters	Number of clusters (default: auto from celltypes)
n_iter	Maximum k-means iterations (default: 1000)
seed	Random seed

Value

A list containing cluster assignments and NMI score

`cluster_silhouette` *Calculate Silhouette Score for Clustering*

Description

Calculate silhouette score to assess clustering quality.

Usage

```
cluster_silhouette(deconv, clusters)
```

Arguments

`deconv`	Deconvolution matrix
`clusters`	Cluster assignments

Value

Average silhouette score

`clustering` *Spatial Clustering Functions for STRIDER*

Description

Functions for neighborhood-aware clustering of spatial spots

combine_plots *Combine Multiple Plots*

Description

Combine multiple ggplot objects into a single figure.

Usage

```
combine_plots(plots, ncol = 2, nrow = NULL, title = NULL)
```

Arguments

plots	List of ggplot objects
ncol	Number of columns
nrow	Number of rows (optional)
title	Overall title

Value

A combined plot

compute_distance_matrix *Compute Spatial Distance Matrix*

Description

Compute pairwise Euclidean distance matrix from coordinates.

Usage

```
compute_distance_matrix(coords)
```

Arguments

coords	Coordinate data frame or matrix

Value

Distance matrix

compute_kl_divergence *Compute KL Divergence Matrix*

Description

Compute pairwise KL divergence between probability distributions. Matches the exact implementation in Python STRIDE's KLDivergency function.

Usage

```
compute_kl_divergence(P, Q, eps = 1e-10)
```

Arguments

P	First matrix (n x k), rows are distributions
Q	Second matrix (m x k), rows are distributions
eps	Small constant to avoid log(0) (default: 1e-10)

Value

KL divergence matrix (n x m)

compute_rotation_svd_cpp *Compute Optimal Rotation via SVD*

Description

Find the optimal rotation matrix R to align X to Y using Procrustes analysis (SVD of cross-covariance).

Usage

```
compute_rotation_svd_cpp(X, Y, weights)
```

Arguments

X	Source points (n x d)
Y	Target points (n x d)
weights	Point weights (n)

Details

Minimizes weighted sum of squared alignment errors. Solution uses SVD: R = U * V' where H = Y' * W * X = U * S * V'

Value

Rotation matrix R (d x d) such that X * R aligns with Y

---

create_mapping_graph *Create Spot-Cell Neighborhood Graph*

Description

Create a bipartite graph connecting spots to their most similar cells.

Usage

```
create_mapping_graph(mapping, threshold = NULL)
```

Arguments

mapping	Mapping result from map_spots_to_cells()
threshold	Distance threshold for edges (optional)

Value

An igraph object (if igraph is available) or edge list

---

create_strider_object *Create STRIDER Object*

Description

Create a STRIDER result object.

Usage

```
create_strider_object(
  deconv,
  topic_spot = NULL,
  model = NULL,
  method = NULL,
  n_topics = NULL,
  metrics = NULL,
  genes_used = NULL,
  celltypes = NULL
)
```

Arguments

deconv	Deconvolution results matrix (spots x celltypes)
topic_spot	Topic-spot distribution matrix
model	Trained LDA model
method	Selected deconvolution method
n_topics	Number of topics
metrics	Model evaluation metrics
genes_used	Genes used in model training
celltypes	Cell types in the model

Value

A STRIDER object

---

data_io *Data Input/Output Functions for STRIDER*

Description

Functions for reading and writing spatial transcriptomics data

---

deconvolve *Cell Type Deconvolution Functions for STRIDER*

Description

Core functions for deconvolving cell type proportions in spatial transcriptomics

Perform cell type deconvolution on spatial transcriptomics data using a trained LDA model.

Usage

```
deconvolve(
  st.matrix,
  model,
  topic_celltype,
  celltype_topic_bayes = NULL,
  method = "Bayes",
  n.iter = 100
)
```

Arguments

st.matrix	Spatial count matrix (genes x spots)
model	Trained LDA model from train_lda()
topic_celltype	Topic-celltype association matrix
celltype_topic_bayes	Bayesian celltype-topic matrix (required for Bayes methods)
method	Deconvolution method: • "Raw": Direct matrix multiplication • "Norm": Column-normalized weights • "NormBySD": SD-normalized weights • "Bayes": Bayesian posterior (default) • "BayesNorm": Normalized Bayesian posterior
n.iter	Inference iterations (default: 100)

Details

The deconvolution computes cell type fractions for each spot by:

1. Inferring topic distributions for spatial spots
2. Applying celltype-topic weights to convert topics to cell types
3. Normalizing to obtain fractions that sum to 1

Value

A list containing:

- celltype_fracCell type fractions (spots x celltypes)
- topic_spotTopic distributions (topics x spots)

---

euclidean_distance_cpp

Compute Euclidean Distance Matrix

Description

Compute pairwise Euclidean distances between points.

Usage

```
euclidean_distance_cpp(X, Y_nullable = NULL)
```

Arguments

X	First point set (n x d)
Y_nullable	Second point set (m x d), optional. If NULL, computes pairwise distances within X.

Value

Distance matrix (n x m or n x n)

`evaluate_model_prediction` *Evaluate Model Using Cell Type Prediction*

Description

Evaluate LDA model quality by predicting cell types and comparing with known annotations.

Usage

```
evaluate_model_prediction(
  topic_cell,
  topic_celltype,
  true_celltypes,
  method = "Raw",
  celltype_topic_bayes = NULL
)
```

Arguments

`topic_cell`	Topic-cell matrix (topics x cells)
`topic_celltype`	Topic-celltype matrix (topics x celltypes)
`true_celltypes`	Named vector of true cell type annotations
`method`	Deconvolution method: "Raw", "Norm", "NormBySD", "Bayes", "BayesNorm"
`celltype_topic_bayes`	Bayesian celltype-topic matrix (for Bayes methods)

Value

A list containing:

- accuracyPrediction accuracy
- predictionsPredicted cell types
- confusionConfusion matrix

Examples

```
## Not run:
eval <- evaluate_model_prediction(topic_cell, topic_ct, true_ct)

## End(Not run)
```

`evaluate_topic_range` *Evaluate Multiple Topic Numbers*

Description

Train and evaluate LDA models with different numbers of topics to find the optimal configuration.

Usage

```
evaluate_topic_range(
  matrix,
  celltypes,
  topic_range,
  methods = c("Raw", "Norm", "NormBySD", "Bayes", "BayesNorm"),
  n.iter = 500,
  seed = 42,
  save.models = FALSE,
  out.dir = NULL
)
```

Arguments

`matrix`	Normalized count matrix (genes x cells)
`celltypes`	Named vector of cell type annotations
`topic_range`	Vector of topic numbers to test
`methods`	Vector of evaluation methods
`n.iter`	LDA iterations (default: 500)
`seed`	Random seed
`save.models`	Save intermediate models (default: FALSE)
`out.dir`	Output directory for models

Value

A list containing:

- metricsData frame of evaluation metrics
- best_modelBest LDA model
- best_n_topicsOptimal number of topics
- best_methodBest evaluation method

Examples

```
## Not run:
results <- evaluate_topic_range(mat, celltypes, topic_range = 5:15)

## End(Not run)
```

filter_cells *Filter Cells/Spots*

Description

Filter cells or spots based on minimum expression criteria.

Usage

```
filter_cells(matrix, min.genes = 200, min.counts = 0, max.counts = Inf)
```

Arguments

matrix	Count matrix (genes x cells/spots)
min.genes	Minimum number of genes detected (default: 200)
min.counts	Minimum total counts (default: 0)
max.counts	Maximum total counts (default: Inf)

Value

Filtered matrix

Examples

```
## Not run:
filtered <- filter_cells(mat, min.genes = 200)

## End(Not run)
```

filter_genes *Filter Genes*

Description

Filter genes based on minimum expression criteria.

Usage

```
filter_genes(matrix, min.cells = 10, min.counts = 0)
```

Arguments

matrix	Count matrix (genes x cells/spots)
min.cells	Minimum number of cells expressing the gene (default: 10)
min.counts	Minimum total counts for a gene (default: 0)

Value

Filtered matrix

Examples

```
## Not run:
filtered <- filter_genes(mat, min.cells = 10)

## End(Not run)
```

`find_cluster_markers` *Find Enriched Cell Types per Cluster*

Description

Identify cell types that are enriched in each cluster compared to the overall distribution using statistical testing.

Usage

```
find_cluster_markers(
  deconv,
  clusters,
  method = "wilcox",
  p.adjust.method = "BH"
)
```

Arguments

`deconv`	Deconvolution matrix (spots x celltypes)
`clusters`	Cluster assignments
`method`	Statistical test: "wilcox" or "t" (default: "wilcox")
`p.adjust.method`	P-value adjustment method (default: "BH")

Value

Data frame with enrichment statistics per cluster-celltype pair

`find_markers` *Find Marker Genes for Each Cell Type*

Description

Identify differentially expressed marker genes for each cell type using Wilcoxon rank-sum test. This is crucial for training the LDA model.

Usage

```
find_markers(
  matrix,
  celltypes,
  n.top = 200,
  min.pct = 0.1,
  logfc.threshold = 0.25,
  only.pos = TRUE,
  test.use = "wilcox",
  adjust.method = "BH",
  ncores = 1
)
```

Arguments

`matrix`	Count matrix (genes x cells), should be normalized
`celltypes`	Named vector of cell type annotations
`n.top`	Number of top markers per cell type (default: 200)
`min.pct`	Minimum fraction of cells expressing the gene (default: 0.1)
`logfc.threshold`	Minimum log fold change threshold (default: 0.25)
`only.pos`	Only return positive markers (default: TRUE)
`test.use`	Statistical test to use: "wilcox" or "t" (default: "wilcox")
`adjust.method`	P-value adjustment method (default: "BH")
`ncores`	Number of cores for parallel computation (default: 1)

Value

A list containing:

- markersData frame of all markers with statistics
- top_markersCharacter vector of top marker genes

Examples

```
## Not run:
markers <- find_markers(norm_mat, celltypes, n.top = 200)

## End(Not run)
```

`find_markers_scanpy` *Find Marker Genes (scanpy-style workflow)*

Description

Identify differentially expressed marker genes for each cell type using the exact scanpy/STRIDE workflow including preprocessing steps.

Usage

```
find_markers_scanpy(
  matrix,
  celltypes,
  n.top = 200,
  min.genes = 200,
  min.cells = 10,
  target.sum = 10000,
  n.hvg = NULL,
  ncores = 1
)
```

Arguments

`matrix`	Raw count matrix (genes x cells)
`celltypes`	Named vector of cell type annotations
`n.top`	Number of top markers per cell type (default: 200)
`min.genes`	Minimum genes per cell for filtering (default: 200)
`min.cells`	Minimum cells per gene for filtering (default: 10)
`target.sum`	Target sum for normalization (default: 1e4)
`n.hvg`	Number of highly variable genes to use (default: NULL, use all)
`ncores`	Number of cores for parallel computation (default: 1)

Details

This function replicates the scanpy workflow used in Python STRIDE:

1. Filter cells with < min.genes expressed genes
2. Filter genes expressed in < min.cells cells
3. Normalize counts to target.sum per cell

4. Log-transform: log1p(x)
5. Optionally select highly variable genes
6. Run rank_genes_groups with Wilcoxon test and tie correction

Value

A list containing:

- markersData frame of all markers with statistics
- top_markersCharacter vector of top marker genes

Examples

```
## Not run:
markers <- find_markers_scanpy(raw_count_mat, celltypes, n.top = 200)

## End(Not run)
```

`find_neighbors` *Find Spatial Neighbors*

Description

Find neighbors for each spot based on spatial coordinates using k-nearest neighbors or distance threshold.

Usage

```
find_neighbors(coords, k = 6, radius = NULL, include.self = FALSE)
```

Arguments

`coords`	Data frame with spatial coordinates (columns: x, y)
`k`	Number of nearest neighbors (default: 6)
`radius`	Distance radius for neighbors (alternative to k)
`include.self`	Include the spot itself as a neighbor (default: FALSE)

Details

For Visium data, k=6 corresponds to the hexagonal grid structure. For other technologies, adjust k based on expected spatial relationships.

Value

A list where each element contains indices of neighbors for each spot

Examples

```
## Not run:
neighbors <- find_neighbors(spatial_coords, k = 6)

## End(Not run)
```

`find_variable_genes` *Find Highly Variable Genes*

Description

Identify highly variable genes using the variance stabilization approach (similar to Seurat's VST method).

Usage

```
find_variable_genes(
  matrix,
  n.top = 2000,
  min.mean = 0.0125,
  max.mean = 3,
  min.var = 0.5,
  span = 0.3
)
```

Arguments

`matrix`	Count matrix (genes x cells)
`n.top`	Number of top variable genes (default: 2000)
`min.mean`	Minimum mean expression (default: 0.0125)
`max.mean`	Maximum mean expression (default: 3)
`min.var`	Minimum variance (default: 0.5)
`span`	Loess span parameter (default: 0.3)

Value

A list containing:

- hvgCharacter vector of highly variable genes
- statsData frame with gene statistics

Examples

```
## Not run:
hvg <- find_variable_genes(mat, n.top = 2000)

## End(Not run)
```

from_seurat *Create STRIDER Object from Seurat*

Description

Extract data from Seurat object for STRIDER analysis.

Usage

```
from_seurat(seurat, celltype.col, assay = "RNA")
```

Arguments

seurat	Seurat object
celltype.col	Column name in metadata containing cell types
assay	Assay to use (default: "RNA")

Value

List with matrix and celltypes

fused_gromov_wasserstein_cpp *Fused Gromov-Wasserstein Distance*

Description

Compute Fused Gromov-Wasserstein (FGW) distance, which combines feature-based Wasserstein distance and structure-based Gromov-Wasserstein distance.

Usage

```
fused_gromov_wasserstein_cpp(
  M,
  C1,
  C2,
  p,
  q,
  alpha = 0.5,
  reg = 0.1,
  max_iter = 100L,
  tol = 1e-07
)
```

Arguments

M	Feature distance/cost matrix (n x m)
C1	Source structural distance matrix (n x n)
C2	Target structural distance matrix (m x m)
p	Source distribution (n), must sum to 1
q	Target distribution (m), must sum to 1
alpha	Trade-off parameter between 0 and 1 (default: 0.5) • alpha = 0: pure Wasserstein (only feature cost) • alpha = 1: pure GW (only structural cost)
reg	Entropic regularization (default: 0.1)
max_iter	Maximum iterations (default: 100)
tol	Convergence tolerance (default: 1e-7)

Details

The FGW distance is defined as: FGW(p,q) = min_T (1-alpha) * <M, T> + alpha * GW(C1, C2, T) subject to T * 1 = p, T^T * 1 = q

Value

List with:

- T: Optimal transport plan (n x m)
- fgw_dist: FGW distance value

References

Vayer, T., et al. (2020). Fused Gromov-Wasserstein distance for structured objects. Algorithms.

---

get_default_colors *Default Color Palette*

Description

A carefully curated color palette for visualizing cell types and clusters. Designed to be colorblind-friendly and distinguishable.

Usage

```
get_default_colors()
```

Value

A character vector of 36 hex color codes

Examples

```
colors <- get_default_colors()
scales::show_col(colors)
```

`get_extended_colors` *Generate Color Palette*

Description

Generate a color palette with a specified number of colors.

Usage

```
get_extended_colors(n, palette = "default")
```

Arguments

`n`	Number of colors needed
`palette`	Base palette: "default" or "viridis"

Value

A character vector of hex color codes

Examples

```
get_extended_colors(10)
get_extended_colors(50, palette = "viridis")
```

`get_mapped_celltypes` *Get Mapped Cell Information*

Description

Extract information about cells mapped to each spot.

Usage

```
get_mapped_celltypes(mapping, celltypes, aggregate = "majority")
```

Arguments

`mapping`	Mapping result from map_spots_to_cells()
`celltypes`	Named vector of cell type annotations
`aggregate`	How to aggregate cell types: "majority", "all", or "weighted"

Value

Data frame with spot-level cell type information

`get_shared_genes` *Get Shared Genes Between Datasets*

Description

Find genes shared between single-cell and spatial datasets, optionally filtered by a provided gene list.

Usage

```
get_shared_genes(sc_genes, st_genes, gene_use = NULL)
```

Arguments

`sc_genes`	Character vector of single-cell genes
`st_genes`	Character vector of spatial genes
`gene_use`	Optional character vector of genes to use, or "All"

Value

Character vector of shared genes

Examples

```
## Not run:
shared <- get_shared_genes(sc_genes, st_genes, marker_genes)

## End(Not run)
```

`integrate_samples` *Integrate Multiple Spatial Transcriptomics Samples*

Description

Integrate multiple spatial transcriptomics samples using Fused Gromov-Wasserstein (FGW) optimal transport. This aligns samples based on both topic distribution similarity and spatial structure.

Usage

```
integrate_samples(
  deconv.list,
  topic.list,
  coords.list,
  sample.ids = NULL,
  alpha = 0.5,
  reg = 0.1,
  max.iter = 100,
  seed = 42
)
```

Arguments

deconv.list	List of deconvolution result matrices (spots x celltypes)
topic.list	List of topic distribution matrices (topics x spots)
coords.list	List of spatial coordinate data frames
sample.ids	Sample identifiers (default: S1, S2, ...)
alpha	Trade-off parameter between feature and spatial similarity (default: 0.5) • 0: pure OT (only topic similarity) • 1: pure GW (only spatial structure)
reg	Sinkhorn regularization parameter (default: 0.1)
max.iter	Maximum iterations for FGW (default: 100)
seed	Random seed

Details

The integration is performed sequentially, aligning each sample to the previous one. The FGW distance balances:

- Feature similarity: KL divergence between topic distributions
- Structural similarity: Gromov-Wasserstein distance on spatial structure

Value

A list containing:

- aligned_coordsData frame of aligned coordinates for all samples
- transport_plansList of optimal transport plans between samples
- deconvCombined deconvolution results
- rotation_matricesList of rotation matrices

References

Vayer, T., et al. (2020). Fused Gromov-Wasserstein distance for structured objects. Algorithms.

Examples

```
## Not run:
integrated <- integrate_samples(
  deconv.list = list(deconv1, deconv2, deconv3),
  topic.list = list(topic1, topic2, topic3),
  coords.list = list(coords1, coords2, coords3)
)

## End(Not run)
```

---

integration *Multi-Sample Integration Functions for STRIDER*

Description

Functions for integrating multiple spatial transcriptomics samples using Fused Gromov-Wasserstein optimal transport

---

integration_quality *Calculate Integration Quality Metrics*

Description

Calculate metrics to assess integration quality.

Usage

```
integration_quality(integration_result)
```

Arguments

integration_result	Result from integrate_samples()

Value

Data frame with quality metrics

`kl_divergence_matrix_cpp` *Compute KL Divergence Matrix*

Description

Compute pairwise KL divergence between probability distributions (rows of P and Q).

Usage

```
kl_divergence_matrix_cpp(P, Q)
```

Arguments

`P`	First matrix (n x k), rows are distributions
`Q`	Second matrix (m x k), rows are distributions

Details

KL(p||q) = sum_k p_k * log(p_k / q_k)

Value

KL divergence matrix (n x m) where entry at row i, col j is KL(P_i || Q_j)

`lda_model` *LDA Topic Model for STRIDER*

Description

Functions for training and applying LDA topic models

map_spots_to_cells *Map Spatial Spots to Similar Single Cells*

Description

Identify the most similar single cells for each spatial spot based on topic distribution similarity.

Usage

```
map_spots_to_cells(topic.spot, topic.cell, n.top = 10, method = "euclidean")
```

Arguments

topic.spot	Topic distribution for spots (topics x spots)
topic.cell	Topic distribution for cells (topics x cells)
n.top	Number of top similar cells per spot (default: 10)
method	Distance metric: "euclidean", "cosine", or "correlation" (default: "euclidean")

Value

A data frame with spot-cell mappings and similarity scores

Examples

```
## Not run:
mapping <- map_spots_to_cells(topic_spot, topic_cell, n.top = 10)

## End(Not run)
```

mapping *Spot-to-Cell Mapping Functions for STRIDER*

Description

Functions for mapping spatial spots to similar single cells

marker_find *Marker Gene Identification for STRIDER*

Description

Functions for identifying cell-type specific marker genes.

model_evaluate *Model Evaluation Functions for STRIDER*

Description

Functions for evaluating and selecting the best LDA model

normalize_counts *Normalize Counts*

Description

Normalize count matrix by total counts per cell/spot.

Usage

```
normalize_counts(matrix, scale.factor = NULL, log.transform = FALSE)
```

Arguments

matrix	Count matrix (genes x cells/spots)
scale.factor	Target sum for normalization (default: NULL, auto-detect)
log.transform	Apply log1p transformation (default: FALSE)

Value

Normalized matrix

Examples

```
## Not run:
normalized <- normalize_counts(mat, scale.factor = 10000)

## End(Not run)
```

plot *Visualization Functions for STRIDER*

Description

Functions for visualizing deconvolution and analysis results

plot_celltype_heatmap *Plot Cell Type Composition Heatmap*

Description

Create heatmap of cell type composition across spots.

Usage

```
plot_celltype_heatmap(
  deconv_mat,
  title = "Cell Type Composition",
  scale.by = "none"
)
```

Arguments

deconv_mat	Deconvolution matrix (spots x celltypes)
title	Plot title
scale.by	Scale values: "row", "column", or "none"

Value

A ggplot object

plot_cluster *Plot Cluster Results*

Description

Visualize spatial clustering results.

Usage

```
plot_cluster(cluster_result, pt.size = 4, colors = NULL, title = NULL)
```

Arguments

cluster_result	Clustering result from spatial_cluster()
pt.size	Point size (default: 4)
colors	Color palette
title	Plot title

Value

A ggplot object

plot_deconv *Plot Deconvolution Results*

Description

Main plotting function for STRIDER deconvolution results. Supports scatter pie plots, scatter plots, and heatmaps.

Usage

```
plot_deconv(
  deconv,
  coords,
  plot.type = "scatterpie",
  sample.id = NULL,
  pt.size = NULL,
  colors = NULL,
  title = NULL,
  coord.flip = FALSE
)
```

Arguments

deconv	Deconvolution result (STRIDER object or matrix)
coords	Spatial coordinates data frame
plot.type	Type of plot: "scatterpie", "scatter", or "heatmap"
sample.id	Sample ID to plot (if multiple samples)
pt.size	Point size (default: auto)
colors	Color palette (default: STRIDER palette)
title	Plot title (default: auto)
coord.flip	Flip x and y coordinates (default: FALSE)

Value

A ggplot object

Examples

```
## Not run:
plot_deconv(strider_result, coords, plot.type = "scatterpie")

## End(Not run)
```

plot_deconv_pie *Scatter Pie Plot for Deconvolution*

Description

Create scatter pie plot showing cell type composition at each spot.

Usage

```
plot_deconv_pie(
  deconv_mat,
  coords,
  pt.size = 8,
  colors = NULL,
  title = "Cell Type Distribution",
  coord.flip = FALSE
)
```

Arguments

deconv_mat	Deconvolution matrix (spots x celltypes)
coords	Spatial coordinates
pt.size	Point/pie size
colors	Color palette
title	Plot title
coord.flip	Flip coordinates

Value

A ggplot object

plot_deconv_scatter *Scatter Plot for Deconvolution*

Description

Create scatter plot colored by dominant cell type.

Usage

```
plot_deconv_scatter(
  deconv_mat,
  coords,
  pt.size = 2,
  colors = NULL,
  title = "Dominant Cell Type",
  coord.flip = FALSE
)
```

Arguments

deconv_mat	Deconvolution matrix (spots x celltypes)
coords	Spatial coordinates
pt.size	Point size
colors	Color palette
title	Plot title
coord.flip	Flip coordinates

Value

A ggplot object

---

plot_integration *Plot Integration Results*

Description

Visualize integrated samples.

Usage

```
plot_integration(
  integration_result,
  color.by = "sample",
  pt.size = 2,
  deconv = NULL
)
```

Arguments

integration_result	Result from integrate_samples()
color.by	Color spots by: "sample", "celltype", or NULL
pt.size	Point size (default: 2)
deconv	Deconvolution results for cell type coloring

Value

A ggplot object

---

plot_mapping *Visualize Spot-Cell Mapping*

Description

Create visualization of spot-cell mapping relationships.

Usage

```
plot_mapping(mapping, celltypes, spots = NULL)
```

Arguments

mapping	Mapping result from map_spots_to_cells()
celltypes	Named vector of cell type annotations
spots	Specific spots to visualize (default: first 10)

Value

A ggplot object

---

plot_model_selection *Plot Model Selection Results*

Description

Visualize model evaluation metrics across topic numbers.

Usage

```
plot_model_selection(metrics, metric = "accuracy")
```

Arguments

metrics	Metrics data frame from evaluate_topic_range()
metric	Which metric to plot: "accuracy", "coherence", or "nmi"

Value

A ggplot object

plot_topic_heatmap *Plot Topic Distribution Heatmap*

Description

Create heatmap of topic distributions across samples.

Usage

```
plot_topic_heatmap(
  topic_mat,
  celltypes = NULL,
  title = "Topic Distribution",
  cluster.rows = TRUE,
  cluster.cols = FALSE
)
```

Arguments

topic_mat	Topic distribution matrix (topics x cells/spots)
celltypes	Named vector of cell type annotations (optional)
title	Plot title
cluster.rows	Cluster rows (default: TRUE)
cluster.cols	Cluster columns (default: FALSE)

Value

A ggplot object

plot_validation *Compare Cell Type Proportions*

Description

Visualize comparison between STRIDER and reference results.

Usage

```
plot_validation(strider_result, reference_result, plot_type = "scatter")
```

Arguments

strider_result	STRIDER result
reference_result	Reference result
plot_type	Type of plot: "scatter", "bland_altman", "correlation"

Value

A ggplot object

`preprocess` *Preprocessing Functions for STRIDER*

Description

Functions for preprocessing single-cell and spatial transcriptomics data

---

`preprocess_sc` *Preprocess Single-Cell Data*

Description

Complete preprocessing pipeline for single-cell RNA-seq data.

Usage

```
preprocess_sc(
  matrix,
  celltypes,
  scale.factor = NULL,
  min.cells = 10,
  min.genes = 200,
  normalize = TRUE,
  scale = FALSE
)
```

Arguments

`matrix`	Count matrix (genes x cells)
`celltypes`	Named vector of cell type annotations
`scale.factor`	Target sum for normalization (default: NULL)
`min.cells`	Minimum cells per gene (default: 10)
`min.genes`	Minimum genes per cell (default: 200)
`normalize`	Whether to normalize (default: TRUE)
`scale`	Whether to scale by SD (default: FALSE)

Value

A list containing preprocessed data

Examples

```
## Not run:
sc_data <- preprocess_sc(mat, celltypes)

## End(Not run)
```

`preprocess_st` *Preprocess Spatial Data*

Description

Complete preprocessing pipeline for spatial transcriptomics data.

Usage

```
preprocess_st(matrix, scale.factor = NULL, normalize = TRUE, scale = FALSE)
```

Arguments

`matrix`	Count matrix (genes x spots)
`scale.factor`	Target sum for normalization (default: NULL)
`normalize`	Whether to normalize (default: TRUE)
`scale`	Whether to scale by SD (default: FALSE)

Value

A list containing preprocessed data

Examples

```
## Not run:
st_data <- preprocess_st(mat)

## End(Not run)
```

print.STRIDER *Print STRIDER Object*

Description

Print STRIDER Object

Usage

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

Arguments

x	A STRIDER object
...	Additional arguments (unused)

print.strider_benchmark *Print Benchmark Results*

Description

Print Benchmark Results

Usage

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

Arguments

x	A strider_benchmark object
...	Additional arguments (unused)

print.strider_integration *Print Method for Integration Result*

Description

Print Method for Integration Result

Usage

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

Arguments

x	A strider_integration object
...	Additional arguments (unused)

print.strider_lda *Print Method for LDA Model*

Description

Print Method for LDA Model

Usage

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

Arguments

x	A strider_lda object
...	Additional arguments (unused)

print.strider_validation *Print Validation Report*

Description

Print Validation Report

Usage

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

Arguments

x	A strider_validation object
...	Additional arguments (unused)

read_10x_h5 *Read 10X Genomics HDF5 File*

Description

Read gene expression data from 10X Genomics HDF5 format. Supports both Cell Ranger v2 and v3 formats.

Usage

```
read_10x_h5(filename, use.names = TRUE, unique.features = TRUE)
```

Arguments

filename	Path to the HDF5 file
use.names	Use gene names instead of IDs (default: TRUE)
unique.features	Make feature names unique (default: TRUE)

Value

A list containing:

- matrixSparse dgCMatrix of counts
- featuresGene names/IDs
- barcodesCell/spot barcodes

Examples

```
## Not run:
data <- read_10x_h5("filtered_feature_bc_matrix.h5")

## End(Not run)
```

`read_celltype_annotations` *Read Cell Type Annotations*

Description

Read cell type annotations from tab-separated text file. File should have two columns: cell ID and cell type.

Usage

```
read_celltype_annotations(filename, sep = "\t", header = FALSE)
```

Arguments

`filename`	Path to the annotation file
`sep`	Field separator (default: tab)
`header`	Whether file has header (default: FALSE)

Value

A named vector mapping cell IDs to cell types

Examples

```
## Not run:
celltypes <- read_celltype_annotations("celltype.txt")

## End(Not run)
```

`read_count_matrix` *Read Count Matrix from Text File*

Description

Read gene expression count matrix from tab-separated text file. First column should contain gene names, first row should contain cell/spot IDs.

Usage

```
read_count_matrix(
  filename,
  sep = "\t",
  header = TRUE,
  row.names = 1,
  sparse = TRUE
)
```

Arguments

`filename`	Path to the text file
`sep`	Field separator (default: tab)
`header`	Whether file has header (default: TRUE)
`row.names`	Column containing row names (default: 1)
`sparse`	Convert to sparse matrix (default: TRUE)

Value

A list containing:

- matrixCount matrix (sparse or dense)
- featuresGene names
- barcodesCell/spot barcodes

Examples

```
## Not run:
data <- read_count_matrix("gene_count.txt")

## End(Not run)
```

read_gene_list *Read Gene List*

Description

Read a list of genes from a text file (one gene per line).

Usage

```
read_gene_list(filename)
```

Arguments

filename	Path to the gene list file

Value

A character vector of gene names

Examples

```
## Not run:
genes <- read_gene_list("markers.txt")

## End(Not run)
```

read_spatial_coords *Read Spatial Coordinates*

Description

Read spatial coordinates from tab-separated text file. First column should contain spot IDs, followed by coordinates.

Usage

```
read_spatial_coords(
  filename,
  sep = "\t",
  header = TRUE,
  spot.col = 1,
  coord.cols = c(2, 3),
  sample.col = NULL
)
```

Arguments

filename	Path to the text file
sep	Field separator (default: tab)
header	Whether file has header (default: TRUE)
spot.col	Column containing spot IDs (default: 1)
coord.cols	Columns containing coordinates (default: c(2, 3))
sample.col	Column containing sample IDs (optional)

Value

A data.frame with columns: spot, x, y (and optionally sample)

Examples

```
## Not run:
coords <- read_spatial_coords("location.txt")

## End(Not run)
```

---

run_strider *Run STRIDER Deconvolution Pipeline*

Description

Main entry point for STRIDER cell type deconvolution. This function performs the complete pipeline including marker identification, LDA training, model selection, and spatial deconvolution.

Usage

```
run_strider(
  sc.data,
  st.data,
  sc.celltype,
  gene.use = NULL,
  n.topics = NULL,
  method = "auto",
  sc.scale.factor = NULL,
  st.scale.factor = NULL,
  normalize = FALSE,
  n.markers = 200,
  lda.iter = 500,
  seed = 42,
  out.dir = ".",
  out.prefix = "STRIDER",
  save.model = TRUE
)
```

Arguments

sc.data	Single-cell data. Can be: • A matrix (genes x cells) • A Seurat object (V4 or V5) • A file path to count matrix (.h5 or .txt)
st.data	Spatial transcriptomics data. Same formats as sc.data.
sc.celltype	Cell type annotations. Can be: • A named vector (cell -> celltype) • Column name in Seurat metadata • A file path to annotation file
gene.use	Genes to use for model training: • NULL: auto-detect markers (default) • "All": use all shared genes • Character vector: specific genes • File path: gene list file
n.topics	Number of topics, or vector of numbers to test. Default: seq(n_celltypes, 3*n_celltypes)
method	Deconvolution method: "auto", "Raw", "Norm", "NormBySD", "Bayes", "BayesNorm"
sc.scale.factor	Scale factor for single-cell normalization (default: auto)
st.scale.factor	Scale factor for spatial normalization (default: auto)
normalize	Whether to scale by SD (default: FALSE)
n.markers	Number of marker genes per cell type (default: 200)
lda.iter	LDA iterations (default: 500)
seed	Random seed (default: 42)
out.dir	Output directory (default: ".")
out.prefix	Output file prefix (default: "STRIDER")
save.model	Save trained model (default: TRUE)

Details

The STRIDER pipeline consists of the following steps:

1. Load and preprocess single-cell and spatial data
2. Identify marker genes for each cell type
3. Train LDA topic model on single-cell data
4. Evaluate models with different topic numbers
5. Select optimal model and deconvolution method
6. Apply model to spatial data for deconvolution

Value

A STRIDER object containing:

- deconvCell type fractions matrix (spots x celltypes)
- topic_spotTopic distributions for spots (topics x spots)
- modelTrained LDA model
- methodSelected deconvolution method
- n_topicsSelected number of topics
- metricsModel evaluation metrics

References

Sun, D., et al. (2022). STRIDE: accurately decomposing and integrating spatial transcriptomics using single-cell RNA sequencing. Nucleic Acids Research.

Examples

```
## Not run:
# From matrices
result <- run_strider(
  sc.data = sc_matrix,
  st.data = st_matrix,
  sc.celltype = celltypes
)

# From Seurat objects
result <- run_strider(
  sc.data = seurat_sc,
  st.data = seurat_st,
  sc.celltype = "celltype"
)

## End(Not run)
```

---

save_plot *Save Plot to File*

Description

Save a ggplot object to file with sensible defaults.

Usage

```
save_plot(plot, filename, width = 8, height = 6, dpi = 300)
```

Arguments

plot	A ggplot object
filename	Output filename
width	Plot width in inches (default: 8)
height	Plot height in inches (default: 6)
dpi	Resolution (default: 300)

---

scale_data *Scale Data by Standard Deviation*

Description

Scale data by dividing by standard deviation (without centering). Uses population SD (N) instead of sample SD (N-1) to match sklearn's StandardScaler(with_mean=False) used in Python STRIDE.

Usage

```
scale_data(matrix, center = FALSE, scale = TRUE)
```

Arguments

matrix	Normalized matrix (genes x cells/spots)
center	Center data (subtract mean) (default: FALSE)
scale	Scale data (divide by SD) (default: TRUE)

Value

Scaled matrix

Examples

```
## Not run:
scaled <- scale_data(mat)

## End(Not run)
```

select_best_model *Select Best Model*

Description

Automatically select the best model from evaluation results.

Usage

```
select_best_model(eval_results, criterion = "accuracy")
```

Arguments

eval_results	Results from evaluate_topic_range()
criterion	Selection criterion: "accuracy", "coherence", or "balanced"

Value

Selected model components

spatial_cluster *Spatial Clustering Based on Cell Type Composition*

Description

Perform clustering of spatial spots based on cell type composition with optional neighborhood information.

Usage

```
spatial_cluster(
  deconv,
  coords,
  n.clusters = 5,
  weight = 0.5,
  k = 6,
  method = "kmeans",
  seed = 42
)
```

Arguments

deconv	Deconvolution result (STRIDER object or matrix)
coords	Spatial coordinates data frame
n.clusters	Number of clusters (default: 5)
weight	Weight for spot's own composition vs neighbors (default: 0.5) • 1.0: only own composition • 0.0: only neighbor composition • 0.5: equal weight (default)
k	Number of neighbors for local composition (default: 6)
method	Clustering method: "kmeans" or "hierarchical" (default: "kmeans")
seed	Random seed

Details

The clustering combines each spot's own cell type composition with the average composition of its spatial neighbors. This produces spatially coherent clusters that reflect local tissue microenvironments.

Value

A data frame with spot coordinates and cluster assignments

Examples

```
## Not run:
clusters <- spatial_cluster(strider_result, coords, n.clusters = 5)

## End(Not run)
```

---

summary.STRIDER *Summary of STRIDER Object*

Description

Summary of STRIDER Object

Usage

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

Arguments

object	A STRIDER object
...	Additional arguments (unused)

`train_lda` *Train LDA Topic Model*

Description

Train a Latent Dirichlet Allocation (LDA) topic model on single-cell gene expression data. The model treats cells as documents and genes as words, following the methodology described in STRIDE.

Usage

```
train_lda(
  matrix,
  n.topics = 10,
  n.iter = 500,
  n.passes = 1,
  convergence.tol = 1e-04,
  alpha = NULL,
  beta = 0.01,
  gamma.threshold = 0.001,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`matrix`	Normalized count matrix (genes x cells)
`n.topics`	Number of topics (default: 10)
`n.iter`	Number of iterations per pass (default: 500)
`n.passes`	Number of passes over the corpus (default: 1)
`convergence.tol`	Convergence tolerance (default: 1e-4)
`alpha`	Dirichlet prior for document-topic distribution. Default is 50/n.topics as in original STRIDE (matching gensim).
`beta`	Dirichlet prior for topic-word distribution (default: 0.01)
`gamma.threshold`	Minimum change in gamma for document convergence (default: 0.001)
`seed`	Random seed for reproducibility
`verbose`	Print progress messages (default: TRUE)

Details

The LDA model is trained using Variational Bayes inference via the text2vec package, matching gensim's LdaModel behavior as closely as possible. The default hyperparameters (alpha = 50/K, beta = 0.01) follow the original STRIDE implementation for optimal performance on single-cell data.

Note: gensim uses Online Variational Bayes by default, while text2vec uses standard Variational Bayes. For single-cell data with moderate corpus size, results are highly correlated.

Value

An LDA model object with components:

- topic_geneTopic-gene distribution matrix (phi)
- topic_cellDocument-topic distribution matrix (theta)
- vocabGene vocabulary
- n_topicsNumber of topics

References

Sun, D., et al. (2022). STRIDE: accurately decomposing and integrating spatial transcriptomics using single-cell RNA sequencing. Nucleic Acids Research.

Examples

```
## Not run:
model <- train_lda(norm_mat, n.topics = 10)

## End(Not run)
```

---

train_model *Train LDA Model (Convenience Wrapper)*

Description

Train an LDA model from single-cell data with automatic topic-celltype association calculation.

Usage

```
train_model(
  sc.matrix,
  celltypes,
  gene.use = NULL,
  n.topics = 10,
  n.iter = 500,
  seed = 42
)
```

Arguments

sc.matrix	Single-cell count matrix (genes x cells)
celltypes	Named vector of cell type annotations
gene.use	Genes to use (optional)
n.topics	Number of topics
n.iter	LDA iterations
seed	Random seed

Value

Trained LDA model object with topic-celltype matrices

---

transfer_labels *Transfer Labels from Single Cells to Spots*

Description

Transfer cell type labels from single-cell reference to spatial spots using mapping results.

Usage

```
transfer_labels(mapping, celltypes, method = "knn", k = NULL)
```

Arguments

mapping	Mapping result from map_spots_to_cells()
celltypes	Named vector of cell type annotations
method	Transfer method: "knn" (k-nearest neighbor voting) or "weighted"
k	Number of neighbors for kNN (default: all mapped cells)

Value

Named vector of transferred labels

transform_lda *Apply LDA Model to New Data*

Description

Apply a trained LDA model to transform new data (e.g., spatial spots) into topic space using variational inference.

Usage

```
transform_lda(model, newdata, n.iter = 100)
```

Arguments

model	Trained LDA model from train_lda()
newdata	Matrix to transform (genes x cells/spots)
n.iter	Number of inference iterations (default: 100)

Details

This function performs inference on new documents using the trained topic-word distributions. Genes not present in the training vocabulary are ignored.

Value

Topic distribution matrix (topics x cells/spots)

Examples

```
## Not run:
topic_dist <- transform_lda(model, st_matrix)

## End(Not run)
```

validate *Scientific Validation Functions for STRIDER*

Description

Functions for validating STRIDER results against benchmarks and assessing scientific equivalence with Python STRIDE.

validate_strider *Validate STRIDER Results Against Reference*

Description

Compare STRIDER deconvolution results with reference results (e.g., from Python STRIDE) to assess scientific equivalence.

Usage

```
validate_strider(
  strider_result,
  reference_result,
  metrics = c("correlation", "dominant", "error", "rank"),
  tolerance = 0.01
)
```

Arguments

strider_result	STRIDER result object or deconvolution matrix (spots x celltypes)
reference_result	Reference result (matrix, data frame, or file path)
metrics	Which metrics to compute (default: all)
tolerance	Numerical tolerance for "exact" comparisons (default: 0.01)

Details

The validation assesses scientific equivalence rather than numerical identity. Two results are considered equivalent if:

- Pearson correlation > 0.95
- Dominant cell type match > 0.90
- RMSE < 0.05

Value

A validation report object with:

- pearson_rPearson correlation of flattened matrices
- spearman_rSpearman correlation of flattened matrices
- rmseRoot mean squared error
- maeMean absolute error
- dominant_matchFraction of spots with same dominant cell type
- rank_correlationMean rank correlation per spot
- verdict"EQUIVALENT" if metrics pass thresholds

Examples

```
## Not run:
validation <- validate_strider(strider_result, "stride_result.txt")
print(validation)

## End(Not run)
```

`write_10x_h5` *Write 10X Genomics HDF5 File*

Description

Write gene expression data to 10X Genomics HDF5 format.

Usage

```
write_10x_h5(matrix, features, barcodes, filename, overwrite = FALSE)
```

Arguments

`matrix`	Sparse matrix (genes x cells/spots)
`features`	Gene names
`barcodes`	Cell/spot barcodes
`filename`	Output file path
`overwrite`	Overwrite existing file (default: FALSE)

Value

Invisible NULL

Examples

```
## Not run:
write_10x_h5(mat, genes, cells, "output.h5")

## End(Not run)
```

`write_results` *Write Results to File*

Description

Write STRIDER results to tab-separated text files.

Usage

```
write_results(x, filename, row.names = TRUE, col.names = TRUE)
```

Arguments

`x`	Data to write (matrix or data.frame)
`filename`	Output file path
`row.names`	Include row names (default: TRUE)
`col.names`	Include column names (default: TRUE)

Value

Invisible NULL

Index