Package: CytoSPACER (via r-universe)

June 1, 2026

Type Package

Title Mapping Single-Cell Transcriptomes to Spatial Transcriptomics Data

Version 1.0.0

Date 2026-01-25

Description CytoSPACER is an R implementation of CytoSPACE, a computational strategy for assigning single-cell transcriptomes to in situ spatial transcriptomics (ST) data. The method solves single cell/spot assignment by minimizing a correlation-based cost function through a linear programming-based optimization routine using the Jonker-Volgenant algorithm. This package provides high-performance implementations using Rcpp and supports parallel processing across all platforms.

License MIT + file LICENSE

URL https://zaoqu-liu.github.io/CytoSPACER/, https://github.com/Zaoqu-Liu/CytoSPACER

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

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.10), data.table (>= 1.14.0), Matrix (>= 1.5.0), future (>= 1.32.0), future.apply (>= 1.11.0), progressr (>= 0.14.0), ggplot2 (>= 3.4.0), methods, stats, utils

Suggests Seurat (>= 4.0.0), SeuratObject (>= 4.0.0), testthat (>= 3.0.0), knitr, rmarkdown, viridis

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

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

Date/Publication 2026-01-24 18:51:37 UTC

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

RemoteRef main

RemoteSha 885560a47a2170304bac1c921277a4633ee7a2e0

Contents

add_cytospace_to_seurat *Add CytoSPACER Results to Seurat Object*

Description

Add CytoSPACER assignment results as metadata to a Seurat spatial object.

Usage

```
add_cytospace_to_seurat(seurat_obj, results)
```

Arguments

seurat_obj	Seurat spatial object.
results	CytoSPACER results from run_cytospace.

Value

Updated Seurat object with new metadata columns.

add_noise_to_cost *Add Noise to Cost Matrix*

Description

Add small random noise to cost matrix for tie-breaking.

Usage

```
add_noise_to_cost(cost_matrix, scale = 1e-10, seed = NULL)
```

Arguments

cost_matrix	Cost matrix.
scale	Scale of random noise. Default is 1e-10.
seed	Random seed. Default is NULL.

Value

Cost matrix with added noise.

cell_sampling *Cell Sampling Functions for CytoSPACER*

Description

Functions for sampling single cells based on cell type composition.

compute_assignment_cost *Compute Total Assignment Cost*

Description

Compute the total cost of an assignment.

Usage

```
compute_assignment_cost(cost_matrix, assignment)
```

Arguments

cost_matrix	The cost matrix.
assignment	Integer vector of column assignments.

Value

The total cost (sum of assigned costs).

Examples

```
## Not run:
cost <- matrix(c(1, 2, 3, 2, 4, 6, 3, 6, 9), nrow = 3)
assignment <- solve_lap(cost)
total <- compute_assignment_cost(cost, assignment)

## End(Not run)
```

`compute_cost_matrix` *Compute Cost Matrix*

Description

Compute the cost matrix between single cells and spatial spots based on expression similarity. The cost matrix is used as input to the linear assignment problem solver.

Usage

```
compute_cost_matrix(
  sc_data,
  st_data,
  method = c("pearson", "spearman", "euclidean"),
  use_cpp = TRUE
)
```

Arguments

`sc_data`	Normalized single-cell expression matrix (genes x cells).
`st_data`	Normalized spatial expression matrix (genes x spots).
`method`	Distance/similarity metric:
	"pearson" Pearson correlation (default). Cost = -correlation.
	"spearman" Spearman rank correlation. Cost = -correlation.
	"euclidean" Euclidean distance.
`use_cpp`	Logical. If TRUE, use C++ implementation. Default is TRUE.

Details

For correlation-based methods, the cost is computed as $-r$ where r is the correlation coefficient. This transforms the similarity measure into a cost for minimization. This matches the Python CytoSPACE implementation exactly.

Value

A cost matrix of dimensions (n_cells x n_spots).

Examples

```
## Not run:
cost <- compute_cost_matrix(sc_norm, st_norm, method = "pearson")

## End(Not run)
```

cost_matrix *Cost Matrix Computation for CytoSPACER*

Description

Functions for computing cost matrices for the linear assignment problem.

data_io *Data Input/Output Functions for CytoSPACER*

Description

Functions for reading and writing CytoSPACER input/output files.

downsample_expression *Downsample Expression Matrix*

Description

Downsample expression matrix to a target number of transcripts per cell. This helps ensure that cell assignments are not biased by overall expression levels.

Usage

```
downsample_expression(data, target_count = 1500, seed = NULL)
```

Arguments

`data`	Gene expression count matrix (genes x cells).
`target_count`	Target number of transcripts per cell. Default is 1500.
`seed`	Random seed for reproducibility. Default is NULL.

Details

For each cell, if the total count exceeds `target_count`, transcripts are randomly sampled without replacement. Cells with fewer transcripts are left unchanged.

Value

Downsampled matrix with the same dimensions as input.

Examples

```
## Not run:
downsampled <- downsample_expression(counts, target_count = 1500)

## End(Not run)
```

`estimate_cells_per_spot` *Estimate Cells per Spot*

Description

Estimate the number of cells in each ST spot based on total RNA content and an expected mean cell count.

Usage

```
estimate_cells_per_spot(st_data, mean_cells = 5, min_cells = 1)
```

Arguments

`st_data`	Spatial expression matrix (genes x spots).
`mean_cells`	Expected mean number of cells per spot. Default is 5 (appropriate for 10x Visium).
`min_cells`	Minimum cells per spot. Default is 1.

Details

The estimation uses a linear model relating total normalized expression to cell counts, anchored at the minimum and mean values.

Value

An integer vector of estimated cell counts per spot.

Examples

```
## Not run:
cell_counts <- estimate_cells_per_spot(st_expr, mean_cells = 5)

## End(Not run)
```

`estimate_fractions` *Estimate Cell Type Fractions*

Description

Estimate cell type fractions from spatial transcriptomics data using reference-based deconvolution.

Usage

```
estimate_fractions(
  sc_data,
  cell_types,
  st_data,
  method = "seurat",
  max_cells = 10000
)

estimate_cell_type_fractions(
  sc_data,
  cell_types,
  st_data,
  method = "seurat",
  max_cells = 10000
)
```

Arguments

`sc_data`	Single-cell expression matrix (genes x cells).
`cell_types`	Named vector of cell types for each cell.
`st_data`	Spatial expression matrix (genes x spots).
`method`	Deconvolution method. Currently "seurat" is supported.
`max_cells`	Maximum cells to use for computational efficiency. Default is 10000.

Details

This function requires the Seurat package (v4 or v5). It uses SCTransform for normalization and FindTransferAnchors/TransferData for deconvolution.

Value

A data.frame with cell type fractions per spot.

Examples

```
## Not run:
fractions <- estimate_fractions(sc_expr, cell_labels, st_expr)

## End(Not run)
```

`extract_seurat_data` *Extract CytoSPACER Input from Seurat Object*

Description

Extract expression data and metadata from a Seurat object.

Usage

```
extract_seurat_data(
  seurat_obj,
  assay = "RNA",
  cell_type_col = NULL,
  output_dir = NULL,
  prefix = "",
  sparse = FALSE
)
```

Arguments

`seurat_obj`	A Seurat object.
`assay`	Assay to extract. Default is "RNA".
`cell_type_col`	Column containing cell type labels.
`output_dir`	Directory to save output files. If NULL, returns data.
`prefix`	File name prefix for output files.
`sparse`	Save expression as sparse matrix. Default is FALSE.

Value

If `output_dir` is NULL, returns a list. Otherwise saves files.

Examples

```
## Not run:
# Extract to files
extract_seurat_data(seurat_obj, "RNA", "celltype", "input/")

# Extract to R objects
data <- extract_seurat_data(seurat_obj, "RNA", "celltype")
```

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

`extract_spatial_data` *Extract Spatial Data from Seurat Object*

Description

Extract spatial expression data and coordinates from a Seurat spatial object.

Usage

```
extract_spatial_data(
  seurat_obj,
  assay = "Spatial",
  image_name = NULL,
  output_dir = NULL,
  prefix = "",
  sparse = FALSE
)
```

Arguments

`seurat_obj`	A Seurat object with spatial data.
`assay`	Assay to extract. Default is "Spatial".
`image_name`	Name of the image. Default is first image.
`output_dir`	Directory to save output files.
`prefix`	File name prefix.
`sparse`	Save as sparse matrix. Default is FALSE.

Value

If `output_dir` is NULL, returns a list. Otherwise saves files.

`lap_solver` *Linear Assignment Problem Solvers for CytoSPACER*

Description

R interface to linear assignment problem (LAP) solvers.

`main` *Main CytoSPACER Analysis Functions*

Description

Core functions for running the CytoSPACER analysis pipeline.

`normalization` *Data Normalization Functions for CytoSPACER*

Description

Functions for normalizing gene expression data.

`normalize_expression` *Normalize Expression Data*

Description

Normalize gene expression data using TPM normalization followed by log2(x+1) transformation.

Usage

```
normalize_expression(data, scale_factor = 1e+06, use_cpp = TRUE)
```

Arguments

`data`	Gene expression matrix (genes x cells/spots). Can be a matrix, data.frame, or sparse Matrix.
`scale_factor`	Scaling factor for normalization. Default is 1e6 (TPM).
`use_cpp`	Logical. If TRUE, use C++ implementation. Default is TRUE.

Details

The normalization performs:

1. TPM normalization: $x_{ij} = x_{ij} / \sum_i x_{ij} \times 10^6$
2. Log transformation: $x_{ij} = \log_2(x_{ij} + 1)$

This is the standard normalization used in CytoSPACE for computing cell-to-spot similarities.

Value

Normalized matrix with the same dimensions as input.

Examples

```
## Not run:
normalized <- normalize_expression(counts_matrix)

## End(Not run)
```

`plot_composition` *Plot Cell Type Composition*

Description

Plot cell type composition as bar charts.

Usage

```
plot_composition(
  results,
  type = c("global", "per_spot"),
  top_spots = 20,
  colors = NULL
)
```

Arguments

`results`	Results from `run_cytospace`.
`type`	Type of plot: **"global"** Overall cell type proportions **"per_spot"** Proportions per spot (stacked bars)
`top_spots`	Number of top spots for per_spot plot. Default is 20.
`colors`	Named vector of colors.

Value

A ggplot2 object.

`plot_cytospace` *Plot CytoSPACER Results*

Description

Generate visualizations of CytoSPACER cell assignment results.

Usage

```
plot_cytospace(
  results,
  type = c("cell_types", "by_spot", "density"),
  coordinates = NULL,
  point_size = 1,
  alpha = 0.8,
  jitter = 0,
  max_cells = 50000,
  ncol = 3,
  colors = NULL,
  title = NULL
)
```

Arguments

`results`	Results from `run_cytospace`.
`type`	Type of plot to generate: **"cell_types"** Cell type spatial distribution (default) **"by_spot"** Cell counts per spot, faceted by cell type **"density"** Density of cell assignments
`coordinates`	Optional coordinates data.frame.
`point_size`	Size of points. Default is 1.
`alpha`	Point transparency. Default is 0.8.
`jitter`	Amount of jitter to add. Default is 0.
`max_cells`	Maximum cells to plot. Default is 50000.
`ncol`	Number of columns for faceted plots. Default is 3.
`colors`	Named vector of colors per cell type.
`title`	Plot title.

Value

A ggplot2 object.

Examples

```
## Not run:
results <- run_cytospace(...)
plot_cytospace(results, type = "cell_types")
plot_cytospace(results, type = "cell_types", jitter = 0.3)

## End(Not run)
```

`plotting` *Visualization Functions for CytoSPACER*

Description

Functions for visualizing CytoSPACER results.

`read_cytospace_input` *Read CytoSPACER Input Files*

Description

Read various input file formats supported by CytoSPACER.

Usage

```
read_cytospace_input(file_path, as_sparse = FALSE)
```

Arguments

`file_path`	Path to the input file. Supported formats: • `.csv` - Comma-separated values • `.txt`, `.tsv` - Tab-separated values • `.mtx`, `.mtx.gz` - Matrix Market format (sparse)
`as_sparse`	Logical. If TRUE, return sparse matrix. Default is FALSE.

Details

For sparse matrix files (.mtx), the function expects companion files:

- `genes.tsv` or `features.tsv` - Gene names
- `barcodes.tsv` or `cells.tsv` - Cell/spot barcodes

Value

A data.frame or dgCMatrix (if sparse) with genes as rows and cells/spots as columns.

Examples

```
## Not run:
expr <- read_cytospace_input("scRNA_data.csv")
expr_sparse <- read_cytospace_input("matrix.mtx", as_sparse = TRUE)

## End(Not run)
```

`read_visium_data` *Read Visium Data from SpaceRanger Output*

Description

Read 10x Visium spatial transcriptomics data.

Usage

```
read_visium_data(path, use_seurat = FALSE)
```

Arguments

`path`	Path to SpaceRanger output directory or .tar.gz archive
`use_seurat`	Logical. If TRUE and Seurat is available, use Load10X_Spatial. Default is FALSE.

Value

A list containing:

expression Gene expression matrix (genes x spots)

coordinates Spot coordinates data.frame

Examples

```
## Not run:
visium_data <- read_visium_data("path/to/spaceranger/output")

## End(Not run)
```

`run_cytospace` *Run CytoSPACE Analysis*

Description

Main function for assigning single cells to spatial transcriptomics spots using the CytoSPACE algorithm. This is a complete R implementation of the Python CytoSPACE package.

Usage

```
run_cytospace(
  sc_data,
  cell_types,
  st_data,
  coordinates,
  cell_type_fractions = NULL,
  cells_per_spot = NULL,
  mean_cells_per_spot = 5,
  single_cell = FALSE,
  st_cell_types = NULL,
  distance_metric = c("pearson", "spearman", "euclidean"),
  sampling_method = c("duplicates", "synthetic"),
  downsample = TRUE,
  downsample_target = 1500,
  sampling_sub_spots = FALSE,
  number_of_selected_spots = 10000,
  number_of_selected_sub_spots = 10000,
  n_workers = 1,
  seed = 1,
  verbose = TRUE
)
```

Arguments

`sc_data`	Single-cell RNA-seq expression matrix (genes x cells). Can be a matrix, data.frame, or sparse Matrix.
`cell_types`	Named character vector of cell type labels. Names should match column names of `sc_data`.
`st_data`	Spatial transcriptomics expression matrix (genes x spots).
`coordinates`	Data.frame with spot coordinates. Must have columns 'row' and 'col' (or 'X' and 'Y'), with rownames matching `st_data`.
`cell_type_fractions`	Optional data.frame of cell type fractions per spot. If NULL, will be estimated using Seurat.
`cells_per_spot`	Optional integer vector of cell counts per spot. If NULL, will be estimated from RNA content.

`mean_cells_per_spot`
: Expected mean cells per spot for estimation. Default is 5 (appropriate for 10x Visium).

`single_cell`
: Logical. If TRUE, treat ST data as single-cell spatial (one cell per spot). Default is FALSE.

`st_cell_types`
: Optional named vector of cell types for single-cell ST. Required when single_cell=TRUE for optimal assignment.

`distance_metric`
: Distance metric for cost matrix. One of "pearson" (default), "spearman", or "euclidean".

`sampling_method`
: Method for handling cell count mismatches. One of "duplicates" (default) or "synthetic" (called "place_holders" in Python).

`downsample`
: Logical. If TRUE, downsample scRNA-seq to equal counts. Default is TRUE.

`downsample_target`
: Target count for downsampling. Default is 1500.

`sampling_sub_spots`
: Logical. If TRUE, use sub-spot sampling for large datasets. Default is FALSE.

`number_of_selected_spots`
: Chunk size for single_cell mode. Default 10000.

`number_of_selected_sub_spots`
: Chunk size for sampling_sub_spots mode. Default is 10000.

`n_workers`
: Number of parallel workers. Default is 1.

`seed`
: Random seed for reproducibility. Default is 1.

`verbose`
: Logical. If TRUE, print progress messages. Default is TRUE.

Details

The CytoSPACE algorithm assigns single cells to spatial spots by:

1. Estimating cell type fractions and cell counts per spot
2. Sampling single cells to match the ST cell composition
3. Computing a cost matrix based on expression similarity
4. Solving the linear assignment problem (LAP) using Jonker-Volgenant

For large datasets, use `sampling_sub_spots=TRUE` to partition the problem into manageable chunks processed in parallel.

Value

A list containing:

assigned_locations Data.frame with cell-to-spot assignments

expression Expression matrix for assigned cells

cell_type_by_spot Cell type counts per spot (bulk ST only)

fractional_abundances Cell type fractions per spot (bulk ST only)

parameters List of input parameters

log Character vector of log messages

runtime Total execution time in seconds

References

Vahid, M.R., et al. (2023). High-resolution alignment of single-cell and spatial transcriptomes with CytoSPACE. Nature Biotechnology 41, 1543-1548.

Examples

```
## Not run:
# Standard bulk ST analysis
results <- run_cytospace(
  sc_data = sc_expr,
  cell_types = cell_labels,
  st_data = st_expr,
  coordinates = coords,
  mean_cells_per_spot = 5
)

# Single-cell ST with known cell types
results <- run_cytospace(
  sc_data = sc_expr,
  cell_types = cell_labels,
  st_data = st_expr,
  coordinates = coords,
  single_cell = TRUE,
  st_cell_types = st_labels
)

# Large dataset with sub-spot sampling
results <- run_cytospace(
  sc_data = sc_expr,
  cell_types = cell_labels,
  st_data = st_expr,
  coordinates = coords,
  sampling_sub_spots = TRUE,
  number_of_selected_sub_spots = 5000,
  n_workers = 4
)

## End(Not run)
```

`run_cytospace_seurat` *Run CytoSPACER from Seurat Objects*

Description

Run CytoSPACER analysis directly from Seurat objects.

Usage

```
run_cytospace_seurat(
  sc_seurat,
  st_seurat,
  cell_type_col = NULL,
  sc_assay = "RNA",
  st_assay = "Spatial",
  image_name = NULL,
  ...
)
```

Arguments

`sc_seurat`	Seurat object containing scRNA-seq data.
`st_seurat`	Seurat object containing spatial transcriptomics data.
`cell_type_col`	Column name in sc_seurat metadata containing cell type labels. Default is NULL (uses Idents).
`sc_assay`	Assay to use from sc_seurat. Default is "RNA".
`st_assay`	Assay to use from st_seurat. Default is "Spatial".
`image_name`	Name of the image in st_seurat. Default is the first image.
`...`	Additional arguments passed to `run_cytospace`.

Details

This function automatically extracts expression matrices, cell types, and coordinates from Seurat objects. Compatible with Seurat v4 and v5.

Value

A list containing CytoSPACER results.

Examples

```
## Not run:
sc <- readRDS("scRNA_seurat.rds")
st <- readRDS("visium_seurat.rds")
results <- run_cytospace_seurat(sc, st, cell_type_col = "celltype")

## End(Not run)
```

`sample_cells` *Sample Single Cells by Cell Type Composition*

Description

Sample single cells from the scRNA-seq dataset to match the estimated cell type composition in the ST data.

Usage

```
sample_cells(
  sc_data,
  cell_types,
  target_counts,
  method = c("duplicates", "synthetic"),
  seed = NULL
)

sample_cells_by_type(
  sc_data,
  cell_types,
  target_counts,
  method = c("duplicates", "synthetic"),
  seed = NULL
)
```

Arguments

`sc_data`	Single-cell expression matrix (genes x cells).
`cell_types`	Named character vector of cell types for each cell.
`target_counts`	Named integer vector of target cell counts per type.
`method`	Sampling method: **"duplicates"** Allow duplicating cells when needed (default). Ensures all original cells are included first. **"synthetic"** Generate synthetic cells by sampling gene expression values from the same cell type.
`seed`	Random seed for reproducibility.

Value

A list containing:

expression Sampled expression matrix

cell_ids Vector of original cell IDs

cell_types Vector of cell types for sampled cells

is_synthetic Logical vector indicating synthetic cells

Examples

```
## Not run:
# Define target counts per cell type
targets <- c(B_cell = 100, T_cell = 200, Macrophage = 50)

# Sample cells
sampled <- sample_cells(sc_expr, cell_labels, targets)

## End(Not run)
```

`save_cytospace_plot` *Save CytoSPACER Plots*

Description

Save CytoSPACER visualization to files.

Usage

```
save_cytospace_plot(
  plot,
  output_dir,
  prefix = "",
  formats = c("png", "pdf"),
  width = 10,
  height = 8,
  dpi = 300
)
```

Arguments

`plot`	A ggplot2 object.
`output_dir`	Output directory.
`prefix`	File name prefix.
`formats`	File formats to save. Default is c("png", "pdf").
`width`	Plot width in inches. Default is 10.
`height`	Plot height in inches. Default is 8.
`dpi`	Resolution for raster formats. Default is 300.

Value

Invisibly returns file paths.

seurat_integration *Seurat Integration Functions for CytoSPACER*

Description

Functions for integrating CytoSPACER with Seurat objects.

solve_lap *Solve Linear Assignment Problem*

Description

Solve the linear assignment problem using the Jonker-Volgenant algorithm. This is the core optimization step in CytoSPACE.

Usage

```
solve_lap(cost_matrix, maximize = FALSE)
```

Arguments

cost_matrix	A numeric matrix of costs. For rectangular matrices, the number of rows must be less than or equal to columns.
maximize	Logical. If TRUE, maximize instead of minimize. Default is FALSE.

Details

The Jonker-Volgenant algorithm is an efficient O(n^3) algorithm for solving the linear assignment problem. It is particularly well-suited for dense cost matrices.

For rectangular matrices where rows < columns, the algorithm pads the matrix with dummy rows and returns only the assignments for the original rows.

Value

An integer vector of column assignments. The i-th element indicates which column is assigned to row i (1-indexed).

References

Jonker, R., & Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing, 38(4), 325-340.

Examples

```
## Not run:
# Create a simple cost matrix
cost <- matrix(c(1, 2, 3, 2, 4, 6, 3, 6, 9), nrow = 3, byrow = TRUE)

# Solve the assignment problem
assignment <- solve_lap(cost)

## End(Not run)
```

`solve_lap_parallel` *Solve Multiple Assignment Problems in Parallel*

Description

Solve multiple assignment subproblems in parallel.

Usage

```
solve_lap_parallel(cost_matrices, n_workers = NULL, progress = TRUE)
```

Arguments

`cost_matrices`	A list of cost matrices.
`n_workers`	Number of parallel workers. Default is auto-detected.
`progress`	Logical. If TRUE, show progress. Default is TRUE.

Details

Uses the future framework for cross-platform parallel processing.

Value

A list of assignment vectors.

Examples

```
## Not run:
cost_list <- list(cost1, cost2, cost3)
assignments <- solve_lap_parallel(cost_list, n_workers = 4)

## End(Not run)
```

`utils` *Utility Functions for CytoSPACER*

Description

Internal utility functions used by CytoSPACER.

`write_cytospace_results` *Write CytoSPACER Results*

Description

Write CytoSPACER results to output files.

Usage

```
write_cytospace_results(
  results,
  output_dir,
  prefix = "",
  save_expression = TRUE,
  sparse_output = FALSE
)
```

Arguments

`results`	A list returned by `run_cytospace`
`output_dir`	Directory to save output files
`prefix`	Prefix for output file names
`save_expression`	Logical. If TRUE, save expression matrix. Default TRUE.
`sparse_output`	Logical. If TRUE, save expression in sparse MTX format.

Value

Invisibly returns the output directory path.

Examples

```
## Not run:
results <- run_cytospace(...)
write_cytospace_results(results, "output/", prefix = "analysis1_")

## End(Not run)
```

Index