

Package ‘RankMap’

May 8, 2026

Title Rank-based reference mapping for fast and robust cell type annotation in spatial and single-cell transcriptomics

Version 1.0.0

Description RankMap is a fast and scalable tool for reference-based cell type annotation of single-cell and spatial transcriptomics data. It uses ranked gene expression and multinomial regression to achieve robust predictions, even with partial gene coverage. Compatible with Seurat, SingleCellExperiment, and SpatialExperiment objects, RankMap offers flexible preprocessing and significantly faster runtime than tools like SingleR, Azimuth, and RCTD.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, glmnet, graphics, magrittr, Matrix, matrixStats, rlang, Seurat, stats, SummarizedExperiment

biocViews Spatial, SingleCell, Transcriptomics, GeneExpression, Annotation, Regression, Preprocessing, Software

URL <https://github.com/jinming-cheng/RankMap>

BugReports <https://github.com/jinming-cheng/RankMap/issues>

Depends R (>= 4.6.0)

Suggests BiocStyle, knitr, rmarkdown, SingleCellExperiment, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

git_url <https://git.bioconductor.org/packages/RankMap>

git_branch RELEASE_3_23

git_last_commit 4a49131

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-05-07

Author Jinming Cheng [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-3806-4694>>)

Maintainer Jinming Cheng <jinming.cheng@outlook.com>

Contents

RankMap-package	2
computeRankedMatrix	3
evaluatePredictionPerformance	4
extractData	5
factorSorted	6
filterLowConfidenceCells	7
maskTopKGenes	8
optimizeConfidenceThreshold	8
predictRankModel	10
RankMap	11
sampleCellsByType	13
trainRankModel	14
Index	16

RankMap-package	<i>RankMap: Rank-based reference mapping for fast and robust cell type annotation in spatial and single-cell transcriptomics</i>
-----------------	--

Description

RankMap is a fast and scalable tool for reference-based cell type annotation of single-cell and spatial transcriptomics data. It uses ranked gene expression and multinomial regression to achieve robust predictions, even with partial gene coverage. Compatible with Seurat, SingleCellExperiment, and SpatialExperiment objects, RankMap offers flexible preprocessing and significantly faster runtime than tools like SingleR, Azimuth, and RCTD.

Author(s)

Maintainer: Jinming Cheng <jinming.cheng@outlook.com> (ORCID)

See Also

Useful links:

- <https://github.com/jinming-cheng/RankMap>
- Report bugs at <https://github.com/jinming-cheng/RankMap/issues>

`computeRankedMatrix` *Compute Ranked Gene Expression Matrix*

Description

This function transforms a gene expression matrix by applying top-k filtering, optional rank transformation, binning, scaling, and/or expression weighting.

Usage

```
computeRankedMatrix(  
  data,  
  weight_by_expr = TRUE,  
  rank_zeros = FALSE,  
  bin_rank = TRUE,  
  scale_rank = TRUE,  
  k = 20,  
  use_data = FALSE  
)
```

Arguments

<code>data</code>	A gene expression matrix with genes as rows and cells (or spatial spots) as columns. Can be a dense numeric matrix, or a sparse <code>dgMatrix</code> .
<code>weight_by_expr</code>	Logical. Whether to weight the ranks by log-transformed expression values. Default is TRUE.
<code>rank_zeros</code>	Logical. Whether to include zero values in the ranking. Default is FALSE.
<code>bin_rank</code>	Logical. Whether to discretize ranks into bins. Default is TRUE.
<code>scale_rank</code>	Logical. Whether to z-score normalize the ranks across columns. Default is TRUE.
<code>k</code>	Integer. Number of top expressed genes to retain per column. Default is 20.
<code>use_data</code>	Logical. If TRUE, returns the full input expression matrix (unfiltered and un-ranked). Default is FALSE.

Details

If `use_data = TRUE`, the function bypasses all transformation steps and returns the full input expression matrix. This is useful for benchmarking performance of rank-based versus raw log-normalized expression models.

Value

A numeric matrix of ranked expression values (or the original expression matrix if `use_data = TRUE`). Output is always returned as a dense matrix.

See Also

[maskTopKGenes](#), [trainRankModel](#), [predictRankModel](#)

Examples

```
mat <- matrix(runif(1000), nrow = 100)
ranked <- computeRankedMatrix(mat, k = 10)
raw_expr <- computeRankedMatrix(mat, use_data = TRUE)
```

evaluatePredictionPerformance

Evaluate Prediction Accuracy and Confusion by Cell Type

Description

Computes overall accuracy, per-class accuracy, and confusion matrix given predicted vs. true labels.

Usage

```
evaluatePredictionPerformance(  
  prediction_df = NULL,  
  truth = NULL,  
  low_conf_label = "unknown"  
)
```

Arguments

`prediction_df` Output from [predictRankModel](#) (`return_confidence = TRUE`) or filtered version.

`truth` A vector of true labels matching `prediction_df`.

`low_conf_label` Character. Label used for low-confidence predictions (e.g., "unknown" or "uncertain"). These will be excluded from evaluation. Set to NULL to include all predictions.

Value

A list with:

`overall_accuracy`
Proportion of correct predictions

`per_class_accuracy`
Accuracy per true cell type

`confusion_matrix`
Contingency table (true × predicted)

Examples

```
# Read in single-cell reference data
seu_sc <- readRDS(system.file("extdata", "seu_sc.rds",
  package = "RankMap"
))

# Read in Xenium spatial data
seu_xen <- readRDS(system.file("extdata", "seu_xen.rds",
  package = "RankMap"
))

# Predict cell type for Xenium data
prediction_df <- RankMap(
  ref_data = seu_sc,
  ref_labels = seu_sc$cell_type,
  new_data = seu_xen
)

performance <- evaluatePredictionPerformance(
  prediction_df = prediction_df,
  truth = seu_xen$cell_type_SingleR
)
performance
```

extractData	<i>Extract Expression Matrix from Seurat, SummarizedExperiment, or Matrix</i>
-------------	---

Description

Extracts a gene expression matrix from a Seurat object (default assay and "data" slot), a SummarizedExperiment object (assay named "logcounts"), or returns the input matrix if it is already a dense or sparse matrix.

Usage

```
extractData(data)
```

Arguments

data	A Seurat object, a SummarizedExperiment object, a dense numeric matrix, or a sparse matrix of class dgCMatix
------	--

Value

A numeric gene expression matrix (dense or sparse), with genes as rows and cells or spots as columns.

Examples

```
seu_sc <- readRDS(system.file("extdata", "seu_sc.rds",
                             package = "RankMap"
                             ))

# From Seurat object:
mat <- extractData(seu_sc)
```

factorSorted	<i>Order factor levels by frequency</i>
--------------	---

Description

Takes a vector, converts it to a factor, and reorders its levels by their frequency of occurrence (most to least frequent by default).

Usage

```
factorSorted(x, decreasing = TRUE)
```

Arguments

x	A vector (character or factor) to be reordered by level frequency.
decreasing	Logical. If TRUE (default), levels are ordered from most to least frequent; if FALSE, from least to most frequent.

Details

Frequencies are computed with `table(x)`, which ignores NA values by default. NA entries in `x` are preserved in the output but are not considered a level.

Value

A factor with levels ordered by frequency.

Examples

```
factorSorted(c("a", "b", "a", "c", "b", "a"))
factorSorted(c("a", "b", "a", "c", "b", "a"), decreasing = FALSE)
factorSorted(factor(c("x", "y", "x", NA)))
```

`filterLowConfidenceCells`*Tag Low-Confidence Predictions and Add Status Column*

Description

Replaces low-confidence cell type predictions with a placeholder label (e.g., "unknown") and adds a confidence status column ("confident" or "uncertain").

Usage

```
filterLowConfidenceCells(  
  prediction_df,  
  threshold = 0.5,  
  low_conf_label = "unknown",  
  keep_confidence = TRUE  
)
```

Arguments

`prediction_df` A data frame from `predictRankModel` (`return_confidence = TRUE`).

`threshold` Numeric. Confidence threshold below which predictions are flagged. Default is 0.5.

`low_conf_label` Character. Replacement for low-confidence predictions. Default is "unknown".

`keep_confidence` Logical. Whether to retain the original confidence column. Default is TRUE.

Value

A data frame with columns: `cell_id`, `predicted_cell_type`, `status`, and optionally `confidence`.

Examples

```
# Simulated predictions with confidence  
pred_df <- data.frame(  
  cell_id = paste0("cell", 1:5),  
  predicted_cell_type = c("B", "T", "B", "Myeloid", "T"),  
  confidence = c(0.92, 0.47, 0.88, 0.33, 0.76)  
)  
  
# Apply threshold of 0.5 to flag low-confidence cells  
result <- filterLowConfidenceCells(pred_df, threshold = 0.5)  
  
# Show result  
print(result)  
  
# Remove confidence column and use custom label  
# for low-confidence predictions
```

```

result2 <- filterLowConfidenceCells(pred_df,
  threshold = 0.6,
  low_conf_label = "low_conf",
  keep_confidence = FALSE
)

```

maskTopKGenes	<i>Mask Non-Top-K Genes in Each Cell or Spot</i>
---------------	--

Description

This function zeroes out all but the top-k most highly expressed genes in each column of a gene expression matrix. It supports both dense and sparse inputs.

Usage

```
maskTopKGenes(data, k = 20)
```

Arguments

data	A gene expression matrix with genes as rows and cells (or spatial spots) as columns. Can be a dense numeric matrix, or a sparse dgMatrix.
k	Integer. Number of top expressed genes to retain per column. Default is 20.

Value

A dense numeric matrix with the same dimensions as data, with only the top-k values retained per column.

Examples

```

mat <- matrix(runif(1000), nrow = 100)
masked <- maskTopKGenes(mat, k = 10)

```

optimizeConfidenceThreshold	<i>Optimize Confidence Threshold for Cell Type Prediction</i>
-----------------------------	---

Description

Finds the confidence threshold that best balances prediction accuracy and retention rate. Requires ground truth labels for comparison.

Usage

```
optimizeConfidenceThreshold(  
  prediction_df,  
  truth,  
  thresholds = seq(0.1, 0.9, by = 0.05),  
  plot = TRUE  
)
```

Arguments

`prediction_df` A data frame from `predictRankModel` (`return_confidence = TRUE`).

`truth` A character or factor vector of true labels, aligned with rows of `prediction_df`.

`thresholds` Numeric vector of thresholds to test. Default is `seq(0.1, 0.9, by = 0.05)`.

`plot` Logical. If TRUE, generates a plot of accuracy vs. threshold. Default is TRUE.

Value

A data frame summarizing accuracy and retained cell count at each threshold.

Examples

```
# Simulated prediction data  
pred_df <- data.frame(  
  cell_id = paste0("cell", 1:10),  
  predicted_cell_type = c(  
    "A", "B", "A", "B", "A",  
    "B", "A", "A", "B", "B"  
  ),  
  confidence = c(  
    0.95, 0.87, 0.65, 0.48, 0.92,  
    0.55, 0.73, 0.33, 0.99, 0.60  
  )  
)  
  
# Ground truth labels  
truth <- c("A", "B", "A", "B", "B", "B", "A", "A", "B", "B")  
  
# Evaluate how accuracy and coverage change with threshold  
summary_df <- optimizeConfidenceThreshold(pred_df, truth, plot = TRUE)  
  
# View result  
print(summary_df)
```

predictRankModel *Predict Cell Types, Probabilities, or Confidence Scores*

Description

Predicts cell type labels or class probabilities using a trained RankMap model. Supports both `glmnet` and `cv.glmnet`. Optionally returns a data frame with predicted cell types and confidence scores.

Usage

```
predictRankModel(
  model,
  new_data,
  lambda = NULL,
  return_probs = FALSE,
  return_confidence = FALSE,
  ...
)
```

Arguments

<code>model</code>	A fitted model from trainRankModel .
<code>new_data</code>	A gene expression matrix with genes as rows and cells (or spatial spots) as columns. Can be a Seurat object, a SummarizedExperiment object, a dense numeric matrix, or a sparse dgCMatix.
<code>lambda</code>	Optional lambda value. If NULL, uses <code>lambda.min</code> (if available), else <code>0.01</code> .
<code>return_probs</code>	Logical. If TRUE, returns the full matrix of class probabilities.
<code>return_confidence</code>	Logical. If TRUE, returns a data frame with predicted labels and confidence scores.
<code>...</code>	Additional arguments passed to computeRankedMatrix .

Value

A character vector (default), a matrix (if `return_probs = TRUE`), or a data frame (if `return_confidence = TRUE`).

Examples

```
# Read in single-cell reference data
seu_sc <- readRDS(system.file("extdata", "seu_sc.rds",
  package = "RankMap"
))

# Read in Xenium spatial data
```

```
seu_xen <- readRDS(system.file("extdata", "seu_xen.rds",
  package = "RankMap"
))

# Extract normalized expression data
common_genes <- intersect(rownames(seu_sc), rownames(seu_xen))
mat <- extractData(seu_sc)[common_genes, ]
new_mat <- extractData(seu_xen)[common_genes, ]

# Train a model
set.seed(42)
model <- trainRankModel(mat, seu_sc$cell_type)

# Predict cell type
pred <- predictRankModel(model, new_mat)

table(predict = pred, truth = seu_xen$cell_type_SingleR)
```

RankMap

RankMap: Train and Predict Cell Types Using Top-Ranked Genes

Description

A unified function that trains a multinomial classifier on reference expression data and predicts cell types for a new dataset using top-k ranked genes. Automatically matches genes between datasets, optionally performs cross-validation, and returns predictions with confidence scores or probabilities.

Usage

```
RankMap(
  ref_data = NULL,
  ref_labels = NULL,
  new_data = NULL,
  n_feature_max = 500,
  k = 20,
  alpha = 0.1,
  cv = FALSE,
  nfolds = 5,
  lambda = NULL,
  return_probs = FALSE,
  return_confidence = TRUE,
  threshold = NULL,
  return_model = FALSE,
  ...
)
```

Arguments

<code>ref_data</code>	Reference gene expression matrix (genes x cells), a Seurat object, or a SummarizedExperiment object.
<code>ref_labels</code>	A character or factor vector of cell type labels for columns of <code>ref_data</code> .
<code>new_data</code>	New data to annotate. Same format as <code>ref_data</code> (matrix, dgCMatix, Seurat object or SummarizedExperiment object).
<code>n_feature_max</code>	Maximum number of genes to use when more than 500 genes are shared. Default is 500.
<code>k</code>	Number of top expressed genes to retain per cell (ranking). Default is 20.
<code>alpha</code>	Elastic net mixing parameter for <code>glmnet</code> . Default is 0.1.
<code>cv</code>	Logical. Whether to use <code>cv.glmnet</code> for cross-validation. Default is FALSE.
<code>nfolds</code>	Number of folds for cross-validation. Default is 5.
<code>lambda</code>	Optional lambda value for prediction. If NULL, uses <code>lambda.min</code> from CV or defaults to 0.01.
<code>return_probs</code>	Logical. If TRUE, returns full class probability matrix. Default is FALSE.
<code>return_confidence</code>	Logical. If TRUE, returns prediction with confidence score and status. Default is TRUE.
<code>threshold</code>	Optional numeric threshold. If set and <code>return_confidence = TRUE</code> , predictions below this confidence are labeled as "unknown".
<code>return_model</code>	Logical. If TRUE, returns a list containing both predictions and the trained model. Default is FALSE.
<code>...</code>	Additional arguments passed to <code>computeRankedMatrix</code> .

Value

A data frame of predictions (by default), or a list with elements `predictions` and `model` if `return_model = TRUE`.

Examples

```
# Read in single-cell reference data
seu_sc <- readRDS(system.file("extdata", "seu_sc.rds",
  package = "RankMap"
))

# Read in Xenium spatial data
seu_xen <- readRDS(system.file("extdata", "seu_xen.rds",
  package = "RankMap"
))

# Predict cell type for spatial data using single-cell data as reference
pred_df <- RankMap(
  ref_data = seu_sc,
  ref_labels = seu_sc$cell_type,
  new_data = seu_xen
```

```
)  
head(pred_df)
```

sampleCellsByType	<i>Sample Cells by Cell Type with Minimum Representation</i>
-------------------	--

Description

This function samples a specified total number of cells from a metadata table, ensuring that each cell type is represented by at least a minimum number of cells.

Usage

```
sampleCellsByType(cell_metadata, n_total_cells = 5000, min_per_type = 50)
```

Arguments

`cell_metadata` A data frame containing at least two columns: `cell_id` and `cell_type`.
`n_total_cells` Total number of cells to sample. Default is 5000.
`min_per_type` Minimum number of cells to sample from each `cell_type`. Default is 50.

Value

A data frame of sampled cells with at least `min_per_type` cells per `cell_type`.

Examples

```
meta <- data.frame(  
  cell_id = paste0("cell", 1:10000),  
  cell_type = sample(c("T", "B", "Mac"), 10000, replace = TRUE)  
)  
set.seed(42)  
sampled <- sampleCellsByType(meta,  
  n_total_cells = 1000,  
  min_per_type = 50  
)
```

trainRankModel	<i>Train Multinomial Rank-Based Model</i>
----------------	---

Description

This function trains a multinomial logistic regression model using ranked gene expression. Optionally performs cross-validation to select the optimal regularization parameter.

Usage

```
trainRankModel(
  data = NULL,
  labels = NULL,
  alpha = 0.1,
  cv = FALSE,
  nfolds = 5,
  ...
)
```

Arguments

data	A gene expression matrix with genes as rows and cells (or spatial spots) as columns. Can be a Seurat object, a SummarizedExperiment object, a dense numeric matrix, or a sparse dgCMatrix.
labels	A vector of cell type labels (one per column of data).
alpha	Elastic net mixing parameter. 1 = LASSO, 0 = Ridge. Default is 0.1.
cv	Logical. If TRUE, performs cross-validation using <code>cv.glmnet</code> . Default is FALSE.
nfolds	Number of folds for cross-validation (only used if <code>cv = TRUE</code>). Default is 5.
...	Additional arguments passed to computeRankedMatrix .

Value

A fitted `glmnet` or `cv.glmnet` model object.

See Also

[computeRankedMatrix](#), [predictRankModel](#)

Examples

```
# Read in single-cell reference data
seu_sc <- readRDS(system.file("extdata", "seu_sc.rds",
  package = "RankMap"
))

# Extract normalized expression data
mat <- extractData(seu_sc)
```

```
# Train a model
set.seed(42)
model <- trainRankModel(mat, seu_sc$cell_type)

# Train a model with cross-validation
model <- trainRankModel(
  data = mat,
  labels = seu_sc$cell_type,
  cv = TRUE,
  nfolds = 3
)
```

Index

* **internal**

- RankMap-package, [2](#)
- computeRankedMatrix, [3](#), [10](#), [12](#), [14](#)
- evaluatePredictionPerformance, [4](#)
- extractData, [5](#)
- factorSorted, [6](#)
- filterLowConfidenceCells, [7](#)
- maskTopKGenes, [4](#), [8](#)
- optimizeConfidenceThreshold, [8](#)
- predictRankModel, [4](#), [7](#), [9](#), [10](#), [14](#)
- RankMap, [11](#)
- RankMap-package, [2](#)
- sampleCellsByType, [13](#)
- trainRankModel, [4](#), [10](#), [14](#)