Package 'SPOTlight'

February 2, 2026

Version 1.15.0

Type Package

Title `SPOTlight`: Spatial Transcriptomics Deconvolution

Description `SPOTlight` provides a method to deconvolute spatial transcriptomics spots using a seeded NMF approach along with visualization tools to assess the results. Spatially resolved gene expression profiles are key to understand tissue organization and function. However, novel spatial transcriptomics (ST) profiling techniques lack single-cell resolution and require a combination with single-cell RNA sequencing (scRNA-seq) information to deconvolute the spatially indexed datasets. Leveraging the strengths of both data types, we developed SPOTlight, a computational tool that enables the integration of ST with scRNA-seq data to infer the location of cell types and states within a complex tissue. SPOTlight is centered around a seeded non-negative matrix factorization (NMF) regression, initialized using cell-type marker genes and non-negative least squares (NNLS) to subsequently deconvolute ST capture locations (spots).

Depends R (>= 4.5.0)

Imports ggplot2, Matrix, SingleCellExperiment, sparseMatrixStats, stats

Suggests BiocStyle, colorBlindness, DelayedArray, DropletUtils, ExperimentHub, ggcorrplot, grDevices, grid, igraph, jpeg, knitr, methods, png, rmarkdown, scater, scatterpie, scran, SpatialExperiment, SummarizedExperiment, S4Vectors, TabulaMurisSenisData, TENxVisiumData, testthat

LinkingTo Rcpp, RcppEigen

biocViews SingleCell, Spatial, StatisticalMethod

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

URL https://github.com/MarcElosua/SPOTlight

BugReports https://github.com/MarcElosua/SPOTlight/issues

git_url https://git.bioconductor.org/packages/SPOTlight

git_branch devel

git_last_commit b112ee3

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-02-01

Author Marc Elosua-Bayes [aut, cre],
Zachary DeBruine [aut],
Helena L. Crowell [aut]

Maintainer Marc Elosua-Bayes <elosua.marc@gmail.com>

Contents

`.filter` *.init_nmf <- function(x, groups, mgs, n_top = NULL, gene_id = "gene", group_id = "cluster", weight_id = "weight") # check validity of input arguments if (is.null(n_top)) n_top <- max(table(mgs[[group_id]])) stopifnot(is.character(gene_id), length(gene_id) == 1, is.character(group_id), length(group_id) == 1, is.character(weight_id), length(weight_id) == 1, c(gene_id, group_id, weight_id) is.numeric(n_top), length(n_top) == 1, round(n_top) == n_top) ng <- nrow(x) nc <- ncol(x) names(ks) <- ks <- unique(groups) # subset 'n_top' features mgs <- split(mgs, mgs[[group_id]]) mgs <- lapply(mgs, function(df) o <- order(df[[weight_id]], decreasing = TRUE) n <- ifelse(nrow(df) < n_top, nrow(df), n_top) df[o,][seq_len(n),]) # subset unique features mgs <- lapply(ks, function(k) g1 <- mgs[[k]][[gene_id]] g2 <- unlist(lapply(mgs[ks != k], '[[', gene_id)) mgs[[k]][!g1) # W is of dimension (#groups)x(#features) with W(i,j) # equal to weight if j is marker for i, and ~0 otherwise W <- vapply(ks, function(k) w <- numeric(ng) + 1e-12 names(w) <- rownames(x) ws <- mgs[[k]][[weight_id]] w[mgs[[k]][[gene_id]]] <- ws return(w) , numeric(ng)) # there is no need to initialize H tp <- paste0("topic_", seq_len(length(ks))) dimnames(W) <- list(rownames(x), tp) return(W) Filter features from expression matrix*

Description

Remove undetected features and optionally keep only shared features between the expression matrix and a reference set of features.

Usage

```
.filter(x, y)
```

Arguments

`x`	Expression matrix to filter
`y`	Vector of feature names to keep (optional)

Details

This function:

- Removes features with zero expression across all samples
- Optionally filters to keep only features present in both datasets
- Ensures a minimum of 10 features remain after filtering

Value

Filtered expression matrix

data *Synthetic single-cell, mixture and marker data*

Description

mockSC/mockSP() are designed to generate synthetic single-cell and spatial mixture data. These data are not meant to represent biologically meaningful use-cases, but are solely intended for use in examples, for unit-testing, and to demonstrate SPOTlight's general functionality. Finally, .get_mgs() implements a statistically naive way to select markers from single-cell data; again, please don't use it in real life.

Usage

```
mockSC(ng = 200, nc = 50, nt = 3)

mockSP(x, ns = 100)

getMGS(x, n_top = 10)
```

Arguments

ng, nc, nt, ns	integer scalar specifying the number of genes, cells, types (groups) and spots to simulate.
x	Single cell experiment object
n_top	integer specifying the number of marker genes to extract for each cluster.

Value

- mockSC returns a SingleCellExperiment with rows = genes, columns = single cells, and cell metadata (colData) column type containing group identifiers.
- mockSP returns a SingleCellExperiment with rows = genes, columns = single cells, and cell metadata (colData) column type containing group identifiers.
- getMGS returns a data.frame with nt*n_top rows and 3 columns: gene and type (group) identifier, as well as the gene's weight = the proportion of counts accounted for by that type.

Examples

```
sce <- mockSC()
spe <- mockSP(sce)
mgs <- getMGS(sce)
```

plotCorrelationMatrix *Plot Correlation Matrix*

Description

This function takes in a matrix with the predicted proportions for each spot and returns a correlation matrix between cell types.

Usage

```
plotCorrelationMatrix(
  x,
  cor.method = c("pearson", "kendall", "spearman"),
  insig = c("blank", "pch"),
  colors = c("#6D9EC1", "white", "#E46726"),
  hc.order = TRUE,
  p.mat = TRUE,
  ...
)
```

Arguments

x	numeric matrix with rows = samples and columns = cell types Must have at least two rows and two columns.
cor.method	Method to use for correlation: c("pearson", "kendall", "spearman"). By default pearson.
insig	character, specialized insignificant correlation coefficients, "pch", "blank" (default). If "blank", wipe away the corresponding glyphs; if "pch", add characters (see pch for details) on corresponding glyphs.
colors	character vector with three colors indicating the lower, mid, and high color. By default c("#6D9EC1", "white", "#E46726").
hc.order	logical value. If TRUE, correlation matrix will be hc.ordered using hclust function.
p.mat	logical value. If TRUE (default), correlation significance will be used. If FALSE arguments sig.level, insig, pch, pch.col, pch.cex are invalid.
...	additional graphical parameters passed to ggcorrplot.

Value

ggplot object

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
set.seed(321)
x <- replicate(m <- 25, runif(10, 0, 1))
rownames(x) <- paste0("spot", seq_len(nrow(x)))
colnames(x) <- paste0("type", seq_len(ncol(x)))

# The most basic example
plotCorrelationMatrix(x = x)

# Showing the non-significant correlatinos
plotCorrelationMatrix(x = x, insig = "pch")

# A more elaborated
plotCorrelationMatrix(
    x = x,
    hc.order = FALSE,
    type = "lower",
    outline.col = "lightgrey",
    method = "circle",
    colors = c("#64ccc9", "#b860bd", "#e3345d"))
```

---

plotImage *Plot JP(E)G/PNG/Raster/RGB images*

Description

This function takes in an image-related object - path to JP(E)G/PNG file, raster object, RGBarray. It returns a ggplot object with the selected image.

Arguments

x	A variety of objects can be passed: character string corresponding to an image file path, valid file types are JPG, JPEG and PNG. It can also take as input objects of class raster and RGB arrays. It can also take a SpatialExperiment from which the image will be extracted.
slice	Character string indicating which image slice to use when SpatialExperiment objects are passed. By default uses the first slice available.

Value

ggplot object

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
# Filename
path <- file.path(
  system.file(package = "SPOTlight"),
  "extdata/SPOTlight.png")
plotImage(x = path)
# array
png_img <- png::readPNG(path)
plotImage(png_img)
# SpatialExperiment
```

`plotInteractions` *Plot group interactions*

Description

This function takes in a matrix with the predicted proportions for each spot and returns a heatmap `which = plotHeatmap` or a network graph `which = plotNetwork` to show which cells are interacting spatially.

Usage

```
plotInteractions(
  x,
  which = c("heatmap", "network"),
  metric = c("prop", "jaccard"),
  min_prop = 0,
  ...
)
```

Arguments

`x`	numeric matrix with rows = samples and columns = groups. Must have at least one row and column, and at least two columns.
`which`	character string specifying the type of visualization: one of "heatmap" or "network".
`metric`	character string specifying which metric to show: one of "prop" or "jaccard".
`min_prop`	scalar specifying the value above which a group is considered to be contributing to a given sample. An interaction between groups i and j is counted for sample s only when both x[s, i] and x[s, j] fall above `min_prop`.
`...`	additional graphical parameters passed to `plot.igraph` when `which = "network"` (see `?igraph.plotting`).

Value

base R plot

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
library(ggplot2)
mat <- replicate(8, rnorm(100, runif(1, -1, 1)))
# Basic example
plotInteractions(mat)

### heatmap ###
# This returns a ggplot object that can be modified as such
plotInteractions(mat, which = "heatmap") +
    scale_fill_gradient(low = "#f2e552", high = "#850000") +
    labs(title = "Interaction heatmap", fill = "proportion")

### Network ###
# specify node names
nms <- letters[seq_len(ncol(mat))]
plotInteractions(mat, which = "network", vertex.label = nms)

# or set column names instead
colnames(mat) <- nms
plotInteractions(mat, which = "network")

# pass additional graphical parameters for aesthetics
plotInteractions(mat,
    which = "network",
    edge.color = "cyan",
    vertex.color = "pink",
    vertex.label.font = 2,
    vertex.label.color = "maroon")
```

---

plotSpatialScatterpie *Spatial scatterpie*

Description

This function takes in the coordinates of the spots and the proportions of the cell types within each spot. It returns a plot where each spot is a piechart showing proportions of the cell type composition.

Usage

```
plotSpatialScatterpie(
  x,
  y,
  cell_types = colnames(y),
  img = FALSE,
  slice = NULL,
```

```
  scatterpie_alpha = 1,
  pie_scale = 0.4,
  degrees = NULL,
  axis = NULL,
  ...
)
```

Arguments

x	Object containing the spots coordinates, it can be an object of class SpatialExperiment, dataframe or matrix. For the latter two rownames should have the spot barcodes to match x. If a matrix it has to of dimensions nrow(y) x 2 where the columns are the x and y coordinates in that order.
y	Matrix or dataframe containing the deconvoluted spots. rownames need to be the spot barcodes to match to x.
cell_types	Vector of cell type names to plot. By default uses the column names of y.
img	Logical TRUE or FALSE indicating whether to plot the image or not. Objects of classes accepted by `plotImage` can also be passed and that image will be used. By default FALSE.
slice	Character string indicating which slice to plot if img is TRUE. By default uses the first image.
scatterpie_alpha	Numeric scalar to set the alpha of the pie charts. By default 1.
pie_scale	Numeric scalar to set the size of the pie charts. By default 0.4.
degrees	From SpatialExperiment rotateImg. For clockwise (degrees > 0) and counterclockwise (degrees < 0) rotation. By default NULL.
axis	From SpatialExperiment mirrorImg. When a SpatialExperiment object is passed as the image return the mirror image. For horizontal (axis = "h") and vertical (axis = "v") mirroring. By default NULL.
...	additional parameters to geom_scatterpie

Value

`ggplot` object

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
set.seed(321)

# Coordinates
x <- replicate(2, rnorm(100))
rownames(x) <- paste0("spot", seq_len(nrow(x)))
colnames(x) <- c("imagecol", "imagerow")
```

```
# Proportions
y <- replicate(m <- 5, runif(nrow(x), 0, 1))
y <- prop.table(y, 1)

rownames(y) <- paste0("spot", seq_len(nrow(y)))
colnames(y) <- paste0("type", seq_len(ncol(y)))

(plt <- plotSpatialScatterpie(x = x, y = y))
```

`plotTopicProfiles` *Plot NMF topic profiles*

Description

This function takes in the fitted NMF model and returns the topic profiles learned for each cell `facet = FALSE` or cell type `facet = TRUE`. Ideal training will return all the cell from the same cell type to share a unique topic profile.

Usage

```
plotTopicProfiles(x, y, facet = FALSE, min_prop = 0.01, ncol = NULL)
```

Arguments

`x`	`list` object obtained from `SPOTlight`.
`y`	vector of group labels. Should be of length `ncol(res_lvl1$NMF$h)`.
`facet`	logical indicating whether to stratify by group. If `FALSE` (default), weights will be the median across cells for each group (point = topic weight for a given cell type). If `TRUE`, cell-specific weights will be shown (point = topic weight of a given cell).
`min_prop`	scalar in [0,1]. When `facet = TRUE`, only cells with a weight > `min_prop` will be included.
`ncol`	integer scalar specifying the number of facet columns.

Value

`ggplot` object

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
library(ggplot2)
x <- mockSC()
y <- mockSP(x)
z <- getMGS(x)

res <- SPOTlight(x, y,
    groups = x$type,
    mgs = z,
    group_id = "type",
    verbose = FALSE)

plotTopicProfiles(res[[3]], x$type, facet = TRUE)
plotTopicProfiles(res[[3]], x$type, facet = FALSE)
```

`runDeconvolution` *Run Deconvolution using NNLS model*

Description

This function takes in the mixture data, the trained model & the topic profiles and returns the proportion of each cell type within each mixture

Usage

```
runDeconvolution(
  x,
  mod,
  ref,
  scale = TRUE,
  min_prop = 0.01,
  verbose = TRUE,
  slot = "counts",
  L1_nnls_topics = 0,
  L2_nnls_topics = 0,
  L1_nnls_prop = 0,
  L2_nnls_prop = 0,
  threads = 0,
  ...
)
```

Arguments

`x`	mixture dataset. Can be a numeric matrix, `SingleCellExperiment` or `SpatialExperiment`
`mod`	object as obtained from trainNMF.
`ref`	object of class matrix containing the topic profiles for each cell type as obtained from trainNMF.

scale	logical specifying whether to scale single-cell counts to unit variance. This gives the user the option to normalize the data beforehand as you see fit (CPM, FPKM, ...) when passing a matrix or specifying the slot from where to extract the count data.
min_prop	scalar in [0,1] setting the minimum contribution expected from a cell type in x to observations in y. By default 0.
verbose	logical. Should information on progress be reported?
slot	If the object is of class SpatialExperiment indicates matrix to use. By default "counts".
L1_nnls_topics, L1_nnls_prop	LASSO penalty in the range (0, 1] for NNLS when computing cell type topic profiles and cell type proportions respectively. Larger values remove "noisy" contributions more aggressively.
L2_nnls_topics, L2_nnls_prop	RIDGE penalty >0 for NNLS when computing cell type topic profiles and cell type proportions respectively. Larger values remove "noisy" contributions more aggressively.
threads	number of threads to use, default 0 (all threads)
...	additional parameters.

Value

base a list where the first element is a list giving the NMF model and the second is a matrix containing the topic profiles learnt.

Author(s)

Marc Elosua Bayes, Zach DeBruine, and Helena L Crowell

Examples

```
set.seed(321)
# mock up some single-cell, mixture & marker data
sce <- mockSC(ng = 200, nc = 10, nt = 3)
spe <- mockSP(sce)
mgs <- getMGS(sce)

res <- trainNMF(
    x = sce,
    y = rownames(spe),
    groups = sce$type,
    mgs = mgs,
    weight_id = "weight",
    group_id = "type",
    gene_id = "gene")
# Run deconvolution
decon <- runDeconvolution(
    x = spe,
    mod = res[["mod"]],
    ref = res[["topic"]])
```

SPOTlight *Deconvolution of mixture using single-cell data*

Description

This is the backbone function which takes in single cell expression data to deconvolute spatial transcriptomics spots.

Usage

```
SPOTlight(
  x,
  y,
  groups = NULL,
  mgs,
  n_top = NULL,
  gene_id = "gene",
  group_id = "cluster",
  weight_id = "weight",
  hvg = NULL,
  scale = TRUE,
  min_prop = 0.01,
  verbose = TRUE,
  slot_sc = "counts",
  slot_sp = "counts",
  L1_nmf = 0,
  L2_nmf = 0,
  maxit = 100,
  threads = 0,
  tol = 1e-05,
  L1_nnls_topics = 0,
  L2_nnls_topics = 0,
  L1_nnls_prop = 0,
  L2_nnls_prop = 0,
  ...
)
```

Arguments

x, y	single-cell and mixture dataset, respectively. Can be a numeric matrix or `SingleCellExperiment`..
groups	character vector of group labels for cells in `x`. When `x` is a `SingleCellExperiment`., defaults to `colLabels(x)` and `Idents(x)`, respectively. Make sure groups is not a Factor.
mgs	`data.frame` or `DataFrame` of marker genes. Must contain columns holding gene identifiers, group labels and the weight (e.g., logFC, -log(p-value) a feature has in a given group.

`n_top` integer scalar specifying the number of markers to select per group. By default NULL uses all the marker genes to initialize the model.

`gene_id`, `group_id`, `weight_id`
character specifying the column in `mgs` containing gene identifiers, group labels and weights, respectively.

`hvg` character vector containing hvg to include in the model. By default NULL.

`scale` logical specifying whether to scale single-cell counts to unit variance. This gives the user the option to normalize the data beforehand as you see fit (CPM, FPKM, ...) when passing a matrix or specifying the slot from where to extract the count data.

`min_prop` scalar in [0,1] setting the minimum contribution expected from a cell type in `x` to observations in `y`. By default 0.

`verbose` logical. Should information on progress be reported?

`slot_sc`, `slot_sp`
If the object is of class `SingleCellExperiment` indicates matrix to use. By default "counts".

`L1_nmf` LASSO penalty in the range (0, 1] for NMF, larger values increase sparsity of each factor

`L2_nmf` RUDGE penalty >0 for NMF, larger values increase angle between factors and thus sparsity.

`maxit` maximum number of NMF iterations for fitting

`threads` number of threads to use, default 0 (all threads)

`tol` tolerance of the NMF model at convergence, the Pearson correlation distance between models across consecutive iterations (1e-5 is publication quality)

`L1_nnls_topics`, `L1_nnls_prop`
LASSO penalty in the range (0, 1] for NNLS when computing cell type topic profiles and cell type proportions respectively. Larger values remove "noisy" contributions more aggressively.

`L2_nnls_topics`, `L2_nnls_prop`
RIDGE penalty >0 for NNLS when computing cell type topic profiles and cell type proportions respectively. Larger values remove "noisy" contributions more aggressively.

`...` additional parameters.

Details

SPOTlight uses a Non-Negative Matrix Factorization approach to learn which genes are important for each cell type. In order to drive the factorization and give more importance to cell type marker genes we previously compute them and use them to initialize the basis matrix. This initialized matrices will then be used to carry out the factorization with the single cell expression data. Once the model has learn the topic profiles for each cell type we use non-negative least squares (NNLS) to obtain the topic contributions to each spot. Lastly, NNLS is again used to obtain the proportion of each cell type for each spot by finding the fitting the single-cell topic profiles to the spots topic contributions.

Value

a numeric matrix with rows corresponding to samples and columns to groups

Author(s)

Marc Elosua Bayes, Zach DeBruine, and Helena L Crowell

Examples

```
library(scater)
library(scran)

# Use Mock data
# Refer to the vignette for a full workflow
sce <- mockSC(ng = 200, nc = 10, nt = 3)
spe <- mockSP(sce)
mgs <- getMGS(sce)

res <- SPOTlight(
    x = counts(sce),
    y = counts(spe),
    groups = as.character(sce$type),
    mgs = mgs,
    hvg = NULL,
    weight_id = "weight",
    group_id = "type",
    gene_id = "gene")
```

`trainNMF` *train NMF model*

Description

This is the training function used by SPOTLight. This function takes in single cell expression data, trains the model and learns topic profiles for each cell type

Usage

```
trainNMF(
  x,
  y = NULL,
  groups = NULL,
  mgs,
  n_top = NULL,
  gene_id = "gene",
  group_id = "cluster",
  weight_id = "weight",
  hvg = NULL,
```

```
  scale = TRUE,
  verbose = TRUE,
  L1_nmf = 0,
  L2_nmf = 0,
  tol = 1e-05,
  maxit = 100,
  threads = 0,
  slot_sc = "counts",
  ...
)
```

Arguments

x	single-cell dataset. Can be a numeric matrix, Can be a numeric matrix or SingleCellExperiment.
y	Null if you want to train the model with all the genes in the SC data or a character vector with the rownames of the mixture dataset to subset the gene set used to the intersection between them.
groups	character vector of group labels for cells in x. When x is a SingleCellExperiment., defaults to colLabels(x) and Idents(x), respectively. Make sure groups is not a Factor.
mgs	data.frame or DataFrame of marker genes. Must contain columns holding gene identifiers, group labels and the weight (e.g., logFC, -log(p-value) a feature has in a given group.
n_top	integer scalar specifying the number of markers to select per group. By default NULL uses all the marker genes to initialize the model.
gene_id, group_id, weight_id	character specifying the column in mgs containing gene identifiers, group labels and weights, respectively.
hvg	character vector containing hvg to include in the model. By default NULL.
scale	logical specifying whether to scale single-cell counts to unit variance. This gives the user the option to normalize the data beforehand as you see fit (CPM, FPKM, ...) when passing a matrix or specifying the slot from where to extract the count data.
verbose	logical. Should information on progress be reported?
L1_nmf	LASSO penalty in the range (0, 1] for NMF, larger values increase sparsity of each factor
L2_nmf	RUDGE penalty >0 for NMF, larger values increase angle between factors and thus sparsity.
tol	tolerance of the NMF model at convergence, the Pearson correlation distance between models across consecutive iterations (1e-5 is publication quality)
maxit	maximum number of NMF iterations for fitting
threads	number of threads to use, default 0 (all threads)
slot_sc	If the object is of class SingleCellExperiment indicates matrix to use. By default "counts".
...	additional parameters.

Value

a list where the first element is a list with the NMF model information and the second is a matrix containing the topic profiles learnt per cell type.

Author(s)

Marc Elosua Bayes & Helena L Crowell

Examples

```
set.seed(321)
# mock up some single-cell, mixture & marker data
sce <- mockSC(ng = 200, nc = 10, nt = 3)
spe <- mockSP(sce)
mgs <- getMGS(sce)

res <- trainNMF(
    x = sce,
    y = rownames(spe),
    groups = sce$type,
    mgs = mgs,
    weight_id = "weight",
    group_id = "type",
    gene_id = "gene")
# Get NMF model
res[["mod"]]
# Get topic profiles
res[["topic"]]
```

Index