Package 'DenoIST'

November 11, 2026

Title DenoIST: Denoising Image-based Spatial Transcriptomics data

Version 1.0.0

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

Description DenoIST identifies and removes contamination in Image-based Spatial Transcriptomics data, using a transposed poisson mixture model with local neighbourhood offsets to infer genes that are likely to be due to neighbourhood contamination rather than endogenous expression.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/aaronkwc/DenoIST

BugReports https://github.com/aaronkwc/DenoIST/issues

Imports flexmix, hexbin, pbapply, sparseMatrixStats, SpatialExperiment, stats, SummarizedExperiment, parallel, Matrix, dbscan, methods

Suggests BiocStyle, knitr, rmarkdown, testthat, ggplot2, patchwork

VignetteBuilder knitr

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

git_branch RELEASE_3_23

git_last_commit 391d87b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Aaron Kwok [aut, cre] (ORCID: <https://orcid.org/0000-0001-7831-4198>), Heejung Shim [aut], Davis McCarthy [aut]

Maintainer Aaron Kwok <akwok@svi.edu.au>

Contents

denoist 2
local_offset_distance_with_background 4
local_offset_distance_with_background_fast 5
solve_poisson_mixture 7

Index **9**

`denoist` *DenoIST*

Description

DenoIST (Denoising Image-based Spatial Transcriptomics) is a method for identifying and removing contamination artefacts in image-based single-cell transcriptomics (IST) data. It uses a transposed Poisson mixture model to identify contamination.

Usage

```
denoist(
  mat,
  tx,
  coords = NULL,
  tx_x = "x",
  tx_y = "y",
  feature_label = "gene",
  distance = 50,
  nbins = 200,
  posterior_cutoff = 0.6,
  n_inits = 10,
  cl = 1,
  out_dir = NULL,
  neighbour_mode = "fast",
  verbose = FALSE
)
```

Arguments

`mat`	A matrix of counts (genes x cells), or a SpatialExperiment object.
`tx`	A data frame of transcript with x, y and qv columns.
`coords`	A data frame of coordinates (n_cells x 2). Only used if not using a SpatialExperiment object as input.
`tx_x`	Column name for the x coordinates in the transcripts dataframe. Default is 'x'.
`tx_y`	Column name for the y coordinates in the transcripts dataframe. Default is 'y'.
`feature_label`	Column name for the gene of each transcript in the transcripts dataframe. Default is 'gene'.

distance	The maximum distance to consider for local background estimation.
nbins	The number of bins to use for hexagonal binning.
posterior_cutoff	The cutoff for posterior probability to determine contamination.
n_inits	The number of initialisations for the mixing proportion in the Poisson mixture model. If input is a vector, directly use the vector of values as init values.
cl	The number of cores to use for parallel processing.
out_dir	The output directory to save the results.
neighbour_mode	The method to use for finding neighbors. Options are 'fast' (default) and 'original'. 'fast' uses a faster implementation with dbscan, while 'original' uses the original implementation.
verbose	Logical, if TRUE, print progress messages.

Details

The function calculates local background using hexagonal binning and applies a Poisson mixture model to identify contamination. It returns a matrix of memberships and adjusted counts for each gene in each cell.

Value

A list containing the following elements:

memberships	A matrix of memberships for each gene in each cell.
adjusted_counts	A matrix of adjusted counts for each gene in each cell.
params	A list of parameters for each gene.

Examples

```
# Load example data
set.seed(42)
mat <- matrix(rpois(1000, lambda = 10), nrow = 10, ncol = 100)
rownames(mat) <- paste0("gene", 1:10)
coords <- data.frame(x = rnorm(100), y = rnorm(100))
tx <- data.frame(x = c(rnorm(500), rnorm(500, 3)),
                 y = c(rnorm(500), rnorm(500, 3)),
                 qv = rep(30, 1000), gene = paste0('gene', 1:10))
# Run DenoIST
result <- denoist(mat, tx, coords, distance = 1, nbins = 50, cl = 1,
                  out_dir = NULL, verbose = TRUE)
# Check results
print(result$memberships[1:5, 1:5])
print(result$adjusted_counts[1:5, 1:5])
print(result$params[[1]])
```

`local_offset_distance_with_background` *Neighbourhood offset*

Description

This function calculates the local neighbourhood offset together with ambient background for each cell in a count matrix.

Usage

```
local_offset_distance_with_background(
  mat,
  tx,
  coords,
  tx_x = "x",
  tx_y = "y",
  feature_label = "gene",
  distance = 50,
  nbins = 200,
  cl = 1,
  verbose = FALSE
)
```

Arguments

`mat`	A count matrix with genes as rows and cells as columns.
`tx`	A transcript dataframe with x, y coordinates and qv values.
`coords`	A dataframe with x, y coordinates of each cell as separate columns.
`tx_x`	Column name for the x coordinates in the transcripts dataframe.
`tx_y`	Column name for the y coordinates in the transcripts dataframe.
`feature_label`	Column name for the gene of each transcript in the transcripts dataframe.
`distance`	The maximum distance to consider for local background estimation.
`nbins`	The number of bins to use for hexagonal binning, used for calculating background transcript contamination.
`cl`	The number of cores to use for parallel processing.
`verbose`	Logical, if TRUE, print progress messages.

Details

The function calculates the offset used for each cell based on their local neighbourhoods.In most cases you do not need to use this as denoist already runs this internally but it is good for debugging if needed.

Value

A matrix of local background counts for each gene in each cell.

Examples

```
# Load example data
set.seed(42)
mat <- matrix(rpois(1000, lambda = 10), nrow = 10, ncol = 100)
rownames(mat) <- paste0("gene", 1:10)
coords <- data.frame(x = rnorm(100), y = rnorm(100))
tx <- data.frame(x = c(rnorm(500), rnorm(500, 3)),
                 y = c(rnorm(500), rnorm(500, 3)),
                 qv = rep(30, 1000), gene = paste0('gene', 1:10))
# Run DenoIST
off_mat <- local_offset_distance_with_background(mat, tx, coords,
                                                 distance = 1, nbins = 50,
                                                 cl = 1, verbose = TRUE)
# Check results
print(off_mat[1:5, 1:5])
```

---

local_offset_distance_with_background_fast

Neighbourhood offset fast

Description

This function calculates the local neighbourhood offset together with ambient background for each cell in a count matrix. This is a faster version of the local_offset_distance_with_background function that uses dbscan for neighbor finding. Code credit to Sam Zimmerman with minor modifications.

Usage

```
local_offset_distance_with_background_fast(
  mat,
  tx,
  coords,
  tx_x = "x",
  tx_y = "y",
  feature_label = "gene",
  distance = 50,
  nbins = 200,
  cl = 1,
  verbose = FALSE
)
```

Arguments

mat	A count matrix with genes as rows and cells as columns.
tx	A transcript dataframe with x, y coordinates and qv values.
coords	A dataframe with x, y coordinates of each cell as separate columns.
tx_x	Column name for the x coordinates in the transcripts dataframe.
tx_y	Column name for the y coordinates in the transcripts dataframe.
feature_label	Column name for the gene of each transcript in the transcripts dataframe.
distance	The maximum distance to consider for local background estimation.
nbins	The number of bins to use for hexagonal binning, used for calculating background transcript contamination.
cl	The number of cores to use for parallel processing.
verbose	Logical, if TRUE, print progress messages.

Details

The function calculates the offset used for each cell based on their local neighbourhoods.In most cases you do not need to use this as denoist already runs this internally but it is good for debugging if needed. Usage is the same as the non-fast version, will replace the original function in denoist in the future if this one is stable and faster.

Value

A matrix of local background counts for each gene in each cell.

Examples

```
# Load example data
set.seed(42)
mat <- matrix(rpois(1000, lambda = 10), nrow = 10, ncol = 100)
rownames(mat) <- paste0("gene", 1:10)
coords <- data.frame(x = rnorm(100), y = rnorm(100))
tx <- data.frame(x = c(rnorm(500), rnorm(500, 3)),
                 y = c(rnorm(500), rnorm(500, 3)),
                 qv = rep(30, 1000), gene = paste0('gene', 1:10))
# Run DenoIST
off_mat <- local_offset_distance_with_background_fast(mat, tx, coords,
                                                      distance = 1, nbins = 50,
                                                      cl = 1, verbose = TRUE)
# Check results
print(off_mat[1:5, 1:5])
```

`solve_poisson_mixture` *Poisson mixture model solver*

Description

This function implements a 2-component Poisson mixture model using the EM algorithm. It estimates the parameters of the model and assigns memberships to each observation based on the posterior probabilities.

Usage

```
solve_poisson_mixture(
  x,
  s,
  max_iter = 5000,
  tol = 1e-06,
  n_inits = 10,
  posterior_cutoff = 0.6,
  verbose = FALSE
)
```

Arguments

`x`	A vector of counts.
`s`	A vector of offsets.
`max_iter`	Maximum number of iterations for the EM algorithm.
`tol`	Tolerance for convergence.
`n_inits`	Number of initial values for the mixing proportion inference. If input is a vector, directly use the vector has init values.
`posterior_cutoff`	Cutoff for posterior probabilities to assign memberships.
`verbose`	Logical, if TRUE, print progress messages.

Details

The function takes a vector of counts and a vector of offsets as input. It uses the EM algorithm to iteratively update the parameters of the model until convergence is reached or the maximum number of iterations is exceeded. The function also allows for multiple initialisations of the mixing proportion to find the best solution.

Value

A list containing the following elements:

Examples

```
x <- rpois(100, lambda = 5)
s <- runif(100, min = 0, max = 1)
result <- solve_poisson_mixture(x, s)
print(result)
```

Index

denoist, 2

local_offset_distance_with_background, 4
local_offset_distance_with_background_fast, 5

solve_poisson_mixture, 7