Package: lisaClust (via r-universe)

October 27, 2026

Type Package

Title lisaClust: Clustering of Local Indicators of Spatial Association

Version 1.21.13

Description lisaClust provides a series of functions to identify and visualise regions of tissue where spatial associations between cell-types is similar. This package can be used to provide a high-level summary of cell-type colocalization in multiplexed imaging data that has been segmented at a single-cell resolution.

License GPL (>=2)

Copyright The C++ port of concaveman in src/core: see inst/COPYRIGHTS.

biocViews SingleCell, CellBasedAssays, Spatial

Encoding UTF-8

Depends R (>= 4.1.0)

VignetteBuilder knitr

BugReports https://github.com/ellispatrick/lisaClust/issues

URL https://ellispatrick.github.io/lisaClust/, https://github.com/ellispatrick/lisaClust

Imports Rcpp, ggplot2, deldir, polyclip, grid, BiocParallel, RhpcBLASctl, spatstat.explore, spatstat.geom, BiocGenerics, S4Vectors, methods, stats, dplyr, SingleCellExperiment, SpatialExperiment, SummarizedExperiment, pheatmap, spatstat.random, lifecycle, rlang (>= 1.0.0)

Suggests SpatialDatasets, spicyR (>= 1.99.38), tidySpatialExperiment, BiocStyle, knitr, Matrix, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp

SystemRequirements C++17

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libmagick++-dev gsfonts libicu-dev libssl-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-10-27 02:36:47 UTC

RemoteUrl https://github.com/bioc/lisaClust

RemoteRef HEAD

RemoteSha a52849b521bde47ad5ffa6ea9cee1fc50dd9af45

Contents

hatchingPlot 2
inhomLocalK 5
lisa 6
lisaClust 8
nameRegions 10
regionBoxPlot 11
regionMap 12
scale_region 13

Index **15**

`hatchingPlot` *hatchingPlot*

Description

The hatchingPlot() function is used to create hatching patterns for representating spatial regions and cell-types.

The hatching geom is used to create hatching patterns for representation of spatial regions.

Usage

```
hatchingPlot(
  cells,
  useImages = NULL,
  region = "region",
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  window = "concave",
  line.spacing = 21,
  hatching.colour = 1,
  nbp = NULL,
  window.length = NULL
)

geom_hatching(
  mapping = NULL,
```

```
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  line.spacing = 21,
  hatching.colour = 1,
  window = "concave",
  window.length = NULL,
  nbp = NULL,
  line.width = 1,
  ...
)
```

Arguments

cells	A data.frame or SingleCellExperiment.
useImages	A vector of images to plot.
region	The region column to plot. Numeric region labels are treated as categories.
imageID	The imageIDs column if using data.frame or SingleCellExperiment.
cellType	The cellType column if using data.frame or SingleCellExperiment.
spatialCoords	The spatial coordinates columns if using data.frame or SingleCellExperiment.
window	Should the window around the regions be 'square', 'convex' or 'concave'.
line.spacing	A integer indicating the spacing between hatching lines.
hatching.colour	A colour for the hatching.
nbp	Not used: regions are outlined by the Voronoi tiles of their cells.
window.length	A tuning parameter for controlling the level of concavity when estimating concave windows.
mapping	Set of aesthetic mappings created by aes() or aes_(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer as a string.
position	adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
line.width	A numeric controlling the width of the hatching lines
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

Details

Each region is outlined by the union of the Voronoi tiles of its cells, clipped to the window, and its hatching is clipped to that outline. Up to 12 regions have their own hatching: none, /, \, -, |, x, +, dots, circles, / with dots, \ with dots and - with dots.

Value

A ggplot object

A ggplot geom

Examples

```
## Generate toy data
set.seed(51773)
x <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
    runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)
cells <- SingleCellExperiment::SingleCellExperiment(colData = cells)

## Generate regions
cells <- lisaClust(cells, k = 2)

## Plot regions
hatchingPlot(cells)

## Generate toy data
set.seed(51773)
library(ggplot2)
x <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
```

```
    runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)
## Generate regions
cells <- lisaClust(cells, k = 2)

# Plot the regions with geom_hatching()
ggplot(
    cells, aes(x = x, y = y, colour = cellType, region = region)
) +
    geom_point() +
    facet_wrap(~imageID) +
    geom_hatching()
```

`inhomLocalK` *Calculate the inhomogenous local K function.*

Description

Calculate the inhomogenous local K function.

Usage

```
inhomLocalK(
  data,
  Rs = c(20, 50, 100, 200),
  sigma = 10000,
  window = "convex",
  window.length = NULL,
  minLambda = 0.05,
  lisaFunc = "K",
  membership = NULL
)
```

Arguments

`data`	The data.
`Rs`	A vector of the radii that the measures of association should be calculated.
`sigma`	A numeric variable used for scaling when filting inhomogeneous L-curves.
`window`	Should the window around the regions be 'square', 'convex' or 'concave'.
`window.length`	A tuning parameter for controlling the level of concavity.

minLambda	Minimum value for density for scaling when fitting inhomogeneous L-curves.
lisaFunc	Either "K" or "L" curve.
membership	Used internally by [lisa()] with ‘cellWeights‘: the soft membership of the cells, already prepared. Leave as ‘NULL‘.

Value

A matrix of LISA curves

Examples

```
## Generate toy data: two images, two cell types that sit in separate bands
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)
cells$cellID <- paste0("cell_", seq_len(nrow(cells)))

# The curves of the first image
inhom <- inhomLocalK(cells[cells$imageID == "s1", ], Rs = c(10, 20, 50))
```

---

lisa *Generate local indicators of spatial association*

Description

Generate local indicators of spatial association

Usage

```
lisa(
  cells,
  r = NULL,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  cores = 1,
  window = "convex",
  window.length = NULL,
```

```
  whichParallel = "imageID",
  sigma = NULL,
  lisaFunc = "K",
  minLambda = 0.05,
  BPPARAM = NULL,
  Rs = r,
  cellWeights = NULL
)
```

Arguments

cells	A SingleCellExperiment, SpatialExperiment or data frame that contains at least the variables x and y, giving the coordinates of each cell, imageID and cellType.
r	A vector of the radii that the measures of association should be calculated.
imageID	The column which contains image identifiers.
cellType	The column which contains the cell types.
spatialCoords	The columns which contain the x and y spatial coordinates.
cores	Number of cores to use for parallel processing, or a BiocParallel param object (such as MulticoreParam, SnowParam or SerialParam).
window	Should the window around the regions be 'square', 'convex' or 'concave'.
window.length	A tuning parameter for controlling the level of concavity when estimating concave windows.
whichParallel	Should the function use parallization on the imageID or the cellType.
sigma	A numeric variable used for scaling when filting inhomogeneous L-curves.
lisaFunc	Either "K" or "L" curve.
minLambda	Minimum value for density for scaling when fitting inhomogeneous L-curves.
BPPARAM	{DEPRECATED} A BiocParallel param object.
Rs	{DEPRECATED} A vector of the radii that the measures of association should be calculated.
cellWeights	Optional cell weights in \[0, 1\], in the order of 'cells': a numeric vector with one weight per cell for its own cell type (for example a quality score), or a numeric matrix (or sparse 'Matrix') of soft cell type membership with one row per cell and one named column per label (for example a classifier's confidence). The curves are then one per label: each neighbour counts towards a label in proportion to its weight, and a label's density is its summed weight per unit area. Labels may overlap, for example a group and its subtypes. Weights of 1 give the curves of the cell types.

Value

A matrix of LISA curves, one row per cell in the order of 'cells' and one column per radius and cell type. The rows are named by the cellID column, or cell_1, cell_2, ... in the order of 'cells' if there is none.

Examples

```
## Generate toy data: two images, two cell types that sit in separate bands
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)

# Generate LISA curves
lisaCurves <- lisa(cells, r = c(10, 20, 50))

# Cluster the LISA curves; the rows are in the order of the cells
kM <- kmeans(lisaCurves, 2)
cells$region <- kM$cluster
```

`lisaClust` *Use k-means clustering to cluster local indicators of spatial association. For other clustering use lisa.*

Description

Use k-means clustering to cluster local indicators of spatial association. For other clustering use lisa.

Usage

```
lisaClust(
  cells,
  k = 2,
  r = NULL,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  regionName = "region",
  cores = 1,
  window = "convex",
  window.length = NULL,
  whichParallel = "imageID",
  sigma = NULL,
  lisaFunc = "K",
```

```
  minLambda = 0.05,
  BPPARAM = NULL,
  Rs = r,
  cellWeights = NULL
)
```

Arguments

cells	A SingleCellExperiment, SpatialExperiment or data frame that contains at least the variables x and y, giving the coordinates of each cell, imageID and cellType.
k	The number of regions to cluster.
r	A vector of the radii that the measures of association should be calculated.
imageID	The column which contains image identifiers.
cellType	The column which contains the cell types.
spatialCoords	The columns which contain the x and y spatial coordinates.
regionName	The output column for the lisaClust regions.
cores	Number of cores to use for parallel processing, or a BiocParallel param object (such as MulticoreParam, SnowParam or SerialParam).
window	Should the window around the regions be 'square', 'convex' or 'concave'.
window.length	A tuning parameter for controlling the level of concavity when estimating concave windows.
whichParallel	Should the function use parallization on the imageID or the cellType.
sigma	A numeric variable used for scaling when filting inhomogeneous L-curves.
lisaFunc	Either "K" or "L" curve.
minLambda	Minimum value for density for scaling when fitting inhomogeneous L-curves.
BPPARAM	{DEPRECATED} A BiocParallel param object.
Rs	A vector of the radii that the measures of association should be calculated.
cellWeights	Optional cell weights, a vector or a soft membership matrix in the order of 'cells'; see [lisa()].

Value

'cells', in the same order, with each cell's region in the column 'regionName'.

Examples

```
## Generate toy data: two images, two cell types that sit in separate bands
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
```

```
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)

# Cluster the cells into two regions
cells <- lisaClust(cells, k = 2)
table(cells$region, cells$cellType)
```

`nameRegions` *Name regions by the cell types they are enriched for*

Description

The regions found by `lisaClust()` are numbered arbitrarily. `nameRegions()` gives each region the name of the marker cell type it is most enriched for, as measured by `regionMap(type = "table")`. Regions most enriched for the same marker get the same name, so several regions can be merged into one domain.

Usage

```
nameRegions(
  cells,
  markers,
  region = "region",
  cellType = "cellType",
  regionName = region
)
```

Arguments

`cells`	A SingleCellExperiment, SpatialExperiment or data frame with a region column.
`markers`	A character vector of cell types. Its names, when given, are used as the names of the regions; otherwise the cell types are.
`region`	The column storing the regions.
`cellType`	The column storing the cell types.
`regionName`	The output column for the named regions; by default the regions are replaced.

Value

`cells` with the named regions in the column `regionName`.

Examples

```
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)
cells <- lisaClust(cells, k = 2)

cells <- nameRegions(cells, c(first = "c1", second = "c2"), regionName = "domain")
table(cells$domain, cells$cellType)
```

---

regionBoxPlot *Compare the share of each region between groups*

Description

For each image, or each patient when `imageID` names a patient column, the share of its cells in each region, drawn as box plots by `condition` with one panel per region. An image or patient with no cells in a region has a share of zero.

Usage

```
regionBoxPlot(
  cells,
  condition,
  region = "region",
  imageID = "imageID",
  subject = NULL,
  regions = NULL
)
```

Arguments

`cells`	A SingleCellExperiment, SpatialExperiment or data frame with a region column.
`condition`	The column storing the group of each image (or patient).
`region`	The column storing the regions.
`imageID`	The column of the images.

subject	Optional: the column of the patient. The shares are then those of each patient's cells, all images pooled, so each patient is one point.
regions	The regions to show; all by default.

Value

A ggplot object.

Examples

```
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID, group = rep(c("A", "B"), c(800, 800)))
cells <- lisaClust(cells, k = 2)

regionBoxPlot(cells, condition = "group")
```

---

regionMap *Plot heatmap of cell type enrichment for lisaClust regions*

Description

Plot heatmap of cell type enrichment for lisaClust regions

Usage

```
regionMap(
  cells,
  type = "bubble",
  cellType = "cellType",
  region = "region",
  limit = c(0.33, 3),
  ...
)
```

Arguments

cells	SingleCellExperiment, SpatialExperiment or data.frame
type	Make a "bubble" or "heatmap" plot, or return the relative frequencies as a "table".
cellType	The column storing the cell types
region	The column storing the regions
limit	limits to the lower and upper relative frequencies
...	Any arguments to be passed to the pheatmap package

Value

A bubble plot or heatmap, or with `type = "table"` a matrix of the relative frequencies with one row per cell type and one column per region: how much more often the cell type is found in the region than if cell types were spread evenly over the regions.

Examples

```
set.seed(51773)
x <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
  runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))

cells <- data.frame(x, y, cellType, imageID)

cells <- lisaClust(cells, k = 2)

regionMap(cells)
regionMap(cells, type = "table")
```

---

scale_region *Scale constructor for regions*

Description

Region scale constructor.

Usage

```
scale_region(aesthetics = "region", ..., guide = "legend")

scale_region_manual(..., values)
```

Arguments

aesthetics	The names of the aesthetics that this scale works with
...	Arguments passed on to discrete_scale
guide	A function used to create a guide or its name. See guides() for more info.
values	a set of aesthetic values to map data values to. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) with the limits of the scale. Any data values that don't match will be given na.value.

Value

a ggplot guide

Examples

```
## Generate toy data
set.seed(51773)
x <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
    runif(200) + 3, runif(200) + 2, runif(200) + 1, runif(200)
), 4) * 100
y <- round(c(
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3,
    runif(200), runif(200) + 1, runif(200) + 2, runif(200) + 3
), 4) * 100
cellType <- factor(paste("c", rep(rep(c(1:2), rep(200, 2)), 4), sep = ""))
imageID <- rep(c("s1", "s2"), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)
cells <- SingleCellExperiment::SingleCellExperiment(colData = cells)

## Generate regions
cells <- lisaClust(cells, k = 2)

# Plot the regions with hatchingPlot()
hatchingPlot(cells) +
    scale_region_manual(
        values = c(1, 4), labels = c("Region A", "Region B"),
        name = "Regions"
    )
```

Index

geom_hatching *(*hatchingPlot*)*, 2

hatchingPlot, 2

inhomLocalK, 5

lisa, 6
lisaClust, 8

nameRegions, 10

regionBoxPlot, 11
regionMap, 12

scale_region, 13
scale_region_manual *(*scale_region*)*, 13