Package ‘lisaClust’

May 17, 2024

**Type**  Package

**Title**  lisaClust: Clustering of Local Indicators of Spatial Association

**Version**  1.12.0

**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)

**biocViews**  SingleCell, CellBasedAssays, Spatial

**Encoding**  UTF-8

**Depends**  R (>= 4.0)

**VignetteBuilder**  knitr

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

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

**Imports**  ggplot2, class, concaveman, grid, BiocParallel,
spatstat.explore, spatstat.geom, BiocGenerics, S4Vectors,
methods, spicyR, purrr, stats, data.table, dplyr, tidyr,
SingleCellExperiment, SpatialExperiment, SummarizedExperiment,
pheatmap, spatstat.random

**Suggests**  BiocStyle, knitr, rmarkdown

**RoxygenNote**  7.2.3

**git_url**  https://git.bioconductor.org/packages/lisaClust

**git_branch**  RELEASE_3_19

**git_last_commit**  580e374

**git_last_commit_date**  2024-04-30

**Repository**  Bioconductor 3.19

**Date/Publication**  2024-05-16
**hatchingPlot**

**Description**

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

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

**Usage**

```r
hatchingPlot(
  data,
  useImages = NULL,
  region = "region",
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  window = "concave",
  line.spacing = 21,
  hatching.colour = 1,
  nbp = 50,
  window.length = NULL
)
```

```r
geom_hatching(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
```

**Contents**

- hatchingPlot .......................................................... 2
- inhomLocalK .......................................................... 5
- lisa ................................................................. 6
- lisaClust ............................................................ 7
- regionMap ............................................................ 9
- scale_region ........................................................ 10

**Index**

<table>
<thead>
<tr>
<th>hatchingPlot</th>
<th>hatchingPlot</th>
</tr>
</thead>
</table>

- hatchingPlot
- inhomLocalK
- lisa
- lisaClust
- regionMap
- scale_region

- hatchingPlot
- inhomLocalK
- lisa
- lisaClust
- regionMap
- scale_region

**Author** Ellis Patrick [aut, cre],
Nicolas Canete [aut]

**Maintainer** Ellis Patrick <ellis.patrick@sydney.edu.au>
hatchingPlot

show.legend = NA,
inherit.aes = TRUE,
line.spacing = 21,
hatching.colour = 1,
window = "concave",
window.length = NULL,
nbp = 250,
line.width = 1,
...

Arguments

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)).

useImages A vector of images to plot.
region The region column to plot.
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 An integer tuning the granularity of the grid used when defining regions
window.length A tuning parameter for controlling the level of concavity when estimating con-
cave 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.
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.
hatchingPlot

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.

Value

A ggplot object

A ggplot geom

Examples

```r
library(spicyR)
# 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)

# Store data in SegmentedCells object
cellExp <- SegmentedCells(cells, cellTypeString = 'cellType')

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

# Plot regions
hatchingPlot(cellExp)
```

```r
library(ggplot2)

# Extract the region information along with x-y coordinates
df <- as.data.frame(cellSummary(cellExp))

# Plot the regions with geom_hatching()
p <- ggplot(df,aes(x = x, y = y, colour = cellType, region = region)) +
geom_point() +
facet_wrap(~imageID) +
geom_hatching()
```
Calculate the inhomogenous local K function.

### Usage

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

### 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 fitting 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.

### Value

A matrix of LISA curves

### Examples

```r
library(spicyR)
# Read in data as a SegmentedCells objects
isletFile <- system.file("extdata", "isletCells.txt.gz", package = "spicyR")
cells <- read.table(isletFile, header = TRUE)
cells$x <- cells$AreaShape_Center_X
cells$y <- cells$AreaShape_Center_Y
cells$cellType <- as.factor(sample(c("big", "medium", "small"), length(cells$AreaShape_Center_Y), replace = TRUE))
```
cells$cellID <- as.factor(cells$ObjectNumber)

inhom <- inhomLocalK(cells[1:100,])

---

**lisa**

*Generate local indicators of spatial association*

**Description**

Generate local indicators of spatial association

**Usage**

```r
lisa(
  cells,
  Rs = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  window = "convex",
  window.length = NULL,
  whichParallel = "imageID",
  sigma = NULL,
  lisaFunc = "K",
  minLambda = 0.05,
  fast = TRUE,
  spatialCoords = c("x", "y"),
  cellType = "cellType",
  imageID = "imageID"
)
```

**Arguments**

- `cells` A SegmentedCells, SingleCellExperiment, SpatialExperiment or data frame that contains at least the variables x and y, giving the coordinates of each cell, imageID and cellType.
- `Rs` A vector of the radii that the measures of association should be calculated.
- `BPPARAM` A BiocParallelParam object.
- `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 parallelization on the imageID or the cellType.
- `sigma` A numeric variable used for scaling when fitting inhomogeneous L-curves.
- `lisaFunc` Either "K" or "L" curve.
- `minLambda` Minimum value for density for scaling when fitting inhomogeneous L-curves.
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,
Rs = NULL,
spatialCoords = c("x", "y"),
cellType = "cellType",
spatialCoords = c("x", "y"),
cellType = "cellType"
)

Value

A matrix of LISA curves

Examples

library(spicyR)
# Read in data as a SegmentedCells objects
isletFile <- system.file("extdata","isletCells.txt.gz", package = "spicyR")
cells <- read.table(isletFile, header=TRUE)
cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

# Cluster cell types
markers <- cellMarks(cellExp)
kM <- kmeans(markers,8)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')

# Generate LISA
lisaCurves <- lisa(cellExp)

# Cluster the LISA curves
kM <- kmeans(lisaCurves,2)
cellAnnotation(cellExp, "region") <- paste('region',kM$cluster,sep = '_')

fast A logical describing whether to use a fast approximation of the inhomogeneous local L-curves.
spatialCoords The columns which contain the x and y spatial coordinates.
cellType The column which contains the cell types.
imageID The column which contains image identifiers.
imageID = "imageID",
regionName = "region",
BPPARAM = BiocParallel::SerialParam(),
window = "convex",
window.length = NULL,
whichParallel = "imageID",
sigma = NULL,
lisaFunc = "K",
minLambda = 0.05,
fast = TRUE
)

Arguments

cells A SegmentedCells, 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.
Rs A vector of the radii that the measures of association should be calculated.
spatialCoords The columns which contain the x and y spatial coordinates.
cellType The column which contains the cell types.
imageID The column which contains image identifiers.
regionName The output column for the lisaClust regions.
BPPARAM A BiocParallelParam object.
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 parallelization on the imageID or the cellType.
sigma A numeric variable used for scaling when fitting inhomogeneous L-curves.
lisaFunc Either "K" or "L" curve.
minLambda Minimum value for density for scaling when fitting inhomogeneous L-curves.
fast A logical describing whether to use a fast approximation of the inhomogeneous local L-curves.

Value
A matrix of LISA curves

Examples
library(spicyR)
# Read in data as a SegmentedCells objects
isletFile <- system.file("extdata","isletCells.txt.gz", package = "spicyR")
cells <- read.table(isletFile, header=TRUE)
cellExp <- SegmentedCells(cells, cellProfiler = TRUE)
# Cluster cell types
markers <- cellMarks(cellExp)
kM <- kmeans(markers, 8)
cellType(cellExp) <- paste('cluster', kM$cluster, sep = '')

# Generate LISA
cellExp <- lisaClust(cellExp, k = 2)

---

### 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 SegmentedCells, SingleCellExperiment, SpatialExperiment or data.frame
type Make a "bubble" or "heatmap" plot.
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
Examples

```r
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)

cellExp <- spicyR::SegmentedCells(cells, cellTypeString = 'cellType')

regionMap(cellExp)
```

---

**scale_region**

*Scale constructor for regions*

**Description**

Region scale constructor.

**Usage**

```r
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

```r
library(spicyR)
## 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(c(1:2), rep(200, 2)), sep = ''))
imageID <- rep(c('s1', 's2'), c(800, 800))
cells <- data.frame(x, y, cellType, imageID)

## Store data in SegmentedCells object
cellExp <- SegmentedCells(cells)

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

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

gem_hatching (hatchingPlot), 2
hatchingPlot, 2
inhomLocalK, 5
lisa, 6
lisaClust, 7
regionMap, 9
scale_region, 10
scale_region_manual (scale_region), 10