Package 'lisaClust'

April 12, 2022

Type Package

Title lisaClust: Clustering of Local Indicators of Spatial Association

Version 1.2.0

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

Encoding UTF-8

Depends R (>= 4.1)

VignetteBuilder knitr

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

Imports ggplot2, class, concaveman, grid, BiocParallel, spatstat.core, spatstat.geom, BiocGenerics, S4Vectors, methods, spicyR, purrr, stats, data.table, dplyr, tidyr

Suggests BiocStyle, knitr, rmarkdown

RoxygenNote 7.1.1

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

git_branch RELEASE_3_14

git_last_commit 94e92c6

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

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

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

R topics documented:

---

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(
  data,
  imageID = NULL,
  window = "concave",
  line.spacing = 21,
  hatching.colour = 1,
  nbp = 250,
  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 = "square",
  window.length = 0,
  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)).
imageID	A vector of imageIDs to be plotted.
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 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.
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.

Value

A ggplot object

A ggplot geom

Examples

```
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 LISA
lisaCurves <- lisa(cellExp)

## Cluster regions
kM <- kmeans(lisaCurves,2)
region(cellExp) <- paste('region',kM$cluster,sep = '_')

## Plot regions
hatchingPlot(cellExp)



library(ggplot2)

# Extract the region information along with x-y coordinates
df <- region(cellExp, annot = TRUE)

# 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()
```

`lisa` *Generate local indicators of spatial association*

Description

Generate local indicators of spatial association

Usage

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

Arguments

cells	A SegmentedCells or data frame that contains at least the variables x and y, giving the coordinates of each cell, 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 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.
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
lisaCurves <- lisa(cellExp)

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

`Region accessors` *Region accessors for SegmentedCells*

Description

Methods to access various components of the 'SegmentedCells' object.

Usage

```
region(x, imageID = NULL, annot = FALSE)

region(x, imageID = NULL) <- value
```

Arguments

`x`	A 'SegmentedCells' object.
`imageID`	A vector of imageIDs to specifically extract.
`annot`	Add cell annotation when selecting region information.
`value`	The relevant information used to replace.

Value

DataFrame or a list of DataFrames

Examples

```
library(spicyR)
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 <- SegmentedCells(cells, cellProfiler = TRUE)
```

```
# Generate LISA
lisaCurves <- lisa(cellExp)

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

`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

```
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 LISA
lisaCurves <- lisa(cellExp)

## Cluster regions
kM <- kmeans(lisaCurves,2)
region(cellExp) <- paste('region',kM$cluster,sep = '_')

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

Index