Package 'spicyR'

January 23, 2021

Type Package

Title Spatial analysis of in situ cytometry data

Version 1.2.0

Description spicyR provides a series of functions to aid in the analysis of both immunofluorescence and mass cytometry imaging data as well as other assays that can deeply phenotype individual cells and their spatial location.

License GPL (>=2)

biocViews SingleCell, CellBasedAssays

Encoding UTF-8

Depends R (>= 4.0.0)

VignetteBuilder knitr

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

Imports ggplot2, class, concaveman, grid, BiocParallel, spatstat, lmerTest, BiocGenerics, S4Vectors, lme4, methods, mgcv, pheatmap, purrr, rlang, grDevices, IRanges, stats

Suggests BiocStyle, knitr, rmarkdown

RoxygenNote 7.1.1

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

git_branch RELEASE_3_12

git_last_commit 6af6c64

git_last_commit_date 2020-10-27

Date/Publication 2021-01-22

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

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

R topics documented:

`Accessors` *Accessors for SegmentedCells*

Description

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

Usage

```
cellSummary(x, imageID = NULL, bind = TRUE)

cellSummary(x, imageID = NULL) <- value

cellMarks(x, imageID = NULL, bind = TRUE)

cellMarks(x, imageID = NULL) <- value

cellMorph(x, imageID = NULL, bind = TRUE)

cellMorph(x, imageID = NULL) <- value

imagePheno(x, imageID = NULL, bind = TRUE, expand = FALSE)

imagePheno(x, imageID = NULL) <- value

imageID(x, imageID = NULL)

cellID(x, imageID = NULL)

cellID(x) <- value

imageCellID(x, imageID = NULL)

imageCellID(x) <- value

cellType(x, imageID = NULL)

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

```
filterCells(x, select)

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

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

Arguments

x	A 'SegmentedCells' object.
imageID	A vector of imageIDs to specifically extract.
bind	When false outputs a list of DataFrames split by imageID
expand	Used to expand the phenotype information from per image to per cell.
value	The relevant information used to replace.
select	A logical vector of the cells to be kept.
annot	Add cell annotation when selecting region information.

Value

DataFrame or a list of DataFrames

Descriptions

'cellSummary': Retrieves the DataFrame containing 'x' and 'y' coordinates of each cell as well as 'cellID', 'imageID' and 'cellType'. imageID can be used to select specific images and bind=FALSE outputs the information as a list split by imageID.

'cellMorph': Retrieves the DataFrame containing morphology information.

'cellMarks': Retrieves the DataFrame containing intensity of gene or protein markers.

'imagePheno': Retrieves the DataFrame containing the phenotype information for each image. Using expand = TRUE will produce a DataFrame with the number of rows equal to the number of cells.

Examples

```
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2,c(n/2,n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types
```

```
intensities <- cellMarks(cellExp)
kM <- kmeans(intensities,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')

cellSummary(cellExp, imageID = 1)
```

`as.data.frame.SegmentedCells` *as.data.frame*

Description

Function to coerce a SegmentedCells object to a data frame.

Usage

```
## S3 method for class 'SegmentedCells'
as.data.frame(x, ...)
```

Arguments

`x`	A SegmentedCells object.
`...`	Other arguments.

Value

A data.frame

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) 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) 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 cellsDF <- as.data.frame(cellExp)

NULL

`diabetesDataSub` *Diabetes IMC data*

Description

This is a subset of the Damond et al 2019 imaging mass cytometry dataset. The data contains cells in the pancreatic islets of individuals with early onset diabetes and healthy controls. The object contains single-cell data of 160 images from 8 subjects, with 20 images per subject.

Usage

```
diabetesDataSub
```

Format

diabetesDataSub a SegmentedCells object

`getPairwise` *Get statistic from pairwise L curve of a single image.*

Description

Get statistic from pairwise L curve of a single image.

Usage

```
getPairwise(cells, from, to, dist = NULL)
```

Arguments

`cells`	A SegmentedCells or data frame that contains at least the variables x and y, giving the location coordinates of each cell, and cellType.
`from`	The 'from' cellType for generating the L curve.
`to`	The 'to' cellType for generating the L curve.
`dist`	The distance at which the statistic is obtained.

Value

Statistic from pairwise L curve of a single image.

Examples

```
data("diabetesDataSub")
pairAssoc <- getPairwise(diabetesDataSub)
```

`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 = 0
)

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

```
## 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)
             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)
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 = "square",
  window.length = NULL,
  whichParallel = "imageID",
  sigma = NULL
)
```

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 fiting inhomogeneous L-curves

Value

A matrix of LISA curves

Examples

```
# 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 = '_')
```

`plot,SegmentedCells,ANY-method`
A basic plot for SegmentedCells object

Description

This function generates a basic x-y plot of the location coordinates and cellType data.

Usage

```
## S4 method for signature 'SegmentedCells,ANY'
plot(x, imageID = NULL)
```

Arguments

`x`	A SegmentedCells object.
`imageID`	The image that should be plotted.

Value

A ggplot object.

usage

‘plot(x, imageID = NULL)‘

Examples

```
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2,c(n/2,n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
```

```
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types
markers <- cellMarks(cellExp)
kM <- kmeans(markers,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')

#plot(cellExp, imageID=1)
```

`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

`SegmentedCells-class` *The SegmentedCells class*

Description

The SegmentedCells S4 class is for storing data from segmented imaging cytometry and spatial omics data. It extends DataFrame and defines methods that take advantage of DataFrame nesting to represent elements of cell-based experiments with spatial orientation that are commonly encountered. This object is able to store information on a cell's spatial location, cellType, morphology, intensity of gene/protein markers as well as image level phenotype information.

Usage

```
SegmentedCells(
  cellData,
  cellProfiler = FALSE,
  spatialCoords = c("x", "y"),
  cellTypeString = "cellType",
  intensityString = "intensity_",
  morphologyString = "morphology_",
  phenotypeString = "phenotype_",
  cellIDString = "cellID",
  imageCellIDString = "imageCellID",
  imageIDString = "imageID"
)
```

Arguments

cellData	A data frame that contains at least the columns x and y giving the location coordinates of each cell.
cellProfiler	A logical indicating that cellData is in a format similar to what cellProfiler outputs.
spatialCoords	The column names corresponding to spatial coordinates. eg. x, y, z...
cellTypeString	The name of the column that contains cell type calls.
intensityString	A string which can be used to identify the columns which contain marker intensities. (This needs to be extended to take the column names themselves.)
morphologyString	A string which can be used to identify the columns which contains morphology information.
phenotypeString	A string which can be used to identify the columns which contains phenotype information.
cellIDString	The column name for cellID.
imageCellIDString	The column name for imageCellID.
imageIDString	The column name for imageIDString.

Value

A SegmentedCells object

Examples

```
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(seq_len(2),c(n/2,n/2))
```

```
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)

### Cluster cell types
intensities <- cellMarks(cellExp)
kM <- kmeans(intensities,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')
cellSummary(cellExp)
```

`show-SegmentedCells` *Show SegmentedCells*

Description

This outputs critical information about aSegmentedCells.

Arguments

`object`	A SegmentedCells.

Value

Information of the number of images, cells, intenisties, morphologies and phenotypes.

usage

‘show(object)‘

Examples

```
### Something that resembles cellProfiler data

set.seed(51773)

n = 10

cells <- data.frame(row.names = seq_len(n))
cells$ObjectNumber <- seq_len(n)
cells$ImageNumber <- rep(1:2,c(n/2,n/2))
cells$AreaShape_Center_X <- runif(n)
cells$AreaShape_Center_Y <- runif(n)
cells$AreaShape_round <- rexp(n)
cells$AreaShape_diameter <- rexp(n, 2)
cells$Intensity_Mean_CD8 <- rexp(n, 10)
cells$Intensity_Mean_CD4 <- rexp(n, 10)

cellExp <- SegmentedCells(cells, cellProfiler = TRUE)
```

```
### Cluster cell types
markers <- cellMarks(cellExp)
kM <- kmeans(markers,2)
cellType(cellExp) <- paste('cluster',kM$cluster, sep = '')

cellExp
```

---

signifPlot *Plots result of signifPlot.*

Description

Plots result of signifPlot.

Usage

```
signifPlot(
  results,
  fdr = FALSE,
  breaks = c(-5, 5, 0.5),
  colors = c("blue", "white", "red"),
  marksToPlot = NULL
)
```

Arguments

results	Data frame obtained from spicy.
fdr	TRUE if FDR correction is used.
breaks	Vector of 3 numbers giving breaks used in pheatmap. The first number is the minimum, the second is the maximum, the third is the number of breaks.
colors	Vector of colours to use in pheatmap.
marksToPlot	Vector of marks to include in pheatmap.

Value

a pheatmap object

Examples

```
data(spicyTest)
signifPlot(spicyTest, breaks=c(-3, 3, 0.5))
```

SpicyResults-class *Performs spatial tests on spatial cytometry data.*

Description

Performs spatial tests on spatial cytometry data.

Usage

```
spicy(
  cells,
  condition = NULL,
  subject = NULL,
  covariates = NULL,
  from = NULL,
  to = NULL,
  dist = NULL,
  integrate = TRUE,
  nsim = NULL,
  verbose = TRUE,
  weights = TRUE,
  ...
)
```

Arguments

cells	A SegmentedCells or data frame that contains at least the variables x and y, giving the location coordinates of each cell, and cellType.
condition	Vector of conditions to be tested corresponding to each image if cells is a data frame.
subject	Vector of subject IDs corresponding to each image if cells is a data frame.
covariates	Vector of covariate names that should be included in the mixed effects model as fixed effects.
from	vector of cell types which you would like to compare to the to vector
to	vector of cell types which you would like to compare to the from vector
dist	The distance at which the statistic is obtained.
integrate	Should the statistic be the integral from 0 to dist, or the value of the L curve at dist.
nsim	Number of simulations to perform. If empty, the p-value from lmerTest is used.
verbose	logical indicating whether to output messages.
weights	logical indicating whether to include weights based on cell counts.
...	Other options to pass to bootstrap.

Value

Data frame of p-values.

Examples

```
data("diabetesDataSub")

# Test with random effect for patient on only one pairwise combination of cell types.
spicy(diabetesDataSub, condition = "condition", subject = "subject",
      from = "Tc", to = "Th")

# Test all pairwise combination of cell types without random effect of patient.
#spicyTest <- spicy(diabetesDataSub, condition = "condition")

# Test all pairwise combination of cell types with random effect of patient.
#spicy(diabetesDataSub, condition = "condition", subject = "subject")

# Test all pairwise combination of cell types with random effect of patient using
# a bootstrap to calculate significance.
#spicy(diabetesDataSub, condition = "condition", subject = "subject", nsim = 10000)
```

---

spicyTest *Results from spicy for diabetesDataSub*

Description

Results from the call: spicyTest <- spicy(diabetesDataSub, condition = "condition", subject = "subject")

Usage

```
spicyTest
```

Format

spicyTest a spicy object

---

spicyTestBootstrap *Results from spicy with bootstrap for diabetesDataSub*

Description

Results from the call: spicyTestBootstrap <- spicy(diabetesDataSub, condition = "condition", subject = "subject", nsim = 199)

Usage

```
spicyTestBootstrap
```

Format

spicyTestBootstrap a spicy object

topPairs *A table of the significant results from spicy tests*

Description

A table of the significant results from spicy tests

Usage

```
topPairs(x, coef = 1, n = 10, adj = "fdr", cutoff = NULL)
```

Arguments

x	The output from spicy.
coef	Which coefficient to list.
n	Extract the top n most significant pairs.
adj	Which p-value adjustment method to use, argument for p.adjust().
cutoff	A p-value threshold to extract significant pairs.

Value

A data.frame

Examples

```
data(spicyTest)
topPairs(spicyTest)
```

Index