Package 'spicyR'

November 28, 2021

Type Package

Title Spatial analysis of in situ cytometry data

Version 1.6.0

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 (>= 4.1)

VignetteBuilder knitr

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

Imports ggplot2, concaveman, BiocParallel, spatstat.core, spatstat.geom, lmerTest, BiocGenerics, S4Vectors, lme4, methods, mgcv, pheatmap, rlang, grDevices, IRanges, stats, data.table, dplyr, tidyr, scam

Suggests BiocStyle, knitr, rmarkdown

RoxygenNote 7.1.1

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

git_branch RELEASE_3_14

git_last_commit ef4f5a5

git_last_commit_date 2021-10-26

Date/Publication 2021-11-28

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)

cellAnnotation(x, variable, imageID = NULL)

cellAnnotation(x, variable, 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.
variable	A variable to add or retrieve from cellSummary.

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

diabetesData *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

```
diabetesData
```

Format

diabetesData 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 = unique(cellType(cells)),
  to = unique(cellType(cells)),
  dist = NULL,
  window = "convex",
  window.length = NULL,
  Rs = c(20, 50, 100),
  sigma = NULL,
  minLambda = 0.05,
  fast = TRUE,
  edgeCorrect = TRUE,
  BPPARAM = BiocParallel::SerialParam()
)
```

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.
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.
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.
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 L-curves.
edgeCorrect	A logical indicating whether to perform edge correction.
BPPARAM	A BiocParallelParam object.

Value

Statistic from pairwise L curve of a single image.

Examples

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

---

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)
```

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",
  cellAnnotations = NULL,
  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.
`cellAnnotations`	A vector of variables that provide additional annotation of a cell.
`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,
  weightsByPair = FALSE,
  weightFactor = 1,
  window = "convex",
  window.length = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  sigma = NULL,
  Rs = NULL,
  minLambda = 0.05,
  fast = TRUE,
  edgeCorrect = 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.
weightsByPair	logical indicating whether weights should be calculated for each cell type pair.
weightFactor	numeric that controls the convexity of the weight function.
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.
BPPARAM	A BiocParallelParam object.
sigma	A numeric variable used for scaling when fitting inhomogeneous L-curves.
Rs	A vector of the radii that the measures of association should be calculated.
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 L-curves.
edgeCorrect	A logical indicating whether to perform edge correction.
...	Other options to pass to bootstrap.

Value

Data frame of p-values.

Examples

```
data("diabetesData")

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

```
# Test all pairwise combination of cell types without random effect of patient.
#spicyTest <- spicy(diabetesData, condition = "stage", subject = "case")

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

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

`spicyTest` *Results from spicy for diabetesData*

Description

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

Usage

```
spicyTest
```

Format

spicyTest a spicy object

`spicyTestBootstrap` *Results from spicy with bootstrap for diabetesData*

Description

Results from the call: spicyTestBootstrap <- spicy(diabetesData, 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 = NULL, 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