Package: Statial (via r-universe)

November 2, 2026

Type Package

Title A package to identify changes in cell state relative to spatial associations

Version 1.14.0

Date 2025-10-14

VignetteBuilder knitr

Encoding UTF-8

biocViews SingleCell, Spatial, Classification

Depends R (>= 4.1.0)

Imports BiocParallel, spatstat.geom, concaveman, data.table, spatstat.explore, dplyr, tidyr, SingleCellExperiment, tibble, stringr, tidyselect, ggplot2, methods, stats, SummarizedExperiment, S4Vectors, plotly, purrr, ranger, magrittr, limma, SpatialExperiment, cluster, treekoR, edgeR

Suggests BiocStyle, knitr, testthat (>= 3.0.0), ClassifyR, spicyR, ggsurvfit, lisaClust, survival

Description Statial is a suite of functions for identifying changes in cell state. The functionality provided by Statial provides robust quantification of cell type localisation which are invariant to changes in tissue structure. In addition to this Statial uncovers changes in marker expression associated with varying levels of localisation. These features can be used to explore how the structure and function of different cell types may be altered by the agents they are surrounded with.

License GPL-3

RoxygenNote 7.3.3

Config/testthat/edition 3

URL https://sydneybiox.github.io/Statial
https://github.com/SydneyBioX/Statial/issues

BugReports https://github.com/SydneyBioX/Statial/issues

Config/pak/sysreqs
libabsl-dev libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglpk-dev make libmagick++-dev gsfonts libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl libproj-dev libsqlite3-dev libudunits2-dev libnode-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:59:24 UTC

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

RemoteRef RELEASE_3_23

RemoteSha f15705a083484b7981fc9f9070e6ca85e6f16304

Contents

calcContamination . . . 2
calcStateChanges . . . 3
distanceCalculator . . . 5
getAbundances . . . 5
getDistances . . . 6
getMarkerMeans . . . 7
getParentPhylo . . . 8
isKontextual . . . 9
kerenKontextual . . . 9
kerenSCE . . . 10
kontextCurve . . . 10
kontextPlot . . . 12
Kontextual . . . 12
makeWindow . . . 14
parentCombinations . . . 15
plotStateChanges . . . 16
prepMatrix . . . 17
relabelKontextual . . . 18

Index **21**

`calcContamination` *Calculate the level of marker contamination of each cell*

Description

Calculates contamination scores using a random forest classification

Usage

```
calcContamination(
  cells,
  markers = NULL,
  num.trees = 100,
  verbose = FALSE,
  missingReplacement = 0,
  assay = "intensities",
  cellType = "cellType",
  redDimName = "contaminations"
)
```

Arguments

cells	A SingleCellExperiment or SpatialExperiment with a cellType column as well as marker intensity information corresponding to each cell.
markers	A vector of markers that proxy a cell's state. If NULL, all markers will be used.
num.trees	Number of trees to be used in the random forest classifier
verbose	A logical indicating whether information about the final random forest model should be outputted.
missingReplacement	A default value to replace missing marker intensities for classification.
assay	The assay in the SingleCellExperiment object that contains the desired marker expressions.
cellType	The name of the column in colData that stores the cell types.
redDimName	The redDimName to store the output in the sce.

Examples

```
data("kerenSCE")

singleCellDataDistancesContam <- calcContamination(
  kerenSCE
)
```

---

calcStateChanges *First layer wrapper function to build linear models measuring state changes*

Description

Builds linear models measuring marker based state changes in a cell type based of the proximity or abundance of another cell type. The function provides the option to build robust and mixed linear model variants

Usage

```
calcStateChanges(
  cells,
  marker = NULL,
  from = NULL,
  to = NULL,
  image = NULL,
  type = "distances",
  assay = 1,
  cellType = "cellType",
  imageID = "imageID",
  contamination = NULL,
  test = "g",
  minCells = 20,
  verbose = FALSE,
  timeout = 10,
  nCores = 1
)
```

Arguments

cells	A dataframe with a imageID, cellType, and marker intensity column along with covariates (e.g. distance or abundance of the nearest cell type) to model cell state changes
marker	A vector of markers that proxy a cell's state. If NULL, all markers will be used.
from	A vector of cell types to use as the primary cells. If NULL, all cell types will be used.
to	A vector of cell types to use as the interacting cells. If NULL, all cell types will be used.
image	A vector of images to filter to. If null all images will be used.
type	What type of state change. This value should be in reduced dimensions.
assay	The assay in the SingleCellExperiment object that contains the marker expressions.
cellType	The column in colData that stores the cell types.
imageID	The column in colData that stores the image ids.
contamination	If TRUE, use the contamination scores that have previously been calculate. Otherwise a name of which reduced dimension contains the scores.
test	The type of test to perform. By default this will assume the data is Gaussian. A value of "nb" will use a negative binomial to model the expression.
minCells	The minimum number of cells required to fit a model.
verbose	A logical indicating if messages should be printed
timeout	A maximum time allowed to build each model. Setting this may be important when building rlm mixed linear models
nCores	Number of cores for parallel processing

Examples

```
library(dplyr)
data("kerenSCE")

kerenSCE <- kerenSCE[, kerenSCE$imageID %in% c(5, 6)]

kerenSCE <- getDistances(kerenSCE,
  maxDist = 200,
)

imageModels <- calcStateChanges(
  cells = kerenSCE,
  from = "Macrophages",
  to = "Tumour"
)
```

---

distanceCalculator *Calculate pairwise distance between cell types*

Description

Calculates the euclidean distance from each cell to the nearest cell of each type for a single image

Usage

```
distanceCalculator(data, maxDist = 200, distFun = "min")
```

Arguments

data	the single cell data of interest
maxDist	Maximum distance between pairs of points to be counted as close pairs.
distFun	How to merge duplicate entries.

---

getAbundances *Wrapper to calculate imhomogenous K function between a cell and surrounding types on each image*

Description

Calculate the imhomogenous K function (a measure of cell type abundance) for each cell to other cell types

Usage

```
getAbundances(
  cells,
  r = 200,
  distFun = "abundance",
  redDimName = "abundances",
  cellType = "cellType",
  imageID = "imageID",
  spatialCoords = c("x", "y"),
  nCores = 1
)
```

Arguments

cells	A dataframe with a cellType column as well as x and y spatial coordinates. The dataframe must contain a imageID column and cellID (unique cell identifier's) column as well
r	Radius to include in that calculation of pairwise abundance (K-function) between cells (can be a numeric or vector of radii)
distFun	What distance function to use.
redDimName	Name of the reduced dimension to store in sce.
cellType	The name of the column in colData that stores the cell types.
imageID	The name of the column in colData that Stores the image ids.
spatialCoords	The names of the columns in colData that store the spatial coordinates.
nCores	Number of cores for parallel processing

Examples

```
library(dplyr)
data("kerenSCE")

singleCellDataCounts <- getAbundances(kerenSCE,
  r = 200,
)
```

---

getDistances *Wrapper to calculate pairwise distance between cell types by image*

Description

Calculates the euclidean distance from each cell to the nearest cell of each type

Usage

```
getDistances(
  cells,
  maxDist = NULL,
  imageID = "imageID",
  spatialCoords = c("x", "y"),
  cellType = "cellType",
  redDimName = "distances",
  distFun = "min",
  nCores = 1
)
```

Arguments

cells	A dataframe with a cellType column as well as x and y spatial coordinates. The dataframe must contain a imageID column and cellID (unique cell identifier's) column as well
maxDist	The maximum distance considered.
imageID	The name of the colData column that stores in the image ID.
spatialCoords	The columns that store the spatial coordinates.
cellType	The name of the colData column that stores the cell types.
redDimName	The name of the reduced dimension to store the distances in.
distFun	What distance function to use. Can be min or abundance.
nCores	Number of cores for parallel processing.

Examples

```
data("kerenSCE")

kerenSCE <- getDistances(kerenSCE,
  maxDist = 200
)
```

---

getMarkerMeans *Extract the average expression for all markers for each cell type in each region defined by lisaClust*

Description

Takes a SingleCellExperiment and outputs a dataframe in a convenient format for cross validation

Usage

```
getMarkerMeans(
  data,
  imageID = NULL,
  cellType = NULL,
  region = NULL,
  markers = NULL,
  assay = 1,
  replaceVal = 0
)
```

Arguments

data	A SingleCellExperiment object with intensities data in the assays slot and regions information in colData generated by lisaClust.
imageID	The colData column that stores the image IDs.
cellType	The colData column that store the cell types.
region	The colData column that stores the regions.
markers	A string vector of markers that proxy a cell's state. If NULL, all markers will be used.
assay	Which assay do you want to use for the expression data.
replaceVal	A value to replace missing values with.

Examples

```
data(kerenSCE)

kerenSCE <- kerenSCE[, kerenSCE$imageID %in% c("5", "6")]

regionSCE <- lisaClust::lisaClust(kerenSCE, k = 5)

lisaClustOutput <- getMarkerMeans(regionSCE)
```

---

getParentPhylo *Extract parent and all children from a Phylo object*

Description

This function takes in a 'phylo' object or a 'treekoR' result from the `getClusterTree` function, and converts its into a named list of each and children to input into `parentCombinations`.

Note: Parent populations with one child will be pruned. Make sure to include this cell type in the 'all' vector when using `parentCombinations` to ensure this cell type is included in pairwise calculations.

Usage

```
getParentPhylo(phylo_tree)
```

Arguments

phylo_tree	a phylo object or a treekoR result.

Value

A named list of parents and their respective children.

---

isKontextual *Test whether an object is a kontextualResult*

Description

Test whether an object is a kontextualResult

Usage

```
isKontextual(kontextualResult)
```

Arguments

kontextualResult	a object to test

Examples

```
data <- data.frame()
if (!isKontextual(data)) print("Not a kontextualResult")
```

---

kerenKontextual *Kontextual results from kerenSCE*

Description

This is a kontextual results data.frame created using Kontextual on the kerenSCE dataset.

Usage

```
data(kerenKontextual)
```

Format

kerenKontextual a kontextual results object.

kerenSCE *MIBI-TOF Breast cancer intensities*

Description

This is a single MIBI-TOF data of breast cancer from patient 6 of the Keren et al 2018 dataset.

Usage

```
data(kerenSCE)
```

Format

kerenSCE a SingleCellExperiment object

References

Keren, L., Bosse, M., Marquez, D., Angoshtari, R., Jain, S., Varma, S., Yang, S. R., Kurian, A., Van Valen, D., West, R., Bendall, S. C., & Angelo, M. (2018). A Structured Tumor-Immune Microenvironment in Triple Negative Breast Cancer Revealed by Multiplexed Ion Beam Imaging. Cell, 174(6), 1373-1387.e1319. ([DOI](https://doi.org/10.1016/j.cell.2018.08.039))

kontextCurve *Evaluation of Kontextual over a range of radii.*

Description

This function obtains 'Kontextual' values over a range of radii, standard deviations for each value can be obtained using permutation for significance testing. To obtain estimates for standard deviations specify 'se = TRUE'.

Usage

```
kontextCurve(
  cells,
  from,
  to,
  parent,
  image = NULL,
  rs = seq(10, 100, 10),
  inhom = FALSE,
  edge = TRUE,
  se = FALSE,
  nSim = 20,
  cores = 1,
  imageID = "imageID",
```

```
  cellType = "cellType",
  ...
)
```

Arguments

cells	A single image from a SingleCellExperiment object
from	The first cell type to be evaluated in the pairwise relationship.
to	The second cell type to be evaluated in the pairwise relationship.
parent	The parent population of the from cell type (must include from cell type).
image	A vector of images to subset the results to. If NULL we default to all images.
rs	A vector of radii to evaluate kontextual over.
inhom	A logical value indicating whether to perform an inhomogeneous L function.
edge	A logical value indicating whether to perform edge correction.
se	A logical value to indicate if the standard deviation of kontextual should be calculated to construct error bars.
nSim	Number of randomisations to perform using relabelKontextual, which will be used to calculated the SE.
cores	Number of cores for parallel processing.
imageID	The column in colData that stores the image ids.
cellType	The column in colData that stores the cell types.
...	Any arguments passed into Kontextual.

Value

A data frame of original L values and Kontextual values evaluated over a range of radii.

Examples

```
data("kerenSCE")

kerenImage6 <- kerenSCE[, kerenSCE$imageID == "6"]

rsDf <- kontextCurve(
  cells = kerenSCE,
  from = "CD4_Cell",
  to = "Keratin_Tumour",
  parent = c("CD4_Cell", "Macrophages"),
  rs = seq(10, 510, 100),
  cores = 2
)
```

kontextPlot *Plotting the original and kontextual L values over a range of radii.*

Description

This function takes outputs from rsCurve and plots them in ggplot. If standard deviation is estimated in rsCurve, then confidence intervals will be constructed based on the standard deviation. If the confidence interval overlaps with 0, then the relationship is insignificant for that radius.

Usage

```
kontextPlot(rsDf)
```

Arguments

rsDf	A data frame from kontextCurve.

Value

A ggplotly object showing the original and kontextual L function values over a range of radii

Examples

```
data("kerenSCE")

kerenImage6 <- kerenSCE[, kerenSCE$imageID == "6"]

rsDf <- kontextCurve(
  cells = kerenImage6,
  from = "p53",
  to = "Immune",
  parent = c("p53", "Keratin+Tumour"),
  rs = seq(10, 510, 100),
  cores = 2
)

kontextPlot(rsDf)
```

Kontextual *Evaluation of pairwise cell relationships, conditional on a 3rd population.*

Description

Kontextual identifies the relationship between two cell types which are conditional on the spatial behaviour of a 3rd cell population, for a particular radius (r).

Usage

```
Kontextual(
  cells,
  r,
  parentDf = NULL,
  from = NULL,
  to = NULL,
  parent = NULL,
  image = NULL,
  inhom = FALSE,
  edgeCorrect = TRUE,
  window = "convex",
  window.length = NA,
  includeOriginal = TRUE,
  spatialCoords = c("x", "y"),
  cellType = "cellType",
  imageID = "imageID",
  cores = 1
)
```

Arguments

cells	A SingleCellExperiment, SpatialExperiment or a list of data.frames containing columns specifying the imageID, cellType, and x and y spatial coordinates.
r	Radii to evaluated pairwise relationships between from and to cells.
parentDf	A data frame from parentCombinations
from	The first cell type to be evaluated in the pairwise relationship.
to	The second cell type to be evaluated in the pairwise relationship.
parent	The parent population of the from cell type (must include from cell type).
image	A vector of images to subset the results to. If NULL we default to all images.
inhom	A logical value indicating whether to account for inhomogeneity.
edgeCorrect	A logical value indicating whether to perform edge correction.
window	Type of window for data, either 'square', 'convex' or 'concave', passed into makeWindow
window.length	A tuning parameter for controlling the level of concavity when estimating concave windows. Passed into makeWindow
includeOriginal	A logical value to return the original L function values along with the kontextual values.
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.
cores	Number of cores for parallel processing.

Value

A kontextualResult object

Examples

```
# Load data
data("kerenSCE")


CD4_Kontextual <- Kontextual(
  cells = kerenSCE,
  r = 50,
  from = "Macrophages",
  to = "Keratin_Tumour",
  parent = c("Macrophages", "CD4_Cell"),
  image = "6"
)


head(CD4_Kontextual)
```

---

makeWindow *Creates a window for a PPP object*

Description

This function creates a window for a 'spatstat::ppp' object, the type of window can be specified using the 'window' argument.

Usage

```
makeWindow(data, window = "square", window.length = NULL)
```

Arguments

data	A single image data frame from a SingleCellExperiment object or PPP object.
window	The shape of window around the regions, can be 'square', 'convex' or 'concave'
window.length	A tuning parameter for controlling the level of concavity when estimating concave windows.

Value

Creates an 'owin' class, representing the observation window for the image.

Examples

```
data <- data.frame(x = rnorm(10), y = rnorm(10))
ow <- makeWindow(data, window = "square")

spatstat.geom::ppp(x = data$x, y = data$y, window = ow)
```

`parentCombinations` *Create all combinations of cell type relationships from a list of parents*

Description

This function takes in named vectors of all the parent populations in the dataset, and creates a data frame containing all pairwise cell relationships, this data frame can be inputed into the 'parentDf' argument in 'Kontextual'.

Usage

```
parentCombinations(all, ..., parentList = NULL)
```

Arguments

`all`	A list of all the 'to' cell types Kontextual is evaluated over.
`...`	Vectors of each parent population.
`parentList`	a named list where the names correspond to parent names and the values contain a vector of children for that parent. Note: If parentList is specified the '...' argument will be ignored, see examples.

Value

A data frame containing all pairwise cell relationships and their corresponding parent

Examples

```
# Example 1, using `parentList`

parentList <- list(
  "tcells" = c("CD4", "CD8"),
  "tissue" = c("epithelial", "stromal")
)

allCells <- c("tumour", "CD4", "CD8", "epithelial", "stromal")

parentCombinations(all = allCells, parentList = parentList)


# Example 2, with `...` operator
tcells <- c("CD4", "CD8")
```

```
tissue <- c("epithelial", "stromal")
allCells <- c("tumour", tissue, tcells)

parentCombinations(all = allCells, tcells, tissue)
```

---

plotStateChanges *Visualise Cell-Cell Marker Relationships*

Description

Helper functions to visualise OLS model fits for image based state models

Usage

```
plotStateChanges(
  cells,
  image,
  from,
  to,
  marker,
  type = "distances",
  assay = 1,
  cellType = "cellType",
  imageID = "imageID",
  spatialCoords = c("x", "y"),
  size = 1,
  shape = 19,
  interactive = FALSE,
  plotModelFit = FALSE,
  method = "lm"
)
```

Arguments

cells	A SingleCellExperiment that has had distances already calculated.
image	An image to subset to.
from	A character indicating the name of the cell type (from the cellType column) whose cell state is being investigated in
to	A character indicating the name of the cell type (from the cellType column) who may be influencing the cell state of another cell type
marker	The marker of interest.
type	The name of the reduced dimension to use for the x-axis.
assay	Name of the assay that stores the marker expression.
cellType	The name of the column in colData that stores the cell types.

imageID	The name of the column in colData that stores the image ids.
spatialCoords	The names of the columns in colData that store the spatial coordinates.
size	Aesthetic numerical variable determining the size of the displayed cells
shape	Aesthetic variable determining the shape grouping of the displayed cells
interactive	Logical indicating if the output visualisation should be a interactive (plotly)
plotModelFit	Logical indicating if fitted values should be plotted or actual intensities for marker specified. The default is to plot actual intensities
method	The method to build the model with. Currently the only option is "lm". However, capabilities may be expanded in the future

Details

image,

Examples

```
library(dplyr)
data("kerenSCE")

kerenSCE <- getDistances(kerenSCE)

p <- plotStateChanges(
  cells = kerenSCE,
  type = "distances",
  image = "6",
  from = "Keratin_Tumour",
  to = "Macrophages",
  marker = "p53",
  size = 1,
  shape = 19,
  interactive = FALSE,
  plotModelFit = FALSE,
  method = "lm"
)

p
```

---

prepMatrix *Convert Kontextual or state changes result to a matrix for classification*

Description

Convert Kontextual or state changes result to a matrix for classification

Usage

```
prepMatrix(result, replaceVal = 0, column = NULL, test = NULL)
```

Arguments

result	a kontextual or state changes result data.frame.
replaceVal	value which NAs are replaced with.
column	The column which contains the scores that you want to select.
test	A column containing which will be the column names of the expanded matrix.

Examples

```
data("kerenSCE")


CD4_Kontextual <- Kontextual(
  cells = kerenSCE,
  r = 50,
  from = "Macrophages",
  to = "Keratin_Tumour",
  parent = c("Macrophages", "CD4_Cell"),
  image = "6"
)


kontextMat <- prepMatrix(CD4_Kontextual)
```

---

relabelKontextual *Cell permutation for Kontextual*

Description

Function which randomises specified cells in an image and calculates the 'Kontextual' value. This can be used to estimate the null distribution, of the parent cell population for significance testing.

This function relabels all specified cells within a single image, to estimate the null distribution of cell population specified.

Usage

```
relabelKontextual(
  cells,
  nSim = 1,
  r,
  from,
  to,
  parent,
```

```
  image = NULL,
  returnImages = FALSE,
  inhom = TRUE,
  edge = FALSE,
  cores = 1,
  spatialCoords = c("x", "y"),
  cellType = "cellType",
  imageID = "imageID",
  ...
)

relabel(image, labels = NULL)
```

Arguments

cells	A single image data frame from a SingleCellExperiment object
nSim	Number of randomisations which will be calculated.
r	Radius to evaluated pairwise relationships between from and to cells.
from	The first cell type to be evaluated in the pairwise relationship.
to	The second cell type to be evaluated in the pairwise relationship.
parent	The parent population of the from cell type (must include from cell type).
image	A single image from a Single Cell Experiment object.
returnImages	A logical value to indicate whether the function should return the randomised images along with the Kontextual values.
inhom	A logical value indicating whether to account for inhomogeneity.
edge	A logical value indicating whether to perform edge correction.
cores	Number of cores for parallel processing.
spatialCoords	A character vector containing the names of the two spatial dimansions in the data. Defaults to ‘c("x", "y")‘.
cellType	The name of the cell type field in the data. Defualts to "cellType".
imageID	The name of the image ID field in the data. Defualts to "imageID".
...	Any arguments passed into Kontextual
labels	A vector of CellTypes labels to be permuted If NULL all cells labels will be radomised.

Value

A data frame containing Kontextual value for each randomised image. If ‘returnImages = TRUE‘ function will return a list with Kontextual values and the randomised images.

A data frame containing all pairwise cell relationships and their corresponding parent

Examples

```
data("kerenSCE")

kerenImage6 <- kerenSCE[, kerenSCE$imageID == "6"]

relabelResult <- relabelKontextual(
  cells = kerenImage6,
  nSim = 5,
  r = 250,
  from = "CD4_Cell",
  to = "Keratin_Tumour",
  parent = c("CD4_Cell", "Macrophages"),
  cores = 2
)

data("kerenSCE")

kerenImage6 <- kerenSCE[, kerenSCE$imageID == "6"]

kerenImage6 <- kerenImage6 |>
  SingleCellExperiment::colData() |>
  data.frame()

# Permute CD8 T cells and T cell labels in the image
relabeledImage <- relabel(kerenImage6, labels = c("p53", "Keratin+Tumour"))
plot(relabeledImage)
```

Index

∗ **datasets**
kerenKontextual, 9
kerenSCE, 10

calcContamination, 2
calcStateChanges, 3

distanceCalculator, 5

getAbundances, 5
getClusterTree, *8*
getDistances, 6
getMarkerMeans, 7
getParentPhylo, 8

isKontextual, 9

kerenKontextual, 9
kerenSCE, 10
kontextCurve, 10, *12*
kontextPlot, 12
Kontextual, *11*, 12, *19*

makeWindow, *13*, 14

parentCombinations, *8*, *13*, 15
plotStateChanges, 16
prepMatrix, 17

relabel *(*relabelKontextual*)*, 18
relabelKontextual, *11*, 18