

Package: spicityR (via r-universe)

October 16, 2026

Type Package

Title Test Changes in the Co-Localisation of Cell Types Between Groups of Patients

Version 1.99.24

Description spicityR tests whether the spatial co-localisation of pairs of cell types differs between groups of patients, or is associated with patient outcome, in cell-resolution spatial omics data (imaging mass cytometry, CODEX, MIBI, Xenium, CosMx and others). Its default method, spicityR Cell, counts the cells of one type within a radius of each cell of another and compares the count with its exact expectation under random labelling of the observed cells. Images are combined within patients by a frailty generalised estimating equation, and differences are tested with a small-sample cluster-robust (CR2) variance with patients as the units. Covariates, several conditions, several radii and survival outcomes are supported, and the original image-level test is kept as an option. The computations run in a C++ core shared with the Python package spicityr.

License GPL (>=2)

Copyright The C++ port of concaveman in src/concaveman.cpp: see inst/COPYRIGHTS.

LazyData false

biocViews SingleCell, CellBasedAssays, Spatial, Survival

Encoding UTF-8

Depends R (>= 4.5.0)

VignetteBuilder knitr

BugReports <https://github.com/SydneyBioX/spicityR/issues>

URL <https://sydneybioX.github.io/spicityR/>,
<https://github.com/SydneyBioX/spicityR>

Imports Rcpp, methods, stats, utils, grDevices, grid, splines, BiocParallel, RhpcBLASctl, ggplot2, S4Vectors, survival

LinkingTo Rcpp, RcppEigen

SystemRequirements C++17

Suggests SummarizedExperiment, plotly, dplyr, SingleCellExperiment,
SpatialExperiment, lmerTest, lme4, Rarr, spatialdataR, scam,
coxme, spatstat.geom, spatstat.explore, SpatialDatasets,
BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Roxygen list(markdown = TRUE)

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-10-16 03:37:08 UTC

RemoteUrl <https://github.com/bioc/spicyR>

RemoteRef HEAD

RemoteSha 9c6becb2ca04ce8627648066c619346ceec02616e

Contents

bind	3
buildKnnGraph	3
colTest	4
convPairs	5
diabetesData	6
getPairwise	7
getPairwiseProp	9
getProp	11
imageCrossPlot	11
plotImage	13
signifPlot	14
spicy	15
spicyBoxPlot	18
SpicyResults-class	20
spicyTest	21
topPairs	21
writeCellStore	22

Index	24
--------------	-----------

bind	<i>The per-image values of every pair, as a data frame</i>
------	--

Description

Returns, for each image, its condition and the value of each pair that `spicy()` tested: for the cell method its effect (by default the extra fraction of to cells with a from cell within `r`; with `effect = "count"`, the extra from cells per to cell), or the L-function summary for the image method. Use it for your own plots or models.

Usage

```
bind(results, pairName = NULL)
```

Arguments

<code>results</code>	The <code>SpicyResults</code> object returned by <code>spicy()</code> .
<code>pairName</code>	A pair, as "from__to". If <code>NULL</code> , all pairs are returned.

Value

A `data.frame` with one row per image: `imageID`, `condition`, `subject` (if given) and one column per pair, named "from__to". NA where the pair could not be measured in an image.

Examples

```
data(spicyTest)
df <- bind(spicyTest)
```

buildKnnGraph	<i>Build a k-nearest-neighbour graph between cells</i>
---------------	--

Description

Finds the `k` nearest neighbours of every cell within its own image and stores them as a `colPair` of cells, ready for `convPairs()`. Distances are Euclidean, a cell is not its own neighbour, and ties are broken as in `spatstat.geom::nnwhich()`. The graph is the one `imcRtools::buildSpatialGraph(type = "knn")` builds, computed by the C++ core that `spicy()` uses for `k`.

Usage

```
buildKnnGraph(
  cells,
  k = 20,
  imageID = "imageID",
  spatialCoords = c("x", "y"),
  name = "knn_interaction_graph",
  cores = 1
)
```

Arguments

cells	A SingleCellExperiment or SpatialExperiment.
k	The number of neighbours of each cell. Cells in images with at most k cells get no neighbours.
imageID	The colData column holding the image of each cell.
spatialCoords	The colData columns holding the coordinates. A SpatialExperiment's own spatialCoords() are used unless these name colData columns.
name	The name of the colPair the graph is stored in.
cores	The number of threads; images are processed in parallel.

Value

cells, with the graph in colPair(cells, name): one edge from each cell to each of its neighbours.

Examples

```
data("diabetesData")
diabetesData <- buildKnnGraph(diabetesData, k = 10)
head(SingleCellExperiment::colPair(diabetesData, "knn_interaction_graph"))
```

colTest	<i>Perform a simple wilcoxon-rank-sum test or t-test on the columns of a data frame</i>
---------	---

Description

Perform a simple wilcoxon-rank-sum test or t-test on the columns of a data frame

Usage

```
colTest(df, condition, type = NULL, feature = NULL, imageID = "imageID")
```

Arguments

df	A data.frame or SingleCellExperiment, SpatialExperiment
condition	The condition of interest
type	The type of test, "wilcox", "ttest" or "survival".
feature	Can be used to calculate the proportions of this feature for each image
imageID	The imageID's if presenting a SingleCellExperiment

Value

A data.frame with one row per column of df (for example per cell type), ordered by p-value: the mean in each condition, the test statistic, the p-value (pval), the Benjamini-Hochberg adjusted p-value (adjPval) and the column name (cluster).

Examples

```
# Test for a difference in cell-type proportions between onset and long-duration diabetes
# (images are treated as independent here, which ignores that each donor has several)
data("diabetesData")
props <- getProp(diabetesData)
images <- unique(as.data.frame(SummarizedExperiment::colData(diabetesData))[, c("imageID", "stage")])
condition <- setNames(as.character(images$stage), images$imageID)
condition <- condition[condition %in% c("Long-duration", "Onset")]
test <- colTest(props[names(condition), ], condition)
head(test)
```

convPairs	<i>Converts colPairs object into an abundance matrix based on number of nearby interactions for every cell type.</i>
-----------	--

Description

Converts colPairs object into an abundance matrix based on number of nearby interactions for every cell type.

Usage

```
convPairs(cells, colPair, imageID = "imageID", cellType = "cellType")
```

Arguments

cells	A SingleCellExperiment that contains objects in the colPairs slot.
colPair	The name of the object in the colPairs slot for which the dataframe is constructed from.
imageID	The image ID if using SingleCellExperiment.
cellType	The cell type if using SingleCellExperiment.

Value

Matrix of abundances

Examples

```
data("diabetesData")
images <- c("A09", "A11", "A16", "A17")
diabetesData <- diabetesData[
  , SummarizedExperiment::colData(diabetesData)$imageID %in% images
]

diabetesData <- buildKnnGraph(diabetesData, k = 20)

pairAbundances <- convPairs(diabetesData,
  colPair = "knn_interaction_graph"
)
```

diabetesData	<i>Imaging mass cytometry of human pancreas in type 1 diabetes (Damond et al. 2019)</i>
--------------	---

Description

Cell types and positions from imaging mass cytometry of pancreas sections from 12 donors at three stages of type 1 diabetes (Damond et al. 2019): 4 non-diabetic donors, 4 at onset and 4 with long-duration disease. The object holds 253,777 cells from 120 images, 10 images per donor. It has no assays: only the cell data used by spicyR.

Usage

```
data("diabetesData")
```

Format

A SingleCellExperiment with one column per cell and these colData columns:

imageID the image (character)
cellID, imageCellID cell identifiers, in the whole data set and within its image
x, y the cell's coordinates in the image, in micrometres
cellType the cell type assigned by the authors (factor)
case the donor (integer)
slide, part the slide, and the part of the pancreas (head, body or tail)
group, stage the stage of type 1 diabetes, as a code and as a factor with levels "Non-diabetic", "Onset" and "Long-duration"

Source

Damond N et al. (2019). Mendeley Data, [doi:10.17632/cydmwsfztj.1](https://doi.org/10.17632/cydmwsfztj.1), under the CC BY 4.0 licence. How the subset was made is described in `inst/scripts/make-diabetesData.R`.

References

Damond N, Engler S, Zanutelli VRT, et al. (2019). A map of human type 1 diabetes progression by imaging mass cytometry. *Cell Metabolism* 29(3), 755-768. [doi:10.1016/j.cmet.2018.11.014](https://doi.org/10.1016/j.cmet.2018.11.014)

Examples

```
data("diabetesData")
table(unique(as.data.frame(SummarizedExperiment::colData(diabetesData))[, c("case", "stage")])$stage)
```

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,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  r = NULL,
  sigma = NULL,
  from = NULL,
  to = NULL,
  cores = 1,
  minLambda = 0.05,
  window = "convex",
  window.length = NULL,
  edgeCorrect = TRUE,
  includeZeroCells = FALSE,
  BPPARAM = NULL,
  imageIDCol = imageID,
  cellTypeCol = cellType,
  spatialCoordCols = spatialCoords,
  nCores = cores,
  Rs = r
)
```

Arguments

cells	A SummarizedExperiment that contains at least the variables x and y, giving the location coordinates of each cell, and cellType.
imageID	The name of the imageID column if using a SingleCellExperiment or SpatialExperiment.
cellType	The name of the cellType column if using a SingleCellExperiment or SpatialExperiment.
spatialCoords	The names of the spatialCoords column if using a SingleCellExperiment.
r	A vector of the radii that the measures of association should be calculated over.
sigma	A numeric variable used for scaling when fitting inhomogenous L-curves.
from	The 'from' cellType for generating the L curve.
to	The 'to' cellType for generating the L curve.
cores	Number of cores to use for parallel processing or a BiocParallel MulticoreParam or SerialParam object.
minLambda	Minimum value density 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 when estimating concave windows.
edgeCorrect	A logical indicating whether to perform edge correction.
includeZeroCells	A logical indicating whether to include cells with zero counts in the pairwise association.
BPPARAM	{DEPRECATED} A BiocParallel MulticoreParam or SerialParam object.
imageIDCol	{DEPRECATED} The name of the imageID column if using a SingleCellExperiment or SpatialExperiment.
cellTypeCol	{DEPRECATED} The name of the cellType column if using a SingleCellExperiment or SpatialExperiment.
spatialCoordCols	{DEPRECATED} The names of the spatialCoords column if using a SingleCellExperiment.
nCores	{DEPRECATED} Number of cores to use for parallel processing or a BiocParallel MulticoreParam or SerialParam object.
Rs	{DEPRECATED} A vector of the radii that the measures of association should be calculated over. calculation.

Value

Statistic from pairwise L-curve of a single image.

Examples

```
data("diabetesData")
# Subset by imageID for fast example
selected_cells <- diabetesData[
  , SummarizedExperiment::colData(diabetesData)$imageID == "A09"
]
pairAssoc <- getPairwise(selected_cells)
```

getPairwiseProp

Observed/expected proportion ratio for a fixed-k neighbourhood

Description

Computes a per-image, per-cell-type-pair spatial localisation statistic as a drop-in alternative to the fixed-radius L-function statistic returned by [getPairwise](#). Instead of a search radius r , each reference cell's neighbourhood is its k nearest neighbours (Euclidean distance, any cell type, the cell itself excluded).

Usage

```
getPairwiseProp(
  cells,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  k = 15,
  from = NULL,
  to = NULL,
  cores = 1,
  includeZeroCells = FALSE,
  BPPARAM = NULL
)
```

Arguments

<code>cells</code>	A <code>SingleCellExperiment</code> , <code>SpatialExperiment</code> or <code>data.frame</code> .
<code>imageID</code>	The name of the <code>imageID</code> column if a column name is provided.
<code>cellType</code>	The name of the <code>cellType</code> column if a column name is provided.
<code>spatialCoords</code>	The names of the spatial coordinates columns if provided.
<code>k</code>	Number of nearest neighbours to search per reference cell.
<code>from</code>	The reference cell types. Defaults to all cell types.
<code>to</code>	The target cell types. Defaults to all cell types.
<code>cores</code>	Number of threads for processing images in parallel (a <code>BiocParallelParam</code> object is also accepted, for backward compatibility).

includeZeroCells

If FALSE (default), image-pairs where the reference or target cell type is absent are returned as NA. If TRUE, image-pairs whose reference type is absent (but whose target type is present, so $p_{0,j} > 0$) are floored at $\hat{\pi}_j = 0$, i.e. $v_j = 0$. Pairs whose target type is entirely absent ($p_{0,j} = 0$) stay NA regardless, since the ratio is undefined.

BPPARAM

A BiocParallelParam object; its number of workers overrides cores. Kept for backward compatibility.

Details

For an ordered pair (from = reference type A, to = target type B), the statistic for image j is the plain observed-over-expected ratio

$$v_j = \hat{\pi}_j / p_{0,j},$$

where $\hat{\pi}_j$ is the mean over reference cells of the fraction of their k nearest neighbours that are type B, and $p_{0,j}$ is the image's background proportion of type-B cells (B count / total cell count).

$v_j = 1$ is the null (complete spatial randomness); $v_j > 1$ indicates attraction/enrichment and $0 \leq v_j < 1$ depletion. No variance-stabilising transform (arcsine, log) is applied: v_j is bounded below at 0 and right-skewed, a rougher fit to the linear model's Gaussian-noise assumption than a transformed statistic, but chosen for direct interpretability ("observed is 1.4x expected").

The returned matrix is shaped identically to `getPairwise()`'s output (images in rows, from__to cell-type pairs in columns, same order), so it can be passed straight to `spicy()` as `alternateResult` to run the standard `spicyR` weighting and linear (mixed) model pipeline on the ratio.

Weights. `spicy()`'s variance-weighting model (`weights = TRUE`, the default) is calibrated to the L-function's numeric scale. It works for this ratio when the dataset has enough cross-pair variance heterogeneity (e.g. several cell types, some rare) to anchor the fit, but on a dataset with few cell types or weak spatial structure it can degenerate and return an all-NA p-value table. If that happens, refit with `weights = FALSE`.

No edge correction is applied: $\hat{\pi}_j$ estimates neighbourhood composition rather than a count within an area, so a boundary cell's truncated neighbourhood carries the same null label distribution as an interior cell's. k controls only the variance of $\hat{\pi}_j$, not the location of its null value.

Value

A matrix with one row per image and one column per from__to cell-type pair, containing the observed/expected proportion ratio.

Examples

```
data("diabetesData")
propAssoc <- getPairwiseProp(diabetesData, k = 15)

# Use as the spicyR response, reusing the standard model:

spicy(diabetesData,
  condition = "stage", subject = "case",
  alternateResult = propAssoc
)
```

getProp	<i>Get proportions from a SummarizedExperiment.</i>
---------	---

Description

Get proportions from a SummarizedExperiment.

Usage

```
getProp(cells, feature = "cellType", imageID = "imageID")
```

Arguments

cells	A SingleCellExperiment, SpatialExperiment or data.frame.
feature	The feature of interest
imageID	The imageID's

Value

Proportions

Examples

```
data("diabetesData")
prop <- getProp(diabetesData)
```

imageCrossPlot	<i>Pairwise cross-plot of spatial associations</i>
----------------	--

Description

Builds a pairwise scatterplot of marker-marker associations for a single image using the output of getPairwise().

Usage

```
imageCrossPlot(
  result,
  image = NULL,
  colourGradient = c("#4575B4", "white", "#D73027"),
  marksToPlot = NULL,
  limits = NULL
)
```

Arguments

<code>result</code>	A matrix or data frame produced by <code>getPairwise()</code> where rownames are image IDs and colnames are concatenated marker pairs in the form " <code>from__to</code> ". Must contain at least one row with name matching <code>image</code> .
<code>image</code>	Character scalar. Which image (row in <code>result</code>) to plot. If <code>NULL</code> , defaults to the first rowname of <code>result</code> .
<code>colourGradient</code>	Character vector of length 3 giving the low, mid, and high colours for a diverging palette used by <code>scale_colour_gradient2()</code> . Default is <code>c("#4575B4", "white", "#D73027")</code> .
<code>marksToPlot</code>	Optional character vector of marker names. If supplied, the plot is restricted to rows and columns where both <code>from</code> and <code>to</code> are in this set.
<code>limits</code>	Numeric length-2 vector giving the lower and upper caps applied to association values for colour mapping. Use <code>NULL</code> to avoid clamping. Point sizes are clamped to <code>c(0, max(abs(limits)))</code> .

Details

The function expects `colnames(result)` to contain "`__`" separating marker names; otherwise an error is thrown. Values are transformed into two aesthetics:

- `value`: the (possibly clamped) signed association used for colour.
- `size`: the (possibly clamped) absolute association used for point size.

The colourbar shows three ticks at lower, 0, and upper with labels "`<lower`", "`0`", and "`>upper`".

Value

A **ggplot2** object.

See Also

[getPairwise](#)

Examples

```
# Minimal example (toy data)
set.seed(1)
mks <- c("A", "B", "C")
pairs <- as.vector(outer(mks, mks, paste, sep="__"))
res <- matrix(rnorm(length(pairs)*2), nrow = 2,
             dimnames = list(c("img1", "img2"), pairs))

p <- imageCrossPlot(res, image = "img1",
                   colourGradient = c("#4575B4", "white", "#D73027"),
                   marksToPlot = c("A", "B", "C"),
                   limits = c(-3, 3))

print(p)
```

plotImage

Plot one image, showing the from and to cells of a pair

Description

The density of all cells is shown in blue, with the from and to cells on top. With `r`, a circle of radius `r` is drawn around each to cell: the from cells inside the circles are those that spacyR counts.

Usage

```
plotImage(
  cells,
  imageToPlot,
  from,
  to,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  r = NULL
)
```

Arguments

<code>cells</code>	A SummarizedExperiment object.
<code>imageToPlot</code>	The ID of the image to be plotted.
<code>from</code>	The "from" cell type.
<code>to</code>	The "to" cell type.
<code>imageID</code>	The name of the imageID column in the SummarizedExperiment object.
<code>cellType</code>	The name of the cellType column in the SummarizedExperiment object.
<code>spatialCoords</code>	The names of the spatialCoords column if using a SingleCellExperiment.
<code>r</code>	Optional radius: draw a circle of this radius around each to cell.

Value

A ggplot object.

Examples

```
data("diabetesData")
plotImage(diabetesData, "A09", from = "acinar", to = "alpha")
plotImage(diabetesData, "A09", from = "acinar", to = "alpha", r = 50)
```

signifPlot

*Plots result of signifPlot.***Description**

Plots result of signifPlot.

Usage

```
signifPlot(
  results,
  fdr = FALSE,
  type = "bubble",
  breaks = NULL,
  comparisonGroup = NULL,
  colours = c("#4575B4", "white", "#D73027"),
  marksToPlot = NULL,
  cutoff = 0.05,
  contextColours = NULL,
  contextLabels = waiver()
)
```

Arguments

results	A spicy results object
fdr	TRUE if FDR correction is used.
type	Whether to make a bubble plot ("bubble") or a heatmap (any other value). Note: For survival results a bubble plot will be used.
breaks	Vector of 3 numbers giving breaks used in legend. The first number is the minimum, the second is the maximum, the third is the number of breaks.
comparisonGroup	A string specifying the name of the outcome group to compare with the base group.
colours	Vector of colours to use to colour legend.
marksToPlot	Vector of marks to include in plot.
cutoff	significance threshold for circles in bubble plot.
contextColours	Used for Kontextual results. A named list specifying the colours for each context. By default the Tableau colour palette is used.
contextLabels	Used for Kontextual results. A named list to change the default labels for each context.

Value

a ggplot object

Examples

```
data(spicyTest)

p <- signifPlot(spicyTest, breaks = c(-3, 3, 0.5))
p
signifPlot(spicyTest, type = "heatmap")
```

spicy

*Test for changes in the co-localisation of cell types between conditions***Description**

spicy() tests, for every ordered pair of cell types from $\rightarrow$ to, whether the co-localisation of the two types differs between conditions, or is associated with survival. A pair asks whether to cells are placed near from cells more than other cells are.

Usage

```
spicy(
  cells,
  condition,
  subject = NULL,
  covariates = NULL,
  imageID = "imageID",
  cellType = "cellType",
  spatialCoords = c("x", "y"),
  r = NULL,
  from = NULL,
  to = NULL,
  method = c("cell", "image"),
  effect = c("allocation", "count"),
  k = NULL,
  combine = c("maxT", "davies", "cauchy"),
  adjustAbundance = FALSE,
  variance = c("cr2", "hartung_knapp"),
  ref = NULL,
  cores = 1,
  tableKey = NULL,
  imageInfo = NULL,
  ...
)
```

Arguments

cells A data.frame, SingleCellExperiment or SpatialExperiment with one row (column) per cell, or (cell method) the path of a SpatialData or AnnData Zarr

store (read with the Rarr package). From a store, the columns are read a block of rows at a time and the coordinates a batch of whole images at a time, so the table is never held in memory; the results are those of the same cells given as a `data.frame`. Column names then refer to the table's obs; `imageID` defaults to the table's region key and `spatialCoords` to its obsm array "spatial". A `SpatialData` object (`spatialdataR`) read from a store is read from that store in the same way; one built in memory is analysed from its table.

condition	The column of the image-level condition: two groups, or a <code>survival::Surv</code> column for a survival outcome (from a store, a column of <code>imageInfo</code>).
subject	The column of the patient (unit) of each image. Images of one patient are combined; if omitted, every image is its own patient.
covariates	Image- or patient-level columns to adjust for (cell method: added to the design of the excess, and the effect of each is reported as <code><column>_effect</code> and <code><column>_p_value</code> ; survival: added to the null Cox model).
imageID, cellType, spatialCoords	Column names of the image, cell type and coordinates. For a Zarr store, <code>spatialCoords</code> is either two obs columns or the name of one obsm array (its first two columns).
r	Radius (or radii) of the neighbourhood, in the units of the coordinates. Cell method: one radius (default 50), or several to be combined by <code>combine</code> . Image method: the radii of the L function (default 20, 50 and 100).
from, to	Cell types to test (all ordered pairs by default).
method	"cell" (spicyR Cell, the default) or "image" (the original spicyR test).
effect	Cell method: "allocation" (the default), the extra fraction of to cells with at least one from cell within <code>r</code> ; or "count", the number of extra from cells within <code>r</code> of each to cell. With <code>k</code> , "within <code>r</code> " means among the cell's <code>k</code> nearest neighbours.
k	Cell method: use the <code>k</code> nearest neighbours instead of a radius.
combine	Cell method with several radii: "maxT" (max-T with the sandwich correlation across radii; up to 6 radii), "davies" (Davies' bound for the same maximum, for many radii) or "cauchy" (Cauchy combination).
adjustAbundance	Cell method: adjust the test for the log share of the from type in each image (default FALSE). Its effect is reported as <code>abundance_effect</code> . It does not separate more from cells from more densely packed ones, and it removes real effects when the share tracks the condition.
variance	Cell method: "cr2" (the default: CR2 on Satterthwaite df) or "hartung_knapp" (the model-based variance, scaled by the Pearson statistic when that exceeds its df and never below CR2, on <code>m - 2</code> df).
ref	Cell method: the reference level of condition.
cores	Number of threads (cell method) or cores (image method).
tableKey	Cell method with a <code>SpatialData</code> store or object: the name of the cell table in tables/ (by default the only table, or the one called "table").
imageInfo	Cell method: a table with one row per image, for condition, subject, covariates or a survival outcome that are not columns of the cell table: a <code>data.frame</code> whose

imageID column, or else its row names, gives the image IDs, or the name of a table of the SpatialData store or object. Every image of the cells needs a row, and every row an image. A survival::Surv column given as condition makes a survival analysis, also from a store.

... Arguments of the image method: sigma, alternateResult, minLambda, weights, weightsByPair, weightFactor, weightZThreshold, window, window.length, edgeCorrect, includeZeroCells, verbose, BPPARAM. Supplying alternateResult selects the image method.

Details

method = "cell" (**the default**). In each image, the share of to cells with at least one from cell within radius r is compared with its exact expectation q if the to cells were a random choice among the cells that are not from cells. The effect (effect = "allocation", the default) is the **fraction of to cells placed next to (or kept away from) from cells**. For a pair that attracts (more to cells next to from cells than q over all images together) it is (observed share - q) / (1 - q): if a fraction f of the to cells were moved next to from cells, the effect is f . For a pair that avoids it is (observed share - q) / q : if a fraction f of the to cells that would have a from cell nearby were moved away, the effect is $-f$. Either way it does not depend on how many from cells there are or how densely they are packed. The side is chosen once per pair from all images, without the conditions, and is reported in the side column. effect = "count" gives the number of extra from cells within r of each to cell instead; it also reflects how many from cells surround a to cell (depth of infiltration), but it grows with how densely the from cells are packed, so a change in packing alone can appear as a change in co-localisation. Images are combined within patients and patients within conditions by a frailty GEE with a second random effect for the images of a patient (each image is weighted by $1 / (\text{its sampling variance} + \text{the between-image variance})$, so one large image cannot outweigh the patient's other images), and the difference between conditions is tested with a CR2 cluster-robust variance on Satterthwaite degrees of freedom, with **patients (subject) as the units**. The difference is adjusted for any covariates, and, with adjustAbundance = TRUE, for the log share of the from type in each image; the unadjusted test is then reported alongside (unadjusted_* columns).

When nearly every cell has a from cell within r (q close to 1), there is little room for attraction and an attracting pair's images carry little information; a smaller r is more informative.

method = "image" (**the original spicyR test**). A per-image L-function summary of each pair is compared between conditions with a weighted linear model, or a mixed model when subject is given (Canete et al. 2022).

Value

A SpicyResults object. topPairs(), signifPlot(), spicyBoxPlot() and bind() work for both methods. For the cell method, \$cellResults holds the full table: for the allocation effect, the pair's side ("attract" or "avoid", which sets the scale of the effect), the effect in each condition (excess_ref, excess_comp, at the average covariates), the difference (excess_difference), its standard error, df, p-value and BH-adjusted p-value, the frailty variance, the between-image variance (tau2_image) and, when the test was adjusted, what for (adjusted_for), the effect and p-value of each adjustment, and the unadjusted test (unadjusted_difference, unadjusted_p_value, unadjusted_p_adj). \$effect records which effect was estimated.

References

- Canete NP et al. (2022). spicyR: spatial analysis of in situ cytometry data in R. *Bioinformatics* 38(11), 3099-3105. doi:[10.1093/bioinformatics/btac268](https://doi.org/10.1093/bioinformatics/btac268)
- Bell RM, McCaffrey DF (2002). Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology* 28(2), 169-181.
- DerSimonian R, Laird N (1986). Meta-analysis in clinical trials. *Controlled Clinical Trials* 7(3), 177-188. doi:[10.1016/01972456\(86\)900462](https://doi.org/10.1016/01972456(86)900462)
- Pustejovsky JE, Tipton E (2018). Small-sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business & Economic Statistics* 36(4), 672-683. doi:[10.1080/07350015.2016.1247004](https://doi.org/10.1080/07350015.2016.1247004)
- Paule RC, Mandel J (1982). Consensus values and weighting factors. *Journal of Research of the National Bureau of Standards* 87(5), 377-385. doi:[10.6028/jres.087.022](https://doi.org/10.6028/jres.087.022)
- Hartung J, Knapp G (2001). A refined method for the meta-analysis of controlled clinical trials with binary outcome. *Statistics in Medicine* 20(24), 3875-3889. doi:[10.1002/sim.1009](https://doi.org/10.1002/sim.1009)
- Liu Y, Xie J (2020). Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures. *Journal of the American Statistical Association* 115(529), 393-402. doi:[10.1080/01621459.2018.1554485](https://doi.org/10.1080/01621459.2018.1554485)

Examples

```
data("diabetesData")
# spicyR Cell: patients ("case") are the units
# the extra fraction of Th cells, and of beta cells, with a Tc cell within 50 units
res <- spicy(diabetesData, condition = "stage", subject = "case", r = 50,
             from = "Tc", to = c("Th", "beta"))
topPairs(res)
res$cellResults

# the extra number of Tc cells within 50 units of each Th cell
resCount <- spicy(diabetesData, condition = "stage", subject = "case", r = 50,
                  from = "Tc", to = "Th", effect = "count")

# the original image-level test
resImage <- spicy(diabetesData, condition = "stage", subject = "case",
                  from = "Tc", to = "Th", method = "image")
topPairs(resImage)
```

spicyBoxPlot

Box plot of one pair, with a point per image

Description

Shows the per-image value of one pair in each condition: a box plot with the images as points behind it. For the cell method the value is the effect of `spicy()` in each image (by default the extra fraction of to cells with a from cell within `r`; with `effect = "count"`, the extra from cells per to cell), and each point is sized by how much the image contributes to the test (its weight in the frailty

model, relative to the average image of its condition). With `interactive = TRUE` the plot is a plotly widget: hover over a point to see its image, patient, excess and weight, which helps to find images worth looking at with `plotImage()`.

Usage

```
spicyBoxPlot(
  results,
  from = NULL,
  to = NULL,
  rank = NULL,
  interactive = FALSE,
  parent = NULL
)
```

Arguments

<code>results</code>	The SpicyResults object returned by <code>spicy()</code> (either method).
<code>from</code>	The from cell type (the neighbours looked for around each to cell).
<code>to</code>	The to cell type (the centre).
<code>rank</code>	Alternatively, the rank of the pair by p-value (1 is the most significant).
<code>interactive</code>	Return an interactive plotly widget instead of a ggplot (needs the plotly package).
<code>parent</code>	For a Kontextual result (<code>Statial::kontextualTest()</code>), the parent population of the triple.

Value

A ggplot, or with `interactive = TRUE` a plotly htmlwidget. Images in which the pair was not tested (no to cells, for example) are not shown.

Examples

```
data(spicyTest)
spicyBoxPlot(spicyTest, rank = 1)

data("diabetesData")
res <- spicy(diabetesData, condition = "stage", subject = "case", r = 50,
            from = "Tc", to = c("Th", "beta"))
spicyBoxPlot(res, from = "Tc", to = "Th")
```

SpicyResults-class *The results of spicy()*

Description

`spicy()` returns a `SpicyResults` object: a list with the elements below. Use `topPairs()`, `signifPlot()`, `spicyBoxPlot()` and `bind()` on it, or take elements with `$`.

Details

Elements for the cell method (`method = "cell"`, the default):

`cellResults` the full table, one row per pair (and per level with more than two conditions): `excess_ref`, `excess_comp`, `excess_difference`, `se`, `df`, `p_value`, `p_adj`, `tau2`; with covariates or `adjustAbundance = TRUE`, what the test was adjusted for (`adjusted_for`), the effect and p-value of each adjustment (`<covariate>_effect`, `<covariate>_p_value`, `abundance_effect`) and the unadjusted test (`unadjusted_difference`, `unadjusted_se`, `unadjusted_df`, `unadjusted_p_value`, `unadjusted_p_adj`). For survival: `score_coefficient`, `p_value`, `p_adj`, `hazard_ratio_sd`, `hr_p_value` and, with `adjustAbundance = TRUE`, the p-value and hazard ratio without it. With several radii: `r`, the radius with the strongest evidence, and `p_value_best_radius`.

`radiusResults` with several radii, the table at every radius.

`pairwiseAssoc` a list with, for every pair, the effect in every image (NA where the pair could not be measured).

`effect` "allocation" (the extra fraction of to cells with a from cell within `r`), "count" (the extra from cells within `r` of each to cell), or "kontextual" (Statial's Kontextual test).

`imageWeights` a list with, for every pair, each image's weight in the test: its share of its condition's information (sums to 1 within each condition).

`coefficient`, `p.value`, `se`, `statistic`, `df` matrices with one row per pair, as for the image method: the reference excess and the differences, and their tests.

`condition`, `imageID`, `subject`, `nCells`, `r`, `k` the condition and patient of every image, the cell counts, and the settings of the analysis.

For the image method (`method = "image"`) the object holds the matrices `coefficient`, `p.value`, `se`, `statistic` and `df`, and `pairwiseAssoc`, as in `spicyR` 1.x.

See Also

[spicy](#), [topPairs](#), [bind](#)

spicyTest

Results of the image-level test on diabetesData

Description

The result of `spicy(diabetesData, condition = "stage", subject = "case", method = "image")`: the original image-level test of `spicyR`, for all pairs of cell types, comparing the onset and long-duration stages with non-diabetic donors. It is used in examples, so that they run quickly.

Usage

```
data("spicyTest")
```

Format

A `SpicyResults` object (see [spicy](#)).

Source

Made by `inst/scripts/make-spicyTest.R` from [diabetesData](#).

Examples

```
data("spicyTest")
topPairs(spicyTest, n = 5)
```

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, figures = NULL)
```

Arguments

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

Value

A data.frame

Examples

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

writeCellStore	<i>Write cells as a SpatialData Zarr store</i>
----------------	--

Description

Writes a data.frame of cells as the cell table of a SpatialData Zarr store (an AnnData table under tables/, in Zarr v3), which `spicy()` reads a block of rows at a time and `spatialdataR`'s `readSpatialData()` reads as a SpatialData object. The coordinates become the obsm array "spatial", the other columns the table's obs (factors as categorical columns), and the image column the table's region key (always a categorical column, its categories the levels of a factor, else the images in order of first appearance). A table with one row per image can be added as a second table. Each column is written in chunks of 2^{20} rows compressed with zstd, one chunk at a time, so writing needs little memory beyond that of cells, however many cells it has.

Usage

```
writeCellStore(
  cells,
  path,
  imageID = "imageID",
  spatialCoords = c("x", "y"),
  tableKey = "table",
  imageInfo = NULL,
  infoKey = "images"
)
```

Arguments

cells	A data.frame with one row per cell. Its columns are factors, character, integer, double or logical vectors.
path	The directory of the store (replaced if it exists).
imageID, spatialCoords	The column of the image and the two columns of the coordinates.
tableKey	The name of the cell table.
imageInfo	Optional: a data.frame with one row per image (row names: the image IDs), written as the table infoKey.
infoKey	The name of the per-image table.

Value

path, invisibly.

Examples

```
cells <- data.frame(imageID = factor(rep(c("a", "b"), each = 50)), cellType = factor(rep(c("T", "B"), 50)),  
                    x = runif(100, 0, 100), y = runif(100, 0, 100))  
if (requireNamespace("Rarr", quietly = TRUE)) {  
  path <- writeCellStore(cells, tempfile(fileext = ".zarr"))  
  list.files(file.path(path, "tables"))  
}
```

Index

- * **datasets**
 - diabetesData, [6](#)
 - spicyTest, [21](#)
- bind, [3](#), [20](#)
- buildKnnGraph, [3](#)
- colTest, [4](#)
- convPairs, [5](#)
- convPairs(), [3](#)
- diabetesData, [6](#), [21](#)
- getPairwise, [7](#), [9](#), [12](#)
- getPairwiseProp, [9](#)
- getProp, [11](#)
- imageCrossPlot, [11](#)
- Kontextual, [14](#)
- plotImage, [13](#)
- plotImage(), [19](#)
- signifPlot, [14](#)
- spicy, [15](#), [20](#), [21](#)
- spicy(), [3](#), [18](#), [22](#)
- spicyBoxPlot, [18](#)
- SpicyResults (SpicyResults-class), [20](#)
- SpicyResults-class, [20](#)
- spicyTest, [21](#)
- topPairs, [20](#), [21](#)
- topPairs, SpicyResults-method
 (topPairs), [21](#)
- writeCellStore, [22](#)