Package: fuzzyCell (via r-universe)

October 29, 2026

Type Package

Title Bootstrap Classification Confidence for Spatial Cell-Type Analysis

Version 0.99.1

Description Cell type labels in spatial single-cell data are uncertain, particularly between closely related subtypes. fuzzyCell estimates a per-cell classification confidence by bootstrapping (multinomial resampling of each cell's counts followed by re-classification) and uses the resulting confidences as soft cell type weights. The weights are propagated through downstream analyses: cell type proportions, co-localisation (spicyR's L-function, and Statial's Kontextual test for relative co-localisation) and niche identification (lisaClust), each compared to the same analysis using hard labels. Cell types are organised with a user-supplied hierarchy of lineages, groups and subtypes. Helpers for testing against sample-level outcomes and for plotting are included.

License GPL (>= 3)

URL https://github.com/shreyarajeshrao/fuzzyCell

BugReports https://github.com/shreyarajeshrao/fuzzyCell/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

biocViews Software, Spatial, SingleCell, Classification, Clustering, StatisticalMethod, Regression, SupportVectorMachine

Depends R (>= 4.5.0)

Imports BiocGenerics, BiocParallel, callr, dplyr, e1071, forcats, ggplot2, glmnet, irlba, lisaClust (>= 1.21.9), MASS, Matrix, methods, pheatmap, randomForest, RhpcBLASctl, rlang, SpatialExperiment, spicyR (>= 1.99.32), stats, SummarizedExperiment, tibble, tidyr, utils, withr

Suggests gdtools, ggalluvial, ggiraph, ggnewscale, knitr, quarto, S4Vectors, SpatialDatasets, testthat (>= 3.0.0)

VignetteBuilder quarto

SystemRequirements quarto

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libmagick++-dev gsfonts libicu-dev libssl-dev zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-10-29 02:36:37 UTC

RemoteUrl https://github.com/BiocStaging/fuzzyCell

RemoteRef HEAD

RemoteSha d171b9a457af9869138d4f998c8d3cec0fd8e6ea

Contents

bootstrapClassify . . . 2
bootstrapLineageConfidence . . . 4
cellTypeTree . . . 6
confidenceWeights . . . 8
crossfitLineageProbabilities . . . 9
getCellData . . . 10
heartCellTypeTree . . . 11
loadAmancherla . . . 11
lungCellTypeTree . . . 12
lungConfList . . . 13
lungSubset . . . 13
nicheIdentification . . . 14
nichePlots . . . 16
plotMarkerSpatial . . . 17
plotMetricBoxplot . . . 19
plotProportionBoxplot . . . 20
plotSpatial . . . 21
runColocalisationModels . . . 23
runProportionsPipeline . . . 25
testNicheOutcome . . . 27

Index **29**

`bootstrapClassify` *Bootstrap classification confidence for one group*

Description

The single-group worker behind bootstrapLineageConfidence(). Trains a classifier on one group's cells, then repeatedly resamples each cell's counts (multinomial resampling) and re-predicts its class. The fraction of resamples assigning a cell to each class is its confidence for that class.

Usage

```
bootstrapClassify(
  countMat,
  labels,
  nBoot = 100,
  nPerClass = 50,
  cores = 1,
  seed = NULL,
  classifier = c("pcaLogistic", "logistic", "rf", "lda", "svm"),
  ntree = 500,
  priors = c("balanced", "observed"),
  correctedCounts = FALSE
)
```

Arguments

countMat	A cells x genes matrix of raw (non-negative integer) counts for the cells of one group. Normalised or log-transformed values are rejected, since the resampling needs each cell's total transcript count.
labels	Character vector of the original cell type label for each row of countMat.
nBoot	Number of bootstrap resamples.
nPerClass	Minimum number of training cells per class; classes smaller than this are up-sampled with replacement.
cores	Number of cores to run the bootstrap resamples on (forked with BiocParallel); the default runs serially. On Windows, where forking is not available, it always runs serially. If seed is given, the results are identical whatever the number of cores.
seed	Random seed for reproducibility, or NULL. Recommended for reproducibility.
classifier	One of "pcaLogistic" (ridge multinomial logistic regression, via glmnet, with the penalty chosen by cross-validation, on the first 50 principal components of the training cells, computed exactly for up to 5,000 genes and with irlba beyond; the default), "logistic" (the same on all genes), "rf" (random forest, via randomForest), "lda" (linear discriminant analysis, via MASS) or "svm" (support vector machine, via e1071).
ntree	Number of trees; only used when classifier = "rf".
priors	The classifier is trained with balanced classes, which gives every class the same prior. "balanced" keeps those equal priors; "observed" weights each class by its share of the cells, so a resample is assigned to its most probable class given how common the classes are.
correctedCounts	FALSE (the default) requires whole-number counts. TRUE also accepts counts that are not whole numbers, such as counts corrected for ambient RNA, and resamples them as they are. They must still be on the scale of molecule counts: a cell's resample draws as many transcripts as its total (rounded), so normalised or log-transformed values would give the wrong amount of resampling noise.

Value

A numeric matrix with one row per cell (in the order of `countMat`) and one column per class in `labels`. Entry `[i, k]` is the fraction of bootstrap resamples in which cell `i` was classified as class `k`, so each row sums to 1.

Examples

```
data(lungSubset)
endo = lungSubset$cellType %in% c("Arteriole", "Venous", "Capillary", "Lymphatic")
countMat = t(as.matrix(SummarizedExperiment::assay(lungSubset, "counts")[, endo]))
freq = bootstrapClassify(countMat, lungSubset$cellType[endo], nBoot = 5, seed = 1)
head(freq)
```

---

bootstrapLineageConfidence

Bootstrap confidence scores for every group in a cell type tree

---

Description

For each group in `cellTypeTree` that has children (see cellTypeTree), runs `bootstrapClassify()` on that group's cells to get, for every cell, a confidence score for each of the group's child cell types. The result is the `confList` used by the downstream analyses, such as `runProportionsPipeline()` and `runColocalisationModels()`. Cell types without a group (ungrouped leaves) are not refined and keep hard labels.

Usage

```
bootstrapLineageConfidence(
  spe,
  cellTypeTree,
  nBoot = 100,
  nPerClass = 50,
  cores = 1,
  seed = 51773,
  skipGroups = character(0),
  subprocess = FALSE,
  classifier = c("pcaLogistic", "logistic", "rf", "lda", "svm"),
  ntree = 500,
  priors = c("balanced", "observed"),
  correctedCounts = FALSE,
  cellType = "cellType",
  cellID = "cellID",
  assayName = "counts"
)
```

Arguments

spe	A SummarizedExperiment (e.g. SpatialExperiment) with a counts assay and per-cell cell type labels and unique cell IDs in colData.
cellTypeTree	A cell type tree, see cellTypeTree.
nBoot	Number of bootstrap resamples.
nPerClass	Minimum number of training cells per class; classes smaller than this are up-sampled with replacement.
cores	Number of cores to run the bootstrap resamples on (forked with BiocParallel); the default runs serially. On Windows, where forking is not available, it always runs serially. If seed is given, the results are identical whatever the number of cores.
seed	Random seed, or NULL for no fixed seed. Each group is bootstrapped with seed plus its position in the list of groups, so every group is reproducible independently of the others.
skipGroups	Names of groups to leave out of the result.
subprocess	If TRUE, each group is bootstrapped in a fresh R session (via callr), which only holds that group's counts. With cores above 1, every worker starts as a copy of the session it is forked from, so this keeps the workers small on large data. It requires the same version of fuzzyCell to be installed.
classifier	One of "pcaLogistic" (ridge multinomial logistic regression, via glmnet, with the penalty chosen by cross-validation, on the first 50 principal components of the training cells, computed exactly for up to 5,000 genes and with irlba beyond; the default), "logistic" (the same on all genes), "rf" (random forest, via randomForest), "lda" (linear discriminant analysis, via MASS) or "svm" (support vector machine, via e1071).
ntree	Number of trees; only used when classifier = "rf".
priors	The classifier is trained with balanced classes, which gives every class the same prior. "balanced" keeps those equal priors; "observed" weights each class by its share of the cells, so a resample is assigned to its most probable class given how common the classes are.
correctedCounts	FALSE (the default) requires whole-number counts. TRUE also accepts counts that are not whole numbers, such as counts corrected for ambient RNA, and resamples them as they are. They must still be on the scale of molecule counts: a cell's resample draws as many transcripts as its total (rounded), so normalised or log-transformed values would give the wrong amount of resampling noise.
cellType	Name of the colData(spe) column holding each cell's cell type label.
cellID	Name of the colData(spe) column holding unique cell IDs. They are returned in the cellID column of each result.
assayName	Name of the assay with the raw counts.

Value

A named list with one element per group (named by group). Groups with no cells, or with only one cell type present, cannot be bootstrapped: they are skipped with a message and left out of the

list. Each element is a `data.frame` with one row per cell in that group, a `cellID` column, and one column per child cell type of the group holding the cell's confidence score for that cell type (between 0 and 1, rows summing to 1 across the child columns). Each data frame also carries a `"settings"` attribute recording how it was computed (`nBoot`, `nPerClass`, `classifier`, `priors`, `correctedCounts`, `ntree`, the group's `seed`, its `children` and the `fuzzyCellVersion`).

Examples

```
data(lungSubset)
data(lungCellTypeTree)
confList = bootstrapLineageConfidence(lungSubset, lungCellTypeTree, nBoot = 5,
    seed = 1)
names(confList)
head(confList[["Endothelial lineage"]])

# the same call with nBoot = 20 gives the confidences in `lungConfList`
```

`cellTypeTree` *Cell type tree utilities*

Description

Helpers for the cell type tree: a nested list describing how cell types are organised into broad lineages and (optionally) groups of closely related subtypes. `bootstrapLineageConfidence()` refines the cells within each group, and the analysis functions compare cell types, groups and lineages.

Usage

```
buildCellTypeMap(cellTypeTree)

annotateCellTypeTree(spe, cellTypeTree, cellType = "cellType")

resolveTreeLeaves(cellTypeTree, nodeName)

autoParentOverrides(cellTypeTree)
```

Arguments

`cellTypeTree`	A nested list describing the cell type hierarchy (see Details and heartCellTypeTree).
`spe`	A `SummarizedExperiment` (e.g. a `SpatialExperiment`), or a `data.frame`/`DataFrame` of per-cell data.
`cellType`	Name of the column of `colData(spe)` (or of the data frame) holding each cell's cell type label. An error is raised if it is not a column.
`nodeName`	A single string: the name of any node in `cellTypeTree`, a lineage, an organisational wrapper, a group or a cell type.

Details

A tree is a named list with one entry per broad lineage (these become the `lineage` column). Each lineage is either a character vector of cell types, or a list whose entries are:

- a bare string: a cell type that is not in any group;
- a named `list(children = c(...))`: a **group** of closely related cell types, the unit that is bootstrap-refined. A group may contain only `children`;
- a named list of further entries: purely organisational nesting (a "wrapper") that is not a group itself, to any depth.

A lineage can also itself be `list(children = c(...))`, in which case the lineage name doubles as the group name. A cell type's group is its nearest `list(children = ...)` ancestor, however many wrappers sit above it. In the example below `CD8_Resting` is in the group `CD8+ T cells`, not `T cells`.

Every cell type must appear exactly once, and lineage, wrapper and group names must be unique and different from every cell type name.

```
list(
    "Immune" = list(
        "T cells" = list(
            "CD8+ T cells" = list(children = c("CD8_Resting", "CD8_Memory")),
            "CD4+ T cells" = list(children = c("CD4_naive", "CD4_Memory"))),
        "NK",
        "Mast"),
    "Endothelial lineage" = c("Endothelial", "Activated endothelial"))
```

`resolveTreeLeaves()` expands any node, however far up the tree, to the cell types beneath it. It is used by relative co-localisation when it conditions on a parent that is broader than a cell type's own group (see `parentOverrides` in `runColocalisationModels()`), and `autoParentOverrides()` finds those parents automatically.

Value

- `buildCellTypeMap()`: a list with `cellMap` (a `data.frame` with columns `cellType`, `group` and `broad`, one row per cell type) and `groupsWithChildren` (names of the groups to be bootstrap-refined). An error is raised if the tree is not a valid tree as described in Details: it must be a non-empty named list, every top-level entry must be named, groups may only contain `children`, cell type names must be non-empty strings, and names must not be repeated or shared between cell types and lineages, wrappers or groups.
- `annotateCellTypeTree()`: `spe` with `group` and `lineage` columns added (a message reports any existing `group` or `lineage` column whose values are replaced). An error is raised if any cell's cell type is not in the tree (including `NA`).
- `resolveTreeLeaves()`: character vector of every cell type beneath `nodeName`.
- `autoParentOverrides()`: named character vector mapping each group that has a named ancestor to that ancestor, for use as `parentOverrides` in `runColocalisationModels()`.

Examples

```
data(lungCellTypeTree)
buildCellTypeMap(lungCellTypeTree)
resolveTreeLeaves(lungCellTypeTree, "Immune")
autoParentOverrides(lungCellTypeTree)

data(lungSubset)
cells = annotateCellTypeTree(getCellData(lungSubset), lungCellTypeTree)
table(cells$group, useNA = "ifany")
```

`confidenceWeights` *Cell type weights from bootstrap confidences*

Description

Builds the soft cell type membership of every cell, the weights that the weighted analyses pass to spicyR and lisaClust: one column per leaf cell type and per group of the tree. A cell of a bootstrapped group has, for each subtype of the group, its bootstrap confidence in that subtype; every other cell has weight 1 for its own cell type. A group's column is 1 for each of its cells, since the bootstrap reclassifies cells only within their group.

Usage

```
confidenceWeights(
  spe,
  cellTypeTree,
  confList,
  weighted = TRUE,
  cellType = "cellType",
  cellID = "cellID"
)
```

Arguments

`spe`	A `SummarizedExperiment` (e.g. `SpatialExperiment`) with a cell ID, cell type, image and coordinates for every cell (see `getCellData()`).
`cellTypeTree`	A cell type tree, see cellTypeTree.
`confList`	Confidence list from `bootstrapLineageConfidence()`.
`weighted`	`TRUE` for the bootstrap confidences, `FALSE` for the original labels (0/1 weights) on the same columns.
`cellType, cellID`	Names of the `colData(spe)` columns holding each cell's cell type label and a unique cell ID (matching the `cellID` column of `confList`).

Value

A numeric matrix with one row per cell of `spe` (in its order) and one named column per leaf cell type and group.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
w = confidenceWeights(lungSubset, lungCellTypeTree, lungConfList)
head(w[, c("Arteriole", "Venous", "Endothelial lineage")])
```

`crossfitLineageProbabilities` *Cross-fitted classification probabilities for every group of a tree*

Description

An alternative to `bootstrapLineageConfidence()`: for each group with children, a classifier is trained on the group's cells with balanced classes and gives each cell the probability of each child cell type. The cells are split into `nFolds` folds and each fold is scored by a classifier trained on the other folds, so no cell is scored by a classifier that saw it. Because the classes are balanced in training, the probabilities give every child cell type the same prior weight.

Usage

```
crossfitLineageProbabilities(
  spe,
  cellTypeTree,
  nFolds = 5,
  nPerClass = 50,
  seed = 51773,
  classifier = c("pcaLogistic", "logistic", "rf", "lda"),
  ntree = 500,
  priors = c("balanced", "observed"),
  cores = 1,
  skipGroups = character(0),
  cellType = "cellType",
  cellID = "cellID",
  assayName = "counts"
)
```

Arguments

`spe`	A `SummarizedExperiment` (e.g. `SpatialExperiment`) with a counts assay and per-cell cell type labels and unique cell IDs in `colData`.
`cellTypeTree`	A cell type tree, see cellTypeTree.
`nFolds`	Number of folds. The folds are stratified by cell type, so every fold's classifier is trained on every child cell type; each child cell type therefore needs at least 2 cells.

nPerClass	Minimum number of training cells per class; classes smaller than this are upsampled with replacement.
seed	Random seed, or NULL for no fixed seed. Each group is bootstrapped with seed plus its position in the list of groups, so every group is reproducible independently of the others.
classifier	"pcaLogistic" (the default), "logistic", "rf" or "lda", as in bootstrapClassify().
ntree	Number of trees; only used when classifier = "rf".
priors	"balanced" (equal priors, as the classifier is trained) or "observed" (each class weighted by its share of the group's cells).
cores	Number of cores to fit the folds on in parallel, as in bootstrapClassify().
skipGroups	Names of groups to leave out of the result.
cellType	Name of the colData(spe) column holding each cell's cell type label.
cellID	Name of the colData(spe) column holding unique cell IDs. They are returned in the cellID column of each result.
assayName	Name of the assay with the raw counts.

Value

A confidence list in the format of bootstrapLineageConfidence(), with probabilities in place of bootstrap confidences.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
probList = crossfitLineageProbabilities(lungSubset, lungCellTypeTree,
    nFolds = 3, seed = 1)
head(probList[["Endothelial lineage"]])
```

---

getCellData *Per-cell data frame from a SummarizedExperiment*

Description

Returns colData(spe) as a data.frame. For a SpatialExperiment whose coordinates are stored in spatialCoords() rather than in colData(), the coordinates are added as columns x and y. The spatial functions in fuzzyCell expect cellID, imageID, cellType, x and y columns.

Usage

```
getCellData(spe)
```

Arguments

spe	A SummarizedExperiment or SpatialExperiment.

Value

A `data.frame` with one row per cell.

Examples

```
data(lungSubset)
head(getCellData(lungSubset))
```

---

heartCellTypeTree *Cell type tree for the heart transplant data*

Description

The cell type hierarchy used with `loadAmancherla()`: broad lineages, groups of closely related subtypes (for example the T cell subtypes), and ungrouped leaf cell types. See cellTypeTree for the format.

Usage

```
data(heartCellTypeTree)
```

Format

A nested list; see cellTypeTree.

Source

Defined in `data-raw/heartCellTypeTree.R`.

Examples

```
data(heartCellTypeTree)
buildCellTypeMap(heartCellTypeTree)$groupsWithChildren
```

---

loadAmancherla *Heart transplant example data*

Description

Downloads the Amancherla et al. (2025) heart transplant Xenium dataset from the SpatialDatasets package (via ExperimentHub) and prepares it for fuzzyCell: renames the columns fuzzyCell expects, keeps patients with cellular rejection, and keeps one randomly chosen image per patient for each biopsy timing. The first call downloads the data, which is large. With `"both"` every patient has an image before and after transplant, so those images are not independent and cannot be passed together to `runColocalisationModels()` with `subject = "subject"`.

Usage

```
loadAmancherla(timing = c("pre", "post", "both"), seed = 51773)
```

Arguments

timing	Which biopsies to keep: "pre" or "post" transplant, or "both".
seed	Random seed used to choose one image per patient.

Details

The prepared object has the columns fuzzyCell functions use: cellID, imageID, cellType, x, y and subject, and the per-patient outcome resolution_broad ("persistence" or "resolved"). The cell type hierarchy for these cell types is in heartCellTypeTree.

Value

A SpatialExperiment.

Examples

```
spe = loadAmancherla("pre")
spe
table(spe$resolution_broad)
```

---

lungCellTypeTree *Cell type tree for the lung data*

Description

The cell type hierarchy for lungSubset (and the full SpatialDatasets::spe_Vannan_2025() data): broad lineages, groups of closely related subtypes (for example the fibroblast subtypes), and ungrouped leaf cell types. See cellTypeTree for the format.

Usage

```
data(lungCellTypeTree)
```

Format

A nested list; see cellTypeTree.

Source

Defined in data-raw/lungCellTypeTree.R.

Examples

```
data(lungCellTypeTree)
buildCellTypeMap(lungCellTypeTree)$groupsWithChildren
```

---

lungConfList *Bootstrap confidences for the lung data*

Description

The bootstrap classification confidences of the cells in lungSubset, from `bootstrapLineageConfidence(lungSubset, lungCellTypeTree, nBoot = 20, classifier = "pcaLogistic", seed = 1)`, so that the examples can skip the bootstrap.

Usage

```
data(lungConfList)
```

Format

A named list with one data frame per group of lungCellTypeTree (see `bootstrapLineageConfidence()` for the format).

Source

Built by `data-raw/lungConfList.R`.

Examples

```
data(lungConfList)
names(lungConfList)
head(lungConfList[["Endothelial lineage"]])
```

---

lungSubset *A small cropped subset of the pulmonary fibrosis lung data*

Description

Xenium data of lung biopsies from Vannan et al. (2025), available in full as `SpatialDatasets::spe_Vannan_2025()`, cut down to ship with the package. It has five control and five pulmonary fibrosis biopsies (the five with the fewest cells of each), each cropped to the roughly 3,000 cells in a square window at its centre, so each image is a window of a larger biopsy. Every image comes from a different subject. Cell types with fewer than 20 cells in the windows are removed. The hierarchy for these cell types is in lungCellTypeTree.

Usage

```
data(lungSubset)
```

Format

A `SpatialExperiment` with the raw counts of 343 genes in 29,971 cells (assay `counts`), the cell coordinates in `spatialCoords()`, and in `colData`:

cellID A unique cell ID.

imageID The image (the window of a biopsy); also `sample_id`.

subject The patient.

cellType The cell type label.

disease_status `"Control"` or `"Disease"`.

Source

Built from `SpatialDatasets::spe_Vannan_2025()` by `data-raw/lungSubset.R`.

Examples

```
data(lungSubset)
lungSubset
table(lungSubset$imageID, lungSubset$disease_status)
```

`nicheIdentification` *Niche identification*

Description

Defines niches by k-means clustering of per-cell feature vectors, once with hard cell type membership and once with bootstrap-confidence weights. The features of each cell are its local indicators of spatial association from lisaClust (`lisaClust::lisa()`) at several radii.

Usage

```
runNicheIdentification(
  spe,
  cellTypeTree,
  confList,
  toLabels = NULL,
  r = c(20, 50, 100),
  k,
  window = "convex",
  lisaFunc = "K",
  cores = 1,
  seed = 51773,
  cellType = "cellType",
  cellID = "cellID",
  imageID = "imageID",
  x = "x",
  y = "y"
)
```

Arguments

spe	A SummarizedExperiment (e.g. SpatialExperiment) with a cell ID, cell type, image and coordinates for every cell (see getCellData()).
cellTypeTree	A cell type tree, see cellTypeTree.
confList	Confidence list from bootstrapLineageConfidence().
toLabels	Labels (leaves and/or groups) to build features from; defaults to every leaf and group in the tree. An error is raised for a label that is not in the tree.
r	Radii of the local curves: one or more positive numbers.
k	Number of niches, a whole number of at least 2 and below the number of cells.
window, lisaFunc	The image window and the local function ("K" or "L"), as in lisaClust::lisa().
cores	Number of threads.
seed	Random seed for the k-means clustering, as in bootstrapLineageConfidence(), or NULL for no fixed seed. The same seed is used for the unweighted and weighted clusterings.
cellType, cellID, imageID	Names of the colData(spe) columns holding each cell's cell type label, a unique cell ID (matching the cellID column of confList), and the image it is in.
x, y	Names of the columns holding the cell coordinates. For a SpatialExperiment whose coordinates are in spatialCoords(), the defaults "x" and "y" refer to its first two coordinates.

Details

For every cell, every label in toLabels and every radius in r, the feature is the local co-localisation of the cell's neighbours with that label (lisaClust's local K or L function, centred so that 0 means no attraction or repulsion). The weighted features count each neighbour with its bootstrap confidence of having the label, instead of 0 or 1 (see confidenceWeights()), and a label's density is its summed weight per unit area.

Niche names (niche1, niche2, ...) are arbitrary. K-means starts from random cluster centres, so the niches depend on the random seed. The unweighted and weighted clusterings are both started from the same seed, so that if the weights made no difference the two sets of niches would be identical.

Value

A list with the feature matrices (lisaMatUnweighted, lisaMatWeighted) and named cell-to-niche vectors nicheUnweighted and nicheWeighted.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
niches = runNicheIdentification(lungSubset, lungCellTypeTree, lungConfList,
    r = 50, k = 6)
table(weighted = niches$nicheWeighted, unweighted = niches$nicheUnweighted)
```

nichePlots *Niche plots*

Description

`plotNicheSpatial()` shows the cells of one niche in one image. `nicheCompositionMap()` shows the relative enrichment of each cell type in each niche as a bubble plot or heatmap. `nicheBarplot()` shows it for a single niche. `nicheProportionBoxplot()` compares a niche's per-image proportion between outcome groups.

Usage

```
plotNicheSpatial(spe, imageID, niche, nicheCol = "nicheWeighted", cellSize = 1)

nicheCompositionMap(
  spe,
  niche,
  cellType = "cellType",
  type = c("bubble", "heatmap"),
  limit = c(0.33, 3),
  ...
)

nicheBarplot(spe, niche, target, cellType = "cellType", limit = c(0.33, 3))

nicheProportionBoxplot(proportions, niche, spe, outcomeCol)
```

Arguments

spe	A `SummarizedExperiment` (e.g. a `SpatialExperiment`) with the cell type, image and coordinates of every cell in the form given by `getCellData()`.
imageID	The image to plot.
niche	For `plotNicheSpatial()` and the proportion boxplot, the niche to show. For `nicheCompositionMap()` and `nicheBarplot()`, the niche assignments: either a vector of niches named by cell ID, such as `niches$nicheWeighted` from `runNicheIdentification()`, or the name of a `colData` column holding them.
nicheCol	The niche assignments for `plotNicheSpatial()`, in either of the forms taken by `niche` in `nicheCompositionMap()`.
cellSize	Size of the highlighted cells in `plotNicheSpatial()`, the cells of the chosen niche. The grey background cells always have size 0.7.
cellType	Column of `colData(spe)` with cell type labels.
type	`"bubble"` or `"heatmap"`.
limit	Lower and upper limit of the relative enrichment.
...	Further arguments passed to `pheatmap::pheatmap()`.

target	The niche to plot in nicheBarplot().
proportions	The propWeighted or propUnweighted element of the results of testNicheOutcome(): the table you pass decides whether the weighted or the unweighted niches are plotted.
outcomeCol	Name of the per-image outcome column in colData(spe).

Value

A ggplot object; for nicheProportionBoxplot() a girafe (interactive) widget; for nicheCompositionMap(type = "heatmap") the result of pheatmap::pheatmap().

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
niches = runNicheIdentification(lungSubset, lungCellTypeTree, lungConfList,
    r = 50, k = 6)
plotNicheSpatial(lungSubset, "VUHD113", "niche1",
    nicheCol = niches$nicheWeighted)
nicheCompositionMap(lungSubset, niches$nicheWeighted)
nicheBarplot(lungSubset, niches$nicheWeighted, "niche1")
res = testNicheOutcome(lungSubset, niches$nicheUnweighted,
    niches$nicheWeighted, outcomeCol = "disease_status", subject = "subject")
if (requireNamespace("ggiraph", quietly = TRUE)) {
    nicheProportionBoxplot(res$propWeighted, "niche1", lungSubset,
        "disease_status")
    nicheProportionBoxplot(res$propUnweighted, "niche1", lungSubset,
        "disease_status")
}
```

---

plotMarkerSpatial *Spatial plots of marker expression within a group*

Description

Plots one image, with the cells of a group shaded by the (library-size normalised, rescaled) expression of one or two markers.

Usage

```
plotOneMarkerSpatial(
  spe,
  imageID,
  group,
  marker,
  colour = "purple",
  cellTypeTree = NULL,
```

```
  cellSize = 1,
  assayName = "counts"
)

plotTwoMarkerSpatial(
  spe,
  imageID,
  groupA,
  markerA,
  colourA = "purple",
  groupB,
  markerB,
  colourB = "green",
  cellTypeTree = NULL,
  cellSize = 1,
  assayName = "counts"
)
```

Arguments

spe	A SummarizedExperiment (e.g. a SpatialExperiment) with a cell ID, cell type, image and coordinates for every cell in the form given by getCellData(), and a counts assay.
imageID	The image to plot.
group, groupA, groupB	Group name(s) whose cells are shaded.
marker, markerA, markerB	Gene name(s), which must be rows of the assay.
colour, colourA, colourB	Colour(s) used for the shading.
cellTypeTree	A cell type tree, used to find the cells of each group. It is needed unless colData(spe) already has a group column.
cellSize	Size of the highlighted cells, the ones drawn in colour. The grey background cells always have size 0.7.
assayName	Name of the assay with the counts. Expression is normalised by each cell's library size.

Value

A ggplot object.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
plotOneMarkerSpatial(lungSubset, "VUHD113", "Endothelial lineage", "HEY1",
    cellTypeTree = lungCellTypeTree)
plotTwoMarkerSpatial(lungSubset, "VUHD113", "Endothelial lineage", "PECAM1",
```

```
    "purple", "Fibroblast lineage", "COL1A1", "green",
    cellTypeTree = lungCellTypeTree)
```

`plotMetricBoxplot` *Plot one pair's co-localisation by outcome*

Description

Shows, for one pair of cell types or groups, the per-image co-localisation measured with the original labels (unweighted) and with the bootstrap confidence weights (weighted), for each level of the outcome. Each point is one image.

Usage

```
plotMetricBoxplot(
  results,
  spe,
  fromType,
  toType,
  outcomeCol,
  metric = c("lfunction", "kontextual")
)
```

Arguments

`results`	Output of `runColocalisationModels()`.
`spe`	A `SummarizedExperiment` (e.g. a `SpatialExperiment`) with `imageID` and the outcome in `colData`.
`fromType, toType`	The pair to show. An error is raised if the pair is not in the results; relative co-localisation (`"kontextual"`) only has pairs whose `toType` has a parent other than itself.
`outcomeCol`	Name of the outcome column in `colData(spe)`.
`metric`	The measure to plot: `"lfunction"` (co-localisation, the cross-type L-function) or `"kontextual"` (co-localisation relative to a parent population, the Kontextual measure).

Value

An interactive plot, a `girafe` widget that needs the ggiraph package.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
res = runColocalisationModels(lungSubset, lungCellTypeTree, lungConfList,
    outcomeCol = "disease_status", subject = "subject")
if (requireNamespace("ggiraph", quietly = TRUE)) {
    plotMetricBoxplot(res, lungSubset, "Mast", "Proliferating FBs",
        outcomeCol = "disease_status")
    plotMetricBoxplot(res, lungSubset, "Mast", "Proliferating FBs",
        outcomeCol = "disease_status", metric = "kontextual")
}
```

---

plotProportionBoxplot *Plot one cell type's proportion by outcome*

Description

Shows, for one cell type (or group), each image's proportion measured with the original labels (unweighted) and with the bootstrap confidence weights (weighted), for each level of the outcome. For the results of `runProportionsPipeline()` the proportion is of all cells in the image; for the results of `runProportionsRelativeToParentPipeline()` it is of the cells of the parent group. Each point is one image.

Usage

```
plotProportionBoxplot(results, cellType, spe, outcomeCol)
```

Arguments

`results`	Output of `runProportionsPipeline()` or `runProportionsRelativeToParentPipeline()`.
`cellType`	The cell type to show: one of the `cellType` values in `results$proportions`. For `runProportionsPipeline()` results this can also be one of the bootstrapped groups.
`spe`	A `SummarizedExperiment` (e.g. a `SpatialExperiment`) with `imageID` and the outcome in `colData`.
`outcomeCol`	Name of the outcome column in `colData(spe)`.

Value

An interactive plot, a `girafe` widget that needs the ggiraph package.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
res = runProportionsPipeline(lungSubset, lungCellTypeTree, lungConfList,
    outcomeCol = "disease_status", subject = "subject")
rel = runProportionsRelativeToParentPipeline(lungSubset, lungCellTypeTree,
    lungConfList, outcomeCol = "disease_status", subject = "subject")
if (requireNamespace("ggiraph", quietly = TRUE)) {
    plotProportionBoxplot(res, "Arteriole", lungSubset, "disease_status")
    plotProportionBoxplot(rel, "Arteriole", lungSubset, "disease_status")
}
```

---

plotSpatial *Spatial plots of cell types and confidence scores*

Description

Plots the cells of one image, highlighting a cell type or group, or showing bootstrap confidence scores.

Usage

```
plotWeightedSpatialPair(
  spe,
  cellTypeTree,
  confList,
  imageID,
  groupA,
  subtypeA,
  groupB,
  subtypeB,
  encoding = c("alpha", "colour"),
  cellSize = 1
)

plotHardSpatialPair(
  spe,
  imageID,
  subtypeA,
  subtypeB,
  cellTypeTree = NULL,
  cellSize = 1
)

plotWeightedSpatial(
  spe,
```

```
  cellTypeTree,
  confList,
  imageID,
  targetGroup,
  targetSubtype,
  cellSize = 1
)

plotOriginalLabelSpatial(spe, cellTypeTree, imageID, targetGroup, cellSize = 1)
```

Arguments

spe	A SummarizedExperiment (e.g. a SpatialExperiment) with a cell ID, cell type, image and coordinates for every cell in the form given by getCellData() (cellID, cellType, imageID, x and y).
cellTypeTree	A cell type tree, see cellTypeTree. Optional for plotHardSpatialPair(), where it is only needed to use a group name as a label.
confList	Confidence list from bootstrapLineageConfidence().
imageID	The image to plot. An error is raised if it is not an image in the data.
groupA, groupB, targetGroup	Group names. An error is raised for a group that has no cells in the data.
subtypeA, subtypeB, targetSubtype	Cell types within the groups, which must have confidence scores in confList (for plotHardSpatialPair(), a cell type or a group name in the data).
encoding	How to show the confidence scores: "alpha" (the default), as transparency, or "colour", as a colour gradient. Either way each of the two cell types has its own legend.
cellSize	Size of the highlighted cells, the ones drawn in colour. The grey background cells always have size 0.7.

Value

A ggplot object.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
plotHardSpatialPair(lungSubset, "VUHD113", "Mast", "Proliferating FBs",
    lungCellTypeTree)
plotWeightedSpatial(lungSubset, lungCellTypeTree, lungConfList, "VUHD113",
    "Endothelial lineage", "Arteriole")
plotOriginalLabelSpatial(lungSubset, lungCellTypeTree, "VUHD113",
    "Endothelial lineage")
if (requireNamespace("ggnewscale", quietly = TRUE)) {
    plotWeightedSpatialPair(lungSubset, lungCellTypeTree, lungConfList,
        "VUHD113", "Granulocytes", "Mast", "Fibroblast lineage",
```

```
        "Proliferating FBs")
}
```

`runColocalisationModels` *Co-localisation analysis with soft cell type weights*

Description

Tests, for every pair of leaf cell types and groups, whether the co-localisation of the two differs between the two conditions of an outcome, once with the original labels and once with the bootstrap confidences as soft cell type weights. The co-localisation of each image is the cross-type L-function, computed by spicyR (`spicyR::spicy(method = "image")`), and relative co-localisation is Statial's Kontextual test (`spicyR::kontextualEngine()`).

Usage

```
runColocalisationModels(
  spe,
  cellTypeTree,
  confList,
  outcomeCol,
  r = 50,
  subject = NULL,
  cores = 1,
  parentOverrides = character(0),
  autoOverrideParents = TRUE,
  cellType = "cellType",
  cellID = "cellID",
  imageID = "imageID",
  x = "x",
  y = "y"
)
```

Arguments

`spe`	A `SummarizedExperiment` (e.g. `SpatialExperiment`) with a cell ID, cell type, image and coordinates for every cell (see `getCellData()`).
`cellTypeTree`	A cell type tree, see cellTypeTree.
`confList`	Confidence list from `bootstrapLineageConfidence()`.
`outcomeCol`	Name of the `colData(spe)` column with the per-image outcome. It must have exactly two levels and a single value per image.
`r`	Radius within which cells count as neighbours, in the units of the coordinates; a single positive number.
`subject`	Name of the `colData(spe)` column identifying the subject (for example the patient) each image comes from. With `NULL` every image is its own subject.

`cores` Number of threads.

`parentOverrides`
Named character vector mapping a `to` label to the ancestor node (a group, wrapper or lineage in `cellTypeTree`) that relative co-localisation should compare against instead of the label's own group, for example `c("Fibroblast lineage" = "Mesenchymal")`.

`autoOverrideParents`
Whether to add overrides found by `autoParentOverrides()`. Use `FALSE` to condition only on the parents you list in `parentOverrides`.

`cellType`, `cellID`, `imageID`
Names of the `colData(spe)` columns holding each cell's cell type label, a unique cell ID (matching the `cellID` column of `confList`), and the image it is in.

`x`, `y` Names of the columns holding the cell coordinates. For a `SpatialExperiment` whose coordinates are in `spatialCoords()`, the defaults `"x"` and `"y"` refer to its first two coordinates.

Details

Co-localisation of a pair `from -> to` in an image is the cross-type L-function: how strongly the `from` cells are attracted to (positive values) or avoid (negative values) the `to` cells within radius `r`, with 0 meaning no attraction or repulsion. The area of an image is that of the square around its cells, and cells near the border are edge corrected. Each pair is then tested by a linear model of its per-image values on the outcome (the images are the units); if some subjects have several images, spicyR fits a mixed model with the subject as a random effect instead.

With weights (see `confidenceWeights()`), a cell of a bootstrapped group belongs to each subtype of the group with its bootstrap confidence: each cell contributes to a label in proportion to its weight, and the density of a label is its summed weight per unit area. With the original labels (weights of 0 and 1) this is the unweighted L-function.

Relative co-localisation compares the co-localisation of `from` with `to` to that of `from` with a parent population of `to`, so that attraction that only arises because both cell types prefer the same region is not counted. By default the parent of a `to` cell type is its own group; `parentOverrides` and `autoOverrideParents` give broader parents. Pairs whose `to` has no parent other than itself are not tested.

A label is not paired with itself, nor with a group that contains it.

Value

A list with the tests `modelWeighted`, `modelUnweighted`, `modelRelativeWeighted` and `modelRelativeUnweighted` (one row per pair: `estimate`, the difference for the second level of the outcome compared with the first, its `se`, `statistic`, `p.value` and BH-adjusted `p.adj`, and for relative co-localisation the `parent` and `side`, `"attract"` or `"avoid"`), the per-image values `weighted`, `unweighted`, `relativeWeighted` and `relativeUnweighted` (columns `imageID`, `from`, `to`, `colocalisation`, and `parent` for the relative version), and the spicyR results themselves (`spicy`, a list of `weighted`, `unweighted`, `relativeWeighted` and `relativeUnweighted`), for spicyR's plotting functions.

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
res = runColocalisationModels(lungSubset, lungCellTypeTree, lungConfList,
    outcomeCol = "disease_status", subject = "subject")
head(res$modelWeighted)
```

`runProportionsPipeline` *Cell type proportion analysis*

Description

Computes per-image proportions of every cell type using original labels (unweighted) and bootstrap confidences (weighted), and tests each against an outcome with a linear model. `runProportionsRelativeToParentPipelir` instead expresses each subtype as a proportion of its parent group.

Usage

```
runProportionsPipeline(
  spe,
  cellTypeTree,
  confList,
  outcomeCol,
  skipGroups = character(0),
  subject = NULL,
  cellType = "cellType",
  cellID = "cellID",
  imageID = "imageID"
)

runProportionsRelativeToParentPipeline(
  spe,
  cellTypeTree,
  confList,
  outcomeCol,
  skipGroups = character(0),
  subject = NULL,
  cellType = "cellType",
  cellID = "cellID",
  imageID = "imageID"
)
```

Arguments

spe	A SummarizedExperiment (e.g. SpatialExperiment) with a cell ID, cell type and image for every cell in colData.
cellTypeTree	A cell type tree, see cellTypeTree.
confList	Confidence list from bootstrapLineageConfidence(): a named list with one data frame per group, whose names must be groups of cellTypeTree.
outcomeCol	Name of the per-image outcome column in colData(spe). It must have exactly two levels (subset your data to two conditions first) and a single value per image: repeated measures and mixed models are not supported. An error is raised otherwise, before any computation.
skipGroups	Names of groups of cellTypeTree to exclude, as a character vector (the default character(0) excludes none). An error is raised for a name that is not a group of the tree, so the same skipGroups can be used here and in bootstrapLineageConfidence().
subject	Name of the colData(spe) column identifying the subject (for example the patient) each image comes from, or NULL. If given, every image must belong to one subject and every subject must contribute only one image, as in runColocalisationModels(). With NULL this cannot be checked, and a message says so.
cellType, cellID, imageID	Names of the colData(spe) columns holding each cell's cell type label, a unique cell ID (matching the cellID column of confList), and the image it is in.

Details

In runProportionsPipeline() the unweighted results cover every cell type and also the whole groups that were bootstrapped. Only the subtypes of those groups have confidences, so for every other cell type (cell types without a group, and the group-level proportions) the weighted proportion is the unweighted one: their labels are not uncertain. The weighted and unweighted models therefore test the same set of cell types, and their adjusted p-values are comparable.

The lineage column of the results is the group a subtype belongs to (not the broad lineage). For cell types without a group and for the group-level proportions it is the cell type or group itself.

Value

A list with proportions, weighted, unweighted (long data.frames; only proportions and not weighted/unweighted for the relative-to-parent version), and modelWeighted and modelUnweighted (per cell type model results, where estimate is the difference for the second level of the outcome compared with the first).

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
res = runProportionsPipeline(lungSubset, lungCellTypeTree, lungConfList,
    outcomeCol = "disease_status", subject = "subject")
head(res$modelWeighted)
```

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
rel = runProportionsRelativeToParentPipeline(lungSubset, lungCellTypeTree,
    lungConfList, outcomeCol = "disease_status", subject = "subject")
head(rel$modelWeighted)
```

`testNicheOutcome` *Test niche proportions against an outcome*

Description

Computes the proportion of each image's cells in each niche and tests every niche against a two-level outcome with a Wilcoxon or t-test.

Usage

```
testNicheOutcome(
  spe,
  nicheUnweighted,
  nicheWeighted,
  outcomeCol,
  test = c("wilcox", "ttest"),
  subject = NULL,
  cellID = "cellID",
  imageID = "imageID"
)
```

Arguments

`spe`	A `SummarizedExperiment` (e.g. a `SpatialExperiment`) with a cell ID, an image and the outcome for every cell in `colData`.
`nicheUnweighted, nicheWeighted`	Named cell-to-niche vectors, as returned by `runNicheIdentification()`.
`outcomeCol`	Name of the per-image outcome column in `colData(spe)`. It must have exactly two levels (subset your data to two conditions first) and a single value per image: repeated measures and mixed models are not supported. An error is raised otherwise, before any computation.
`test`	`"wilcox"` or `"ttest"`.
`subject`	Name of the `colData(spe)` column identifying the subject (for example the patient) each image comes from, or `NULL`. If given, every image must belong to one subject and every subject must contribute only one image, as in `runColocalisationModels()`. With `NULL` this cannot be checked, and a message says so.
`cellID, imageID`	Names of the `colData(spe)` columns holding a unique cell ID (matching the names of the niche vectors) and the image each cell is in.

Details

The `estimate` is the difference in the niche proportion for the second level of the outcome compared with the first (alphabetical order for text, factor order for factors), for both tests, as in `runColocalisationModels()`. For `test = "wilcox"` this is the Hodges-Lehmann estimate of the shift in location. Each image is one observation, so every image must come from a different subject if `subject` is given.

Value

A list with `testUnweighted`, `testWeighted` (one row per niche) and `propUnweighted`, `propWeighted` (per image niche proportions, to pass to `nicheProportionBoxplot()`).

Examples

```
data(lungSubset)
data(lungCellTypeTree)
data(lungConfList)
niches = runNicheIdentification(lungSubset, lungCellTypeTree, lungConfList,
    r = 50, k = 6)
res = testNicheOutcome(lungSubset, niches$nicheUnweighted,
    niches$nicheWeighted, outcomeCol = "disease_status", subject = "subject")
res$testWeighted
```

Index

∗ **datasets**
heartCellTypeTree, 11
lungCellTypeTree, 12
lungConfList, 13
lungSubset, 13

annotateCellTypeTree *(*cellTypeTree*)*, 6
autoParentOverrides *(*cellTypeTree*)*, 6
autoParentOverrides(), *24*

bootstrapClassify, 2
bootstrapClassify(), *4*, *10*
bootstrapLineageConfidence, 4
bootstrapLineageConfidence(), *6*, *8–10*, *13*, *15*, *22*, *23*, *26*
buildCellTypeMap *(*cellTypeTree*)*, 6

cellTypeTree, *4*, *5*, 6, *8*, *9*, *11*, *12*, *15*, *22*, *23*, *26*
confidenceWeights, 8
confidenceWeights(), *15*, *24*
crossfitLineageProbabilities, 9

getCellData, 10
getCellData(), *8*, *15*, *16*, *18*, *22*, *23*

heartCellTypeTree, *6*, 11, *12*

loadAmancherla, 11
loadAmancherla(), *11*
lungCellTypeTree, 12, *13*
lungConfList, 13
lungSubset, *12*, 13, *13*

nicheBarplot *(*nichePlots*)*, 16
nicheCompositionMap *(*nichePlots*)*, 16
nicheIdentification, 14
nichePlots, 16
nicheProportionBoxplot *(*nichePlots*)*, 16
nicheProportionBoxplot(), *28*

pheatmap::pheatmap(), *16*, *17*
plotHardSpatialPair *(*plotSpatial*)*, 21
plotMarkerSpatial, 17
plotMetricBoxplot, 19
plotNicheSpatial *(*nichePlots*)*, 16
plotOneMarkerSpatial *(*plotMarkerSpatial*)*, 17
plotOriginalLabelSpatial *(*plotSpatial*)*, 21
plotProportionBoxplot, 20
plotSpatial, 21
plotTwoMarkerSpatial *(*plotMarkerSpatial*)*, 17
plotWeightedSpatial *(*plotSpatial*)*, 21
plotWeightedSpatialPair *(*plotSpatial*)*, 21

resolveTreeLeaves *(*cellTypeTree*)*, 6
runColocalisationModels, 23
runColocalisationModels(), *4*, *7*, *11*, *19*, *26–28*
runNicheIdentification *(*nicheIdentification*)*, 14
runNicheIdentification(), *16*, *27*
runProportionsPipeline, 25
runProportionsPipeline(), *4*, *20*
runProportionsRelativeToParentPipeline *(*runProportionsPipeline*)*, 25
runProportionsRelativeToParentPipeline(), *20*

testNicheOutcome, 27
testNicheOutcome(), *17*