Package: spiDE (via r-universe)

November 11, 2026

Title Context-specific, neighbourhood-dependent differential expression

Version 0.99.38

Description spiDE identifies context-specific, neighbourhood-dependent differential expression in spatial transcriptomics data. Within an index cell type, it asks whether a gene's expression changes with the local density (niche) of another cell type within patients, and whether that dependence differs between two conditions. Niche covariates are Gaussian kernel density estimates of each cell type. For every index cell type a per-gene negative binomial GLM with a patient intercept is fitted with SpaNorm's Newton solver; its niche slopes are then tested across patients, either from each patient's own slopes with weighted limma (the slopes engine) or with a patient-clustered CR2 sandwich (the sandwich engine). The pooled test asks whether a slope is consistently non-zero; the condition-specific test whether it differs between conditions.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

biocViews Software, Spatial, DifferentialExpression, GeneExpression, Transcriptomics, Regression, CellBiology

Depends R (>= 4.5.0)

Imports ggplot2, Matrix, scales, SpaNorm (>= 1.7.15), SpatialExperiment, SingleCellExperiment, SummarizedExperiment, S4Vectors, spatstat.geom, spatstat.explore, BiocParallel, limma, methods, splines, stats, utils

Suggests testthat (>= 3.1.2), knitr, rmarkdown, BiocStyle, ggrepel, patchwork, RhpcBLASctl, clubSandwich

URL https://bhuvalab.github.io/spiDE, https://github.com/BhuvaLab/spiDE

BugReports https://github.com/BhuvaLab/spiDE/issues

VignetteBuilder knitr

Config/testthat/edition 3

LazyData false

Config/Needs/website ggplot2, patchwork, bslib

Config/pak/sysreqs
libglpk-dev libmagick++-dev gsfonts libicu-dev libxml2-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-11-11 00:30:10 UTC

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

RemoteRef HEAD

RemoteSha caa61fdeeb6015de53edfafcae9e171f46e49945

Contents

buildNiches . . . 3
fitSpiDE . . . 4
mergeNiches . . . 7
patientIntercepts . . . 8
patientSlopes . . . 9
plotCallMap . . . 10
plotNicheMap . . . 11
plotNicheResponse . . . 12
plotPatientEffects . . . 13
plotPatientSlopes . . . 15
plotPValues . . . 16
plotSpillover . . . 16
plotTopTriplets . . . 17
plotTripletHeatmap . . . 19
results . . . 20
spiDE . . . 21
spiDEColours . . . 22
SpiDEFit-class . . . 23
SpiDEResults-class . . . 24
spiGSEA . . . 25
testNicheAbundance . . . 26
testSpiDE . . . 29
theme_spiDE . . . 31
toySpiDE . . . 31

Index **33**

buildNiches *Build niche covariate matrices*

Description

Computes, for each biological sample and each bandwidth `sigma`, a Gaussian kernel density estimate of every cell type evaluated at each cell's location. The resulting cells x cellTypes matrices are stored as `reducedDim(spe, "Niche<sigma>")`, one per bandwidth, with a consistent set of cell-type columns (cell types absent from a sample are set to 0).

Usage

```
buildNiches(spe, ...)

## S4 method for signature 'ANY'
buildNiches(
  spe,
  sigma = c(10, 30, 50, 70),
  cell_type = "cell_type",
  sample_id = "sample_id",
  edge = TRUE,
  diggle = TRUE,
  name = "Niche",
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`spe`	a SpatialExperiment with spatial coordinates, cell type labels, and sample identifiers.
`...`	ignored.
`sigma`	a numeric vector of kernel bandwidths (in the units of `spatialCoords`), default `c(10, 30, 50, 70)`.
`cell_type`	a character, the colData column holding cell type labels.
`sample_id`	a character, the colData column identifying samples.
`edge, diggle`	logicals passed to `spatstat.explore::densityfun` for edge correction (default TRUE).
`name`	a character, the prefix for the stored reducedDims (default "Niche").
`BPPARAM`	a BiocParallelParam for parallelising over samples.

Value

the input `spe` with one `reducedDim` per bandwidth added.

Examples

```
data(toySpiDE)
spe <- toySpiDE
spe <- buildNiches(spe, sigma = c(10, 20))
SingleCellExperiment::reducedDimNames(spe)
```

`fitSpiDE` *Fit spiDE's per-patient niche models*

Description

Fits, for every index cell type, one negative binomial GLM per gene of its expression on the local density of every other (niche) cell type, with an intercept per patient so that each niche effect is a within-patient slope. The slopes are then tested across patients by `testSpiDE()`, pooled or between conditions.

Usage

```
fitSpiDE(spe, condition = NULL, ...)

## S4 method for signature 'SpatialExperiment'
fitSpiDE(
  spe,
  condition = NULL,
  engine = c("slopes", "sandwich"),
  index = NULL,
  niche = NULL,
  sigma = NULL,
  cell_type = "cell_type",
  sample_id = "sample_id",
  section = NULL,
  covariates = character(),
  depth = c("loglib", "nonlinear", "spatial_spline", "none"),
  offset = NULL,
  strata = NULL,
  assay = "counts",
  name = "Niche",
  min.cells = 10L,
  min.patients = 6L,
  min.detect = 0.1,
  genes = NULL,
  tile = NULL,
  min.tiles = 5L,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = TRUE,
  ...
)
```

Arguments

spe	a SpatialExperiment with raw counts and a niche reducedDim from buildNiches().
condition	NULL or a character, the colData column holding a two-level, patient-level condition. The slopes engine only records it (it can also be given to testSpiDE()); the sandwich engine fits the condition-specific model only when it is given.
...	unused.
engine	"slopes" or "sandwich".
index	NULL (every cell type with at least min.cells cells in at least min.patients patients) or a character vector of index cell types.
niche	NULL (test every niche column) or a character vector of niche columns to test; the others stay in the model as adjustments. An index type is never tested against its own niche (see mergeNiches()).
sigma	the niche bandwidth (needed only if spe carries several).
cell_type, sample_id	the colData columns of the cell type and of the patient (the unit that is replicated: one sample per patient).
section	NULL (= sample_id) or the colData column of the tissue section, used to lay the spatial tiles of the variance estimate.
covariates	a character vector of cell-level colData columns to adjust for (centred; patient-level ones are absorbed by the patient intercepts).
depth	how sequencing depth enters: "loglib" (default, the centred log library size as a covariate with a slope per gene), "nonlinear" (a natural cubic spline of it with 3 df, knots at the index type's quantiles, so the depth response of each gene may bend), "spatial_spline" (a library-size spline within each patient: per section, log library size and its product with a smooth tensor spline of position, absorbed with the patient's intercept, so each gene's depth effect may vary over the section as in SpaNorm's $\log LS \cdot h_g(x, y)$ term, fitted jointly with the niche terms) or "none" (e.g. when offset carries depth).
offset	NULL or the name of an assay holding a log-scale offset per gene and cell (added to every linear predictor with its coefficient fixed at 1), e.g. the library-size component of a SpaNorm fit. Held dense per index type (genes x index cells).
strata	NULL or a patient-level colData column (e.g. slide or batch) whose niche slopes are adjusted for in the condition test. Needed when the condition is confounded with slide; used by the sandwich engine's condition model here and by the slopes engine in testSpiDE(). The pooled test never uses it: its slope is the average over all patients.
assay	the counts assay (raw integer counts).
name	the niche reducedDim prefix.
min.cells	minimum cells of an index type a patient must contribute.
min.patients	minimum patients an index type needs.
min.detect	minimum fraction of an index type's cells in which a gene is detected for it to be tested in that type.

genes	NULL (every gene) or a character vector of the genes to test. Library size is always computed from every row of the assay, so a panel can be tested in part without changing each cell's depth.
tile	side of the square tiles of the spatial variance estimate, in coordinate units (default 3 x the bandwidth).
min.tiles	minimum tiles a patient needs for its own spatial variance.
BPPARAM	a BiocParallelParam; genes are fitted in parallel.
verbose	report progress.

Details

Two engines estimate the between-patient error the tests need:

"slopes" **(default)** fits the shared model once, without the condition, then gives every patient a one-step NB estimate of its own niche slopes with a within-patient spatial sandwich variance. testSpiDE() combines them across patients with weighted limma. It is the more powerful engine when effects vary between patients, and its fit never sees the condition, so any condition can be tested later.

"sandwich" fits the condition-specific model directly (the condition x niche columns) and tests its coefficients with a patient-clustered CR2 sandwich and Bell-McCaffrey degrees of freedom. It is simpler, and strongest for effects shared by all patients, but needs the condition at fit time.

The slopes engine passes every calibration null of five comparisons in four spatial cohorts, the sandwich engine all but one borderline tail; see the calibration vignette.

Value

a SpiDEFit.

See Also

testSpiDE(), spiDE(), testNicheAbundance()

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
fit <- fitSpiDE(spe, index = "A", sigma = 30)
fit
```

`mergeNiches` *Merge niche cell-type columns into groups*

Description

Sums the density columns of grouped cell types in each niche `reducedDim`, producing coarser niche covariates (e.g. merging Macrophage subtypes into a single "Macrophage" niche). Cell types absent from `groups` are retained as their own (singleton) niche.

Usage

```
mergeNiches(spe, groups, ...)

## S4 method for signature 'ANY'
mergeNiches(spe, groups, sigma = NULL, name = "Niche", ...)
```

Arguments

`spe`	a SpatialExperiment with niche reducedDims (see `buildNiches()`).
`groups`	a named list mapping each merged niche name to a character vector of cell-type column names to sum.
`...`	ignored.
`sigma`	a numeric vector of bandwidths to update; `NULL` (default) updates every `name<sigma>` reducedDim present.
`name`	a character, the reducedDim name prefix (default "Niche").

Details

The group membership is recorded in `metadata(spe)$spiDE_niche_groups` (per niche reducedDim) so that a subsequent `fitSpiDE()` never tests an index cell type against a merged niche it is a member of (its own density would otherwise contaminate the niche).

Value

the input `spe` with merged niche reducedDims.

Examples

```
data(toySpiDE)
spe <- toySpiDE
spe <- buildNiches(spe, sigma = 30)
spe <- mergeNiches(spe, groups = list(AC = c("A", "C")), sigma = 30)
colnames(SingleCellExperiment::reducedDim(spe, "Niche30"))
```

`patientIntercepts` *Each patient's intercept*

Description

Every index type's shared fit has an intercept per patient and gene: the patient's expression level of the gene in that type, at zero niche density and a reference depth: the index type's mean log library size under `depth = "loglib"` or `"nonlinear"`, each patient's own mean under `"spatial_spline"`. The intercepts absorb every difference between patients – the condition's main effect, composition, batch – which is why the niche slopes are within-patient slopes. `plotPatientEffects()` shows what they capture. Kept by `fitSpiDE()` from spiDE 0.99.34 on, for both engines.

Usage

```
patientIntercepts(object, ...)

## S4 method for signature 'SpiDEFit'
patientIntercepts(object, gene = NULL, index = NULL, ...)

## S4 method for signature 'SpiDEResults'
patientIntercepts(object, ...)
```

Arguments

`object`	a SpiDEFit or SpiDEResults.
`...`	unused.
`gene, index`	optional character vectors to subset.

Value

a data.frame: `gene`, `index`, `patient`, `intercept` (log scale), `ncells` (the patient's cells of the index type), plus the patient-level colData columns recorded by `fitSpiDE()`.

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
fit <- fitSpiDE(spe, index = "A", sigma = 30, verbose = FALSE)
head(patientIntercepts(fit, gene = "G1"))
```

patientSlopes *Each patient's niche slopes*

Description

The slopes engine estimates every patient's own niche slopes (a one-step negative binomial estimate from the shared fit); this returns them in long form, e.g. to plot how a gene's slope varies between patients and conditions.

Usage

```
patientSlopes(object, ...)

## S4 method for signature 'SpiDEFit'
patientSlopes(object, gene = NULL, index = NULL, niche = NULL, ...)

## S4 method for signature 'SpiDEResults'
patientSlopes(object, ...)
```

Arguments

`object`	a SpiDEFit or SpiDEResults from the slopes engine.
`...`	unused.
`gene`, `index`, `niche`	optional character vectors to subset.

Value

a data.frame: `gene`, `index`, `niche`, `patient`, `slope`, `var_spatial` (the within-patient spatial sandwich variance), `var_model`, `ncells`, plus the patient-level colData columns recorded by `fitSpiDE()`.

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
fit <- fitSpiDE(spe, index = "A", sigma = 30)
head(patientSlopes(fit, gene = "G1"))
```

`plotCallMap` *Where the calls are*

Description

The number of genes called for every (index, niche) pair, split by direction: for the pooled test, a slope that rises or falls with the niche's density; for the condition-specific test, a slope higher in one condition than in the other. The heatmap gives exact counts and marks untested pairs (an index type against its own niche) in grey; the graph draws cell types on a fixed circle with an arrow from niche to index type, its width the number of genes – it reads best when calls are few (the condition test), the heatmap when every pair has some.

Usage

```
plotCallMap(
  x,
  test = NULL,
  fdr = x@fdr,
  style = c("heatmap", "graph"),
  min.calls = 1L
)
```

Arguments

`x`	a SpiDEResults.
`test`	`NULL` (the condition-specific test if run, else the pooled test), `"pooled"` or `"condition"`.
`fdr`	the q-value threshold of a call.
`style`	`"heatmap"` or `"graph"`.
`min.calls`	the graph's smallest drawn edge.

Details

Read it for which cell-type pairs carry the signal, and for pairs whose calls all go one way toward one niche type (the shape segmentation spillover gives; see `plotSpillover()`). An arrow is an association, not signalling.

Value

a ggplot. For the heatmap, `p$data` has one row per (index, niche, direction): `calls` and `tested`.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, verbose = FALSE)
plotCallMap(res, test = "pooled")
plotCallMap(res, style = "graph")
```

plotNicheMap *The niche covariate in tissue*

Description

One sample's cells, coloured by the log1p density of a niche type: the covariate spiDE regresses on. With `index`, only the index type's cells are coloured – the covariate as the index type's model reads it – and the others are dim. The panel is dark (the sticker's ink) so that the lowest densities stay visible; the legend, titles and axes keep the page's style.

Usage

```
plotNicheMap(
  spe,
  niche,
  sample,
  index = NULL,
  sigma = NULL,
  cell_type = "cell_type",
  sample_id = "sample_id",
  section = NULL,
  name = "Niche",
  point.size = NULL,
  scale.bar = TRUE,
  unit = NULL
)
```

Arguments

`spe`	a SpatialExperiment with niche reducedDims (`buildNiches()`).
`niche`	a niche column (a cell type or a `mergeNiches()` group).
`sample`	the sample (patient) to show.
`index`	`NULL` or an index cell type to highlight.
`sigma`	the bandwidth (needed only if `spe` carries several).
`cell_type, sample_id`	the colData columns of cell type and sample.
`section`	`NULL` or the colData column of the section; a sample with several sections gets one panel each.
`name`	the niche reducedDim prefix.
`point.size`	the point size (default: from the number of cells).
`scale.bar`	draw a scale bar of about a fifth of the width.
`unit`	the coordinates' unit, for the scale bar's label (e.g. `"µm"`).

Details

Read it to judge the bandwidth: a speckled field that follows single cells is too small; a field nearly flat within a section is too large (the patient intercept then absorbs it). It also shows where the index cells sit relative to the niche.

Value

a ggplot; `p$data` has one row per cell of the sample: `x`, `y` (from each section's own origin), `cell_type`, `density` (log1p), `section` and `focal` (coloured).

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
plotNicheMap(spe, niche = "B", sample = "S1", index = "A")
```

---

plotNicheResponse *Expression against niche density: the cells behind a slope*

Description

The index type's cells' expression of each gene (counts per 10,000, log axis) against the log1p density of the niche type, binned, one panel per gene.

Usage

```
plotNicheResponse(
  spe,
  x,
  gene,
  index,
  niche,
  adjust = TRUE,
  bins = 8L,
  assay = "counts",
  name = "Niche"
)
```

Arguments

`spe`	the SpatialExperiment the fit was made from.
`x`	a SpiDEResults or SpiDEFit; `adjust = TRUE` needs one from spiDE >= 0.99.34. Under the sandwich engine every patient's and group's line has the pooled slope.
`gene`	one or more genes, one panel each.
`index, niche`	the index and niche cell types.
`adjust`	divide out each patient's intercept (see Details).

bins	the number of density bins (adjust = TRUE; bins are quantiles over all patients). Raw data use fifths of each patient's own density range.
assay	the counts assay.
name	the niche reducedDim prefix.

Details

With adjust = TRUE (the default) each cell's expression is divided by its patient's fitted intercept, so the patients share one reference level and overlay: faint points are each patient's mean in a density bin, points and bars the mean across patients and its 95\ the dashed line each group's fitted slope. This is what the tests compare. With adjust = FALSE the data are raw: each patient's own slope is a segment through its mean (open circle), and the dashed line through the patients' means is the between-patient association that the intercepts absorb – testNicheAbundance()'s question, not a niche slope.

Read it for the data behind a slope, whether the log1p scale is roughly linear, and whether a few extreme cells drive it. The display normalises depth by library size (the model fits a slope on it). Every layer carries patient, so + facet_wrap(~ patient) splits it by patient.

Value

a ggplot; p$data has one row per (gene, patient, bin): gene, patient, bin, expr (mean CP10k), density (mean log1p density), condition and gene_lab. Bins without counts are left out of that layer (not out of the means).

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
res <- spiDE(spe, condition = "condition", sigma = 30, index = "A", verbose = FALSE)
plotNicheResponse(spe, res, gene = "G1", index = "A", niche = "B")
plotNicheResponse(spe, res, gene = "G1", index = "A", niche = "B", adjust = FALSE)
```

---

plotPatientEffects *What the patient intercepts, or slopes, capture*

Description

Every index type's shared fit gives each patient an intercept per gene (see patientIntercepts()): they absorb every difference between patients – the condition's main effect, composition, batch – which is what makes the niche slopes within-patient slopes. This relates the principal components of the patients' gene-centred intercepts to patient covariates: every patient-level colData column, plus the patients' index cells (log10), mean log library size, and mean log1p density of each tested niche type. Each tile is the adjusted R^2 of a component on a covariate.

Usage

```
plotPatientEffects(
  x,
  index,
  niche = NULL,
  covariates = NULL,
  type = c("association", "pca"),
  colour.by = NULL,
  n.pcs = 5L,
  ntop = 500L
)
```

Arguments

x	a SpiDEResults or SpiDEFit (spiDE >= 0.99.34).
index	the index cell type.
niche	NULL (the intercepts) or a niche type (its slopes; slopes engine).
covariates	NULL (all) or the covariates to show, by the names in the plot.
type	"association" (the R^2 tiles) or "pca" (the first two components, coloured by colour.by).
colour.by	for type = "pca": a covariate (default the condition, else the first).
n.pcs	the number of components shown.
ntop	the number of genes, most expressed first, whose intercepts enter the components (a gene with any non-finite value is left out).

Details

Structure in the intercepts is harmless to the tests: it is absorbed by design. With niche, the same analysis runs on the patients' slopes on that niche (each gene's slopes scaled by its between-patient spread, as in spiGSEA()): a slide or batch that organises the slopes can reach the condition test, and belongs in strata.

Value

a ggplot. For "association", p$data has pc, covariate, r2 and group (patient or derived); for "pca", patient, PC1, PC2 and the covariate.

See Also

patientIntercepts(), testNicheAbundance() for the composition association gene by gene.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, index = "A", verbose = FALSE)
plotPatientEffects(res, index = "A")
plotPatientEffects(res, index = "A", niche = "B")
```

`plotPatientSlopes` *Each patient's niche slope for a triplet*

Description

The evidence the tests weigh: every patient's own slope of the gene's expression in the index type on the niche type's density (log1p), with an interval from its within-patient variance – the variance the tests use – and a point sized by the patient's weight in them. The pooled estimate and its confidence interval are the grey band. With a condition the patients are split by it, each group's weighted mean is a coloured line, and each strip gives the condition-specific slope difference and its q-value.

Usage

```
plotPatientSlopes(x, gene, index, niche, ci = 0.95)
```

Arguments

`x`	a SpiDEResults from the slopes engine.
`gene`	one or more genes, one column each.
`index`, `niche`	the index and niche cell types.
`ci`	the confidence level of the intervals.

Details

Read it for whether a call rests on patients that agree or on one or two heavily weighted ones, and for why a condition test is missing (each group needs three effective patients). Patients are ordered by group, then by their mean slope over the genes shown, so a row is one patient across genes.

Value

a ggplot. `p$data` has one row per (gene, patient) with a slope: `gene`, `patient`, `slope`, `se` (within patient), `weight` (the patient's share of the pooled test's weights), `condition` and `gene_lab` (the strip label, plotmath). Change it like any ggplot, e.g. `+ theme(axis.text.y = element_blank())`.

See Also

`patientSlopes()` for the numbers, `plotNicheResponse()` for the cells.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, index = "A", verbose = FALSE)
plotPatientSlopes(res, gene = c("G1", "G2"), index = "A", niche = "B")
```

plotPValues *p-values by test and expression band*

Description

Histograms of every triplet's p-value, one row per test and one column per fifth of each index type's genes by mean expression, with the level of a uniform distribution dashed. A spike at 0 on a flat floor is signal. A floor that rises toward 0 only in the brightest genes, or a hump at 1 (low power, conservative p-values), deserves a look at the calibration vignette. This is a sanity check, not a calibration test: real signal also moves p-values toward 0, and calibration needs data where there is nothing to find.

Usage

```
plotPValues(x, bands = 5L)
```

Arguments

x	a SpiDEResults.
bands	the number of expression bands.

Value

a ggplot; `p$data` has one row per tested triplet with `p`, `test` and `band`.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, procedure = "all", verbose = FALSE)
plotPValues(res)
```

plotSpillover *Are the calls the niche type's own markers?*

Description

Every tested triplet, placed by how much higher the gene is in the niche type than in the index type (log2, from each cell type's mean counts per 10,000) against the signed z of its test; calls are coloured by direction. A gene several-fold higher in the niche type whose measured level in the index cells rises with that niche's density is what transcripts leaking across segmentation boundaries would produce: the association is real in the data, but it needs orthogonal evidence (nuclear counts, a spillover-corrected segmentation) before it is read as biology.

Usage

```
plotSpillover(
  x,
  spe,
  test = "pooled",
  fold = 4,
  label = 6L,
  fdr = x@fdr,
  assay = "counts",
  name = "Niche"
)
```

Arguments

x	a SpiDEResults.
spe	the SpatialExperiment the results came from.
test	"pooled" or "condition".
fold	the niche-marker threshold (shaded).
label	the number of markers labelled (strongest first).
fdr	the q-value threshold of a call.
assay, name	the counts assay and the niche reducedDim prefix.

Value

a ggplot; p$data is the test's rows of the results table with fold (log2), z and status.

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
res <- spiDE(spe, condition = "condition", sigma = 30, verbose = FALSE)
plotSpillover(res, spe)
```

---

plotTopTriplets *The strongest triplets, or those of chosen genes*

Description

A forest plot: each triplet's estimate (the pooled slope, or the slope difference between conditions) with its 95\ where it is not called. gene, index and niche filter the table before the n smallest p-values are taken, so gene = c("CD74", "HLA-DRA") shows those genes' triplets. With spe, a hollow triangle marks a gene at least fold-fold higher in the niche type than in the index type (see plotSpillover()).

Usage

```
plotTopTriplets(
  x,
  test = NULL,
  n = 30L,
  gene = NULL,
  index = NULL,
  niche = NULL,
  spe = NULL,
  fold = 4,
  fdr = x@fdr,
  assay = "counts",
  name = "Niche"
)
```

Arguments

x	a SpiDEResults.
test	NULL (the condition-specific test if run, else the pooled test), "pooled" or "condition".
n	the number of triplets.
gene, index, niche	optional filters.
spe	NULL or the SpatialExperiment, for the niche-marker flag.
fold	the niche-marker threshold.
fdr	the q-value threshold of a call.
assay, name	the counts assay and the niche reducedDim prefix.

Value

a ggplot; p$data is the plotted rows of the results table with lo, hi, called, direction, marker and label (plotmath).

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
res <- spiDE(spe, condition = "condition", sigma = 30, verbose = FALSE)
plotTopTriplets(res, test = "pooled", n = 15, spe = spe)
plotTopTriplets(res, gene = "G1")
```

`plotTripletHeatmap` *Niche-response profiles of genes or gene sets*

Description

Features (genes, or gene sets from `spiGSEA()`) against every (index, niche) column: the fill is the test's t (gold: the slope rises with the niche's density, or is higher in the second condition; violet: the reverse), a dot marks a call, grey a feature not tested in that index type. Rows are clustered on their t profiles. Read it for a gene's or pathway's response across all cell-type pairs, and for pairs that share responses.

Usage

```
plotTripletHeatmap(
  x,
  features = NULL,
  test = NULL,
  n = 30L,
  fdr = NULL,
  limit = NULL,
  cluster = TRUE
)
```

Arguments

`x`	a SpiDEResults (genes) or the table `spiGSEA()` returns (sets; calls by `q.global`).
`features`	`NULL` (the `n` features with the smallest p-value in any column) or the features to show.
`test`	`NULL` (the condition-specific test if present, else the pooled test), `"pooled"` or `"condition"`.
`n`	the number of features when `features` is `NULL`.
`fdr`	the q-value threshold of a call (default: the results', or 0.05 for sets).
`limit`	the |t| at which the fill saturates (default: the 98th percentile).
`cluster`	order rows by hierarchical clustering of their t profiles.

Value

a ggplot; `p$data` has the plotted rows with `feature` and `called`.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, verbose = FALSE)
plotTripletHeatmap(res, test = "pooled", n = 10)
```

`results` *Extract spiDE's test results*

Description

Extract spiDE's test results

Usage

```
results(object, ...)

## S4 method for signature 'SpiDEResults'
results(object, test = NULL, fdr = NULL, ...)
```

Arguments

`object`	a SpiDEResults.
`...`	unused.
`test`	`NULL` (the condition-specific test if one was run, else the pooled test), `"condition"`, `"pooled"` or `"both"`.
`fdr`	`NULL` (every tested triplet) or a level: keep rows with `q <= fdr`.

Value

a data.frame, one row per (gene, index, niche, test), ordered by p-value: `estimate` (the pooled slope, or the difference in slope between conditions, on the log scale per unit log1p niche density), `se`, `t`, `df`, `p`, `q` (BH within the test's family), `in_family` and, for the slopes engine, `p.heterogeneity` (the label-free test that the triplet's slopes vary between patients, the filter of `procedure = "heterogeneity"`). Condition-specific rows outside the family have `q = NA`.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, index = "A",
             procedure = "all")
results(res, test = "pooled")
```

spiDE *Run spiDE end to end*

Description

Builds the niche covariates at bandwidth `sigma` (unless `spe` already carries them), fits the per-patient niche models (`fitSpiDE()`) and tests them (`testSpiDE()`): the pooled niche test always, and the condition-specific test when a `condition` is given.

Usage

```
spiDE(spe, condition = NULL, ...)

## S4 method for signature 'SpatialExperiment'
spiDE(
  spe,
  condition = NULL,
  sigma,
  engine = c("slopes", "sandwich"),
  procedure = c("pooled_or_heterogeneity", "heterogeneity", "filtered", "all"),
  fdr = 0.05,
  cell_type = "cell_type",
  sample_id = "sample_id",
  name = "Niche",
  BPPARAM = BiocParallel::SerialParam(),
  verbose = TRUE,
  ...
)
```

Arguments

`spe`	a SpatialExperiment with raw counts.
`condition`	`NULL` or the colData column of a two-level, patient-level condition.
`...`	further arguments to `fitSpiDE()`.
`sigma`	the niche bandwidth.
`engine`	`"slopes"` or `"sandwich"`, see `fitSpiDE()`.
`procedure, fdr`	passed to `testSpiDE()`; `procedure` defaults to the engine's default there.
`cell_type, sample_id`	the colData columns of cell type and patient.
`name`	the niche reducedDim prefix: the niches are read from, or built into, `reducedDim(spe, paste0(name, sigma))`.
`BPPARAM`	a BiocParallelParam.
`verbose`	report progress.

Value

a SpiDEResults.

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, index = "A",
             procedure = "all")
head(results(res))
```

`spiDEColours` *The spiDE colours*

Description

spiDE gives each role one colour and uses it in every plot: gold and violet for a slope that rises or falls with niche density (and for the diverging t scale), dark violet and pink for the first and second level of the condition, the viridis "magma" ramp for magnitudes (light to dark on the page), a brighter part of the same ramp (dark to bright) for plots in tissue coordinates, drawn on a dark panel, and the Okabe-Ito colours for a nuisance factor such as slide. The colours were checked under simulated deuteranopia, protanopia and tritanopia. Use them to make your own figures match, or pass others to a ggplot2 scale to override a spiDE plot's.

Usage

```
spiDEColours(role = NULL, n = 9L)
```

Arguments

`role`	`NULL` (the named fixed colours) or one of `"direction"`, `"condition"`, `"magnitude"` (the light-panel ramp, light to dark), `"spatial"` (the dark-panel ramp of tissue plots, dark to bright), `"diverging"` and `"nuisance"`.
`n`	the number of colours of a ramp, or of nuisance levels.

Value

a character vector of colours.

See Also

`theme_spiDE()`

Examples

```
spiDEColours("condition")
spiDEColours("magnitude", 5)
spiDEColours("spatial", 5)
```

SpiDEFit-class *The fitted per-patient niche models of spiDE*

Description

Returned by `fitSpiDE()`; tested with `testSpiDE()`.

Usage

```
## S4 method for signature 'SpiDEFit'
x$name
```

Arguments

`x`	a SpiDEFit or SpiDEResults.
`name`	a slot name.

Details

For the slopes engine each element of `index` holds `genes`, `niches` (tested niche columns), `patients`, `beta` (genes x patients x niches: each patient's slope), `v_tile` (the within-patient spatial sandwich variance of each slope), `v_model` (the model variance), `factor` (patients x niches: the per-patient variance factor), `pooled` (genes x niches: the shared fit's slope), `mean_expr` and `ncells`. For the sandwich engine it holds `genes`, `niches` and a data.frame `coef` of the tested coefficients (`test` = "pooled" or "condition") with their CR2 standard errors and Bell-McCaffrey degrees of freedom. Both hold `status` (genes fitted; genes left out because the solver could not fit them or their coefficients ran off, the latter also counted as `runaway`; and genes whose dispersion is on its search bound) and, under `depth = "spatial_spline"`, `depth_r2` (patients x niches: the R^2 of each niche column on the patient's library-size spline, a diagnostic of how much of the niche covariate the spline could absorb). From spiDE 0.99.34 both also hold `intercept` (genes x patients: each patient's intercept in the shared, condition-free fit; see `patientIntercepts()`), `niche_mean` (patients x niches: the mean log1p density of each tested niche over the patient's index cells), `loglib_mean` (the patients' mean log library size) and `ncells` (the patients' index cells).

Value

An object of class `SpiDEFit`, created by `fitSpiDE()`; its `$` accessor returns a slot, and `show()` prints a summary.

Slots

`engine` a character, `"slopes"` or `"sandwich"`.

`sigma` a numeric, the niche bandwidth the fit used.

`condition` a character (length 0 or 1), the condition the fit was given (the sandwich engine fits condition-specific slopes only if it was given one; the slopes engine only records it).

`patients` a data.frame, one row per patient (sample): its id, the number of cells it contributes, and every colData column that is constant within patients (so a condition can be named at `testSpiDE()` time).

`index` a named list, one element per index cell type, holding the engine's estimates (see Details).

`params` a list of the settings the fit used.

`SpiDEResults-class` *The results of spiDE's niche tests*

Description

Returned by `testSpiDE()` and `spiDE()`; read it with `results()`.

Usage

```
## S4 method for signature 'SpiDEResults'
x$name
```

Arguments

`x`	a SpiDEResults.
`name`	a slot name.

Value

An object of class `SpiDEResults`, created by `testSpiDE()`; its `$` accessor returns a slot, and `show()` prints a summary.

Slots

`table` a data.frame, one row per (gene, index, niche, test).

`condition` a character (length 0 or 1), the condition tested.

`contrast` a character, e.g. `"Responder - Non-responder"`.

`procedure` a character, the condition test's family: `"pooled_or_heterogeneity"`, `"heterogeneity"`, `"filtered"` or `"all"`.

`fdr` a numeric, the FDR level of the filter.

`fit` the SpiDEFit tested.

spiGSEA *Gene-set tests on spiDE's niche results*

Description

Asks, for every gene set and every (index cell type, niche cell type) column, whether the set's genes as a group change with the niche (the pooled test) or change with it differently between the two conditions (the condition-specific test).

Usage

```
spiGSEA(object, genesets, ...)

## S4 method for signature 'SpiDEResults'
spiGSEA(
  object,
  genesets,
  test = NULL,
  type = c("competitive", "self-contained"),
  min.size = 5L,
  max.size = 500L,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`object`	a SpiDEResults from the slopes engine.
`genesets`	a named list of character vectors of gene names.
`...`	unused.
`test`	`"pooled"`, `"condition"` or both (the default: every test the results hold).
`type`	`"competitive"` (default) or `"self-contained"`.
`min.size`, `max.size`	set size limits after intersecting with the genes tested in each index type.
`BPPARAM`	a BiocParallelParam; the condition-specific test is run in parallel over sets.

Details

A gene set is tested exactly like a gene. Every patient gets a slope for the set: the average of its genes' niche slopes in that patient, each gene's slope first put on a common scale (divided by the gene's typical spread of slopes between patients, so a highly expressed gene does not dominate). By default the set's slope is *competitive*: the same average over the column's other tested genes is subtracted, so the test asks whether the set moves *more* than the typical gene. This guards against a shift that moves every gene, e.g. from segmentation spillover. With `type = "self-contained"` nothing is subtracted.

The patients' set slopes then go through the same pooled and condition tests as the genes' slopes (see `testSpiDE()` and the model vignette). Because the patient is the unit of replication, the correlation between the genes of a set is accounted for without being estimated: it is already in how much the set's slope varies from patient to patient.

Needs the slopes engine, whose fit holds each patient's slopes. For gene sets in the between-patient test, use `testNicheAbundance()` with `genesets`.

Value

a data.frame, one row per (set, index, niche, test): `size` (genes of the set tested there), `estimate` (the mean set slope, pooled, or its difference between conditions, in units of the genes' between-patient spread), `se`, `t`, `df`, `p`, `q` (Benjamini-Hochberg within the (index, niche, test) column), `q.global` (over every row of that test), `n_patients` and `direction`.

See Also

`testSpiDE()`

Examples

```
data(toySpiDE)
res <- spiDE(toySpiDE, condition = "condition", sigma = 30, index = "A",
             procedure = "all")
sets <- list(first = paste0("G", 1:5), second = paste0("G", 6:12))
spiGSEA(res, sets, min.size = 3)
```

---

testNicheAbundance *Test whether patients with more of a niche cell type express genes differently*

Description

A between-patient question that the within-patient engines cannot see: does a gene's expression in index cell type k differ between patients whose type-k cells sit, on average, among more type-n cells? (`fitSpiDE()` and `testSpiDE()` instead ask whether expression changes with niche density WITHIN each patient.)

Usage

```
testNicheAbundance(spe, ...)

## S4 method for signature 'ANY'
testNicheAbundance(
  spe,
  condition = NULL,
  sigma,
  index = NULL,
  niche = NULL,
```

```
  covariates = character(),
  assay = "counts",
  cell_type = "cell_type",
  sample_id = "sample_id",
  name = "Niche",
  min.cells = 10L,
  prior.count = 1,
  max.leverage = 3,
  genesets = NULL,
  type = c("competitive", "self-contained"),
  min.size = 5L,
  max.size = 500L,
  verbose = TRUE
)
```

Arguments

spe	a SpatialExperiment with niche reducedDims (see buildNiches()).
...	further arguments passed to the method.
condition	a character, the colData column of the condition (constant within sample), or NULL.
sigma	a numeric, the bandwidth (one value).
index, niche	character vectors restricting the index / niche cell types (NULL = all). An index type is never tested against its own niche.
covariates	a character vector of sample-level colData columns to adjust for (constant within sample; the first non-missing value per sample is used).
assay	a character, the counts assay.
cell_type, sample_id	the colData columns of cell type and sample.
name	the niche reducedDim prefix.
min.cells	minimum cells of the index type a sample must contribute.
prior.count	the pseudocount in the log2-CPM.
max.leverage	a number above 1: samples whose leverage in a design exceeds this multiple of the average leverage are down-weighted until none does (Inf turns this off).
genesets	a named list of character vectors (gene identifiers as in rownames(spe)), or NULL to test genes.
type	with genesets: "competitive" (the set against the other genes) or "self-contained" (the set alone).
min.size, max.size	with genesets: set size limits after intersecting with each index type's genes.
verbose	report progress.

Details

For each index cell type k and niche cell type n, regresses the per-sample pseudobulk log2-CPM of every gene in the type-k cells on the per-sample mean log1p niche density of type n around those cells, across samples, with a `limma` moderated t-test. Without a condition the design is `~ niche + covariates` and one term is reported, `"niche"`: the association across all samples. With a `condition` the design is `~ niche * condition + covariates` with the condition coded $-1/2, +1/2$, and two terms are reported: `"niche"`, the association averaged over the two conditions, and `"condition:niche"`, its difference between them (second level minus first) – the patient-level counterpart of the condition-specific niche test that `testSpiDE()` runs within patients. Where the conditions leave fewer than two samples each, only `"niche"` is reported, across all samples.

Influential samples. With tens of samples, one sample at the edge of a niche type's abundances can carry the whole slope, and a gene whose expression is extreme in that same sample then gets a far smaller p-value than the moderated t allows. Samples whose leverage (hat value) in the design exceeds `max.leverage` times the average are therefore down-weighted until none does. The weights come from the design alone, never from expression, so no gene's own values decide how much a sample counts and the estimates stay unbiased; the permutation nulls set the default. Each row reports the largest leverage before the cap and how many samples were down-weighted.

Gene sets. With `genesets`, sets are tested instead of genes, each like a gene. Every gene of the index type is standardised across samples, and a set's score in a sample is the mean of its genes' standardised pseudobulk expression there. With `type = "competitive"` (the default), the mean over all other genes is subtracted, so a shift that moves every gene together does not make a set call. The scores then go through the same regression, leverage cap and multiple testing as genes. This is the between-patient counterpart of `spiGSEA()`, which tests sets on the within-patient slopes. The set version has not yet been checked against permutation nulls.

Multiple testing. `q.global` is a Benjamini-Hochberg adjustment over every row of the same term, so the two terms are separate families: a strong `"niche"` signal no longer loosens the threshold for `"condition:niche"`.

This is the association that `fitSpiDE()`'s per-patient intercepts deliberately absorb. It is a between-patient effect with S experimental units; it is not neighbourhood-dependent differential expression, and the two must not be conflated (see the model vignette). It was `compositionTest()` up to spiDE 0.99.22. Samples contributing fewer than `min.cells` cells of the index type are dropped for that index type, and an index type with fewer than three remaining samples is skipped.

Value

a data.frame with one row per (gene, index type, niche type, term): `gene`, `index`, `niche`, `term`, `estimate` (log2-CPM per unit log1p density), `t`, `p`, `n_patients`, `leverage` (the largest sample leverage before the cap, as a multiple of the average), `downweighted` (the samples down-weighted), `q` (BH within each (index, niche, term) over genes) and `q.global` (BH over every row of the same term). With `genesets`, one row per (set, index type, niche type, term): the column `gene` becomes `set`, followed by `size`, and a `direction` column is added; `estimate` is in standard deviations of the set score per unit log1p density.

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
```

```
na <- testNicheAbundance(spe, condition = "condition", sigma = 30)
head(na[order(na$p), ])
sets <- list(first = paste0("G", 1:6), second = paste0("G", 7:12))
nas <- testNicheAbundance(spe, condition = "condition", sigma = 30,
                          genesets = sets, verbose = FALSE)
head(nas)
```

`testSpiDE` *Test niche-dependent differential expression*

Description

Always runs the **pooled** test: for every (gene, index cell type, niche cell type), whether the gene's expression in the index type changes with the density of the niche type, within patients, consistently across patients. If a `condition` is given (here, or recorded by `fitSpiDE()`), it also runs the **condition-specific** test: whether that slope differs between the two conditions – a between-patient contrast of within-patient slopes.

Usage

```
testSpiDE(object, condition = NULL, ...)

## S4 method for signature 'SpiDEFit'
testSpiDE(
  object,
  condition = NULL,
  procedure = c("pooled_or_heterogeneity", "heterogeneity", "filtered", "all"),
  fdr = 0.05,
  strata = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  ...
)
```

Arguments

`object`	a SpiDEFit from `fitSpiDE()`.
`condition`	`NULL` (use the condition recorded by `fitSpiDE()`, if any) or a character naming a two-level, patient-level colData column. With the sandwich engine it must be the condition the model was fitted with. The contrast is second level minus first (factor levels, else sorted values).
`...`	unused.
`procedure`	`"pooled_or_heterogeneity"`, `"heterogeneity"`, `"filtered"` or `"all"`, see Details. The default is `"pooled_or_heterogeneity"` for the slopes engine and `"filtered"` for the sandwich engine.
`fdr`	the FDR level of the filter.

`strata`	`NULL` (the fit's) or a patient-level column adjusted for in the slopes engine's condition test (e.g. slide, where the condition is confounded with it).
`BPPARAM`	a BiocParallelParam; the slopes engine's condition-specific test is run in parallel over genes.

Details

The condition-specific test is carried out within a family of triplets chosen without looking at the condition, so the choice is exact under any relabelling of patients and the multiplicity budget is spent on triplets with something for the condition to explain:

- `procedure = "pooled_or_heterogeneity"` (the slopes engine's default): the triplets that pass either filter below.
- `procedure = "heterogeneity"`: the triplets whose slopes vary between patients more than their sampling variances allow (Cochran's Q across patients, the statistic behind the between-patient variance; Benjamini-Hochberg over every triplet at `fdr`). A condition effect is such variation, including one whose two conditions' slopes cancel in the pooled slope. With few patients the pooled test passes little, and this filter carries the condition test.
- `procedure = "filtered"` (the sandwich engine's default, and the slopes engine's to spiDE 0.99.36): the triplets whose pooled test passes at `fdr`. It keeps a modest condition difference on a strong pooled slope, which the heterogeneity test, spread over every patient, can miss.
- `procedure = "all"`: every triplet.

The heterogeneity filter needs each patient's slopes, so it is the slopes engine's only. The calibration vignette has the measurements behind the default.

The slopes engine's pooled test refers its moderated t statistic to limma's degrees of freedom scaled by the Kish effective share of the patients (the proportional rule), so a column whose weights rest on a few patients is not credited with the others. To spiDE 0.99.36 the df were the smaller of limma's df and the Kish effective number of patients less one, which is conservative with few patients.

Value

a SpiDEResults; read it with `results()`.

Examples

```
data(toySpiDE)
spe <- buildNiches(toySpiDE, sigma = 30)
fit <- fitSpiDE(spe, index = "A", sigma = 30)
res <- testSpiDE(fit, condition = "condition", procedure = "all")
res
head(results(res))
```

theme_spiDE *The spiDE plot theme*

Description

The theme of every spiDE plot: a black panel border, no grid lines, black axis ticks, boxed facet strips and italic legend titles, with text scaled by `rl`. Add it to your own ggplots so that they match spiDE's, and change any element with a further `+ theme(...)`.

Usage

```
theme_spiDE(rl = 1.1)
```

Arguments

`rl`	a positive number, the text size relative to ggplot2's default.

Value

a ggplot2 theme.

See Also

`spiDEColours()`

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) + geom_point() + theme_spiDE()
```

toySpiDE *A small synthetic SpatialExperiment for spiDE examples*

Description

A seeded toy dataset of 20 genes across 2,400 cells from 16 patients (eight Responders, eight Non-responders; one sample of 150 cells each) with three cell types (A, B, C) laid out on a 500 x 500 field. Cell type B is concentrated in the right of the field, and a neighbourhood-dependent signal is planted: gene `G1` is up-regulated in the index cell type A, in Responders, in proportion to the local density of the niche cell type B. Used throughout the examples and vignette.

Usage

```
data(toySpiDE)
```

Format

A SpatialExperiment with a `counts` assay and colData columns `sample_id`, `condition`, `cell_type`, `Age`, `Area`, and `nCount`.

Source

Generated by `data-raw/make_toySpiDE.R`.

Examples

```
data(toySpiDE)
toySpiDE
```

Index

∗ **datasets**
    toySpiDE, 31
$,SpiDEFit-method *(*SpiDEFit-class*)*, 23
$,SpiDEResults-method
        *(*SpiDEResults-class*)*, 24

buildNiches, 3
buildNiches(), *5*, *7*, *11*, *27*
buildNiches,ANY-method *(*buildNiches*)*, 3

fitSpiDE, 4
fitSpiDE(), *7–9*, *21*, *23*, *26*, *28*, *29*
fitSpiDE,SpatialExperiment-method
        *(*fitSpiDE*)*, 4

mergeNiches, 7
mergeNiches(), *5*, *11*
mergeNiches,ANY-method *(*mergeNiches*)*, 7

patientIntercepts, 8
patientIntercepts(), *13*, *14*, *23*
patientIntercepts,SpiDEFit-method
        *(*patientIntercepts*)*, 8
patientIntercepts,SpiDEResults-method
        *(*patientIntercepts*)*, 8
patientSlopes, 9
patientSlopes(), *15*
patientSlopes,SpiDEFit-method
        *(*patientSlopes*)*, 9
patientSlopes,SpiDEResults-method
        *(*patientSlopes*)*, 9
plotCallMap, 10
plotNicheMap, 11
plotNicheResponse, 12
plotNicheResponse(), *15*
plotPatientEffects, 13
plotPatientEffects(), *8*
plotPatientSlopes, 15
plotPValues, 16
plotSpillover, 16
plotSpillover(), *10*, *17*
plotTopTriplets, 17
plotTripletHeatmap, 19

results, 20
results(), *24*, *30*
results,SpiDEResults-method *(*results*)*,
        20

SpatialExperiment, *32*
spiDE, 21
spiDE(), *6*, *24*
spiDE,SpatialExperiment-method *(*spiDE*)*,
        21
spiDEColours, 22
spiDEColours(), *31*
SpiDEFit, *6*, *8*, *9*, *12*, *14*, *24*, *29*
SpiDEFit-class, 23
SpiDEResults, *8–10*, *12*, *14–20*, *22*, *25*, *30*
SpiDEResults-class, 24
spiGSEA, 25
spiGSEA(), *14*, *19*, *28*
spiGSEA,SpiDEResults-method *(*spiGSEA*)*,
        25

testNicheAbundance, 26
testNicheAbundance(), *6*, *13*, *14*, *26*
testNicheAbundance,ANY-method
        *(*testNicheAbundance*)*, 26
testSpiDE, 29
testSpiDE(), *4–6*, *21*, *23*, *24*, *26*, *28*
testSpiDE,SpiDEFit-method *(*testSpiDE*)*,
        29
theme_spiDE, 31
theme_spiDE(), *22*
toySpiDE, 31