Package 'PACE'

October 30, 2026

Type Package

Title Proximity-Associated Changes in Expression for Spatial Transcriptomics

Version 0.99.20

Description PACE (Proximity-Associated Changes in Expression) is a hierarchical empirical Bayes framework that quantifies how a cell's gene expression changes with proximity to specific neighbouring cell types in imaging-based spatial transcriptomics such as Xenium and CosMx. It fits negative binomial mixed models with partial pooling across cell types, corrects for transcript contamination between adjacent cells with a per-cell ambient term, and decomposes expression variance into cell-type identity, spatial cell state, contamination, and residual components. Input and output follow SpatialExperiment conventions.

License GPL-3

Encoding UTF-8

biocViews Spatial, Transcriptomics, GeneExpression, SingleCell, Regression, Bayesian, DifferentialExpression, Software

Depends R (>= 4.6.0)

Imports methods, stats, utils, Rcpp, Matrix, SpatialExperiment, S4Vectors, SummarizedExperiment, BiocParallel, dplyr, tidyr, tibble, spatstat.geom, MASS, dbscan, lme4, mashr, ashr, ggplot2, patchwork, scales, stringr

Suggests DelayedArray, glmmTMB, broom.mixed, glue, testthat (>= 3.0.0), R6, knitr, rmarkdown, BiocStyle

URL https://github.com/ecool50/PACE, https://ecool50.github.io/PACE

BugReports https://github.com/ecool50/PACE/issues

LinkingTo Rcpp, RcppEigen

SystemRequirements GNU GSL

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

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

git_branch devel

git_last_commit 571c226

git_last_commit_date 2026-10-13

Repository Bioconductor 3.24

Date/Publication 2026-10-29

Author Elijah Willie [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5773-3649>),
Shreya Rajesh Rao [aut],
John Ormerod [aut],
Ellis Patrick [aut],
Australian Government Research Training Program [fnd]

Maintainer Elijah Willie <ewil3501@uni.sydney.edu.au>

Contents

PACE-package . . . 3
ambientField . . . 3
anchorGenes . . . 4
buildNeighbourhood . . . 5
cellContamination . . . 6
cellTypes . . . 7
compactFit . . . 8
fitState . . . 9
neighbourSlopes . . . 9
paceDecompose . . . 10
paceDrivers . . . 11
PACEFit . . . 12
paceFit . . . 13
paceModel . . . 14
paceModelGaussian . . . 16
paceParams . . . 17
paceShrink . . . 18
pace_fit_streaming_gaussian . . . 19
pairVariance . . . 20
plotDecomposition . . . 21
plotDrivers . . . 21
plotPairHeatmap . . . 22
plotProximity . . . 23
plotResponseCurve . . . 24
plotResponseMap . . . 25
stashPACE . . . 26
topDrivers . . . 27
varianceDecomposition . . . 28

Index **29**

PACE-package *PACE: Proximity-Associated Changes in Expression*

Description

PACE quantifies how a cell's gene expression changes with proximity to specific neighbouring cell types in imaging-based spatial transcriptomics. It fits hierarchical negative binomial mixed models with partial pooling across cell types, corrects for transcript contamination between adjacent cells with a per-cell ambient term, and decomposes expression variance into cell-type identity, spatial cell state, contamination, and residual components.

Details

The user-facing entry point is `paceFit()`, which takes a SpatialExperiment::SpatialExperiment and returns a PACEFit object. The results are read out with `neighbourSlopes()`, `varianceDecomposition()`, and `topDrivers()`.

Author(s)

Maintainer: Elijah Willie <ewil3501@uni.sydney.edu.au> (ORCID)

Authors:

- Shreya Rajesh Rao
- John Ormerod
- Ellis Patrick

Other contributors:

- Australian Government Research Training Program [funder]

See Also

Useful links:

- https://github.com/ecool50/PACE
- https://ecool50.github.io/PACE
- Report bugs at https://github.com/ecool50/PACE/issues

ambientField *Build the ambient contamination field*

Description

Returns the sparse cross-cell-type ambient weight matrix that defines the per-cell contamination term (the short-range technical field a cell receives from its heterotypic neighbours).

Usage

```
ambientField(object, ...)

## S4 method for signature 'SpatialExperiment'
ambientField(
  object,
  celltype_col,
  image_col = NULL,
  h_tech = 5,
  assay_name = "counts",
  ...
)
```

Arguments

object	A SpatialExperiment::SpatialExperiment.
...	Passed to the ambient-field builder.
celltype_col	colData column with the cell-type annotation.
image_col	Optional colData column grouping cells into images.
h_tech	Technical bandwidth in micrometres (default 5).
assay_name	Counts assay name (default "counts").

Value

The ambient-field object (sparse weight matrix and image index).

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
af <- ambientField(spe, celltype_col = "cellType")
names(af)
```

---

anchorGenes *Contamination anchor genes per cell type*

Description

Returns, for each cell type, the genes used as negative-control anchors when identifying the per-cell contamination loading: genes owned by another cell type (high mean expression there) that are near-absent in the focal type. Anchors are recomputed with the same selection as the fit, and ranked by expression in their owning cell type. This is the source of the manuscript's supplementary anchor-gene tables.

Usage

```
anchorGenes(object, spe, top_n = 10L, assay_name = "counts")
```

Arguments

object	A PACEFit (supplies the cell-type and image columns used at fit time).
spe	The SpatialExperiment::SpatialExperiment that was fitted.
top_n	Number of top anchors to list per cell type (default 10).
assay_name	Counts assay name (default "counts").

Value

A data frame with one row per cell type: cellType, n_anchors (total anchors), and top_anchors (the top_n strongest, comma-separated).

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
anchorGenes(fit, spe)
```

---

buildNeighbourhood *Build the neighbourhood kernels*

Description

Returns the per-cell kernel-weighted neighbour abundances: the Gaussian biological kernel K_bio (used for the proximity coefficients) and the short-range exponential technical kernel K_tech. Useful for inspecting the neighbourhood before or independently of a fit.

Usage

```
buildNeighbourhood(object, ...)

## S4 method for signature 'SpatialExperiment'
buildNeighbourhood(
  object,
  celltype_col,
  h_bio = 30,
  h_tech = 5,
  eps = NULL,
  ...
)
```

Arguments

object	A SpatialExperiment::SpatialExperiment.
...	Passed to the kernel builder.
celltype_col	colData column with the cell-type annotation.
h_bio, h_tech	Bandwidths in micrometres (defaults 30, 5).
eps	Truncation radius; defaults to 3 * h_bio.

Value

A list with K_bio and K_tech (cells x cell types).

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
nb <- buildNeighbourhood(spe, celltype_col = "cellType")
dim(nb$K_bio)
```

---

cellContamination *Per-cell contamination*

Description

Returns the per-cell contamination loading rho_i and the fraction of a cell's expected counts attributed to its local ambient field. Under the per-cell contamination model the expected count is mu_ig = mu_bio_ig + rho_i * a_ig, where a_ig is the cross-cell-type ambient field at cell i and rho_i is a single empirical-Bayes shrunken loading shared across genes. The contamination fraction summarises that second term over genes, contamFraction_i = sum_g mu_spill_ig / sum_g mu_ig, and is the quantity reported as contam_frac in the solver's fitting trace.

Usage

```
cellContamination(object, ...)

## S4 method for signature 'PACEFit'
cellContamination(object, ...)
```

Arguments

object	A PACEFit fitted with contamination = "percell_hc".
...	Unused.

Details

A high contamination fraction marks a cell whose profile is substantially explained by its neighbours rather than by its own cell type, which is the expected signature of a segmentation or transcript-assignment error. It is not a complete cell-quality score: because rho_i carries a fixed gene-direction (the local ambient), it will under-report contamination whose direction departs from the neighbourhood average, and a cell that is wholly a segmentation artefact may fit some incorrect cell type with a low rho_i. Pair it with ordinary per-cell quality control.

Read contamFraction rather than rho on its own. The ambient field is cross-cell-type, so a cell with no differently-typed neighbour inside the technical kernel has a_ig = 0 for every gene and therefore no contamination at all. Its rho_i is then unidentified and shrinks to the empirical Bayes prior mean, giving every such cell the same apparently middling loading. On the shipped example fit this is 4,589 of 7,898 cells, 84% of the tumour cells in a tumour-dominated crop. contamFraction reports them as zero, which is correct; rho alone does not.

Value

A data frame with one row per cell, in fitting order: `cell` (the working-frame cell identifier, or the row index if the frame is unnamed), `celltype` (the label the cell was fitted under), `rho` (the contamination loading) and `contamFraction`.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
cc <- cellContamination(fit)
head(cc)
# Contamination is highest where cells sit against a different type.
tapply(cc$contamFraction, cc$celltype, median)
```

---

cellTypes *Cell types modelled*

Description

The cell types the fit was built on, in fitting order. The order matters: it indexes the rows of the random-effect design, so the readouts and the solver state agree on what a neighbour column refers to.

Usage

```
cellTypes(object, ...)

## S4 method for signature 'PACEFit'
cellTypes(object, ...)
```

Arguments

`object`	A PACEFit.
`...`	Unused.

Value

A character vector of cell type names.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
cellTypes(fit)
```

compactFit *Drop the per-cell matrices from a fit*

Description

A fit stores the fitted means and the contamination offsets as dense cells by genes matrices. On a real cohort those dominate its size, and nothing downstream needs them kept: `paceDecompose()` and `paceDrivers()` read per-cell-type summaries the fit also holds, and rebuild the matrices from the counts when they are absent.

Usage

```
compactFit(
  object,
 which = c("mu", "mu_bio", "mu_spill", "technical_offset_mat", "bleed_offset_mat")
)
```

Arguments

`object`	A PACEFit.
`which`	The matrices to drop. The default is all of them.

Details

Use this before saving a fit. Every readout still works afterwards, and re-decomposing a compacted fit reproduces the tables it already carries.

Value

`object`, without the named matrices.

See Also

`stashPACE()` for keeping the readouts with the data.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
small <- compactFit(fit)
c(before = object.size(fit), after = object.size(small)) |> format(units = "MB")
```

fitState *Raw solver state*

Description

The penalised quasi-likelihood solver's output: fixed and random effects and their standard errors, gene-wise overdispersions, variance components, the per-cell contamination loadings, and the convergence history.

Usage

```
fitState(object, ...)

## S4 method for signature 'PACEFit'
fitState(object, ...)
```

Arguments

`object`	A PACEFit.
`...`	Unused.

Details

This is the fit itself rather than a readout of it, and it is exposed for inspection and diagnostics. Prefer `neighbourSlopes()`, `varianceDecomposition()` and `topDrivers()` for interpretation: those apply the shrinkage and attribution the method is defined by, and the raw BLUPs here do not.

Value

A named list of solver state.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
dim(fitState(fit)$U)
fitState(fit)$converged
```

neighbourSlopes *Shrunken neighbour proximity slopes*

Description

Returns the shrunken per-(gene, focal, neighbour) proximity coefficients with their local false sign rate (lfsr). For condition cohorts the responder interaction terms are included.

Usage

```
neighbourSlopes(object, ...)

neighbourSlopes(object) <- value

## S4 method for signature 'PACEFit'
neighbourSlopes(object, ...)

## S4 replacement method for signature 'PACEFit'
neighbourSlopes(object) <- value
```

Arguments

object	A PACEFit.
...	Unused.
value	A data frame of shrunken proximity coefficients.

Value

A data frame of shrunken slopes.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
head(neighbourSlopes(fit))
```

---

paceDecompose *Variance decomposition*

Description

Partitions per-gene expression variance into cell-type identity, spatial cell state, contamination, and residual (plus a responder block for condition cohorts), populating `varianceDecomposition()`. Needs the fitted object plus the same `SpatialExperiment` used for `paceModel()` (to read the counts).

Usage

```
paceDecompose(object, ...)

## S4 method for signature 'PACEFit'
paceDecompose(object, spe, ...)
```

Arguments

object	A PACEFit from `paceModel()`.
...	Unused.
spe	The SpatialExperiment::SpatialExperiment that was fitted.

Details

The decomposition needs per-cell-type statistics of the fitted means, which the fit stores; it never builds a `cells x genes` matrix. An older fit without those statistics has them computed from its stored fitted means, or, if those were dropped, rebuilt exactly from the fit and `spe` a block of genes at a time.

Value

The `PACEFit` with the variance decomposition added.

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
fit <- paceDecompose(fit, spe)
head(varianceDecomposition(fit))
```

`paceDrivers` *Per-pair driver scores*

Description

Ranks the genes mediating each focal-neighbour relationship by driver score, populating `topDrivers()`. Requires `paceShrink()` and `paceDecompose()` to have run first.

Usage

```
paceDrivers(object, ...)

## S4 method for signature 'PACEFit'
paceDrivers(object, spe = NULL, pairs = NULL, ...)
```

Arguments

`object`	A PACEFit with shrunken slopes and a decomposition.
`...`	Unused.
`spe`	The SpatialExperiment::SpatialExperiment that was fitted. Needed only for an older fit that stores neither its fitted means nor their per-cell-type statistics.
`pairs`	Optional list of focal-neighbour pairs to score; `NULL` scores all pairs.

Details

The driver scores read each cell type's mean fitted mean, which the fit stores. For an older fit saved without its fitted means or those statistics they are rebuilt exactly from the fit and `spe`, as in `paceDecompose()`, a block of genes at a time; pass `spe` for such a fit.

Value

The `PACEFit` with the driver tables added.

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
fit <- paceDrivers(fit, spe)
names(topDrivers(fit))
```

PACEFit *PACEFit: a fitted PACE model*

Description

An S4 container for the result of a PACE fit. It holds the raw penalised quasi-likelihood solver state together with the three interpretable output layers: shrunken per-(gene, focal, neighbour) proximity slopes, the per-gene variance decomposition, and the per-pair driver tables. Construct one with `paceFit()`; read it out with `neighbourSlopes()`, `varianceDecomposition()`, and `topDrivers()`.

Usage

```
## S4 method for signature 'PACEFit'
show(object)
```

Arguments

`object`	A PACEFit object.

Value

The `PACEFit` class itself is a container and is not returned by a function documented here; construct one with `paceFit()`. The show method is called for its side effect of printing a one-screen summary of the fit – cell types, kernel bandwidths, contamination and dispersion settings, the condition term if there is one, and which pipeline stages have run – and returns `NULL` invisibly.

Slots

`fit` The raw streaming PQL solver state (fixed and random effects, per-cell contamination loadings, gene-wise overdispersions, variance components).

`neighbourSlopes` A data frame of shrunken proximity coefficients, one row per gene, focal cell type and neighbour cell type, plus the condition term for condition cohorts, with the raw estimate, shrunken estimate, and local false sign rate (lfsr).

`varianceDecomposition` A list of per-gene, per-focal variance decomposition tables (cell-type identity, spatial cell state, contamination, residual; and, for condition cohorts, a responder spatial block).

`topDrivers` A list of per-pair driver-score tables ranking the genes that mediate each focal-neighbour spatial relationship.

`cellTypes` The cell types modelled, in fitting order.

`params` The fitting parameters actually used (bandwidths, contamination and dispersion settings, condition column, and so on).

`context` The working frame and fixed-effect design retained from the fit so that the downstream stages (`paceShrink()`, `paceDecompose()`, `paceDrivers()`) can run without refitting.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
fit
neighbourSlopes(fit)
```

---

paceFit *Fit a PACE model to a SpatialExperiment*

Description

paceFit() runs the full pipeline in one call: paceModel() -> paceShrink() -> paceDecompose() -> paceDrivers(). For step-by-step control (inspecting the fitted model, re-running the downstream without refitting), call those stages directly.

Usage

```
paceFit(object, ...)

## S4 method for signature 'SpatialExperiment'
paceFit(object, ..., pairs = NULL)
```

Arguments

object	A SpatialExperiment::SpatialExperiment.
...	Arguments passed to paceModel() (celltype_col, image_col, contamination, dispersion, ...).
pairs	Optional list of focal-neighbour pairs for the driver tables.

Value

A fully populated PACEFit.

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))

fit <- paceFit(spe, celltype_col = "cellType", verbose = FALSE)
head(neighbourSlopes(fit))
```

paceModel *Fit the PACE model*

Description

paceModel() builds the biological and technical neighbourhood kernels from the spatial coordinates and cell-type labels and fits the hierarchical negative binomial mixed model by streaming penalised quasi-likelihood (with the per-cell contamination correction). It returns a PACEFit carrying only the fitted model; the reporting layers are added by paceShrink(), paceDecompose(), and paceDrivers() (or all at once by paceFit()).

Usage

```
paceModel(object, ...)

## S4 method for signature 'SpatialExperiment'
paceModel(
  object,
  celltype_col,
  image_col = NULL,
  condition_col = NULL,
  assay_name = "counts",
  h_bio = 30,
  h_tech = 5,
  contamination = c("percell_hc", "none"),
  dispersion = c("nb1", "nb2"),
  kernel_per_image = FALSE,
  image_re = c("none", "intercept", "slopes", "condition_slopes"),
  types = NULL,
  resp_term = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	A SpatialExperiment::SpatialExperiment with a counts assay and two-dimensional spatial coordinates.
...	Further arguments passed to the streaming fitter (n_iter, threads, chunk_size, drop_sparse_neff, within_image, edge_correct, data_informed_tau, tau_shrinkage, and the memory and approximation settings described below).
celltype_col	colData column with the discrete cell-type annotation.
image_col	Optional colData column grouping cells into images/samples. NULL (default) treats the object as one section.
condition_col	Optional colData column with a binary condition; enables the responder spatial block for multi-sample cohorts.
assay_name	Counts assay name (default "counts").
h_bio, h_tech	Biological and technical kernel bandwidths in micrometres (defaults 30 and 5).
contamination	"percell_hc" (default) or "none".

`dispersion` `"nb1"` (default) or `"nb2"`.

`kernel_per_image`
If `TRUE`, neighbourhoods are built within each image.

`image_re` Second image-level random-effect block: `"none"`, `"intercept"`, `"slopes"`, or `"condition_slopes"`.

`types` Cell types in the order they should index the random-effect blocks. `NULL` (default) uses every observed type in alphabetical order; pass an explicit vector to reproduce a locked fit, whose block order the caller chose.

`resp_term` Responder interaction term prefix for condition cohorts; if `NULL` it is derived from `condition_col`.

`verbose` Whether to print progress.

Value

A PACEFit with the fitted model (reporting layers empty).

Memory and approximation settings

Three pass-through arguments change how much memory a fit needs, or trade exactness for time. Two of them are approximations that are ON by default.

`ambient_mode` `"cache"` (default) or `"stream"`. The contamination model needs an ambient field per cell and gene; `"cache"` materialises that n by G product, `"stream"` recomputes each chunk's columns from the weights and the counts. The numbers are identical. On a 1.2M cell, 5,001 gene panel the cached product alone is over 5 GB, so `"stream"` is what makes a full transcriptome panel fit in memory at all.

`alpha_warmup` Default `6`. The per-gene dispersion MLE is re-fitted only on the first `alpha_warmup` iterations and on the last one; in between, alpha is frozen at its warmed-up value. Alpha typically settles within five iterations while the MLE is a large share of each iteration's cost, so this is on by default. **It is an approximation**: pass `Inf` to re-fit the dispersion on every iteration. Lowering it below the default does move results – on the breast cancer cohort 4 changes the number of calls at `lfsr < 0.05`.

`alpha_max_n` Default `Inf`, meaning the dispersion MLE sees every cell. A finite value caps it at an even, deterministic subsample; the estimator, the Brent search and its tolerance are unchanged. The dispersion is one scalar per gene and its standard error falls as `1/sqrt(n)`, so most cells add little. **Validate any cap on your own cohort before trusting it.** A cap of 50,000 reproduced two cohorts of roughly 10^5 cells exactly, and on a 1.2M cell panel – where the same cap is a far smaller fraction of the data – it left the number of calls unchanged while swapping the identity of 72 of them. The binding quantity is not the absolute subsample.

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))

fit <- paceModel(spe, celltype_col = "cellType", verbose = FALSE)
fit
```

paceModelGaussian *Fit the PACE model to continuous intensities*

Description

The Gaussian, identity-link form of `paceModel()`, for continuous measurements such as imaging mass cytometry protein intensities rather than counts.

Usage

```
paceModelGaussian(object, ...)

## S4 method for signature 'SpatialExperiment'
paceModelGaussian(
  object,
  celltype_col,
  image_col = NULL,
  condition_col = NULL,
  assay_name = "intensity",
  h_bio = 30,
  h_tech = 5,
  kernel_per_image = FALSE,
  image_re = c("none", "intercept", "slopes", "condition_slopes"),
  types = NULL,
  resp_term = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	A SpatialExperiment::SpatialExperiment.
`...`	Passed to `pace_fit_streaming_gaussian()`.
`celltype_col`	Column of `colData` holding the cell type labels.
`image_col`	Column of `colData` identifying the image. Defaults to the single image when there is only one.
`condition_col`	Optional column of `colData` giving a condition contrast.
`assay_name`	The assay holding the intensities.
`h_bio, h_tech`	Bandwidths of the biological and technical kernels.
`kernel_per_image`	Build the kernels within each image separately.
`image_re`	An optional second random-effect block over images.
`types`	Cell types, in the order the fit should use them.
`resp_term`	Optional response term passed to the model frame.
`verbose`	Print the fitting trace.

Details

The working response on the identity link is the intensity itself and the working weight is the per-gene residual variance, so the mean converges in a single inner solve and the outer iteration only moves the variance components.

This path does not model technical contamination. Correct the intensities first, against an ambient field built with `ambientField()`, and pass the corrected matrix here.

Value

A PACEFit.

---

paceParams *Fitting parameters*

Description

The settings the fit actually used, rather than the ones requested: kernel bandwidths, the contamination and dispersion models, the column names read from the `SpatialExperiment`, and the condition term if there is one. Worth reading before comparing two fits, since several of these change the answer.

Usage

```
paceParams(object, ...)

## S4 method for signature 'PACEFit'
paceParams(object, ...)
```

Arguments

`object`	A PACEFit.
`...`	Unused.

Value

A named list of parameters.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
paceParams(fit)$h_bio
```

paceShrink *Shrink the neighbour slopes*

Description

Stabilises the fitted per-(gene, focal, neighbour) proximity slopes with multivariate adaptive shrinkage, populating `neighbourSlopes()`.

Usage

```
paceShrink(object, ...)

## S4 method for signature 'PACEFit'
paceShrink(object, ...)
```

Arguments

`object`	A PACEFit from `paceModel()`.
`...`	Further arguments passed to the shrinkage step, notably `null_correlation` (default `TRUE`: estimate the correlation between focal cell types under the null and pass it to mash as `V`; `FALSE` treats them as independent), `data_driven`, and `shrink_threads`. `shrink_threads` (default 1) is how many R processes shrink the neighbour slices at once. The slices are independent and carry about 90% of the work, so this is where the time goes: on the full breast cancer cohort it takes the shrinkage from 55.4 to 21.1 seconds, and the called slopes agree with the serial path to 8.3e-13 with identical calls and no sign flips. It is not, however, the serial computation. With `data_driven = TRUE` the parallel path cannot reproduce the serial one: `cov_pca()` draws its starting vectors from the RNG, and a serial run consumes that stream slice by slice. Above 1 each slice is seeded with its own index, so the parallel result is reproducible run to run but differs from the serial one – by ~1e-12 at cohort scale, but by up to ~1e-6 on small slices, where there is less data to swamp the difference. Serial is the default because it is the stream the package's fixtures were made with, and because the parallel path raised a BiocParallel reducer error on one supported R version that could not be reproduced on any other. Raise it when a large cohort's shrinkage is the bottleneck. With `data_driven = FALSE` nothing draws from the stream and the two are identical.

Details

One mash model is fitted per neighbour cell type, and every one of them is fitted over the same genes. mash calibrates against the genes it is given, so a common gene set keeps the lfsr comparable across neighbours; a gene that cannot be used for one neighbour is dropped for all of them, with a message.

Value

The `PACEFit` with the shrunken neighbour slopes added.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
fit <- paceShrink(fit)
head(neighbourSlopes(fit))
```

`pace_fit_streaming_gaussian` *Gaussian streaming orchestrator*

Description

Builds the neighbour kernels and the random-effect design for continuous intensities, then fits them with the identity-link engine. The Gaussian counterpart of `pace_fit_streaming()`; see the notes at the top of this file for what differs and why they are kept apart.

Usage

```
pace_fit_streaming_gaussian(
  Y,
  df,
  types = NULL,
  celltype_col,
  image_col,
  coord_cols = c("x", "y"),
  h_bio = 30,
  h_tech = 5,
  eps = NULL,
  condition_col = NULL,
  kernel_per_image = FALSE,
  image_re = c("none", "intercept", "slopes", "condition_slopes"),
  drop_sparse_neff = 30,
  within_image = TRUE,
  n_iter = 32L,
  threads = 4L,
  chunk_size = 128L,
  tau_shrinkage = "adaptive",
  early_stop_tol = 0.02,
  min_iter = 12L,
  tau_max = 100,
  verbose = TRUE
)
```

Arguments

`Y`, `df`, `types`, `celltype_col`, `image_col`, `coord_cols`
: Intensities, the model frame, the cell types and the columns naming them.

`h_bio`, `h_tech`, `eps`
: Kernel bandwidths and the neighbour radius.

`condition_col`, `kernel_per_image`, `image_re`, `drop_sparse_neff`, `within_image`
: Design options, as `pace_fit_streaming()` takes them.

`n_iter`, `threads`, `chunk_size`, `tau_shrinkage`, `early_stop_tol`, `min_iter`, `tau_max`
: Fitting control, as `pace_fit_streaming()` takes them.

`verbose`
: Print the fitting trace.

Value

A list with the fit, the model frame, the fixed-effect design, the intensities, the kernels and the cell types.

`pairVariance` *Pairwise spatial variance table*

Description

Returns the tidy per-(focal, neighbour) spatial variance table that `plotPairHeatmap()` draws: for each focal cell type, the percentage of its total expression variance contributed by spatial interaction with each neighbour, obtained by splitting the focal's spatial share across neighbours by a normalised (off-diagonal-renormalised) Pratt attribution. Off-diagonal pairs only.

Usage

```
pairVariance(object, ...)

## S4 method for signature 'PACEFit'
pairVariance(object, block = c("spatial", "responder"), ...)
```

Arguments

`object`
: A PACEFit.

`...`
: Unused.

`block`
: Either `"spatial"` (default) or `"responder"` (condition cohorts).

Value

A data frame with columns `focal`, `neighbour`, and `val` (spatial % of the focal type's total variance).

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
head(pairVariance(fit))
```

plotDecomposition *Per-focal variance decomposition plot*

Description

Reproduces the manuscript per-focal decomposition figure: a full stacked bar of the cell-type identity, spatial cell state, and contamination (spillover) components, with a zoomed grouped bar of the two small spatial components.

Usage

```
plotDecomposition(object, title = "Per-focal decomposition")
```

Arguments

object	A PACEFit.
title	Plot title.

Value

A patchwork / ggplot object.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
plotDecomposition(fit)
```

plotDrivers *Per-pair driver composite*

Description

Reproduces the manuscript per-pair driver figure: the top genes ranked by their driver score for a focal-neighbour pair, alongside their per-gene single-frame variance decomposition. The score is carried in the driver_score column of topDrivers(object).

Usage

```
plotDrivers(
  object,
  focal,
  neighbour,
  n_top = 5,
  panels = c("both", "score", "gene")
)
```

Arguments

object	A PACEFit.
focal, neighbour	The focal and neighbour cell types.
n_top	Number of top genes to show.
panels	One of "both", "score", or "gene".

Value

A patchwork / ggplot object.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
plotDrivers(fit, "Macrophage", "Tumour")
```

---

plotPairHeatmap *Pairwise spatial-percent heatmap*

Description

Reproduces the manuscript pairwise heatmap: each cell is the percentage of the focal type's total variance contributed by spatial interaction with that neighbour, computed as the focal's spatial share split across neighbours by the off-diagonal-renormalised Pratt attribution.

Usage

```
plotPairHeatmap(object, block = c("spatial", "responder"), title = NULL)
```

Arguments

object	A PACEFit.
block	Which spatial block to map: "spatial" (default, the baseline spatial cell state) or "responder" (the responder-by-proximity block, for condition cohorts).
title	Plot title; a sensible default is chosen per block.

Value

A ggplot object.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
plotPairHeatmap(fit)
```

plotProximity *Expression versus number of neighbours (per-bin boxplot)*

Description

Reproduces the manuscript proximity figure (verbatim style port of `density_bin_boxplot` in the analysis helpers): for each focal cell the number of `neighbour`-type cells within `radius` micrometres is counted and binned, and each gene's raw counts are drawn as a boxplot per bin with the per-bin mean overlaid as a point. Because these genes are zero-inflated the box median sits at zero in most bins, so the mean point carries the trend.

Usage

```
plotProximity(
  object,
  spe,
  genes,
  focal,
  neighbour,
  radius = 30,
  breaks = c(0, 2, 4, 6, 8, Inf),
  condition = NULL,
  condition_colours = c("#3B6FB6", "#C0392B"),
  box_colour = "#4F8B5E",
  assay_name = "counts"
)
```

Arguments

`object`	A PACEFit (supplies the cell-type column used at fit time).
`spe`	The SpatialExperiment::SpatialExperiment that was fitted.
`genes`	Character vector of genes (shown as facets).
`focal, neighbour`	Focal and neighbour cell types.
`radius`	Neighbour search radius in micrometres (default 30).
`breaks`	Neighbour-count bin breaks (default `c(0, 2, 4, 6, 8, Inf)`).
`condition`	Optional `colData` column splitting the boxes by group, so the two arms of a condition cohort can be compared within each bin. `TRUE` uses the fit's own `condition_col`; `NULL` (default) pools all cells.
`condition_colours`	Colours for the condition levels.
`box_colour`	Box and mean-point colour, used when `condition` is `NULL`.
`assay_name`	Counts assay name (default `"counts"`).

Value

A `ggplot` object.

Examples

```
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
g <- head(subset(neighbourSlopes(fit),
                 focal == "Macrophage" & neighbour == "Tumour")$gene, 2)
plotProximity(fit, spe, g, "Macrophage", "Tumour")
```

`plotResponseCurve` *Response curve: binned means with the PACE slope overlaid*

Description

Population view of one gene in `focal` cells against `neighbour` density, split by the fit's condition. The solid line and ribbon are binned means and standard errors of the observed data; the dashed line is the PACE slope read straight from the fitted model, anchored at each arm's data centroid. It is the model's estimate drawn over the data, not a smoother refitted to it.

Usage

```
plotResponseCurve(
  object,
  spe,
  gene,
  focal,
  neighbour,
  n_bins = 10,
  min_bin_n = 10,
  colours = c("#3B6FB6", "#C0392B")
)
```

Arguments

`object`	A PACEFit fitted with a `condition_col`.
`spe`	The SpatialExperiment::SpatialExperiment that was fitted.
`gene`, `focal`, `neighbour`	Gene, focal cell type and neighbour cell type.
`n_bins`	Number of density quantile bins (default 10).
`min_bin_n`	Bins with fewer focal cells than this are dropped.
`colours`	Colours for the two arms.

Value

A `ggplot` object.

Examples

```
spe <- readRDS(system.file("extdata", "mel_cosmx_subset.rds", package = "PACE"))

fit <- paceFit(spe, celltype_col = "cellType", condition_col = "Responder",
               image_col = "image", kernel_per_image = TRUE,
               image_re = "intercept", verbose = FALSE)
plotResponseCurve(fit, spe, "SPP1", "Macrophage", "Tumour")
```

`plotResponseMap` *Tissue maps and scatters for one exemplar image per arm*

Description

The spatial view behind `plotResponseCurve()`. For one image per condition arm: a smooth kernel-density map of the neighbour type with the focal cells overlaid and coloured by expression (top), and expression against neighbour density for the same cells with a fitted line (bottom). Colour scales are shared across the panels so the arms are directly comparable.

Usage

```
plotResponseMap(
  object,
  spe,
  gene,
  focal,
  neighbour,
  images = NULL,
  kde_h = 60,
  ngrid = 200,
  scalebar_len = 100
)
```

Arguments

`object`	A PACEFit fitted with a `condition_col`.
`spe`	The SpatialExperiment::SpatialExperiment that was fitted.
`gene`, `focal`, `neighbour`	Gene, focal cell type and neighbour cell type.
`images`	Optional named character vector of image identifiers, one per arm. The default picks, for each arm, the image with the most focal cells.
`kde_h`, `ngrid`	Kernel bandwidth and grid size for the density map.
`scalebar_len`	Scale-bar length in micrometres.

Value

A `patchwork` object.

Examples

```
spe <- readRDS(system.file("extdata", "mel_cosmx_subset.rds", package = "PACE"))

fit <- paceFit(spe, celltype_col = "cellType", condition_col = "Responder",
               image_col = "image", kernel_per_image = TRUE,
               image_re = "intercept", verbose = FALSE)
plotResponseMap(fit, spe, "SPP1", "Macrophage", "Tumour")
```

`stashPACE` *Stash PACE readouts in a SpatialExperiment*

Description

Copies a fit's interpretable output into `metadata(spe)`, so that results travel with the data they came from and downstream code can find them without being handed a second object.

Usage

```
stashPACE(spe, fit, name = "PACE", overwrite = FALSE)
```

Arguments

`spe`	A SpatialExperiment::SpatialExperiment – the one the fit was built on. The cell count is checked against the fit.
`fit`	A PACEFit.
`name`	The `metadata()` element to write. Defaults to `"PACE"`.
`overwrite`	Whether to replace an existing element of that name.

Details

The solver state is deliberately not copied. A fit is large, it is often saved and reloaded independently of the data, and nothing downstream reads it; see `fitState()` if you need it. What is stashed is the three reporting layers plus enough provenance to tell which fit produced them: the cell types, the fitting parameters, and the package version.

Value

`spe`, with `metadata(spe)[[name]]` populated.

See Also

`neighbourSlopes()`, `varianceDecomposition()`, `topDrivers()`

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
spe <- readRDS(system.file("extdata", "bc_xenium_subset.rds", package = "PACE"))
spe <- stashPACE(spe, fit)
names(S4Vectors::metadata(spe)$PACE)
head(S4Vectors::metadata(spe)$PACE$neighbourSlopes, 3)
```

topDrivers *Per-pair driver tables*

Description

Returns the driver-score tables ranking the genes that mediate each focal-neighbour spatial relationship.

Usage

```
topDrivers(object, ...)

topDrivers(object) <- value

## S4 method for signature 'PACEFit'
topDrivers(object, ...)

## S4 replacement method for signature 'PACEFit'
topDrivers(object) <- value
```

Arguments

object	A PACEFit.
...	Unused.
value	A named list of per-pair driver tables, each with scores.

Details

Each pair's genes are called at lfsr < 0.05. The lfsr is the posterior probability that a call has the wrong sign, so its sum over a pair's calls, expected_false_sign, is the expected number of those calls with the wrong sign, and false_sign_rate is that count as a fraction of the calls (Stephens 2017). Read a pair's call count against it: many ordered pairs are tested, and a pair with three calls and an expected 1.5 wrong is weak evidence. Both are only as well calibrated as the lfsr itself.

On a fit with a condition, the table carries two different slopes and they are not a raw-and-shrunken pair of the same quantity. b_clean and lfsr are the condition-interaction term, the one the shrinkage was applied to, and the driver score is built from b_clean. u_raw is the *baseline* slope, the unshrunken BLUP for the neighbour on its own, and it is there because the variance split needs it: V_S comes from u_raw and V_RxS from b_clean. On a fit without a condition there is no interaction to separate, so u_raw is absent and b_clean is the only slope.

Value

A named list with one entry per focal-neighbour pair, each a list with scores (the driver table), status, expected_false_sign and false_sign_rate.

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
names(topDrivers(fit))
```

`varianceDecomposition` *Per-gene variance decomposition*

Description

Returns the per-gene, per-focal variance decomposition. By default the observed single-frame decomposition (cell-type identity, spatial cell state, contamination, residual; and a responder spatial block for condition cohorts) is returned; set `which = "blocks"` for the underlying link-scale block tables.

Usage

```
varianceDecomposition(object, ...)

varianceDecomposition(object) <- value

## S4 method for signature 'PACEFit'
varianceDecomposition(object, which = c("perGene", "blocks"), ...)

## S4 replacement method for signature 'PACEFit'
varianceDecomposition(object) <- value
```

Arguments

`object`	A PACEFit.
`...`	Unused.
`value`	Either the per-gene table, as the getter returns by default, or the whole list with its `perGene` and `blocks` entries. Given a table, the existing `blocks` are kept, so that the setter inverts the getter.
`which`	Either `"perGene"` (default, the observed single-frame table) or `"blocks"` (the raw link-scale decomposition).

Value

A data frame (for `"perGene"`) or a list (for `"blocks"`).

Examples

```
fit <- readRDS(system.file("extdata", "pace_fit_example.rds", package = "PACE"))
head(varianceDecomposition(fit))
```

Index

∗ **internal**
PACE-package, 3
pace_fit_streaming_gaussian, 19

ambientField, 3
ambientField(), *17*
ambientField,SpatialExperiment-method *(*ambientField*)*, 3
anchorGenes, 4

buildNeighbourhood, 5
buildNeighbourhood,SpatialExperiment-method *(*buildNeighbourhood*)*, 5

cellContamination, 6
cellContamination,PACEFit-method *(*cellContamination*)*, 6
cellTypes, 7
cellTypes,PACEFit-method *(*cellTypes*)*, 7
compactFit, 8

fitState, 9
fitState(), *26*
fitState,PACEFit-method *(*fitState*)*, 9

neighbourSlopes, 9
neighbourSlopes(), *3*, *9*, *12*, *18*, *26*
neighbourSlopes,PACEFit-method *(*neighbourSlopes*)*, 9
neighbourSlopes<- *(*neighbourSlopes*)*, 9
neighbourSlopes<-,PACEFit-method *(*neighbourSlopes*)*, 9

PACE *(*PACE-package*)*, 3
PACE-package, 3
pace_fit_streaming_gaussian, 19
paceDecompose, 10
paceDecompose(), *8*, *11–14*
paceDecompose,PACEFit-method *(*paceDecompose*)*, 10
paceDrivers, 11
paceDrivers(), *8*, *12–14*
paceDrivers,PACEFit-method *(*paceDrivers*)*, 11
PACEFit, *3*, *5–12*, 12, *13–15*, *17*, *18*, *20–28*
paceFit, 13
paceFit(), *3*, *12*, *14*
paceFit,SpatialExperiment-method *(*paceFit*)*, 13
PACEFit-class *(*PACEFit*)*, 12
paceModel, 14
paceModel(), *10*, *13*, *16*, *18*
paceModel,SpatialExperiment-method *(*paceModel*)*, 14
paceModelGaussian, 16
paceModelGaussian,SpatialExperiment-method *(*paceModelGaussian*)*, 16
paceParams, 17
paceParams,PACEFit-method *(*paceParams*)*, 17
paceShrink, 18
paceShrink(), *11–14*
paceShrink,PACEFit-method *(*paceShrink*)*, 18
pairVariance, 20
pairVariance,PACEFit-method *(*pairVariance*)*, 20
plotDecomposition, 21
plotDrivers, 21
plotPairHeatmap, 22
plotPairHeatmap(), *20*
plotProximity, 23
plotResponseCurve, 24
plotResponseCurve(), *25*
plotResponseMap, 25

show,PACEFit-method *(*PACEFit*)*, 12
SpatialExperiment::SpatialExperiment, *3–5*, *10*, *11*, *13*, *14*, *16*, *23–26*
stashPACE, 26
stashPACE(), *8*

topDrivers, 27
topDrivers(), *3*, *9*, *11*, *12*, *26*
topDrivers,PACEFit-method *(*topDrivers*)*, 27
topDrivers<- *(*topDrivers*)*, 27
topDrivers<-,PACEFit-method *(*topDrivers*)*, 27

varianceDecomposition, 28
varianceDecomposition(), *3*, *9, 10*, *12*, *26*
varianceDecomposition,PACEFit-method
(varianceDecomposition), 28
varianceDecomposition<-
(varianceDecomposition), 28
varianceDecomposition<-,PACEFit-method
(varianceDecomposition), 28