Package 'crossarrayEWAS'

November 25, 2026

Type Package

Title Design-Aware Region and Block Detection for Cross-Array Longitudinal EWAS

Version 1.0.0

Description Estimation and region-calling routines for epigenome-wide association studies that combine Illumina HumanMethylation450 and MethylationEPIC samples and follow subjects over more than one visit. Probe-level effects are fitted within subject, so subject identity, array type and chip are absorbed rather than modelled; adjacent probes are grouped by within-subject co-methylation; regions are obtained by total-variation denoising with a distance-decaying penalty, which places boundaries where the effect track jumps instead of thresholding a smooth curve; and long-range blocks are called from a three-state hidden semi-Markov model whose transition probabilities decay with genomic distance. Selection stability and permutation nulls resample subjects, not samples, so a subject's visit series is never split.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.5.0)

Imports GenomicRanges, IRanges, Matrix, S4Vectors, SummarizedExperiment, jsonlite, limma, methods, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle, minfi, IlluminaHumanMethylation450kanno.ilmn12.hg19, IlluminaHumanMethylationEPICanno.ilm10b4.hg19

VignetteBuilder knitr

biocViews DNAMethylation, DifferentialMethylation, Epigenetics, MethylationArray, Regression, Software

BiocType Software

URL https://github.com/bioinfbrad/crossarrayEWAS

BugReports https://github.com/bioinfbrad/crossarrayEWAS/issues

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

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

git_branch RELEASE_3_24

git_last_commit 4ba7bf9

git_last_commit_date 2026-11-23

Repository Bioconductor 3.24

Date/Publication 2026-11-24

Author Katarzyna Kamieniecka [aut, cre] (ORCID: <https://orcid.org/0009-0004-2454-5950>), Samantha McLean [aut] (ORCID: <https://orcid.org/0000-0001-9328-8928>), Krzysztof Poterlowicz [aut] (ORCID: <https://orcid.org/0000-0001-6173-5674>)

Maintainer Katarzyna Kamieniecka <kkamieni@bradford.ac.uk>

Contents

crossarrayEWAS-package . . . 2
as_granges . . . 3
call_blocks . . . 4
call_regions . . . 5
comethylation_clusters . . . 6
distance_penalty . . . 7
ewas_blocks . . . 8
ewas_regions . . . 10
fit_block_hsmm . . . 12
fit_within . . . 14
fit_within_se . . . 15
moderate_variance . . . 16
naive_permutation . . . 17
probe_coords . . . 18
read_f64 . . . 18
sort_probes . . . 19
stability_selection . . . 20
tv_denoise . . . 21
within_subject_permutation . . . 22

Index **23**

`crossarrayEWAS-package` *crossarrayEWAS: design-aware EWAS region and block detection*

Description

Numerical core of the cross-array harmonisation pipeline for longitudinal epigenome-wide association studies. The exported functions cover the two stages the pipeline replaces: probe-level within-subject estimation with region calling (`fit_within`, `comethylation_clusters`, `distance_penalty`, `tv_denoise`, `call_regions`), and long-range block detection (`fit_block_hsmm`, `call_blocks`).

Provenance

These routines are an R port of the pipeline's Python reference module, and parts of both were written with AI assistance (Assisted-by: Claude). The port's deliberate divergences from the reference are recorded at each function under 'Notes carried over from the pipeline source'.

Author(s)

Maintainer: Katarzyna Kamieniecka <kkamieni@bradford.ac.uk> (ORCID)

Authors:

- Katarzyna Kamieniecka <kkamieni@bradford.ac.uk> (ORCID)
- Samantha McLean <S.L.Mclean@bradford.ac.uk> (ORCID)
- Krzysztof Poterlowicz <K.Poterlowicz1@bradford.ac.uk> (ORCID)

See Also

Useful links:

- https://github.com/bioinfbrad/crossarrayEWAS
- Report bugs at https://github.com/bioinfbrad/crossarrayEWAS/issues

---

`as_granges` *Region or block calls as a GRanges*

Description

Converts the table returned by `call_regions()` or `call_blocks()` into a GenomicRanges::GRanges, with every remaining column as metadata.

Usage

```
as_granges(x)
```

Arguments

`x`	The list or table from `call_regions()`, or the table from `call_blocks()`.

Details

Given the whole list from `call_regions()`, the elements beside `table` — `max_abs_z` — are kept in the object's `metadata()` rather than dropped, since that value is the one a permutation null is compared against. The table's `width` column is dropped: `GRanges` computes it, and `width` is not a permitted metadata name.

Coordinates pass through unchanged. Both callers report the first and last probe or cluster position, 1-based and inclusive, which is what `GRanges` means by `start` and `end`.

Value

A `GRanges` with one range per called region or block.

Examples

```
tab <- data.frame(chr = "chr1", start = 100L, end = 400L, width = 301L,
                  n_probes = 4L, effect_M = 0.3, se = 0.02, z = 15,
                  p_pointwise = 1e-50, theta = 0.3,
                  probe_start = 0L, probe_end = 3L)
as_granges(list(table = tab, max_abs_z = 15))
```

`call_blocks` *Call blocks from the state posterior*

Description

Merges runs of clusters whose posterior for a non-neutral state exceeds `min_post`, so each block carries a calibrated confidence rather than a binary permutation call. Pass `neutral` from the fit: assuming the middle column is neutral mislabels every block when all cluster effects fall on one side of zero.

Usage

```
call_blocks(
  chrom,
  start,
  end,
  post,
  min_post = 0.8,
  min_clusters = 3L,
  neutral = 2L
)
```

Arguments

`chrom`	chromosome per cluster.
`start`	cluster start position.
`end`	cluster end position.
`post`	clusters x 3 posterior matrix from `fit_block_hsmm()`.
`min_post`	minimum posterior for a cluster to join a block.
`min_clusters`	minimum clusters per reported block.
`neutral`	1-based neutral column, `fit_block_hsmm()$neutral`.

Details

Notes carried over from the pipeline source:

Maximal runs whose posterior for a non-neutral state exceeds `min_post`.

The reported block posterior is the mean over the block's clusters, so a block carries a calibrated confidence rather than a binary permutation call.

`neutral` is the column of `post` holding the state pinned at mu = 0, i.e. fit_block_hsmm()$neutral – pass it. Columns are ordered by mu, so a state left of that column is hypomethylated relative to no change and one to the right is hypermethylated. Assuming the middle column is neutral mislabels every block when all cluster effects fall on one side of zero: the pinned state then sorts to an end and the middle column is a genuine effect state.

Value

data frame with `chr`, `start`, `end`, `width` (inclusive, `end - start + 1`), `n_clusters`, `direction` and `posterior`; zero rows if nothing passes.

Examples

```
set.seed(1)
chrom <- rep("chr1", 40)
pos <- as.integer(seq(1e5, 5e6, length.out = 40))
y <- c(rnorm(15, 0.5, 0.1), rnorm(25, 0, 0.1))
se <- rep(0.1, 40)
fit <- fit_block_hsmm(y, se, pos, chrom, n_iter = 20L)
call_blocks(chrom, pos, pos + 500L, fit$posterior,
            neutral = fit$neutral, min_clusters = 3L)
```

`call_regions` *Turn a denoised effect track into regions*

Description

Region boundaries are the jumps of the piecewise-constant solution (or a change of chromosome or cluster), so they are estimated rather than thresholded off a smooth curve. The region statistic is the precision- weighted mean effect over its own standard error.

Usage

```
call_regions(
  chrom,
  pos,
  beta,
  se,
  theta,
  min_probes = 3L,
  min_effect = 0.05,
  tol = 1e-06,
  cluster = NULL
)
```

Arguments

`chrom`	chromosome per probe.
`pos`	genomic position per probe.
`beta`	effect estimate per probe.
`se`	standard error per probe.
`theta`	denoised track from `tv_denoise()`.
`min_probes`	minimum probes per reported region.
`min_effect`	minimum absolute weighted mean effect.
`tol`	tolerance for treating two `theta` values as equal.
`cluster`	optional cluster id per probe; a change of id forces a boundary.

Details

Notes carried over from the pipeline source:

Turn the denoised track into regions.

Segment boundaries are where the piecewise-constant solution jumps (or the chromosome/cluster changes), so boundaries are estimated, not thresholded off a smooth curve. The region statistic is the precision-weighted mean effect over its own standard error, which accounts for probes in a region carrying very different amounts of information.

Value

list with `table` (one row per region: `chr`, `start`, `end`, effect, standard error, z, probe count and 0-based probe index range) and `max_abs_z`.

Examples

```
chrom <- rep("chr1", 12)
pos <- as.integer(seq(1000, 3200, length.out = 12))
beta <- c(rep(0.4, 6), rep(-0.1, 6))
se <- rep(0.1, 12)
lam <- distance_penalty(pos, lam0 = 1, w = 1 / se^2)
theta <- tv_denoise(beta, w = 1 / se^2, lam = lam, n_iter = 200L)
res <- call_regions(chrom, pos, beta, se, theta, min_probes = 3L)
res$table[, c("chr", "start", "end")]
```

---

comethylation_clusters

Group adjacent probes that co-vary within subject

Description

Clusters genomically adjacent probes whose within-subject residuals correlate, so a region means co-variation of the longitudinal signal rather than between-subject or between-array variation.

Usage

```
comethylation_clusters(
  chrom,
  pos,
  resid,
  max_gap = 1000L,
  rho_min = 0.3,
  chunk = 50000L
)
```

Arguments

`chrom`	chromosome per probe.
`pos`	genomic position per probe.
`resid`	probes x samples matrix of within-subject residuals.
`max_gap`	largest gap (bp) across which two probes may be linked.
`rho_min`	minimum residual correlation for a link.
`chunk`	probes per correlation chunk; memory/speed trade-off only.

Details

Notes carried over from the pipeline source:

Cluster probes that are both physically close and empirically co-methylated.

Returns an integer cluster id per probe (-1 = singleton). `resid` is the within-subject residual matrix, so the correlation is between-visit co-variation – not between-subject or between-array variation, which is what a region is supposed to mean.

Only genomically adjacent pairs are ever linked, so the union-find in the Python reduces to run-length grouping over the sorted order. Cluster ids are assigned in genomic order; the Python numbers them by the smallest input index in each group, which is the same thing whenever the input is sorted (stage 04 always sorts).

Value

integer vector of cluster ids, one per input probe, in input order; probes in no multi-probe cluster are returned as -1.

Examples

```
set.seed(1)
n_probes <- 20; n_sub <- 6
subject <- rep(sprintf("S%02d", seq_len(n_sub)), each = 2)
exposure <- rep(c(0, 1), times = n_sub)
M <- matrix(rnorm(n_probes * length(subject)), nrow = n_probes)
M[1:3, exposure == 1] <- M[1:3, exposure == 1] + 1.2
# residuals of a model carrying subject and exposure, which is what
# ewas_regions() builds internally
design <- stats::model.matrix(~ subject + exposure)
resid <- t(stats::lm.fit(design, t(M))$residuals)
chrom <- rep("chr1", n_probes)
pos <- as.integer(seq(1000, 1000 + 200 * n_probes, length.out = n_probes))
table(comethylation_clusters(chrom, pos, resid, max_gap = 5000L))
```

`distance_penalty` *Distance-decaying penalty for the denoising step*

Description

Builds the per-gap penalty `lam0 * scale * exp(-d / decay_bp)`. When precision weights are supplied `scale` is their median, which makes `lam0` dimensionless and the same grid usable on M-values, beta values or another cohort.

Usage

```
distance_penalty(pos, lam0, decay_bp = 1000, w = NULL, lam_floor = 0)
```

Arguments

`pos`	genomic position per probe, sorted.
`lam0`	dimensionless penalty scale.
`decay_bp`	distance (bp) over which the penalty decays by 1/e.
`w`	optional precision weights, used only to set `scale`.
`lam_floor`	lower bound on the returned penalty.

Details

Notes carried over from the pipeline source:

lam_i = lam0 * scale * exp(-d_i / decay_bp) over successive distances.

`scale` is median(w) when precision weights are supplied, which makes lam0 DIMENSIONLESS: lam0 ~ 1 means "one probe's worth of precision resists one unit of jump", so the same grid is usable on M-values, beta values or a different cohort. Without it lam0 must be retuned whenever the noise level changes – the practical reason a fixed loess span does not transfer between studies.

Value

numeric vector of penalties, one per gap (`length(pos) - 1`).

Examples

```
pos <- as.integer(c(1000, 1100, 1400, 9000))
distance_penalty(pos, lam0 = 1, decay_bp = 1000)
```

`ewas_blocks` *Call long-range methylation blocks from a methylation object*

Description

Collapses co-methylated probes into clusters, fits the within-subject exposure effect per cluster, and calls blocks from a three-state hidden semi-Markov model whose transition probabilities decay with genomic distance. This is what `05_blocks_hsmm.R` does in the pipeline.

Usage

```
ewas_blocks(
  object,
  exposure,
  subject,
  covars = NULL,
  assay = NULL,
  max_gap = 1500L,
  rho_min = 0.2,
  length_scale = 250000,
  min_post = 0.8,
  min_clusters = 3L,
  min_fit_clusters = 200L,
  var_method = "limma",
  seed = 1L,
  verbose = FALSE
)
```

Arguments

`object`	A `SummarizedExperiment`, `RangedSummarizedExperiment` or `GenomicRatioSet` with probe coordinates in `rowRanges()` or `rowData()`.
`exposure`	Character scalar naming a `colData` column, or a vector of length `ncol(object)`. Must vary within subject.

subject	Character scalar naming a colData column, or a vector of length ncol(object).
covars	Optional character vector naming colData columns, or a matrix with ncol(object) rows.
assay	Assay index or name; NULL takes the assay named M, or the only assay. Beta values are converted with a message.
max_gap, rho_min	Clustering, as in ewas_regions(), with the looser defaults blocks are called at.
length_scale	Distance scale of the state-transition decay, in base pairs. The 250 kb default matches the loess window it replaces.
min_post, min_clusters	Block floors, passed to call_blocks().
min_fit_clusters	Refuse to fit the block model on fewer clusters than this.
var_method	Variance moderation, "limma" or "mom", passed to fit_within().
seed	Seed for the deterministic subject-fold split used when lambda is cross-validated. The permutation and subsampling replicates are drawn from the session's RNG stream, so call set.seed() before ewas_regions() to make those reproducible; the function does not reset a stream it does not own.
verbose	Report each stage as it runs.

Details

Blocks are a cluster-level statement, so the probe matrix is reduced first: probes that are close and co-methylated within subject are averaged into one profile, which is the cpgCollapse() idea with clusters defined by the data rather than by fixed widths. The three states are hypomethylated, neutral and hypermethylated with respect to the exposure; which column is neutral is read from the fit rather than assumed, and when every cluster effect falls on one side of zero only one direction can be called, which is reported in metadata().

The model needs enough clusters to be identifiable. min_fit_clusters refuses the fit below a floor rather than returning numbers from an unidentified model; relaxing the clustering merges probes further and yields fewer clusters, so the way out is to split clusters, not to relax them.

Value

A GenomicRanges::GRanges of called blocks with metadata columns from call_blocks(). The object's metadata() carries clusters, a GRanges of the collapsed clusters with their effects and state posteriors, and hsmm, the fitted state means, standard deviations, stationary distribution, neutral column, log-likelihood and iteration count.

See Also

ewas_regions() for regions, and fit_block_hsmm() and call_blocks() for the building blocks.

Examples

```
set.seed(1)
subject <- rep(sprintf("s%02d", 1:10), each = 3)
dose <- rep(c(0, 1, 2), times = 10)

# 60 clusters of three probes, so the block model has units to run along
pos <- sort(as.integer(outer(c(0L, 150L, 300L),
```

```
                              seq(1e5, by = 8000, length.out = 60L), "+")))
n_probes <- length(pos)

M <- matrix(rnorm(n_probes * length(subject), sd = 0.2), nrow = n_probes)
M <- M + rnorm(length(subject), sd = 0.4)[col(M)]
M <- M + rnorm(n_probes, sd = 1)[row(M)]
shallow <- 60:120                                    # planted block
M[shallow, ] <- M[shallow, ] + 0.3 * rep(dose, each = length(shallow))

se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(M = M),
  rowData = data.frame(chr = "chr1", pos = pos),
  colData = data.frame(dose = dose, subject = subject))
blocks <- ewas_blocks(se, "dose", "subject", max_gap = 1000L,
                      rho_min = 0.2, min_fit_clusters = 20L)
blocks
```

`ewas_regions` *Call differentially methylated regions from a methylation object*

Description

The whole region analysis in one call: within-subject probe effects, co-methylation clusters, smoothness chosen by held-out subjects, total-variation region calling, and, when asked for, a permutation null and subject-level selection stability. This is what `04_dmr_ml.R` does in the pipeline.

Usage

```
ewas_regions(
  object,
  exposure,
  subject,
  covars = NULL,
  assay = NULL,
  max_gap = 1000L,
  rho_min = 0.3,
  lambda = c(0.05, 0.1, 0.25, 0.5, 1, 2, 4),
  cv_folds = 5L,
  decay_bp = 1000,
  min_probes = 3L,
  min_effect = 0.05,
  n_perm = 0L,
  perm_scheme = c("within", "naive"),
  n_boot = 0L,
  n_iter = 3000L,
  perm_iter = 800L,
  var_method = "limma",
  seed = 1L,
  verbose = FALSE
)
```

Arguments

object	A SummarizedExperiment, RangedSummarizedExperiment or GenomicRatioSet with probe coordinates in rowRanges() or rowData().
exposure	Character scalar naming a colData column, or a vector of length ncol(object). Must vary within subject.
subject	Character scalar naming a colData column, or a vector of length ncol(object).
covars	Optional character vector naming colData columns, or a matrix with ncol(object) rows.
assay	Assay index or name; NULL takes the assay named M, or the only assay. Beta values are converted with a message.
max_gap, rho_min	Clustering: maximum distance in base pairs between adjacent probes, and the minimum within-subject co-methylation needed to join them. Passed to comethylation_clusters()
lambda	Total-variation penalty. A single value is used as given; a vector is cross-validated over held-out subjects.
cv_folds	Number of subject folds used when lambda is a vector.
decay_bp	Distance scale over which the penalty decays, passed to distance_penalty().
min_probes, min_effect	Region floors, passed to call_regions().
n_perm	Permutation replicates for the family-wise p-value. 0, the default, skips the null.
perm_scheme	"within" permutes the exposure within subject; "naive" permutes across all samples.
n_boot	Subject half-sample replicates for selection stability. 0, the default, skips it.
n_iter, perm_iter	ADMM iterations for the observed track and for the permutation and stability replicates.
var_method	Variance moderation, "limma" or "mom", passed to fit_within().
seed	Seed for the deterministic subject-fold split used when lambda is cross-validated. The permutation and subsampling replicates are drawn from the session's RNG stream, so call set.seed() before ewas_regions() to make those reproducible; the function does not reset a stream it does not own.
verbose	Report each stage as it runs.

Details

The input is a SummarizedExperiment::SummarizedExperiment or anything extending it, which includes minfi's GenomicRatioSet. Rows are ordered by chromosome and position internally, so an object in array order is handled correctly; the returned ranges are in genomic order.

Effects are identified within subject: subject means are differenced out of both the data and the design, so subject identity, array generation and chip are absorbed rather than modelled, and only covariates that vary between a subject's visits contribute. The exposure must vary within subject; a subject with one visit contributes nothing, which is reported rather than silently dropped.

lambda of length one is used as the penalty. A vector, the default, is cross-validated over cv_folds folds of whole subjects, and the selected value and the score table are returned in the object's metadata().

With n_perm > 0 the exposure is permuted n_perm times, the whole region call is repeated, and each region's family-wise p-value is the fraction of null replicates whose maximum |z| reaches the

region's own, with the Monte Carlo standard error of that fraction beside it. The default scheme permutes the exposure among the visits of the same subject, which leaves subject, array, chip and the within-subject covariance untouched; `"naive"` shuffles across subjects, the scheme `bumphunter` uses, and is provided to quantify its mis-calibration rather than to be used.

With `n_boot > 0`, regions are re-called on repeated half-samples of subjects and `stability` reports the fraction of subsamples in which an overlapping region was called again.

Value

A GenomicRanges::GRanges of called regions, in genomic order, with per-region metadata columns `n_probes`, `effect_M`, `se`, `z`, `p_pointwise`, `theta`, and `p_fwer`, `p_fwer_mcse` and `stability` when those were requested. The object's `metadata()` carries `max_abs_z`, `lambda`, `lambda_cv`, `n_clusters`, `fit` (degrees of freedom and subject and sample counts), `perm_scheme` and `null_max_abs_z`.

See Also

`ewas_blocks()` for long-range blocks, and `fit_within()`, `comethylation_clusters()`, `tv_denoise()` and `call_regions()` for the building blocks this assembles.

Examples

```
set.seed(1)
subject <- rep(sprintf("s%02d", 1:8), each = 3)
dose <- rep(c(0, 1, 2), times = 8)

# probes in tight clusters separated by wide gaps, as on an array
pos <- as.integer(outer(c(0L, 150L, 300L), seq(1e5, by = 8000,
                                               length.out = 30L), "+"))
pos <- sort(pos)
n_probes <- length(pos)

M <- matrix(rnorm(n_probes * length(subject), sd = 0.2), nrow = n_probes)
M <- M + rnorm(length(subject), sd = 0.5)[col(M)]
M <- M + rnorm(n_probes, sd = 1)[row(M)]
M[25:30, ] <- M[25:30, ] + 0.6 * rep(dose, each = 6)   # planted region

se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(M = M),
  rowData = data.frame(chr = "chr1", pos = pos),
  colData = data.frame(dose = dose, subject = subject))
reg <- ewas_regions(se, "dose", "subject", lambda = 0.25, min_probes = 3L)
reg
```

`fit_block_hsmm` *Distance-aware three-state hidden semi-Markov block model*

Description

Fits hypo / neutral / hyper states over cluster-level effects with transition probabilities that decay with genomic distance, so block boundaries are not limited by a fixed smoothing window. The neutral state is pinned at mu = 0. Baum-Welch from these fixed starting values is deterministic.

Usage

```
fit_block_hsmm(
  y,
  se,
  pos,
  chrom,
  length_scale = 250000,
  n_iter = 60L,
  tol = 1e-05,
  seed = 1L
)
```

Arguments

y	effect estimate per cluster.
se	standard error per cluster.
pos	representative genomic position per cluster.
chrom	chromosome per cluster.
length_scale	distance (bp) over which state persistence decays.
n_iter	maximum EM iterations.
tol	relative log-likelihood tolerance.
seed	accepted and ignored; the fit is deterministic.

Details

Notes carried over from the pipeline source:

Three-state block HMM over collapsed open-sea cluster effects.

States are hypomethylated / neutral / hypermethylated; emissions are heteroscedastic, y_k ~ N(mu_s, sigma_s^2 + se_k^2), so a cluster estimated from few probes cannot drag a block on its own. Transitions are an explicit function of genomic distance. Fitted by Baum-Welch; posterior decoding gives soft boundaries and a per-block posterior in place of the loess block finder's thresholded curve.

DIVERGENCE from the Python, and the only one here: A(d) is never materialised. Because A = p I + (1 - p) 1 pi', the forward and backward updates collapse to alpha' A = p * alpha + (1 - p) * sum(alpha) * pi_stat A (B * beta) = p * (B * beta) + (1 - p) * sum(pi_stat * B * beta) which is the same arithmetic in a different association order (~1e-15 relative) and avoids an n x 3 x 3 array per EM iteration. The Python is left as the reference implementation because the GSE237561 results were produced with it; tests/test_equivalence.R measures the resulting difference in the posterior and in the called blocks.

seed is accepted and ignored, as in the Python: Baum-Welch from these fixed starting values is deterministic, so this fit – unlike stage 04's resampling – is reproducible across languages exactly.

Value

list with mu, sigma, pi (states ordered by mu), length_scale, posterior (clusters x 3), loglik, n_iter and neutral, the 1-based column pinned at mu = 0.

Examples

```
set.seed(1)
chrom <- rep("chr1", 40)
pos <- as.integer(seq(1e5, 5e6, length.out = 40))
y <- c(rnorm(15, 0.5, 0.1), rnorm(25, 0, 0.1))
se <- rep(0.1, 40)
fit <- fit_block_hsmm(y, se, pos, chrom, n_iter = 20L)
round(fit$mu, 3)
fit$neutral
```

`fit_within` *Within-subject (fixed-effects) probe-level fit*

Description

Fits the exposure effect within subject for every probe at once, eliminating subject intercepts by demeaning rather than estimating them, which is what makes the permutation and subsampling loops affordable.

Usage

```
fit_within(
  M,
  exposure,
  subject,
  covars = NULL,
  shrink_var = TRUE,
  var_method = "limma"
)
```

Arguments

`M`	numeric matrix, probes x samples (M-values).
`exposure`	numeric exposure, one value per sample.
`subject`	subject identifier, one per sample.
`covars`	optional samples x k matrix of TIME-VARYING covariates only; time-invariant ones are removed exactly by the transform and including them makes the design rank-deficient.
`shrink_var`	apply empirical-Bayes variance moderation.
`var_method`	method passed to `moderate_variance()`.

Details

Notes carried over from the pipeline source:

Within-subject (fixed-effects) estimate of the exposure effect per probe.

M probes x samples matrix of M-values exposure length-samples exposure subject length-samples subject identifier covars samples x k TIME-VARYING covariates only; the within transform removes time-invariant ones exactly and including them makes the design rank-deficient.

Subject intercepts are eliminated by demeaning rather than estimated, which is what makes the permutation and subsampling loops affordable.

Value

list with `beta`, `se`, `z` (one per probe), `df`, `n_subjects` and `n_samples`.

Examples

```
set.seed(1)
n_probes <- 20; n_sub <- 6
subject <- rep(sprintf("S%02d", seq_len(n_sub)), each = 2)
exposure <- rep(c(0, 1), times = n_sub)
M <- matrix(rnorm(n_probes * length(subject)), nrow = n_probes)
M[1:3, exposure == 1] <- M[1:3, exposure == 1] + 1.2
fit <- fit_within(M, exposure, subject)
str(fit[c("df", "n_subjects", "n_samples")])
head(fit$z, 3)
```

`fit_within_se` *Fit the within-subject exposure effect on a SummarizedExperiment*

Description

Extracts an assay and its design columns from a SummarizedExperiment::SummarizedExperiment (or anything extending it, including minfi's `GenomicRatioSet`) and passes them to `fit_within()`. The fit itself is the same code the pipeline runs; this function only resolves the object.

Usage

```
fit_within_se(
  object,
  exposure,
  subject,
  covars = NULL,
  assay = NULL,
  shrink_var = TRUE,
  var_method = "limma"
)
```

Arguments

`object`	A `SummarizedExperiment`, or any object with `assay()` and `colData()` methods.
`exposure`	Character scalar naming a `colData` column, or a vector of length `ncol(object)`. Must vary within subject.
`subject`	Character scalar naming a `colData` column, or a vector of length `ncol(object)`, giving the subject each sample belongs to.
`covars`	Optional character vector naming `colData` columns, or a matrix with `ncol(object)` rows.
`assay`	Assay index or name. `NULL`, the default, takes the assay named `M`, or the only assay when the object has one.
`shrink_var, var_method`	Passed to `fit_within()`.

Details

`exposure`, `subject` and `covars` may name `colData` columns or be given as vectors and a matrix. Named covariates are expanded with `stats::model.matrix()` and the intercept is dropped, since the within-subject transform removes it; a covariate with missing values is an error rather than a silently shortened design.

The estimators assume M-values. With `assay = NULL` the assay named `M` is used when the object has one, which is what makes a `GenomicRatioSet` behave as expected: `minfi` stores `Beta` first, so taking the first assay would analyse beta values. A beta-valued assay is converted to M-values with a message.

Value

The list returned by `fit_within()`: `beta`, `se`, `z`, `df`, `n_subjects`, `n_samples`. `beta`, `se` and `z` carry the assay's row names when it has them.

See Also

`fit_within()` for the matrix interface and for what the within-subject transform does to the design.

Examples

```
set.seed(1)
M <- matrix(rnorm(40), nrow = 10,
            dimnames = list(paste0("cg", 1:10), NULL))
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(M = M),
  colData = data.frame(dose = c(0, 1, 0, 1),
                       subject = c("s1", "s1", "s2", "s2")))
fit <- fit_within_se(se, "dose", "subject", shrink_var = FALSE)
head(fit$z)
```

---

moderate_variance *Empirical-Bayes shrinkage of per-probe residual variance*

Description

Shrinks per-probe residual variances towards a fitted prior, which is what keeps a probe with a fluke-small variance out of the top of the region ranking. With fewer than 100 finite positive variances the prior cannot be fitted and the input is returned unchanged.

Usage

```
moderate_variance(s2, df, method = c("limma", "mom"))
```

Arguments

`s2`	numeric vector of per-probe residual variances.
`df`	residual degrees of freedom of the fit that produced `s2` (scalar).
`method`	`"limma"` for limma::squeezeVar (default, preferred in R), `"mom"` for the method-of-moments fit ported from the Python reference.

Details

Notes carried over from the pipeline source:

Empirical-Bayes shrinkage of per-probe residual variance.

method = "mom" : the method-of-moments fit ported from ewasml.py method = "limma" : limma::squeezeVar, i.e. the reference implementation of the same model. Preferred in R – it is maintained, handles the degenerate cases, and this repository already depends on limma at stage 02.

Value

numeric vector of shrunken variances, same length as `s2`.

Examples

```
set.seed(1)
s2 <- rchisq(300, df = 8) / 8
summary(moderate_variance(s2, df = 8, method = "mom"))
```

`naive_permutation` *Permute the exposure across all samples*

Description

Free permutation of the exposure, the scheme bumphunter uses on the design column. Provided so the mis-calibration of that null on a longitudinal, multi-array panel can be demonstrated rather than asserted.

Usage

```
naive_permutation(exposure, subject = NULL)
```

Arguments

`exposure`	numeric exposure, one value per sample.
`subject`	ignored; accepted so the two permutation schemes are interchangeable as arguments.

Details

Notes carried over from the pipeline source:

Free permutation of the exposure across all samples – the scheme bumphunter uses on the design column. Retained only so the pipeline can demonstrate that it is mis-calibrated here (it breaks the subject pairing and reassigns exposures across array types).

Value

numeric vector, the exposure permuted across samples.

Examples

```
set.seed(1)
naive_permutation(c(0, 1, 0, 1))
```

probe_coords *Probe chromosome and position from an annotated object*

Description

Reads probe coordinates from `rowRanges()` when the object has them and from `rowData()` otherwise, and returns them in the form `comethylation_clusters()` and `call_regions()` take.

Usage

```
probe_coords(object)
```

Arguments

`object`	A `SummarizedExperiment` or `RangedSummarizedExperiment`.

Details

Coordinates come back in the object's own row order, unsorted: the region caller treats adjacent rows as adjacent probes, so ordering is the caller's decision and `sort_probes()` is how to make it. `rowData()` is searched for a chromosome column named `chr`, `chrom` or `seqnames` and a position column named `pos`, `position` or `start`, case-insensitively.

Value

A list with `chrom` (character) and `pos` (integer).

Examples

```
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(M = matrix(0, nrow = 3, ncol = 2)),
  rowData = data.frame(chr = "chr1", pos = c(10L, 20L, 30L)))
probe_coords(se)
```

read_f64 *Read a raw float64 matrix written by the pipeline*

Description

Reads the `<stem>.f64` / `<stem>_dims.json` pair that the Nextflow stages use to pass probe-level matrices between processes. This is a transport format, not an analysis format: it exists because a column-major double dump plus a JSON header round-trips between R and Python without either language's serialisation getting involved.

Usage

```
read_f64(stem)
```

Arguments

stem	path prefix without extension; <stem>.f64 holds the doubles and <stem>_dims.json the dimensions and dimnames.

Value

list with mat (numeric matrix, probes x samples), rownames and colnames (character or NULL).

Examples

```
tmp <- tempfile()
m <- matrix(as.double(1:6), nrow = 2)
writeBin(as.vector(m), paste0(tmp, ".f64"))
writeLines(jsonlite::toJSON(list(nrow = 2L, ncol = 3L), auto_unbox = TRUE),
           paste0(tmp, "_dims.json"))
read_f64(tmp)$mat
```

---

sort_probes *Order an object by probe position*

Description

Order an object by probe position

Usage

```
sort_probes(object)
```

Arguments

object	A SummarizedExperiment or RangedSummarizedExperiment.

Details

call_regions() and call_blocks() read runs of adjacent rows as runs of adjacent probes, so an object whose rows are in array order — which is not genomic order — produces regions that are not contiguous on the genome. This applies the ordering once, before the fit, so every downstream stage sees the same one.

Value

object with rows ordered by chromosome then position. The chromosome order is order()'s on the chromosome names as characters, which is stable but not karyotypic.

Examples

```
se <- SummarizedExperiment::SummarizedExperiment(
  assays = list(M = matrix(0, nrow = 3, ncol = 2)),
  rowData = data.frame(chr = "chr1", pos = c(30L, 10L, 20L)))
probe_coords(sort_probes(se))$pos
```

`stability_selection` *Subject-level stability selection*

Description

Meinshausen-Buhlmann stability selection resampling SUBJECTS, not samples: resampling samples would split a subject's visit series across train and test and leak the within-subject effect under test. Draws come from the caller's RNG stream: call set.seed() beforehand to make a run reproducible. There is deliberately no seed argument, because a library function must not reset its caller's stream.

Usage

```
stability_selection(fit_fn, subjects, n_boot = 100L, frac = 0.5)
```

Arguments

`fit_fn`	function of a subject subset returning a character vector of selected region keys.
`subjects`	vector of subject identifiers.
`n_boot`	number of subsamples.
`frac`	fraction of subjects per subsample.

Details

Notes carried over from the pipeline source:

Meinshausen-Buhlmann stability selection at the SUBJECT level.

fit_fn(subject_subset) must return a character vector of region keys selected on that subsample. Subsampling subjects (not samples) is required: resampling samples would split a subject's visit series across train and test and leak the within-subject effect being tested.

Draws come from the caller's RNG stream: call set.seed() before this function to make a run reproducible. It takes no seed argument because a library function must not reset the stream its caller is using – doing so silently changes every later draw in the session, and BiocCheck flags it. The Python twin keeps its `seed` argument: numpy's default_rng() builds a generator local to the call and never touches global state, so the hazard this avoids does not exist there.

Value

list with `freq` (named selection frequencies) and `n_boot`.

Examples

```
subjects <- sprintf("S%02d", 1:12)
fit_fn <- function(sub) if (length(sub) > 3) c("chr1:100-200") else character(0)
stability_selection(fit_fn, subjects, n_boot = 20L)$freq
```

`tv_denoise` *Total-variation denoising of a probe-level effect track*

Description

Fits a piecewise-constant effect track by total-variation regularisation, then reports each segment as the precision-weighted mean of the input over that segment, so a region effect stays an unbiased weighted mean in M-value units.

Usage

```
tv_denoise(
  y,
  w,
  lam,
  n_iter = 3000L,
  rho = NULL,
  tol = 1e-06,
  over_relax = 1.7
)
```

Arguments

`y`	numeric effect estimate per probe.
`w`	precision weight per probe, normally `1 / se^2`.
`lam`	penalty per gap; length `length(y) - 1`, as returned by `distance_penalty()`.
`n_iter`	maximum ADMM iterations.
`rho`	ADMM step size; `NULL` picks a scale-free default.
`tol`	convergence tolerance on the primal iterate.
`over_relax`	ADMM over-relaxation factor.

Details

Notes carried over from the pipeline source:

Weighted total-variation denoising by ADMM.

Solves min_theta 0.5*sum_i w_i (y_i - theta_i)^2 + sum_i lam_i |d theta_i| with y in genomic order within one contiguous run.

lam is length n-1. A distance-decayed lam is what makes this a *genomic* smoother: a large gap gets lam ~ 0 and is free to break, which is precisely the behaviour loess with a fixed span cannot express.

The returned track is DEBIASED: the l1 penalty chooses where the breakpoints are, and each segment is then reported as the precision-weighted mean of y over that segment (the relaxed-lasso convention). The reported region effect is therefore an unbiased weighted mean in M-value units, directly comparable with a per-probe coefficient.

Unlike the Python, the tridiagonal system is factorised once: (W + rho D'D) does not change across iterations, so re-solving it 3000 times with a fresh LAPACK call is wasted work.

Value

numeric vector of fitted values, one per probe.

Examples

```
chrom <- rep("chr1", 12)
pos <- as.integer(seq(1000, 3200, length.out = 12))
beta <- c(rep(0.4, 6), rep(-0.1, 6))
se <- rep(0.1, 12)
lam <- distance_penalty(pos, lam0 = 1, w = 1 / se^2)
theta <- tv_denoise(beta, w = 1 / se^2, lam = lam, n_iter = 200L)
round(theta, 3)
```

`within_subject_permutation`

Permute the exposure within subject

Description

The null that matches a longitudinal design: subject identity, array type, cohort, chip, visit count and the within-subject covariance are untouched, and only the pairing between visit and exposure is broken.

Usage

```
within_subject_permutation(exposure, subject)
```

Arguments

`exposure`	numeric exposure, one value per sample.
`subject`	subject identifier, one per sample.

Details

Notes carried over from the pipeline source:

Shuffle the exposure among the visits of the same subject.

This is the null that matches a longitudinal design: subject identity, array type, cohort, chip, number of visits and the within-subject covariance are all untouched, and only the pairing between visit and exposure is broken. A subject with a single usable visit contributes nothing under this null, exactly as it contributes nothing to the within-subject estimator.

Value

numeric vector, the exposure permuted within each subject.

Examples

```
set.seed(1)
within_subject_permutation(c(0, 1, 0, 1), c("A", "A", "B", "B"))
```

Index

∗ **internal**
    crossarrayEWAS-package, 2

as_granges, 3

call_blocks, *2*, 4
call_blocks(), *3*, *9*, *19*
call_regions, *2*, 5
call_regions(), *3*, *11, 12*, *18, 19*
comethylation_clusters, *2*, 6
comethylation_clusters(), *11, 12*, *18*
crossarrayEWAS
        *(*crossarrayEWAS-package*)*, 2
crossarrayEWAS-package, 2

distance_penalty, *2*, 7
distance_penalty(), *11*

ewas_blocks, 8
ewas_blocks(), *12*
ewas_regions, 10
ewas_regions(), *9*

fit_block_hsmm, *2*, 12
fit_block_hsmm(), *9*
fit_within, *2*, 14
fit_within(), *9*, *11, 12*, *15, 16*
fit_within_se, 15

GenomicRanges::GRanges, *3*, *9*, *12*

moderate_variance, 16

naive_permutation, 17

order(), *19*

probe_coords, 18

read_f64, 18

sort_probes, 19
sort_probes(), *18*
stability_selection, 20
stats::model.matrix(), *16*
SummarizedExperiment::SummarizedExperiment, *11*, *15*

tv_denoise, *2*, 21
tv_denoise(), *12*

within_subject_permutation, 22