

Package: **tewas** (via **r-universe**)

September 15, 2026

Type Package

Title Unit-Level Multiple Testing for Epigenome-Wide Association
Studies with an Anchor-Calibrated Knowledge Prior

Version 0.99.2

Date 2026-09-15

Description A multiple-testing engine for DNA methylation arrays (engine v1.6) and a frame layer for the designs of behavioral epigenetics. Probes are grouped into genomic-run units; each unit is tested with four complementary statistics (dense sum, coherence-gated flip, Simes, kernel matched filter) whose null distributions come from one shared permutation stream with a calibrated exponential tail below the permutation floor; the arm p-values are combined by the Cauchy combination and units are selected by weighted Benjamini-Hochberg with cross-fitted structural weights and, optionally, a per-probe knowledge prior from longitudinal within-person change measured in an external cohort. Anchor probes carried in the scan measure whether the prior is licensed (the lens) and whether the scan's null is calibrated (the inflation guard, read on an alive null ruler); the prior is tempered toward zero when it is not, and nothing is corrected. The frame layer declares a design (moderation, curvature, threshold, acceleration, methylation as predictor or as outcome), runs every arm through the same engine, merges selected units into regions with coordinates, annotates the probes with their target genes, and writes tables, trajectory figures and a self-declaring summary. Unit-level FDR is controlled under the stated assumptions; probe-level false discovery and the calibration of every arm were measured in registered simulations whose readings the package prints beside its results. The anchor panel and the alive ruler are shipped under inst/extdata as per-probe summaries only. Input is a beta matrix or a SummarizedExperiment (a minfi GenomicRatioSet included); selected regions, units and probes are returned as GRanges on request.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.5.0)

Imports stats, utils, graphics, grDevices, methods, data.table,
GenomicRanges, IRanges, SummarizedExperiment

Suggests viridisLite, cpgdirection, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

biocViews DNAMethylation, MethylationArray, Epigenetics,
MultipleComparison, StatisticalMethod, DifferentialMethylation,
GeneRegulation, Software

URL <https://github.com/teindor/tewas>

BugReports <https://github.com/teindor/tewas/issues>

RoxygenNote 7.3.2

Config/pak/sysreqs zlib1g-dev

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

Date/Publication 2026-09-15 11:59:02 UTC

RemoteUrl <https://github.com/BiocStaging/tewas>

RemoteRef HEAD

RemoteSha a806dea26df498f4db1036cd8c4034d19352bdf3

Contents

tewas_anchor_table	3
tewas_engine2	3
tewas_flip_J	5
tewas_focal_stats	6
tewas_frame	7
tewas_genes	9
tewas_granges	10
tewas_inverse_normal	11
tewas_panel	12
tewas_removability	12
tewas_report	13
tewas_ruler_stats	14
tewas_run_units	15
tewas_scan_stats	15
tewas_simulate	16
tewas_v16	17

Index	19
--------------	-----------

tewas_anchor_table	<i>Anchor table for the engine from the R11 panel (v1.6 roles)</i>
--------------------	--

Description

Roles as the registered kits assign them: N_null -> null_ref; A_core / B_core / C_core -> class_A / class_B / class_C; a positive control that passes the IQR rule is an ordinary tested probe (v1.6), one that was re-admitted is carried as excluded.

Usage

```
tewas_anchor_table(cpg, readmitted, panel = tewas_panel("anchors"))
```

Arguments

cpg	probe ids in matrix order.
readmitted	logical (length m): TRUE for anchors re-admitted below the IQR rule.
panel	the R11 panel.

Value

list anchor (data.frame row / role / readmitted), prior (score_A), family, positive_rows.

Examples

```
P <- tewas_panel("anchors")
cpg <- c(head(P$cpg, 50), sprintf("cg%08d", 1:10)) # 50 panel probes and 10 others, in matrix order
AT <- tewas_anchor_table(cpg, readmitted = rep(FALSE, 60), panel = P)
head(AT$anchor); table(AT$family); AT$positive_rows
```

tewas_engine2	<i>The tewas engine (v1.6): units, four arms, one permutation stream, weighted BH with the lens-tempered knowledge prior and the inflation guard</i>
---------------	--

Description

The engine of the methods paper. With prior = NULL the selection is the v1.4 selection (rej equals rej_noprior); with guard = NULL it is bit-identical to tewas 1.3.1. Nothing in the package corrects the statistics handed in.

Usage

```
tewas_engine2(z, p, yr, Xs, dfr, units, weights, chr, pos, unit_type = NULL,
  B = 400L, alpha = 0.05, alpha_local = 0.05, seed = 1L,
  lambdas = c(120, 230, 450), coh_gate = 0.15, ihw = TRUE,
  combine = c("acat", "minp"), prior = NULL, anchor = NULL, family = NULL,
  cap_prior = 10, lens = TRUE, guard = NULL, guard_stats = NULL)
```

Arguments

z, p, yr, Xs, dfr	the per-probe statistics of the scan, as returned by tewas_scan_stats (see tewas_engine for their definitions).
units, weights	the unit set (tewas_run_units) and its per-probe flip multipliers (all 0 for run units).
chr, pos	probe chromosome and position (length m); chromosomes are the cross-fitting folds of the lens gain.
unit_type	character parallel to units (e.g. "run500", "run2000", "gene"); the structural stratum label.
B	permutation draws (250 in every registered run).
alpha, alpha_local	unit-level BH level and the level of the local descent to probes.
seed	permutation-stream seed.
lambdas	matched-filter kernel dictionary in bp.
coh_gate	the flip arm participates when raw coherence is below this.
ihw	cross-fitted structural weighting (TRUE) or plain BH over units.
combine	how the arm p-values are combined: "acat" (the registered engine) or "minp" (the v1.4.1 candidate, not adopted).
prior	v1.5: per-probe knowledge prior (length m, NA = unknown), e.g. the R11 tone table's score_A; aggregated per unit, normalized, capped by cap_prior, tempered by the lens gain. NULL reproduces v1.4.
anchor	v1.5: data.frame row, role (null_ref, class_A, class_B, class_C, excluded, ...). Anchors are removed from every unit, never tested, never rejected (asserted); the lens is read from their z.
family	optional per-probe family label ("A", "B", "C", "none").
cap_prior	weight-ratio cap for the prior and the combined weight.
lens	compute the gain from the anchors (TRUE) or apply the prior untempered.
guard	v1.6: c(rate = 0.10, inflation = 1.25); when the null ruler's raw p < .05 rate or Efron inflation exceeds a bar the gain is 0 for every unit and the scan is flagged lens\$inflated. NULL reproduces 1.3.1 bit for bit.
guard_stats	optional c(rate =, inflation =) measured by the caller on the raw scan (the registered kits read them on the alive null ruler, tewas_ruler_stats).

Value

list: unit (data.frame: k, has_flip, stratum, the four arm p and z, p_unit, ihw_w, prior_w, gain, family, selected, selected_bh, selected_noprior, credit), rej (logical per probe, the v1.5/v1.6 selection), rej_bhonly (plain-BH ablation), rej_noprior (the v1.4 selection on the same permutation stream), combine, lens (calibration, gain, inflated, guard statistics), anchor_stats, n_anchor, n_units, n_selected, n_probes_rejected.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
M <- log2(sim$beta / (1 - sim$beta))
S <- tewas_scan_stats(M, tewas_inverse_normal(sim$y), sim$X) # per-probe statistics
U <- tewas_run_units(sim$chr, sim$pos) # adjacency units
fit <- tewas_engine2(S$z, S$p, S$yr, S$Xs, S$dfr, U$units, U$weights, sim$chr, sim$pos,
                    unit_type = U$unit_type, B = 30L, lens = FALSE)
sum(fit$rej); which(fit$rej)
head(fit$unit[fit$unit$selected, c("k", "stratum", "p_unit", "ihw_w", "credit")])
```

tewas_flip_J

Flip-arm amplitudes by evidence tier

Description

The direction-evidence amplitudes used to build the per-probe flip weights of gene units (weights of [tewas_engine2](#)).

Usage

```
tewas_flip_J()
```

Value

a named numeric vector of amplitudes in [0, 1] by evidence tier (measured / DIRECT_eQTM 0.988, S1 0.868, S2 0.643, all others 0 = abstain).

Examples

```
tewas_flip_J()
```

tewas_focal_stats	<i>Per-probe statistics of a focal regressor after a nuisance design</i>
-------------------	--

Description

The generalized scan of the frame layer: the OLS t of each probe on focal with Z partialled out of both (Frisch-Waugh-Lovell). With $Z = \text{cbind}(1, X)$ and $\text{focal} = y$ it is [tewas_scan_stats](#) exactly; with $\text{focal} = y * \text{mod}$ and $Z = \text{cbind}(1, X, y, \text{mod})$ it is the interaction t of $M \sim y * \text{mod} + X$; with $\text{focal} = (y - c)^2$ and $Z = \text{cbind}(1, X, y)$ the incremental curvature t . The engine's residual-permutation stream then permutes the residualized focal term under the reduced model (the Freedman-Lane form).

Usage

```
tewas_focal_stats(M, focal, Z)
```

Arguments

<code>M</code>	probes x samples matrix (M-values).
<code>focal</code>	the focal regressor (length <code>ncol(M)</code>).
<code>Z</code>	the nuisance design matrix (samples x q), intercept included; lower-order terms of the design belong here.

Value

list `p`, `z`, `t`, `yr`, `Xs`, `dfr`, `beta_m`, `beta_e`, `se_m`: `yr` is the residualized focal term and `Xs` the unit-norm residualized probe matrix (what [tewas_engine2](#) consumes); `beta_m` is the slope of `M` on the focal term (M-value units per unit focal), `beta_e` the slope of the focal term on `M`.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
M <- log2(sim$beta / (1 - sim$beta)); yy <- tewas_inverse_normal(sim$y)
## focal = the phenotype, Z = intercept and covariates: the registered scan
S <- tewas_focal_stats(M, yy, cbind(1, sim$X))
all.equal(S$t, tewas_scan_stats(M, yy, sim$X)$t)
## focal = the interaction, Z = intercept, covariates and both lower-order terms
md <- sim$mod - mean(sim$mod)
Si <- tewas_focal_stats(M, yy * md, cbind(1, sim$X, yy, md))
head(Si$t)
```

tewas_frame	<i>Declare and run a tewas design: DMR-level answers with the probes within</i>
-------------	---

Description

Runs the registered v1.6 engine on a declared design. The linear arm is the registered scan (asserted bit-identical in `tests/test_frame.R`). The moderation, departure and exponential arms were calibrated by the registered null report NR2 (V18, run 2026-09-14) on a real substrate: the moderation (continuous or binary moderator), quadratic, threshold and moderated-curvature arms held their null bands; the exponential arm did not and is opt-in, its NR2 reading printed beside any result; a moderator with excess kurtosis above 1 is reported with a caution. `fr$notes` and the Methods paragraph of `tewas_report` carry the standing of every arm that ran.

Usage

```
tewas_frame(beta, y, X = NULL, chr, pos, frame_role = c("predictor", "outcome"),
  moderation = FALSE, mod = NULL, mod2 = NULL, type = c("linear", "non-linear", "exponential"), center =
  knowledge = TRUE, mask = NULL, B = 250L, alpha = 0.05, alpha_local = 0.05, lambdas = c(120, 230, 450),
  coh_gate = 0.15, cap_prior = 10, guard = c(rate = 0.10, inflation = 1.25), iqr_min = 0.05, seed = 1L,
  diagnostic = TRUE, panel = if (knowledge) tewas_panel("anchors") else NULL, ruler = tewas_panel("ruler"))
```

Arguments

beta	probes x samples beta matrix, rownames = probe ids; or a SummarizedExperiment (a minfi GenomicRatioSet included): the beta matrix is taken from its assay, chr and pos from rowRanges() when not given, and y, X, mod, mod2 may name columns of colData(). The object is unpacked into the matrix-path inputs; the run is the same run.
y	the phenotype (0/1 or continuous; a continuous phenotype is rank-inverse-normal transformed, the registered rule); or a colData column name.
X	covariate matrix or data frame (samples x q) or NULL; or colData column names (factor and character columns are expanded to indicators).
chr, pos	probe chromosome and position in the order of rownames(beta).
frame_role	"predictor": methylation predicts the phenotype; "outcome": the phenotype is an exposure and methylation the outcome. The statistics are identical (partial-correlation symmetry); the report language and the reported slope differ.
moderation, mod, mod2	add the moderation arm: the probe-level y x mod (optionally x mod2) interaction with every lower-order term in the model, tested and pooled by the same engine on its own permutation stream.
type	"linear" (the registered engine); "non-linear" adds the quadratic and threshold departure arms, each an incremental test over the linear term; "exponential" adds the exp arm as well. Departure arms are combined per unit by the Cauchy combination into the shape omnibus (\$shape).

center	centering value of the transformed phenotype for the departure and moderation terms.
knowledge	use the R11 anchor panel (prior, lens, guard) and the alive ruler; FALSE runs the no-prior engine without anchors.
mask, B, alpha, alpha_local, lambdas, coh_gate, cap_prior, guard, iqr_min, seed, diagnostic, panel, ruler	as in tewas_v16 .
labels	named list of display labels (y, mod, mod2) used by tewas_report .
assay	for a SummarizedExperiment, the assay to take the beta values from (name or index); by default "Beta" when present, else "M" (converted to beta), else the first assay.

Value

an object of class `tewas_frame`: `$arms` (one entry per arm: `$units`, the unit table with genomic coordinates; `$dmrs`, the selected units merged into DMRs when they overlap; `$probes`; `$fit`; `$ruler`; `$lens`; `$diagnostic`), `$shape` (the departure omnibus per unit, with the probe-level Sasabuchi / Fieller follow-up as its "probes" attribute and the merged DMRs as "dmrs"), `$design`, `$notes` (what the run declares about itself), `$counts`, `$versions`.

See Also

[tewas_report](#), [tewas_granges](#), [tewas_focal_stats](#), [tewas_v16](#)

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
fr <- tewas_frame(sim$beta, sim$y, sim$X, sim$chr, sim$pos, moderation = TRUE, mod = sim$mod,
  type = "non-linear", knowledge = FALSE, ruler = NULL, B = 30L, iqr_min = 0,
  diagnostic = FALSE, labels = list(y = "phenotype", mod = "moderator"))
fr
names(fr$arms)
fr$arms$moderation$dmrs[, .(rank, coord, n_probes, p_unit, n_rejected)]
sim$planted$interaction # the rows of the planted interaction
fr$notes                # what the run declares about itself, including the NR2 standing of each arm
tewas_granges(fr, "dmrs", "moderation") # the regions as a GRanges
## the same run from a SummarizedExperiment: beta from the assay, coordinates from rowRanges(), y / X / mod from colData
se <- SummarizedExperiment::SummarizedExperiment(assays = list(Beta = sim$beta),
  rowRanges = GenomicRanges::GRanges(sim$chr, IRanges::IRanges(sim$pos, sim$pos)),
  colData = data.frame(y = sim$y, sim$X, mod = sim$mod))
fr2 <- tewas_frame(se, "y", c("age", "sex"), moderation = TRUE, mod = "mod", type = "non-linear",
  knowledge = FALSE, ruler = NULL, B = 30L, iqr_min = 0, diagnostic = FALSE)
identical(fr2$arms$linear$units$selected, fr$arms$linear$units$selected)
```

tewas_genes	<i>Annotate the probes, units and DMRs of a tewas_frame with their target genes</i>
-------------	---

Description

Calls `cpdirection::cpg_gene_pairs()` (or any function with the same contract) on the probes in scope and writes, for every probe, the genes it is related to, how each gene entered the candidate set (EPICv2 manifest, tissue catalog, measured eQTM, SMR) and the pair-specific direction of the methylation-expression association where the evidence is usable (GENE:+, GENE:-, GENE:?). Mapping provenance is distinct from direction evidence: a manifest annotation says Illumina annotates the CpG to the gene, not that the CpG regulates it. `tewas_report(genes = "auto")` runs this step when `cpdirection` is installed.

Usage

```
tewas_genes(fr, tissue = "blood", scope = c("selected", "rejected", "all"),
            probe_qc = "flag", include_brain = FALSE, universal = TRUE,
            annotator = NULL, ...)
```

Arguments

<code>fr</code>	a <code>tewas_frame</code> .
<code>tissue</code>	tissue context for the catalog layer: "blood", "nasal_epithelium" or "solid_tissue".
<code>scope</code>	which probes to query: "selected" (every member probe of a selected unit in any arm plus every rejected probe; default), "rejected" (rejected probes only) or "all" (every tested probe; slow on a full array).
<code>probe_qc, include_brain, universal</code>	passed to <code>cpdirection::cpg_gene_pairs()</code> . The default <code>probe_qc = "flag"</code> keeps masked probes and marks them; the frame's own mask has already removed the general-mask probes.
<code>annotator</code>	a function (<code>cpgs, tissue, probe_qc, include_brain, universal, verbose, ...</code>) returning one row per CpG x gene with at least <code>cpg_id</code> and <code>target_gene</code> ; default <code>cpdirection::cpg_gene_pairs</code> .
<code>...</code>	passed to the annotator.

Value

The frame with `fr$genes` (pairs: the full pair table; by_probe: one row per queried probe with genes, n_genes, gene_mapping, gene_direction, n_usable, probe_masked; counts) and the columns genes, n_genes, gene_direction, gene_mapping attached to every probe table, genes to every unit and DMR table (union of the member probes' genes) and to the shape tables.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
fr <- tewas_frame(sim$beta, sim$y, sim$X, sim$chr, sim$pos, knowledge = FALSE, ruler = NULL,
                 B = 30L, iqr_min = 0, diagnostic = FALSE)
## an annotator with the cpg_gene_pairs() contract (here a stand-in; with cpgdirection installed)
## leave annotator = NULL and the real catalogue is queried)
stub <- function(cpg_ids, tissue = "blood", ...) {
  P <- data.frame(cpg_id = cpg_ids, target_gene = paste0("GENE", seq_along(cpg_ids) %% 3),
                 mapping_primary = "EPICv2_manifest", mapping_sources = "EPICv2_manifest",
                 best_direction = 1, best_evidence = "catalogue", direction_tier = "B",
                 usable = TRUE, probe_masked = FALSE, stringsAsFactors = FALSE)
  attr(P, "qc_excluded_cpgs") <- character(0); P }
fg <- tewas_genes(fr, annotator = stub)
fg$arms$linear$dmrs[, .(coord, genes)]
head(fg$genes$by_probe)

## fr <- tewas_genes(fr, tissue = "blood") # cpgdirection::cpg_gene_pairs()
```

tewas_granges

Selected regions, units or probes of a tewas result as a GRanges

Description

Re-expresses the tables a run returned (`$dmrs`, `$units` or `$probes` of the arm; for a `tewas_v16()` result the unit and DMR tables are built from its `fit` and `probes`) as a `GRanges`, so that results can be overlapped with annotation, exported or plotted with the Bioconductor tools. Nothing is recomputed and no selection changes.

Usage

```
tewas_granges(x, what = c("dmrs", "units", "probes"), arm = "linear")
```

Arguments

<code>x</code>	a <code>tewas_frame</code> (from tewas_frame) or the list returned by tewas_v16 .
<code>what</code>	"dmrs": the selected units merged into regions (one range per DMR, ranked by unit p); "units": every tested unit with its statistics and selected; "probes": every probe in the scan (width-1 ranges, named by probe id) with its statistics and rejected.
<code>arm</code>	for a frame, the arm whose tables are used ("linear", "moderation", "quadratic", "threshold", "exponential", "mod_quadratic", or "shape" for the departure omnibus); ignored for a <code>tewas_v16()</code> result.

Value

a `GenomicRanges::GRanges` without `seqinfo` (the genome is the one the coordinates were given in), with the columns of the corresponding table as metadata columns (list columns excluded). An empty `GRanges` when nothing was selected.

See Also

[tewas_frame](#), [tewas_v16](#)

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
fr <- tewas_frame(sim$beta, sim$y, sim$X, sim$chr, sim$pos, knowledge = FALSE, ruler = NULL,
                  B = 30L, iqr_min = 0, diagnostic = FALSE)
tewas_granges(fr, "dmrs")
gr <- tewas_granges(fr, "probes"); gr[gr$rejected]
length(tewas_granges(fr, "units"))
```

tewas_inverse_normal	<i>Rank-inverse-normal transform (the v1.7 rule for continuous outcomes)</i>
----------------------	--

Description

Heavy-tailed continuous outcomes were the one outcome type that broke the exchangeability of the residual permutation in the null reports (methods section 7.5); continuous outcomes are transformed before the scan.

Usage

```
tewas_inverse_normal(y)
```

Arguments

`y` numeric vector; ties get the average rank.

Value

`qnorm((rank - 0.5) / n)`; a 0/1 vector is returned unchanged.

Examples

```
y <- rexp(50)
z <- tewas_inverse_normal(y)
round(range(z), 2); cor(y, z, method = "spearman")
tewas_inverse_normal(c(0, 1, 1, 0)) # binary: unchanged
```

tewas_panel	<i>Load a shipped anchor panel</i>
-------------	------------------------------------

Description

The two panels every registered run since R12 used, shipped with the package.

Usage

```
tewas_panel(which = c("anchors", "ruler"))
```

Arguments

which	"anchors": the R11 anchor panel v3 (families A / B / C, the null ruler N read by the lens, positive controls P). "ruler": the alive null ruler, panel v4 (R11 Amendment 3): 1,439 catalog-null, 450k-present probes from the least-changing decile of alive probes, read for the inflation guard and the removability diagnostic.
-------	---

Value

a data.table; attribute "sha256" carries the sha256 of the uncompressed csv as recorded in the registrations. Per-probe summaries only; no participant-level data.

Examples

```
P <- tewas_panel("anchors"); nrow(P); table(P$role); attr(P, "sha256")
R <- tewas_panel("ruler"); nrow(R); head(R$cpg)
```

tewas_removability	<i>Removability diagnostic (the v1.7 rule, report-only)</i>
--------------------	---

Description

The two-way cross-fitted held-out sigma0 path over ruler factors and the branch it implies. No correction is applied by anything in this package; the branch is reported (methods section 2.3).

Usage

```
tewas_removability(M, y, X = NULL, rows, idx = NULL, kmax = 10L, bar = 1.10,
  rho_bar = 0.75, guard_inflation = 1.25, seed = 160L)
```

Arguments

M, y, X, idx as in [tewas_scan_stats](#).
 rows the ruler's rows (the alive ruler present on the array).
 kmax, bar, rho_bar, guard_inflation the registered constants (10; 1.10; 0.75; 1.25).
 seed seed of the half split.

Value

list path (held-out sigma0 for $k = 0..kmax$, the two directions averaged), path12, path21, rho, branch (bar, partial, lit, mild), k_bar, sigma0.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
M <- log2(sim$beta / (1 - sim$beta))
d <- tewas_removability(M, tewas_inverse_normal(sim$y), sim$X, rows = 1:150, kmax = 3L)
d$branch; round(d$path, 3); d$rho
```

tewas_report

*Write the tables, figures and summary of a tewas_frame***Description**

For every arm: the unit table with coordinates (unit_table_<arm>), the selected units, the merged DMRs (dmrs_<arm>), the probe table and the rejected probes, each as csv, html and md; the genome-wide unit scan (fig_genome_<arm>.png) and the locus panels of the top DMRs (fig_loci_<arm>.png). For the run: the calibration table and figure (alive-ruler histogram against $N(0, 1)$ and the empirical null; QQ of the tested probes), the moderation figure (simple slopes at the moderator's mean and ± 1 SD with the Johnson-Neyman share), the shape figure (fitted curve of the DMR score with the Sasabuchi / Fieller verdict), design_notes.txt, run_meta.rds, and summary.md: design, the calibration of the scan (read first), results by arm, the DMR tables with coordinates, how to read, and a Methods and a Results paragraph written for the actual run.

Usage

```
tewas_report(fr, outdir = "tewas_out", top = 6, genes = c("auto", "none"),
  tissue = "blood", annotator = NULL)
```

Arguments

fr a [tewas_frame](#).
 outdir output directory (created).
 genes "auto" (default): if the frame carries no gene annotation and cpdirection is installed (or annotator is given), run [tewas_genes](#) first, so that every probe, unit and DMR table, the loci panels and the summary carry the target genes; "none" skips it.

tissue, annotator
passed to [tewas_genes](#).
top number of DMRs drawn in the locus, moderation and shape panels.

Value

the written files (invisibly).

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
fr <- tewas_frame(sim$beta, sim$y, sim$X, sim$chr, sim$pos, knowledge = FALSE, ruler = NULL,
                  B = 30L, iqr_min = 0, diagnostic = FALSE)
out <- file.path(tempdir(), "tewas_example")
files <- tewas_report(fr, out, top = 2, genes = "none")
basename(files)
cat(head(readLines(file.path(out, "summary.md")), 12), sep = "\n")
```

tewas_ruler_stats	<i>Null-ruler statistics on the raw scan</i>
-------------------	--

Description

The guard’s input (guard_stats of [tewas_engine2](#)) and the calibration report of every scan (methods sections 2.3 and 6.5).

Usage

```
tewas_ruler_stats(z, rows)
```

Arguments

z signed z statistics (length m).
rows the ruler’s rows.

Value

c(rate, bias, inflation, pi0, n): raw $p < .05$ rate, Efron’s central-matching median(z) and $IQR(z)/1.349$, Storey’s π_0 at 0.5, the number of finite ruler statistics.

Examples

```
z <- c(rnorm(700), rnorm(300, sd = 1.3)) # 1,000 probes; rows 701-1000 form the ruler
tewas_ruler_stats(z, rows = 701:1000)
```

tewas_run_units	<i>Run units by genomic adjacency (the registered unit set)</i>
-----------------	---

Description

Probes are sorted by chromosome and position; a run breaks where the gap to the previous probe exceeds the threshold (methods section 3).

Usage

```
tewas_run_units(chr, pos, tested = NULL, gaps = c(500L, 2000L))
```

Arguments

chr, pos	probe chromosome and position (length m).
tested	logical (length m): probes eligible for units (re-admitted anchors are not).
gaps	the gap thresholds in bp; both are used, so a probe can sit in a run500 and a run2000 unit.

Value

list units (row-index vectors of length ≥ 2), weights (all 0), unit_type.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
U <- tewas_run_units(sim$chr, sim$pos)
length(U$units); table(U$unit_type); head(lengths(U$units))
```

tewas_scan_stats	<i>Per-probe association statistics (the registered scan)</i>
------------------	---

Description

OLS of every probe on the outcome and covariates by one residualization (methods section 2.1).

Usage

```
tewas_scan_stats(M, y, X = NULL, idx = NULL)
```

Arguments

M	numeric matrix, probes x samples (M-values).
y	outcome (length ncol(M)), 0/1 or continuous (already transformed if that is the rule).
X	covariate matrix (samples x q) or NULL.
idx	optional sample indices (split halves, permutations).

Value

list p, z, t, yr, Xs, dfr - what `tewas_engine2` needs.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
M <- log2(sim$beta / (1 - sim$beta))
S <- tewas_scan_stats(M, tewas_inverse_normal(sim$y), sim$X)
head(S$z); S$dfr
```

tewas_simulate	<i>Simulate a small methylation array with planted loci</i>
----------------	---

Description

A synthetic beta matrix with a shared factor structure, two covariates, a moderator, and three planted loci: a linear association, a pure y x mod interaction and a pure curvature. It exists so that every example in the package runs in seconds on data whose truth is known; it is not a model of a real array (a real array carries the anchor panel and the alive ruler, which a synthetic array does not, so the knowledge layer is off on it).

Usage

```
tewas_simulate(m = 1500L, n = 120L, seed = 1L, effect = 0.7)
```

Arguments

m	number of probes (spread over three chromosomes; a multiple of 3, at least 300).
n	number of samples (at least 40).
seed	random seed.
effect	the planted effect size on the M-value scale.

Value

a list: beta (probes x samples, CpG rownames), y (continuous phenotype), X (covariate matrix: age, sex), mod (a continuous moderator), chr, pos, and planted, the row indices of the three planted loci (linear, interaction, curvature).

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
dim(sim$beta); lengths(sim$planted)
```

tewas_v16

*The registered v1.6 pipeline on a beta matrix***Description**

In the order of the registered kits: probe QC (autosomal cg probes, $\leq 5\%$ missing, beta IQR $\geq$ iqr_min; anchors of roles N / A / B / C re-admitted below the rule; positive controls tested), the logit transform, the rank-inverse-normal rule for a continuous outcome, the per-probe statistics, the run units, the alive ruler's statistics (guard input), tewas_engine2() with the R11 prior, the lens and the guard, and the removability diagnostic. Nothing is corrected; every calibration quantity is reported.

Usage

```
tewas_v16(beta, y, X = NULL, chr, pos, mask = NULL, B = 250L, alpha = 0.05,
  alpha_local = 0.05, lambdas = c(120, 230, 450), coh_gate = 0.15,
  cap_prior = 10, guard = c(rate = 0.10, inflation = 1.25), iqr_min = 0.05,
  seed = 1L, diagnostic = TRUE, panel = tewas_panel("anchors"),
  ruler = tewas_panel("ruler"), assay = NULL)
```

Arguments

beta	numeric matrix probes x samples, rownames = probe ids (cg...); or a SummarizedExperiment (a minfi GenomicRatioSet included): the beta matrix is taken from its assay, chr and pos from rowRanges() when not given, and y and X may name columns of colData(). The object is unpacked into the matrix-path inputs; the run is the same run.
y	outcome (length ncol(beta)): 0/1 or continuous; or a colData column name.
X	covariate matrix (samples x q) or NULL; cell fractions belong here; or colData column names.
chr, pos	probe chromosome ("chr1" ... "chr22") and position in the order of rownames(beta).
mask	optional probe ids to remove (the array's SeSAmE general mask).
B, alpha, alpha_local, lambdas, coh_gate, cap_prior	the registered engine constants.
guard	the inflation guard bars.
iqr_min	the alive rule (beta IQR).
seed	permutation-stream seed.
diagnostic	compute the removability diagnostic.
panel, ruler	the shipped panels.
assay	for a SummarizedExperiment, the assay to take the beta values from (name or index); by default "Beta" when present, else "M" (converted to beta), else the first assay.

Value

list: probes (data.table cpg / chr / pos / z / p / bh / rej_v16 / rej_noprior / family / tested / readmitted / ruler), units, fit (the [tewas_engine2](#) object), ruler (alive-ruler statistics on the raw scan), lens (gain, inflated, applied gain, guard), diagnostic, up, counts, versions.

Examples

```
sim <- tewas_simulate(m = 600, n = 100)
## a synthetic array carries no anchor-panel probe, so the knowledge layer is off (a message says so)
res <- tewas_v16(sim$beta, sim$y, sim$X, sim$chr, sim$pos, B = 30L, iqr_min = 0)
res$counts
res$probes[rej_v16 == TRUE, .(cpg, chr, pos, z, p)]
sim$planted$linear # the rows of the planted linear locus
## the same scan from a SummarizedExperiment (a GenomicRatioSet works the same way)
se <- SummarizedExperiment::SummarizedExperiment(assays = list(Beta = sim$beta),
  rowRanges = GenomicRanges::GRanges(sim$chr, IRanges::IRanges(sim$pos, sim$pos)),
  colData = data.frame(y = sim$y, sim$X))
res2 <- tewas_v16(se, "y", c("age", "sex"), B = 30L, iqr_min = 0)
identical(res2$probes$rej_v16, res$probes$rej_v16)
tewas_granges(res, "probes")[res$probes$rej_v16] # the rejected probes as a GRanges

## a real 450k / EPIC array: the shipped anchor panel and alive ruler are found by probe id
## res <- tewas_v16(beta, y, X, chr = manifest$chr, pos = manifest$pos, mask = sesame_general_mask)
## res$ruler; res$lens; res$diagnostic$branch
```

Index

`print.tewas_frame(tewas_frame)`, [7](#)

`tewas_anchor_table`, [3](#)

`tewas_engine`, [4](#)

`tewas_engine2`, [3](#), [5](#), [6](#), [14](#), [16](#), [18](#)

`tewas_flip_J`, [5](#)

`tewas_focal_stats`, [6](#), [8](#)

`tewas_frame`, [7](#), [10](#), [11](#), [13](#)

`tewas_genes`, [9](#), [13](#), [14](#)

`tewas_granges`, [8](#), [10](#)

`tewas_inverse_normal`, [11](#)

`tewas_panel`, [12](#)

`tewas_removability`, [12](#)

`tewas_report`, [7](#), [8](#), [13](#)

`tewas_ruler_stats`, [4](#), [14](#)

`tewas_run_units`, [4](#), [15](#)

`tewas_scan_stats`, [4](#), [6](#), [13](#), [15](#)

`tewas_simulate`, [16](#)

`tewas_v16`, [8](#), [10](#), [11](#), [17](#)